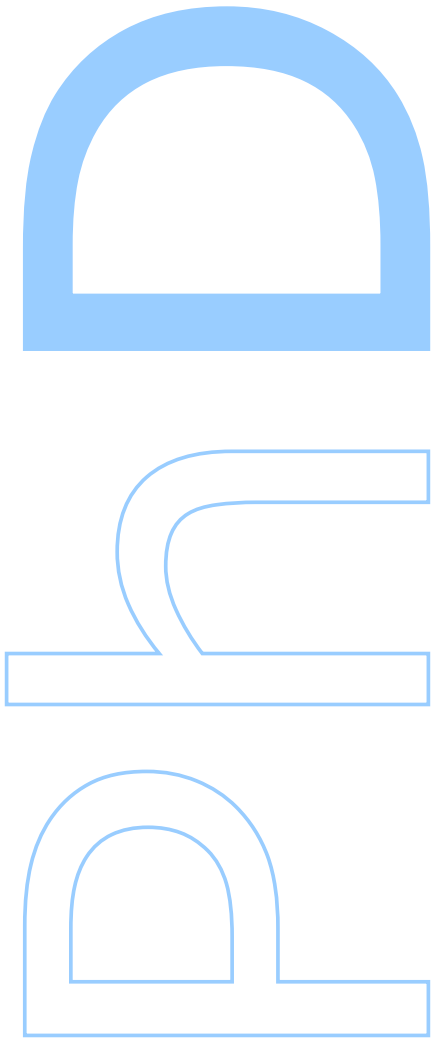
Towards early detection of faults and failures in complex systems

Christopher Harrison
Doutoramento em Ciência de Computadores
Departamento de Ciência de Computadores
2024

Orientador
Prof. Dra. Inês Dutra, Faculdade de Ciências, Departamento de Ciência de Computadores

Coorientador
Prof. Dr. Vitor Santos Costa, Faculdade de Ciências, Departamento de Ciência de Computadores



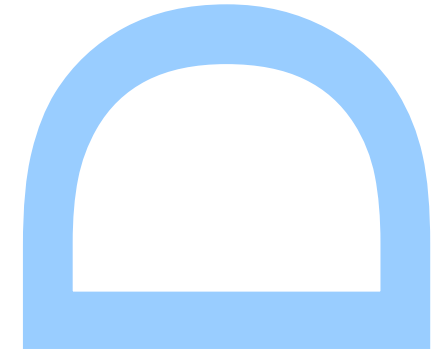
U.PORTO

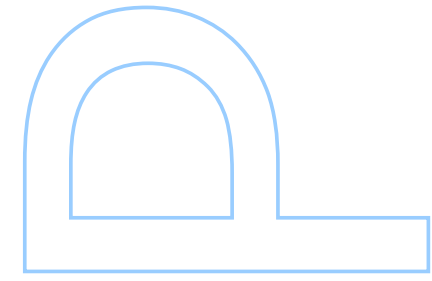
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____





UNIVERSIDADE DO PORTO

DOCTORAL THESIS

Towards early detection of faults and failures in complex systems

Author:
Christopher Harrison

Supervisor:
Inês Dutra

Co-supervisor:
Vitor Santos Costa

A thesis submitted in fulfilment of the requirements
for the degree of Programa Doutoramento em Ciência de Computadores

at the

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

September 2024

"Everything must be made as simple as possible "

Albert Einstein

" Life is a journey thats not measured in miles or years, but in experiences"

Jimmy Buffett

"I can't change the direction of the wind, but I can adjust my sails
to always reach my destination"

Jimmy Buffett

Acknowledgements

What an incredible journey filed with growth and life experiences that shaped me in profound ways. My journey would not have been possible without support.

To my rock, Huimin: Your sacrifices throughout my journey have been immense. While I haven't always expressed how deeply I appreciate it, you are, and always will be, my inspiration, joy, and light. May these words be a humble beginning towards expressing my true appreciation for your unwavering support. **I love you and thank you!**

Sophia and Alexis, my constant inspiration: I embarked on this PhD journey when Sophia was just 6 and Alexis wasn't even born. Alexis, you've only ever known me as a PhD student. This journey has always been for both of you. I've never lost this sight. You will both learn and grow, and I too will continue to learn and grow alongside you (hopefully not physically wider!). This journey has been an experience for all of us. I thank you for sharing it with me, allowing me to share this journey with you. Here's to our continuous growth together as amazing individuals. **Alexis and Sophia, I love you both with all my heart!**

To my advisor's Inês Dutra and Vitor Santos Costa, you both are amazing. From my first steps in Porto, to a case of MRSA, walks in the city and movies, you both provided an environment that goes way beyond an advisors scholastic support and have become like family. Obrigado!

To my lab mates, friends and collaborators along the way, especially: Sündüz Keleş, Henish Balu, David Aparìcio, Çağrl Zafer Soylu, Christine Kirkpatrick and Paul Gunther. Your friendship, support and occasional shared libations carried me through the valleys. Thank you!

To my parents; My gratitude goes to you for the support you've provided throughout the years. You've played a vital role in shaping who I am today, and I hope my accomplishment brings you pride. To my siblings, Tim, Jamie, and Lindsay; Thank you!

To the Faculdade de Ciências da Universidade do Porto for having me as a student during all these years. Last but not least, to my external reviewers Barton Miller and Irene Ong, Thank you!

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

Programa Doutoramento em Ciência de Computadores

Towards early detection of faults and failures in complex systems

by Christopher Harrison

Our research investigates the operational challenges of building and managing computational clusters. Through experience in constructing a cluster for searching the atSNP dataset, we evaluate databases for big data, develop data loading methods, and implement genomic visualizations. Operational issue emerge, in managing our cluster, that leading us to investigate hard drive failure prediction and explore user-submitted cluster task failures. Using machine learning, we try to understand and develop proactive mitigation techniques to address operational issues. Overall, our research aims to enhance the efficiency and reliability of computational cluster operations. Our contributes are in the fields of genomics and computational cluster operations.

Contents

List of Figures

List of Tables

Abbreviations

BCG Biomedical Computing Group. xv, 26, 27, 74, 105, 113, 118

BMI Department of Biostatistics and Medical Informatics. 26, 27, 105, 113

CPG Cost Per Gigabyte. 58

CPU Central Processing Unit. 2, 25–27, 37, 47, 49, 52, 75–77, 113

DNA DeoxyriboNucleic Acid. 2, 3, 5–9, 12, 13, 15, 31, 37–39, 44, 45, 53

ECC Error Correction Codes. 72

ETL Extract Transform Load. 29, 32, 33, 38–44, 48, 49, 51–53, 107, 108

FAR False Alarm Rate. 73

FDR Failure Detection Rate. 73

FMEA Failure Mode and Effective Analysis. 21, 22, 122, 123

GPGPU General Purpose Graphical Processing Unit. 123

GWAS Genomic Wide Association Study. 5, 10, 15, 123

HDD Hard Disk Drive. 21, 40, 54, 55, 57–61, 63, 72, 73, 81, 82, 84, 87–90, 92, 102, 103, 110, 118, 122, 155

HDDM Hard Disk Drive Model. xiii, xv, xvi, 4, 54, 63, 72, 87, 88, 90–94, 97–99, 101–103, 118, 122

HPC High-Performance Computing. xv, 25, 74–78, 118, 121

HPE Hewlett Packard Enterprise. 27, 78

HTC High-Throughput Computing. xv, 25, 74–77, 118, 121

HTCondor High Throughput Condor. 3, 25, 26, 33, 38, 47, 49, 51–53, 105–108, 110, 114, 118, 119

JSON JavaScript Object Notation. 51, 106

LMM Linear Mixed effect Model. 15

MLM Machine Learning Method. 4, 59, 72, 73, 91, 92, 97, 101–103, 113, 115, 118, 119, 122, 155

MTTF Mean Time To Failure. 75

PCA Principal Component Analysis. 111

PWM Position Weight Matrix. xiii, 2, 3, 5, 11–13, 15, 37–39, 45–47, 49, 50, 52, 53, 117

RAID Redundant Array of Inexpensive Drives. 17, 72

RCA Root Cause Analysis. 23, 71, 122, 123

RDBMS Relational DataBase Management System. 39, 40, 43

S.M.A.R.T. Self-Monitoring, Analysis, and Reporting Technology. 21, 57, 60, 61, 63, 72, 84, 91, 97, 101–103, 118, 121

SNP Single Nucleotide Polymorphism. 2, 3, 5, 7–13, 15, 29, 30, 37–40, 45–49, 52, 53, 117

SNP-PWM Single Nucleotide Polymorphism - Position Weight Matrix. 37, 46, 47

SQL Structured Query Language. 43

SSD Solid-State Disks. 21, 35, 58

SVG Scalable Vector Graphics. 50

SVM Support Vector Machine. 65–67, 72, 92

TB TeraByte. 27, 37, 39, 40, 50

TF Transcription Factors. 2, 3, 5, 7–10, 12, 13, 29, 37, 38, 45, 52, 53

With deepest gratitude, I dedicate this thesis to my family for their unwavering love and support.

Chapter 1

Introduction

Computational clusters, which combine the processing power of multiple computers, have expanded the boundaries of knowledge. Clusters enable previously impossible scientific inquiries and have fundamentally altered the scientific landscape. They achieve this by their size and parallel processing capabilities.

We examine the architectural and operational nuances of building a computational cluster that supports the atSNP search [1] database. In our work, we compare database options; identify methods to load billions of records; and provide a method to display the data easily to genomic researchers. A cluster is a powerful tool because of its ability to handle numerous tasks and search billions of records.

Although the potential of computational clusters is undeniable, the inner workings of their operations remain somewhat of a mystery due to their complex nature. There exists a significant knowledge gap in the field of cluster operations management, with numerous critical open questions. Although specific hardware, storage systems, and use cases may differ, the operational issues that plague these clusters often share a common thread. A research theme emerges, as the effective management of these complex systems presents a series of significant challenges.

We address computational cluster operational challenges through motivating events and tie these events to our research. Our work is motivated by actual events and the operational issues presented are personal experiences.

1.1 Motivation

We investigate the challenges of constructing and maintaining a computational cluster. To do so, we initially build a cluster that provides genomic researchers with the ability to search genomic data.

Building on work done by atSNP [2], our computational cluster helps genomic researchers identify and quantify the best gene sequence match to Transcription Factors (TF) and PWMs in a small window around the Single Nucleotide Polymorphism (SNP) location [1]. Addressing a demonstrated need within the genomic research community, we build a cluster that supports the advancement of genomic research.

We relied on the atSNP R package to sample the statistical significance of a given SNP-PWM within a DeoxyriboNucleic Acid (DNA) segment of two major human genomic repositories, JASPER and ENCODE. The SNP-Position Weight Matrix for Transcription Factors was calculated during the generation of the atSNP data set. This analysis matched all 132,946,852 known SNPs to the JASPAR and ENCODE, resulting in 307 billion SNP-PWM records. Generating the data set on the Open Science Grid required a total of 115,000 Central Processing Unit (CPU) hours to complete.

Our work is motivated by personally experienced actual events. We observed computational clusters are subjected to faults and failures that affect system integrity. For example, system integrity requires reliability in data disk storage systems, as they are the cornerstone of data integrity in computing. But in the course of our work, we experienced hard disk failures that lead to a data loss event. Thus, we explore historical data patterns and failure indicators to prevent these types of failures.

Preventing data loss is a key motivation of our work. Developing predictive models to anticipate hard disk failures can significantly enhance the resilience and reliability of data storage systems, thereby safeguarding critical data.

Beyond disk failures, we extend our research scope to understanding why user-submitted cluster tasks fail. In our work, we experienced failures in our genomic cluster data loading process, which was done as a user-submitted cluster task. We noticed that not all the data that we expected to be loaded was actually loaded. In trying to understand what happened, we found some of our cluster-scheduled data loading tasks failed without explanation.

A task failure event can result in computational inefficiencies and, more importantly, wasted user time trying to understand the nature of the failure event. Thus, we seek to understand these types of cluster tasks failures. Specifically, we examine the corresponding compute task failures from the cluster user submissions. Our work investigates various factors that contribute to task failures and employs a machine learning model to identify these factors in the hope of proactively mitigating them.

Motivated by the operational needs and challenges of computational cluster operations, we address cascading disk failures by improving hard disk drive reliability and investigate compute task efficiencies in computational clusters. This research contributes to the field of genomics and computer science.

1.2 Research Contribution

Our research introduces several novel contributions. Our contributions have three major areas. The first area is searching and making genomic research faster by implementing the genomic atSNP search database. The second, is understanding and predicting hard disk drive failures. The last, is understanding computational cluster tasks failures. We have a chapter for each area that discusses our contributions and the Appendix contains our published papers. We can summarize our contributions in this area as the following:

1.2.1 atSNP Search

- a genomic search tool that provides the knowledge of human SNPs on TF-DNA interactions based on PWM
- a comparison of MySQL, Apache Cassandra and Elasticsearch for use as a genomic SNP-PWM motif databases
- a big data loading method to extract transform and load genomic data using High Throughput Condor (HTCondor) [3] for ingestion into a database
- a generation visualization library producing composite motif logo plots

1.2.2 Hard disk drive faults and failures

- a technique to partition by Hard Disk Drive Models (HDDM), instead of the common all-in-one bucket approach

- a new Machine Learning Method (MLM) that produces accurate predictive model for HDDM
- identified failure attribute predictors which have not been mentioned previously in the literature

1.2.3 Computational cluster task failure prediction

- provide an overview of task submission success/failure rates within a production computational cluster
- identified key contributors to cluster task failures through MLM techniques of feature extraction

1.3 Organization of the Dissertation

Our work is organized to highlight our contributions and showcase that this body of work is the beginning of a journey, not the end. Although this work showcases our novel contributions to the field, it should be viewed as a step towards answering a larger question in the field of failure and fault prediction in computational systems. As such, we start our story by first explaining terms that will be used in later chapters. These definitions will be used in later chapters. We hope that the definition chapter will provide context to the importance of our work. This context is important, especially since our work spans numerous disciplines, from life sciences to distributed computational infrastructure to failure and fault prediction.

Our body of work is organized as follows: Concepts and Terminology; atSNP background and related work; atSNP search contributions and results; Hard disk drive faults and failures and cluster task failures; background and related work; Hard disk drive faults and failures contributions and results; Computational cluster task failure contributions and results; Conclusions; Future Work; Appendices; and Bibliography.

The appendices contain our peer reviewed published papers, with the exception being the last paper in the appendix. The last appendix is a work in progress, but has been published at arxiv.

Chapter 2

Concepts and Terminology

Preamble

In this chapter, we present concepts and terminology related to our work. Our work encompasses numerous domains, so this work will use terms spanning these domains including: Computer Science, Computer Engineering, Genetics, and Bioinformatics. This chapter will help in understanding our work through knowledge of existing domain-specific terms and background.

This chapter is organized as follows: atSNP Search; hard disk drive failures, faults; and computational cluster task failures.

2.1 atSNP Search

Our first contribution, *atSNP search* [1], is a genomic search engine database that allows researchers to better understand gene expression. Specifically, this genomic database helps researchers identify and quantify the best DNA sequence matches to a TF PWM. Our work attempts to simplify many of the biological terms and focuses on the computational terms used by the atSNP search project. Although biological terms and processes have little to do with computing, the terms are used as a way to refer to complex relationships and describe the results of the biological processes and the corresponding genomic data analysis. As such, the key concepts are: DNA, Genomic Wide Association Study (GWAS), SNP, TF, DNA sequencing, binding affinities, and PWM.

2.1.1 atSNP Biological Terms

2.1.1.1 Nucleotide

The most basic building block of the hereditary material used by all living species. Each nucleotide consists of three key components:

- **Nitrogenous base**: This can be either a purine (double-ringed structure) or a pyrimidine (single-ringed structure). The specific type of base determines how nucleotides pair with each other in DNA and RNA.
- **Pentose sugar**: This is a five-carbon sugar molecule, either ribose (in RNA) or deoxyribose (in DNA). The presence or absence of an oxygen atom in the second carbon distinguishes these two sugars.
- **Phosphate group**: This phosphate group provides a chemical linkage between nucleotides, allowing them to form long chains, which is the essential structure of nucleic acids.

The specific combination of these three components determines the unique identity and function of each nucleotide [4]. They play a vital role in the storage of genetic information. Each nucleotide is composed of one of four nitrogen-containing nucleobases (cytosine [C], guanine [G], adenine [A], or thymine [T]), a sugar called deoxyribose, and a phosphate group.

2.1.1.2 DeoxyriboNucleic Acid (DNA)

DNA is the foundational hereditary material used by all living species [5]. Often described as the biological library or blueprint of life [6], DNA is a polymer composed of two poly-nucleotide chains that twist around each other to form a double helix, as described by Watson and Crick [7]. This double helix carries the genetic information of all organisms. The length of the double helix is not fixed and is broken down into physical units called chromosomes.

DNA encode genes. Genes are the instructions for building proteins and molecules. Genes are not simply continuous stretches of DNA. They are organized into regions with distinct functions:

- **exons**: These are the coding regions of a gene, containing the instructions for protein synthesis.

- **introns**: These are non-coding regions that interrupt exons and are spliced out during gene expression.
- **regulating regions**: These regions flank genes (Section 2.1.1.11) and contain short specific DNA sequences called motifs. These motifs act as binding sites for proteins called TF.

TF bind to specific motifs based on the unique arrangement of nucleotides within the motif. This binding can either activate or repress the gene's transcription (copying DNA into RNA). By coordinating the binding of multiple TF to different motifs within regulating regions, cells can fine-tune gene expression, ensuring that the right proteins are produced at the right time and in the right amounts.

2.1.1.3 Transcription Factors (TF)

Transcription Factors are specialized proteins, also known as sequence-specific DNA binding factor, that bind to specific short DNA sequences called motifs within the regulatory regions of genes. These motifs act as control elements, influencing whether or not a gene is transcribed into RNA. By binding to motifs, transcription factors can activate or repress gene expression, regulating the production of cellular products encoded by genes.

2.1.1.4 Single Nucleotide Polymorphism (SNP)

A DNA sequence variation that occurs when a single nucleotide is altered in the genome sequence and the particular alteration is present in at least 1% of the population. It is important to understand that a SNP is a single letter change in a DNA sequence. SNP provide the basis of genetic variation and impacts the heterogeneity of a given species.

Most DNA of a given species is almost all identical to each other, except for small variations in the DNA sequence locations. Individuals within a species or subspecies can be traced and tracked by the unique SNP in their DNA sequence and comparing these changes with another individual of the same species.

Some single nucleotide changes can also exhibit phenotypic variation, such as skin, eye, or hair color. SNPs are the most common type of genetic variation, where a single nucleotide differs between individuals at the same position on the chromosome. These

variations are identified by unique IDs prefixed with "rs," followed by a number indicating the specific change in the DNA sequence. The impact of an SNP on the phenotype depends on its location within a gene and its variant allele:

- **coding region (exon)**: If a SNP alters a nucleotide within a coding region (exon), it can change the coded amino acid in the resulting protein. This change in amino acid sequence can affect the protein's shape, stability, or function, potentially leading to altered cellular processes and ultimately influencing an individual's phenotype. For example, five SNPs have been associated with human hair color in the Polish population [8].
- **regulatory region**: SNPs located outside the coding region but within regulatory regions upstream of the gene can affect gene expression by altering the binding affinity of TF. TF are proteins that bind to specific DNA sequences (motifs) and regulate the initiation of gene transcription. A SNP within a motif can strengthen or weaken the binding of a TF, leading to changes in the level of gene expression and potentially affecting the production of the encoded protein. This, in turn, can influence various cellular processes and contribute to phenotypic variation. An example of this can be seen in the association between brown eye color in rs12913832 individuals and SNPs in genes: TYR, TYRP1, and SLC24A4 [9].

Therefore, the location of a SNP within a gene plays a crucial role in determining its potential impact on protein function and, ultimately, on an individual's phenotype.

2.1.1.5 Exon

The coding region that can change the sequence of coded amino acids in the resulting protein. This change in amino acid sequence can affect the protein's shape, stability, or function, potentially leading to altered cellular processes and ultimately influencing an individual's observable traits.

2.1.1.6 Introns

These are non-coding regions that interrupt exons and are spliced out during gene expression.

2.1.1.7 Regulatory region

SNPs located outside the coding region, but within regulatory regions upstream of the gene, can affect gene expression by altering transcription factor binding.

2.1.1.8 Motifs

Short, specific DNA sequences located within regulatory regions of genes. These motifs act as binding sites for TF. The specific sequence of a motif determines which TF can bind to it, allowing targeted regulation of gene expression.

2.1.1.9 Alleles

Gene variants and are usually inherited from each parent. These variations arise from differences in the DNA sequence. Variations can range from SNPs to larger insertions or deletions of DNA segments. a variant of the sequence of nucleotides at a particular location, or locus, on a DNA molecule. The coded region provided the sequence to build a protein or an RNA molecule. An allele can differ in a single position through SNP or can also have insertions and deletions of up to several thousand DNA base pairs. Alleles

2.1.1.10 Phenotypic variation

Refers to the observable differences in traits among individuals within a population of the same species. These traits can encompass an organism's physical form (morphology), physiological and biochemical properties, behavior, and even the products of behavior. The underlying causes of phenotypic variation can be attributed to two main factors: genetic variation and environmental influences.

2.1.1.11 Genes

The fundamental units of heredity, residing on chromosomes and composed of deoxyribonucleic acid (DNA). Genes carry instructions for building proteins or RNA molecules. The coded instructions, the specific sequence of nucleotides on a DNA molecule to build a protein or an RNA molecule. They exist in different versions, called alleles. Imagine a gene as a blueprint for building a protein. Alleles represent variations in this blueprint that arise from differences in the DNA sequence at a specific location (locus) on a chromosome.

These variations can be as subtle as a single misplaced nucleotide or as dramatic as the insertion or deletion of larger DNA segments. An individual inherits two alleles for each gene, one from each biological parent. The specific combination of alleles that an individual has for a particular gene can influence the expression and function of the gene, which can lead to phenotypic variation.

2.1.1.12 Biological term relationships summary

- Genes are composed of DNA and exist in different versions called alleles
- Alleles can arise due to variations in the DNA sequence, including SNPs
- SNPs within regulatory regions can influence how TF bind to motifs, ultimately affecting gene expression
- TF regulate gene expression by binding to specific motifs within regulatory regions of genes
- The interactions between DNA sequence variation (SNPs and alleles) and transcription factor binding to motifs plays a crucial role in determining which genes are expressed and at what level, influencing cellular function and phenotypic variation

2.1.2 atSNP statistical model

The **a**ffinity **t**esting for impact of a SNP's (atSNP) on the TF binding in human genome is a statistical algorithm for binding affinity scores. atSNP was originally developed by Zuo [2], refined by Sunyoung Shin and Sündüz Keleş. This work was motivated by GWAS, that revealed that most disease-associated SNPs are located in regulatory regions within introns or in regions between genes [10]. Regulatory SNPs (rSNPs) are SNPs that affect gene regulation by changing the TF binding affinities of genomic sequences. Identifying potential rSNPs is crucial to understanding the mechanisms of disease. *In silico* methods that evaluate the impact of SNPs on TF binding affinities are not scalable for large-scale analysis.

Specifically, the atSNP statistical method is an affinity testing technique of regulatory SNP's. It was developed as an R bioconductor package [2] for identifying regulatory SNPs to test and identify regulatory SNPs. This method implements an importance sampling algorithm coupled with a first-order Markov model for background nucleotide sequences

to test the significance of affinity scores and SNP-driven changes in these scores [2], as can be seen in Figure 2.1. It offers three key functionalities:

- Binding Affinity Score Calculation: atSNP calculates binding affinity scores for both the reference allele (the most common DNA sequence at a specific location) and the SNP allele (a variation of the reference allele) using position weight matrices (PWMs). These PWMs represent the likelihood of specific DNA sequences binding to transcription factors (proteins that regulate gene expression)
- Allele-Specific p-value Estimation: atSNP computes p-values for the binding affinity scores of each allele. A p-value indicates the probability of observing a score as extreme as the one calculated, assuming there's no real difference between the reference and SNP alleles
- Affinity Score Change Significance Testing: atSNP calculates p-values specifically for the changes in binding affinity scores between the reference and SNP alleles. This allows researchers to identify SNPs that potentially disrupt or enhance the binding of transcription factors to regulatory regions in DNA.

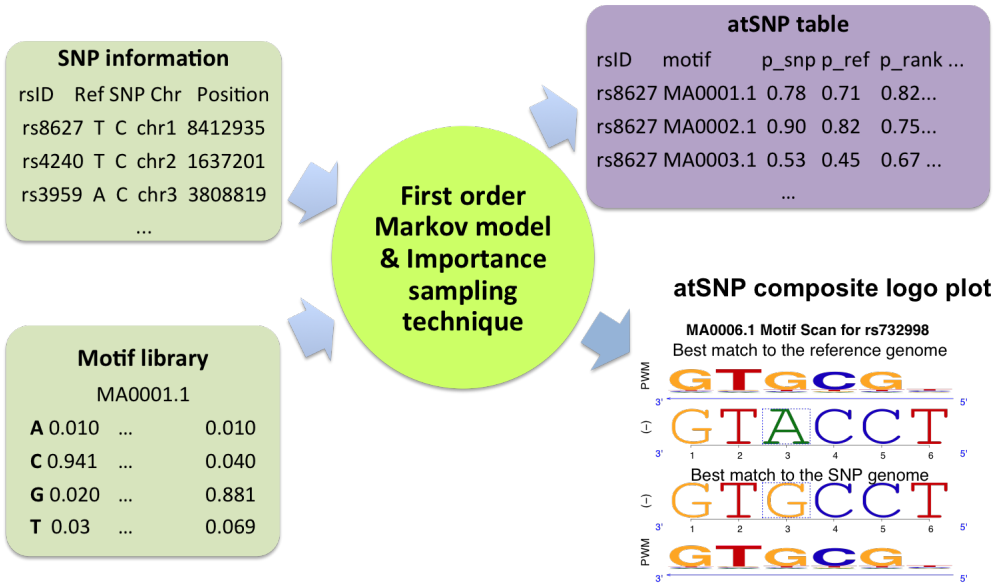


FIGURE 2.1: SNP plus motif results in a composite logo plot with the atSNP Markov model

2.1.2.1 Logo Plots

The PWM is visualized as a sequence motif, where the letter heights are scaled based on their information content. The thick letters are the best match and the thin letters are from

the reference or SNP allele. These plots reveal a SNP gain or loss of function based on their height and stacked letters.

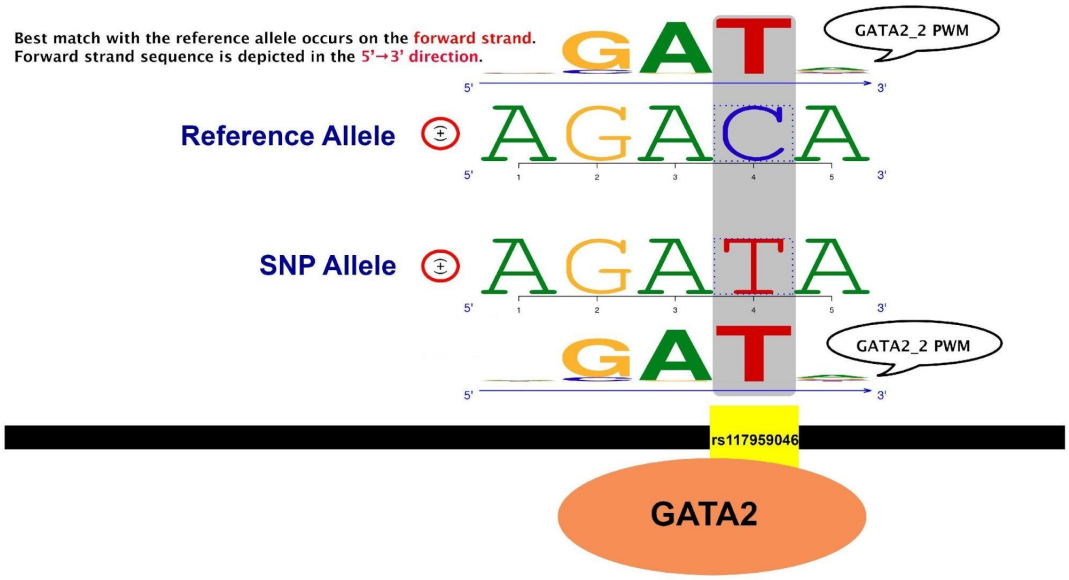


FIGURE 2.2: How to read the PWM of a logo plot, rs117959046 is the position of the reference gene, GATA2 is the Transcription Factor

2.1.2.2 JASPER

The JASPAR database contains a curated, non-redundant set of profiles, derived from published collections of experimentally defined transcription factor binding sites for eukaryotes. The primary difference to similar resources such as TRANSFAC [11]. For our work, we are specifically using the 2014 homosapien (aka human) dataset from JASPER [12] mainly due to the open data access, non-redundancy and quality [12]. JASPAR contains 205 motifs that are used to calculate PWM of SNPs potentially affecting TF bindings.

2.1.2.3 ENCODE

The **ENC**yclopedia **O**f DNA **E**lements (ENC-O-D-E) is a public research consortium of human and mouse genomes. The encyclopedia has gone through numerous revisions. For our research, we used ENCODE revision 3. ENCODE contains 2065 motifs that are used to calculate PWM of SNPs potentially affecting TF bindings.

2.1.2.4 Affinity Testing

Affinity Testing is a method to score individual SNP based on the TF binding affinity of a given SNP to a motif.

2.1.2.5 Position Weight Matrix (PWM)

Also known as a position-specific weight matrix (PSWM) or a position-specific scoring matrix (PSSM), PWM is a commonly used representation of motifs in biological sequences. In this context, they are calculated from data generated from a DNA sequencer. The resulting data is based on the sequence surrounding a SNP and represents the distribution of nucleotides (A,T,C,G) as a position-specific scoring matrix. A motif descriptor attempts to capture the intrinsic variability characteristic of position-specific sequence patterns. A sequence weighted position profile is derived from the resulting set of aligned sequences that is functionally related, see Figure 2.2.

2.1.2.6 DNA sequencing

Obtaining an individual organism's unique DNA fingerprint or sequence is done through a process called DNA sequencing. The process follows these steps: cellular DNA extraction, library preparation, DNA sequencing on a Next Generation DNA Sequencer (NGS), alignment and data analysis.

The science and the bench science process of NGS DNA sequencing fall outside the scope of this body of work. Although bench science is outside the scope of this work, the end result of the NGS is a digital representation file containing fragments of DNA in no specific order. A typical digital file format produced by an NGS is FastQ [13]. An example of this file type and content can be seen in Figure 2.3. FastQ files contain thousands of outputs for short read sequences and their corresponding quality scores. (Short read sequences are an artifact of the methods used in DNA sequencing.)

Sequence reads from FastQ files are aligned to a reference genome, which is a collection of merged genomes of the same species. A reference genome is the basis for understanding the resulting Variant Call File [15] (VCF), which is a collection of Single Nucleotide Polymorphism sequence variants or mutations.

Example FastQ file contents

@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACCAAGTTACCC
TTAACAACTTAAGGGTTTTCAAATAGA
+SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIII9IG9ICIIIIIIIIIIIIIIIIIIDIIIIIII IIIIII
@SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
GTTCAGGGATACGACGTTTGTATTTTAAGAATCTGAAGCAGAAGT
CGATGATAATACGCGTCGTTTTATCAT
+SRR001666.2 071112_SLXA-EAS1_s_7:5:1:801:338 length=72
IIIIIIIIIIIIIIIIIIIIIIIIIIII6IBIIIIIIIIIIIIIIIIIIIGII IIIII-I)8I

FIGURE 2.3: Example from Illumina sequencing of Escherichia coli str. K-12 substr. MG1655 genomic paired-end library[14]

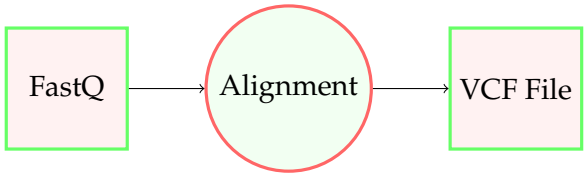


FIGURE 2.4: A typical analysis process pipeline for primary analysis of the FastQ file follows the sequence

2.1.3 Sequence alignment

Sequence alignment is a way to compare the order of nucleotides in two or more DNA sequences. There are two main types of alignment methods:

- Pairwise alignment: Compares only two DNA sequences. This is useful for detailed comparison of closely related genes
- Multiple sequence alignment: Aligns multiple sequences simultaneously. This helps identify conserved regions across a group of related genes or species

Alignment follows this generalized algorithmic order:

1. Arranging Sequences: DNA sequences are displayed in rows, with corresponding nucleotides aligned in columns
2. Gaps: Insertions and deletions (indels) in one sequence compared to another are represented by dashes (-) to maintain alignment
3. Scoring: Alignment algorithms consider matches, mismatches, and gaps when scoring the alignment. A good alignment maximizes matches and minimizes mismatches and gaps

2.1.3.1 Genome Wide Association Studies

GWAS are an analytical technique for understanding the impact of genetic variation on the phenotypic traits of individuals. These studies use hundreds, thousands or even millions of individuals DNA sequences and associate SNP's to individuals with particular phenotypic traits. To obtain results, statistical analysis techniques, such as Linkage disequilibrium estimation [16], are performed on all individuals in a studied population. With a large enough population, Linear Mixed Effect Models estimations can be applied to determine statistically significant direct associations to a SNP or multiple nucleotide polymorphisms.

2.1.3.2 Linear Mixed effect Models (LMM)

Generalized Linear Mixed Effect Models are a combination of a Generalized Linear model with Monte Carlo Markov Chain functionality [17]. These models are used for analyzing data that have fixed and random effects. The fixed effects are constants and represent the systematic part of the model, such as the overall mean or the effect of a specific treatment, while the random effects account for random variability, typically associated with individual subjects or experimental units. They are particularly useful in dealing with clustered or hierarchical data or in situations where observations are not independent, but grouped. By incorporating fixed and random components, Linear Mixed effect Model (LMM)s provide a more nuanced understanding of the data, allowing analysis of complex data structures and exploration of variation at both the population and group / individual levels. This approach also helps in handling missing data and balancing the trade-off between bias and variance, making it a powerful tool for longitudinal and multi-level data analysis. This statistical method provides Bayesian analysis functionality and is suitable for random sampling errors [17].The linear mixed effect model is especially useful for GWAS analysis.

2.1.3.3 Binding Affinity Score

The Binding Affinity Score is a statistical weight used by atSNP for calculating a PWM as parameter to compute log likelihoods for all sub-sequences overlapping of SNP locations that have the same length as the input motif. The best matches are those with the highest logarithmic likelihood with both the SNP and reference alleles.

2.1.4 atSNP Computational Infrastructure Terms

2.1.4.1 Computational Cluster

Computational clusters are complex systems organized to optimize performance. These clusters consist of numerous systems arranged in a distributed manner such that each member of the distributed system is independent but acts in synchrony as a group. These systems are typically managed through a cluster scheduler that is responsible for sending tasks to nodes and having these tasks run on the nodes in parallel. The behaviors of these computational cluster systems mimic the behaviors of other complex systems in nature [18]. Computational clusters, as well as other complex systems, are specifically susceptible to connectivity loss and synchrony issues. This connectivity loss and synchrony issue is typically referred to as the Consistency, Availability, and Partition-able Theorem, aka Brewer's Theorem [19]. According to the theorem, any distributed system or data store can simultaneously provide only two of three guarantees: consistency, availability, and partition tolerance. Various cluster schedules attempt to address Brewer's Theorem by making the trade-off between consistency, availability, and partition tolerance.

2.1.4.2 RPM based Linux system

Based on the Linux operating system, originally developed by Linus Torvalds [20]. Redhat Package Manager (RPM) is a way of compressing and packing application code in a contained verifiable process complete with a full listing or manifest of all the files within the package. These packages are often versioned and often contain, within the manifest file, a listing of package sub-dependencies.

2.1.4.3 Divide and conquer

The process of dividing a problem or application into a smaller and more manageable domain space. For each broken apart, smaller, more manageable pieces that are then analyzed independently. Utilizing this method provides a pathway towards scalable computing by utilizing the parallel and independent characteristics of each broken apart piece. Several strategies for designing divide-and-conquer algorithms arise and are used to derive algorithms for sorting a list of numbers, forming the Cartesian product of two sets, and finding the convex hull of a set of planar points [21]. This technique is used heavily in computational clusters.

2.1.4.4 Replication

Replication is used when data is copied to another location, whether it is another system or a different location on the same computational system. Replication is used to provide a level of resilience for data by helping to recover from failures by providing multiple independent data copies. Other examples include Redundant Array of Inexpensive Drives (RAID) [22] and Erasure coding [23]. Erasure coding is a forward type error coding using separate locations [24]. Replication is a common practice for disaster recovery; data that have been replicated can be used should a disaster occur. Replication can occur at multiple levels within a system. These levels include the data block (i.e., lowest OS level), file system level, or application level.

2.1.4.5 Data sharding

Data sharding is a partitioning architecture technique used to manage extremely large data sets. The data is broken into small pieces, or partitions, that form the basis of a data shard. These shards, in turn, are stored in locations on separate database servers and perhaps in different physical locations [25]. There are numerous advantages to data partitioning through data sharding; specifically, databases and files are divided and distributed among multiple servers, reducing the total number of rows of files retained on each server. Multiple shards can be placed on multiple machines, forming a many-to-many relationship. This enables the distribution of data over a large number of machines, greatly improving performance. The disadvantages of data sharding include complexity, data consistency, and slower performance during shard rebalancing.

2.1.4.6 Extraction Transformation Loading

Extraction Transformation Loading is a technique used frequently to populate databases and data warehouses. This technique is used to **E**xtract the data from the original dataset, **T**ransform it into a database format, and **L**oad it into the database. Map Reduce is a specific case of *ETL*. Map Reduce will map the entire dataset to a format that is expected and then filter the mapped data set to a smaller size (which is why it reduces) [26].

2.1.5 atSNP Database Types

2.1.5.1 Databases for large scale datasets

Large datasets are defined as any dataset that cannot be stored, indexed, and searched by one system in a practical manner. Large datasets may contain billions or more records and the insert, search, and indexing capabilities are beyond the scope of any single system. Therefore, multiple computers are needed to distribute and search records from these data sources.

2.1.5.2 Relational databases System (RDBMS)

A relational database system is defined as a database that stores data in tables and tables use these tables to implement relational algebraic equations for queries and provides ACID (**A**tomicity, **C**onsistency, **I**solation, **D**urability). The ACID properties of database transactions that are intended to guarantee the validity of the data despite errors, power failures, and other mishaps[27]. Data is accessed in these systems using the algebraic relationship method originally proposed by Codd[28]. Functionality exists in these RDBMs systems, such as select, joins, unions, excludes, through relationship joins from multiple data tables. Each data table is connected or "joined" to other data tables through a specialized first order logic algebric based language known as Standard Query Language (SQL). Using SQL results in a dataset list from the joined data tables. For our work, we focus on the MySQL RDBMS systems [29].

2.1.5.3 MyISAM

MyISAM is a MySQL storage engine and it was the original MySQL database engine until version 5.5 [30]. It uses table-level locking, where operations on one part of the table prevent access to the rest, potentially hindering performance in table updates in high-write environments. MyISAM does not support transactions, meaning that it cannot group operations into single atomic units, which is a drawback for applications that require strong data integrity. However, it excels at full text indexing and searching, making it suitable for applications such as search engines. MyISAM stores data and indices in separate files that require applications to manage data relationships. Due to these characteristics, MyISAM is often chosen for scenarios for high performance, heavy read operations, and minimal

concurrent writes, but its limitations in transaction support and concurrency have led many to prefer other engines like InnoDB for more complex applications.

2.1.5.4 InnoDB

InnoDB is a MySQL storage engine, known for its robust data integrity and support for ACID-compliant transactions. It became the default storage engine from MySQL version 5.5 onward. In contrast to MyISAM, InnoDB supports row-level locking, allowing for more efficient processing of concurrent data operations, making it well suited for high-transaction environments. It also provides foreign key constraints, ensuring referential integrity across different tables. InnoDB's design focuses on reliability and consistency, with features such as crash recovery and automatic rollback to safeguard data [31]. Its ability to handle large volumes of data and support for full ACID compliance make InnoDB a preferred choice for applications that require secure and reliable data management, particularly for business and enterprise-level applications.

2.1.5.5 NoSQL Document Stores

NoSQL document store databases are non-tabular databases that store data outside of a traditional relational database table structure. A NoSQL database provides a flexible schema often based on a grouping of key, value objects and typically scales with data size and high user loads.

2.1.5.6 NoSQL Big Table Stores

NoSQL Big Table stores are largely based on the original Google BigTable [32]. These databases are presented as matrix style spreadsheet tables with data structures based on row, column, and data. The row, column, data; data structure format is called a triple, and this structure can scale beyond what can be hosed within a given desktop spreadsheet application. Current example databases include: Apache Cassandra [33], Apache HBase [34], and Google BigTable.

2.2 Hard disk drive faults and failures

2.2.1 Disk failure

A hard disk failure event refers to the malfunctioning or breakdown of a hard disk drive (*HDD*), which is a primary storage device in computers and servers. This type of failure can result from a variety of factors and causes the hard disk to be unable to perform its normal operations, potentially resulting in the loss of data stored on the disk.

A common type of disk failure event is block corruption. Block corruption can manifest itself as either a physical or a logical failure event. This type of corruption can be caused by physical damage or by software bugs that miscalculate block location boundaries.

There are two main types of hard disk failure, physical and logical.

2.2.2 Physical (or Mechanical) Failure

Physical failures occur when the physical components of the hard disk, such as the spindle motor, read/write head, or platters (where data is stored), become damaged or wear out over time. Causes can include mechanical wear, exposure to extreme temperatures, physical shocks (such as dropping the device), or manufacturing defects. Symptoms may include unusual noises (like clicking or grinding sounds), overheating, or complete nonfunctionality.

2.2.3 Logical Failure

Logical failures occur when the problem is related to the software or file system of the disk, rather than its physical components. It can be caused by malware, accidental deletion of critical system files, software corruption, or errors in the disk's filesystem. Although the disk hardware might be physically intact, the data on the disk can become inaccessible or corrupt.

When a hard disk failure event occurs, it can have significant consequences and can result in data loss. Additionally, failure events can produce events beyond the immediate hard disk failure. Often hard disk drives share communication pathways, aka bus, with other hard disk drives and a failure disk drive may cause other disks on the same communication pathway to become unresponsive or have unreliable communication patterns.

2.2.4 S.M.A.R.T

Self-**M**onitoring, **A**nalysis, and **R**eporting **T**echnology (S.M.A.R.T) is a monitoring system that is integrated into Hard Disk Drive (HDD) and Solid-State Disks (SSD). This technology is designed to detect and report various indicators of drive reliability with the intention of anticipating imminent hardware failures.

Self-Monitoring, Analysis, and Reporting Technology (S.M.A.R.T.) functions by continuously monitoring several key indicators of drive performance and health, such as read error rates, spin-up time, and temperature. Each of these indicators is referred to as a S.M.A.R.T. attribute, and each attribute has a specific threshold value determined by the drive manufacturer.

S.M.A.R.T. data can be accessed through various software tools, allowing system administrators to monitor the health status of HDDs and SSDs. This technology is widely used in both personal and enterprise computing environments as part of a comprehensive approach to hardware reliability and data integrity management.

2.2.5 Backblaze dataset

The Backblaze data set consists of a collection of S.M.A.R.T. attributes snapshot. These snapshots are taken once a day from every operational hard disk on all servers hosted at Backblaze data centers located in Sacramento, California; Stockton, California; Phoenix, Arizona; Reston, Virginia; and Amsterdam, The Netherlands. The data includes basic drive information along with S.M.A.R.T. attributes for each drive. The information of each disk drive is collected and homogenized into a comma separated value file with each row representing one disk drives data per day.

Backblaze is a company founded in 2007 as a business-to-business cloud storage and computer backup service. They provide services to clients in both business and direct-to-customer markets. In 2013, Backblaze started publishing their dataset and statistics for insights to the general public. Their data sets are available here [35].

2.2.6 Failure Mode and Effective Analysis (FMEA)

Failure Mode and Effective Analysis is a powerful technique used to proactively manage risk in computer systems. Imagine that you are building a new server to store customer data. A Failure Mode and Effective Analysis (FMEA) analysis would not wait for the

server to crash before taking action; instead, it would help you identify potential weaknesses beforehand.

Here's how FMEA works in this scenario:

- Identifying failure modes: The first step involves brainstorming all the ways the server could potentially fail. This might include hardware malfunctions such as a failed hard drive, software bugs that corrupt data, or even a power outage that disrupts operations.
- Analyzing effects: For each identified failure mode, FMEA looks at the potential consequences. A failing hard drive could lead to data loss, while a software bug might cause the server to crash and become unavailable. Understanding these effects helps prioritize which failures are most critical.
- Assessing likelihood and detectability: FMEA goes beyond just listing problems. It also considers how likely each failure mode is to occur (occurrence ranking) and how easily we can detect it before it causes a major issue (detection ranking). For instance, a server with redundant hard drives might have a lower likelihood of data loss from a single drive failure compared to a server with just one drive. Similarly, having monitoring systems in place can help detect a software bug before it leads to a crash.
- Prioritizing risks: By taking into account the severity of the effects, the likelihood of occurrence, and the ease of detection, FMEA assigns a risk priority number (RPN) to each failure mode. This score helps focus our efforts on the weaknesses that pose the greatest threat.
- Developing corrective actions: Once the most critical failure modes are identified, FMEA guides the development of corrective actions to prevent them or minimize their impact. This could involve using redundant storage systems for critical data, implementing regular software updates to fix bugs, or having backup power supplies to ensure uninterrupted operation during outages.

By following these steps, FMEA helps create a more robust and reliable computer system. It is a proactive approach that can prevent costly downtime, data loss, and security breaches. FMEA is a valuable tool used throughout the computing industry, from designing hardware components to developing complex software applications.

2.2.7 Root Cause Analysis (RCA)

Root Cause Analysis is a structured investigation that aims to identify the true cause of a problem and the actions necessary to eliminate it [36]. The primary goal of Root Cause Analysis (RCA) [36] is to determine the underlying reasons why a problem occurred in the first place. By pinpointing these root causes, rather than simply addressing the immediate symptoms, more effective and long-lasting solutions can be developed to prevent the recurrence of the problem. RCA is a post-event reactive type of analysis instead of a proactive type, where an event has not yet occurred.

In practice, RCA involves collecting data, analyzing information, and identifying the relationships between the contributing factors and the problematic outcome. The process often involves asking a series of "why" questions to drill down to the core issue. This method helps to move beyond the treatment of superficial symptoms and instead focuses on implementing changes that address the deeper issues within a system or process.

RCA is widely used in various fields, including engineering, healthcare, information technology, and business process improvement. It is a key component of problem-solving and quality management initiatives, as it helps organizations learn from failures or near-misses; thereby, improving operations and reducing the likelihood of future problems.

2.2.8 Failure

Failure refers to the state or condition in which a system or component ceases to perform its intended function correctly or completely. These disruptions can occur at various levels, from the hardware components themselves to the software applications running on them. Failures are unrecoverable error states.

Failure in computing systems can manifest itself in various forms, such as system crashes, data loss, incorrect data processing, or unresponsiveness [37]. The nature of these failures can be transient, recoverable, or permanent, depending on the severity of the issue and the system's ability to manage or rectify the problem.

2.2.9 Faults

A "fault" refers to an abnormal condition or defect at the component, equipment, or subsystem level which may lead to failure. This concept is crucial in understanding and managing the reliability and stability of computing systems. A fault in a computing system

can arise from various sources, such as errors in code, hardware malfunctions, incorrect user input, or environmental factors impacting the system's operation.

When including time as a factor, faults fall into the following three categories [38]:

- Transient faults: This type of fault occurs for a short time period and then disappears without physical damage to the processor. It is often induced by electromagnetic interference and cosmic radiation.
- Intermittent faults: This type of fault occurs frequently, and it is difficult to detect because after its occurrence the system operates correctly.
- Permanent faults: This type of fault results from hardware component failure or manufacturing defects. Recovery from this kind of fault is only possible by replacing or repairing the faulty component.

Understanding and mitigating faults is crucial to ensure the reliability, availability, and security of computational systems. Here we breakdown the fault by type:

- Hardware Faults: These are physical malfunctions within the hardware components of a computer system, such as memory errors, processor malfunctions, or disk drive failures [39].
- Software Faults: These are errors or bugs in the software code that can lead to unexpected behavior or system crashes. Software faults can be introduced during the development process or arise due to external factors like unexpected inputs or deliberately [40].
- Environmental Faults: These are external factors that can disrupt the operation of a computer system, such as power outages, extreme temperatures, or electromagnetic interference [38].

Faults are typically considered the root causes of failure of a system to perform its intended function. They can be latent, exist in the system without immediate effect, or active, where their presence is immediately apparent through the system behavior. In the context of software, faults might include bugs or glitches in programming, while in hardware, they could encompass a wide range of physical or electronic issues.

Identification and fault management are a critical part of system design and maintenance. This involves not only detecting and correcting existing faults, but also anticipating potential faults through robust design and testing practices. In many computing

systems, particularly critical applications, fault tolerance is an essential feature. This involves designing the system in such a way that it can continue to operate, possibly at a reduced level, even in the presence of faults.

2.3 Computational cluster task failures

2.3.1 Tasks

A computational task, otherwise known as a job or process, is an operation or a series of operations assigned to a computer for execution. These tasks are designed with specific objectives, such as data processing, calculation, or system management. Each task comes with defined input data or instructions and is expected to produce specific outputs or results. The execution of these tasks requires the consumption of computational resources, including CPU time, memory, and storage.

The duration of a computational task can vary significantly, ranging from a few milliseconds for simple operations to several hours or days for more complex ones. Tasks can be independent, capable of being executed in isolation, or interdependent, where they form part of a larger set of operations and may rely on the output of other tasks. In resource-limited environments such as distributed networks and cloud computing, tasks must be efficiently scheduled and prioritized based on resource availability, urgency, and dependencies.

2.3.2 High Throughput Computing

HTC is a computing paradigm focused on the efficient execution of a large number of independent tasks. Unlike HPC, which is optimized to execute a few highly intensive computations with task inter-dependencies requiring synchronization (like in simulations), HTC is designed to process a vast number of tasks that, individually, may not require much computational power but collectively represent a significant workload.

The most recognized system for implementing HTC is HTCondor [41]. Developed by the University of Wisconsin-Madison, HTCondor is a workload management system for compute-intensive jobs. It effectively harnesses underutilized CPU power from networked desktop workstations and dedicated clusters to perform distributed batch processing. It provides job queueing, scheduling, submission, and monitoring, aimed primarily at the effective use of distributed computing resources.

The system is designed to dynamically allocate and realocate computational tasks. HTCondor operates by managing a pool of processors, directing jobs to available resources while ensuring that tasks are completed in a timely manner.

A key feature of HTCondor is its ability to schedule jobs based on the resource requirements and policies defined by users and administrators. Implements a matchmaking algorithm that pairs submitted jobs with available resources, making efficient use of the computing infrastructure. This matchmaking process takes into account the specific needs of each job and the current state of available resources, with the aim of maximizing overall throughput.

2.3.3 Computational Cluster task scheduling

A computational cluster task schedule is a system that orchestrates the execution of tasks on a cluster of interconnected computers. This schedule determines how tasks are assigned to various nodes, considering the specific requirements of each task and the capabilities of each node.

Effective task scheduling ensures that the workload is evenly distributed across the cluster, maintaining a balance, and preventing scenarios where some nodes are overwhelmed while others are underutilized. It involves managing the priorities and dependencies of tasks and ensuring that tasks of higher importance or those that are prerequisites for others are executed in an appropriate sequence.

The task schedule is responsible for the efficient use of the clusters resources, including CPU , memory, and storage. This involves not just maximizing the performance of individual tasks, but also improving the overall throughput of the system, which is particularly crucial in high-throughput environments where the goal is to process a large number of tasks within a given time frame.

In addition, the schedule might incorporate mechanisms for fault tolerance and resilience, enabling the system to handle node failures or other disruptions smoothly by rerouting tasks as necessary. This capability ensures consistent operation and minimizes downtime.

2.3.4 Biomedical Computing Group's Computational Cluster

Our work depended on a computational cluster designed and build by BCG within the Department of Biostatistics and Medical Informatics (BMI) at the University of Wisconsin

Madison. This cluster was used as the source of data for the failure predictions of the cluster computation jobs.

The cluster comprises 240 nodes of 64 bit multicore x86 systems, with a mixture of hardware vendors and CPU brands. Hardware manufacturers used in the cluster include: Dell, Cisco, Hewlett Packard Enterprise (HPE) and Supermicro. Here is a list of cluster resources:

- 2,875 CPU cores
- 11.8 TeraByte (TB) system RAM
- 590.2 TB disk storage
- 6 compute racks
- OS - Redhat Package Manager based Linux

All BMI BCG computational cluster tasks are submitted by biostatisticians or medical informaticians and require compliance or have data set restrictions that prevent them from running on other infrastructures such as the Open Science Grid [42].

Chapter 3

atSNP Search Background and Related Work

In this chapter, we present background and related work to atSNP search. We will discuss the current state-of-the-art and how the current state-of-the-art relates to the later showcasing of our contributions.

We discuss computational systems, their implementation, and work being done to address these issues. We examine the prior art in distributed NoSQL databases for genomic storage. We examine SNP, TF, and logo plot motifs, Extract Transform Load (ETL) data, and methods for a genomic search engine we built called atSNP search.

Bio-informatics human genomic SNP databases are a relatively small but important cross-domain knowledge intersection between statistics, high-performance computing and genetics. This emerging domain used genetic data sets, statistical methodologies, and scalable computing algorithms. The computing power and statistical knowledge required to perform variant assessments and variant alterations of SNP interactions is generally beyond the reach of the average genetic researcher.

"Affinity Test for regulatory SNP detection (atSNP) is a bioinformatics tool to calculate and test large-scale motif-SNP interactions" [2]. Tools such as FIMO [43] and is-rSNP [44] are computationally expensive analytical methods to calculate the p-value. atSNP uses the importance sampling algorithm. To estimate p-values accurately without evaluating the entire probability space, which grows exponentially with motif length. This significantly reduces the analysis time, making atSNP a powerful tool for large-scale studies of regulatory SNP interactions.

3.1 SNP motif databases: Comparison

Several SNP motif-based resources are currently available to quantify the regulatory impacts of human SNPs. We identified and evaluated three SNP databases with compatible attributes SNP2TFBS [45], Raven [46] and OncoCis [47]. SNP2TFBS, which is a SNP catalog from the 1000 Genomes project [48], was our closest comparable. SNP2TFBS itself contains a much smaller subset of variant-motif pairs that survive at the p-value cutoff 3106. Furthermore, the limitation of SNP2TFBS is a function of the evaluated SNP dataset used, JASPAR. We also evaluated Raven, which pre-computed the results and made the results available on a web server. OncoCis implements motif searches using the Possum tool by Haverty *et al.* [49] for their data generation.

Numerous other examples of SNP databases [50] exist across the web with examples ranging from dbSNP [51] to dbGAP [52]. For example, the Genome Variation Map database [53] recently published its current implementation, which contains a total of about 4.9 billion variations using a MySQL database [29]. dbSNP also utilizes a Relational Database Management System (RDBMS) [54], an entity-relation database [55], which consists of well over 100 relational tables. These SNP databases differ from SNP motif databases as they are not focused on the regulatory regions and their potential impact on gene expression.

3.2 Infrastructure: database comparisons

Real-time searching of large datasets has been an active area of research, with numerous NoSQL big data databases being produced for different use cases. These NoSQL databases vary in type and purpose, with four main types being column-oriented, document stores, key-value, and graph databases [56]. Each type of NoSQL database has benefits and limitations, with individual NoSQL implementations choosing specific optimizations based on the database domain.

NoSQL databases often house hundreds of billions of records. Netflix has a NoSQL Cassandra [57] deployment consisting of 2,500 nodes hosting 420 TB of data. The Netflix database handles over a trillion requests per day [58]. LinkedIn utilized Voldemort [59], a key/value store. Google, often credited with starting the big data movement, created the original BigTable NoSQL Engine [60]. It has been demonstrated that NoSQL databases have been successfully deployed with hundreds of millions or billions of records and ”can

result in a dramatic performance difference against RDBMS while dealing with hundreds of millions or billions of records" [61]. It should be noted that none of these NoSQL databases has been used for genomic genomic data.

Although the algorithmic implementation for each NoSQL sharded database varies, each shares a divide-and-conquer approach through their distributed data storage engine [62].

3.3 Prior Performance Comparisons: A Mixed Bag

Previous research comparing RDBMS and NoSQL databases [63] has yielded inconclusive results [64]. Some studies, such as Puangsaijai *et al.* [65] found that NoSQL databases can offer significant performance advantages. Their work showed that inserting data using ETL pipelines into Redis (a NoSQL database) was more than 20 times faster than MariaDB (an RDBMS similar to MySQL). Interestingly, the same study found that select operations were comparable between both systems for most queries, except for a few simple equality checks.

Another comparison, between MySQL and Cassandra, highlighted specific use cases where Cassandra excelled [66]. However, this study did not explore ETL times. Puangsaijai *et al.* also investigated range queries (greater than/less than) and found that the NoSQL database outperformed MariaDB by a factor of 2.

3.4 Focus on ETL Performance

While surveys have evaluated performance characteristics like replication and query times for databases like MySQL (considered similar to MariaDB) and Elasticsearch, they have not explicitly focused on data ingestion (ETL) properties and processing times. Our study aims to address this gap by specifically investigating the performance of ETL processes in RDBMS and NoSQL databases.

3.5 Motif Logo plots libraries

DNA motif logo plot are frequently used to convey important information about the DNA sequence characteristics. These motif libraries have been used largely by bioinformatics

professionals who frequently use tools such as R [67] and bioconductor [68]. Numerous libraries provide easy access to programmatically generate motif logo plots from within an R program. Examples of such libraries include EDLogo plot [69], motifStack Vignette [70], memes [71]. Work has also been done with applications such as weblogo [70], which uses a server-side scripting language and Ghostscript [72] to generate motif logo plots.

In our development of the atSNP search web application platform, we searched the literature for a JavaScript library to generate motif logo plots. At the time, no library for web motif logo plot generation existed. Therefore, we developed a novel method for displaying motif logo plots for consumption by our end users without relying on a sub-application of R to generate each plot independently. Our work in 2017 included developing a library to dynamically generate motif logo plots. This plot library is similar to LogoJS [73] of today. However, LogoJS was first released in 2019 a full 2 years after our atSNP search platform was released.

3.6 Extraction Transformation Loading data

BigDimETL [74] proposed a method of big dimensional ETL through a big data ETL methodology; however, this work focused mainly on ETL of unstructured data through Map-Reduce.

Recently, ETL for NoSQL has gained popularity especially in big dimensional ETL. This was discussed in Mallek [74] in which they applied an adaptive ETL process to multidimensional data structures.

More recent research by Yulianto *et al.* [75] provided a demonstration of an ETL process for distributed databases not built on Hadoop [76]. This work was built on Simitsis *et al.* [77], which proposed a real-time partitioning method for ETL workflows. This partitioning workflow provided an original algorithmic approach for real-time data processing. This work is notable because it addresses a specific challenge in academic data warehouses (slow ETL with distributed faculty data) and proposes a detailed ETL process with data quality checks and a multidimensional model creation approach.

3.7 Hadoop

This body of work would be incomplete if we did not explicitly mention the Hadoop framework for Map Reduce. Hadoop's Map Reduce [78] is a big data framework for

data manipulation that sits atop the Hadoop storage layer. Hadoop's storage layer is a clustered file system that provides a library for directly manipulating data within the clustered file system to applications using this library at a storage node layer. The idea is to reduce data movement on a network by combining compute and storage into a more powerful system after realizing that data movement costs are the limiting factor for data manipulations [78].

High-throughput methods have been extensively used within the Hadoop community, but this requires applications to use the Hadoop library to manipulate the data within the clustered file system. Hadoop Map-Reduce has been compared in greater detail to other ETL tools and has proven to be significantly more efficient than other commercial technologies [79] [80].

One major criticism of Hadoop is the administrative cost and expertise required for setup and deployment. Therefore, using Hadoop for an ETL that is a one-off seems excessive. Investigating ETL tools, we found that no one has used HTCondor as a platform for ETL even though it is well suited for the task as a massively parallel ETL tool for NoSQL databases.

3.8 Supportability and System Selection

Product supportability is crucial for project success. Therefore, we only consider systems with an install base exceeding 7,000 units and a paid support option available.

- Apache Cassandra: According to Enlyft [81], Cassandra boasts an install base of 7,668
- MySQL: Chowdera reports a massive install base of 360,600,000 for MySQL [82]
- Elasticsearch: Based on data from paid customers, Elasticsearch has an install base exceeding 17,000 [83]

3.9 Cluster Failure and Distributed Metadata

Previous research by Cha *et al.* [84] explored the prediction of faults and failures within clusters, specifically focusing on distributed metadata. Their study revealed that a metadata server cluster with initially uniform metadata distribution becomes unbalanced over time due to constant changes within the cluster. This imbalance worsens progressively. In

addition, restoring balanced metadata placement requires rebalancing among the servers. However, rebalancing presents several challenges:

- Performance degradation of the metadata service during rebalance operations
- Rebalancing typically requires stopping applications on the filesystem, causing inconvenience
- Potential cluster saturation leading to unexpected termination.
- Minimizing failed metadata operations during rebalance becomes increasingly difficult as the database size increases

3.10 Load Balancing and Failure Prediction in NoSQL Databases

Literature suggests a strong focus on load-balancing techniques for NoSQL databases [85] [86]. Object storage systems, a related field, show relevant work in hard disk failure prediction for CephFS [87]. However, understanding faults and failures within the NoSQL database system itself reveals a research gap [88] [89].

Yuan *et al.* [90] investigated user-reported failures in distributed systems, including NoSQL databases such as HBase and Cassandra. Their findings highlight the importance of simple testing to prevent critical failures. They further demonstrate the complex nature of error manifestation sequences, which often require unusual combinations of events and specific input parameters to trigger system failures [90].

Although Yuan's work acknowledges the possibility of system failures arising from complex sequences of events, it also emphasizes that 92% of the examined failures stemmed from improper handling of non-fatal errors [90]. This suggests potential benefits in improving the monitoring and response to events. Machine learning might provide a valuable tool for handling non-fatal faults that could otherwise overwhelm system administrators with excessive notifications.

In particular, 8% of these failures were deemed unavoidable by standard deterministic monitoring systems [90].

3.11 Addressing Batch-Correlated Disk Failures

Recent work by Ke *et al.* [91] proposes a "Fractional-Overlap Declustered Parity" policy to address batch-correlated disk failures (see Figure 3.1). Their approach involves adding

an additional drive to each clustered server, potentially reducing the chance of cluster failure by 99%. However, concerns exist about the feasibility of this method due to consolidation within the HDD storage market. After a recent merger of Hitachi and Western Digital, there are now only three major HDD manufacturers (Seagate, Toshiba, and Western Digital) [92]. This limited vendor landscape weakens the effectiveness of Ke *et al.*'s multi-vendor approach, as Original Equipment Manufacturers (OEMs) often share similar hardware across their drives.

The SSD market presents a slightly more diverse landscape, with seven major vendors controlling flash memory controllers (Greenliant Systems, Hyperstone, Kioxia, Micro, Samsung, SanDisk, and SK hynix) [92]. However, this limited competition still raises concerns about the generalizability of Ke *et al.*'s multivendor strategy.

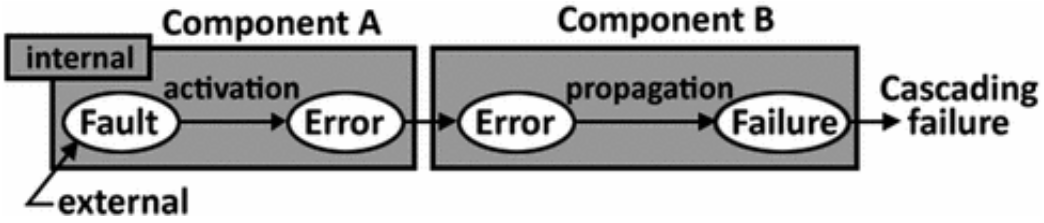


FIGURE 3.1: Batch correlated disk failure event sequence [93]

Chapter 4

atSNP Search

In this chapter, we present our atSNP search and attempt to answer the following question: Is it possible to build a motif genomic database for SNP to PWM analysis that can search and retrieve data from the entire human genome using atSNP data in real time? This chapter also lays the foundation for work in other chapters. Specifically, in the process of managing the atSNP Search system, we experienced a cascading failure event that caused us to investigate additional areas of cluster operations.

We introduce the Affinity Testing SNP (atSNP) Search, a system designed for searching billions of motif Single Nucleotide Polymorphism - Position Weight Matrix (SNP-PWM) for helping genomic researchers determine TF binding affinities.

atSNP [2] can accurately estimate the p-values without evaluating the entire probability space, so it is possible to evaluate the full human genome. The search engine data set was produced using the atSNP R package. This package of atSNP R was used to sample the statistical significance of a given within a DNA segment of two human genomic databases JASPER and ENCODE. The SNPPWM for TF was calculated during the generation of the atSNP data set. The analysis matched all 132,946,852 SNPs to the JASPAR and ENCODE 2,270 motifs, totaling 307 billion (396 TB) SNP-PWM records. The data set generated on the Open Science Grid required 115,000 CPU hours which is 13.12 computing years to complete.

The functional goals of atSNP Search included the search affinity scores by SNP_ID, chromosome, P-values, and SNPlocations. Within our data, we represented P-values as floating-point numeric which needed to be searched as a range around a given P-values. Thus, for atSNP Search the data set calculated the P-values and the affinity testing score of each SNPPWM motif for each SNP. Specifically, we focus on the human genome from

dbSNP [51] Build 144 [94] of the human genome assembly GRCh38hg38 [95]. Therefore, to access the result set for computing all these SNPs we created atSNP search infrastructure.

atSNP Search is tailored to explore, analyze, and interpret 37 billion SNP records. The infrastructure is characterized by its commitment to precision, its ability to manage large datasets, and its emphasis on user accessibility. As the genomic data landscape continues to accumulate data at an exponential rate, the relevance of infrastructure such as atSNP Search is becoming increasingly critical.

The atSNP Search web resource platform was motivated by the need for:

- providing genomic researchers atSNP statistics of the entire human genome through a web interface
- a more comprehensive motif-based discovery of regulatory variants beyond the current state-of-the-art
- statistical quantification of SNPPWM motif matches and changes in motif matches and
- a convenient graphical depiction of potential TF on SNP-PWM via motif logo plots

This chapter describes the core components of the atSNP Search, outlining its design principles, architectural framework, and functional capabilities. It also contextualizes the infrastructure within the broader landscape of genomic research, highlighting the specific needs and challenges that it addresses.

At the onset of our building atSNP Search, our goal was to provide access to atSNP human genomic statistics with motif match without the user needing to individually compute their own affinity testing of each SNP.

This chapter discusses the challenges of atSNP, a survey of databases, a novel ETL method, and an overview of the final search engine. Our novel contributions to the field include:

- a genomic search tool that provides the knowledge of human SNPs on TF-DNA interactions based on PWM
- a comparison of MySQL, Apache Cassandra and Elasticsearch for use as a genomic SNP-PWM motif databases
- a big data loading method to extract transform and load genomic data using HTCondor [3] for ingestion into a database

- a generation visualization library producing composite motif logo plots

4.1 atSNP data

The search of genetic and genomic datasets at the SNP level presents considerable challenges due to the vast domain space. These data sets contain billions of minor sequence variations, each contributing to the multitude of observed phenotypic variations [96]. DNA analysis in this context is heavily based on statistical probabilities, focusing on specific sub regions within the DNA strand to understand phenotypic traits at a population level. Notably, a single DNA base, known as a SNP, is pivotal in this analysis. These SNPs are individual base-pairs in DNA segments that are produced from sequencing techniques such as the shotgun approach to DNA sequencing [97].

Our data set evaluated 133 million SNPs from the National Center for Biotechnology Information's (NCBI) database of genetic variation, against 2,270 PWMs sourced from two primary transcription factor position weight matrix libraries: JASPAR and ENCODE motif libraries. The analysis utilized *in-silico* calculations based on the atSNP statistical R package, a method developed by Zuo *et al.*

We encountered significant challenges with data set size, rendering it impractical to load from a single system. The JASPAR datasets, when compressed, had a compression of 19.8x. However, the uncompressed size of the raw datasets necessitated a reevaluation of standard ETL methods for a traditional Relational DataBase Management System (RDBMS). The uncompressed raw data set from JASPAR was approximately 35.64 TB, which, under our infrastructure constraints, would have required approximately eight hours per day for a single system to simply transfer the data between systems using a 10-gigabit network without any ETL processing. The actual wire speed of the data transfer rate was further hindered by rate limits imposed by the data storage system, affecting our ability to achieve wire speed for data ingestion.

The JASPAR dataset presented unique challenges to traditional RDBMS systems. The sheer number of records posed an additional challenge. The upper bound on the number of rows that could be stored in a single table of a traditional RDBMS was exceeded in our case [98].

The ENCODE motif library, another critical component of our dataset, is a compressed dataset of 19 TB. Decompressed, this expands to 362.39 TB, based on a compression ratio of 19.07x. This ENCODE dataset result consists of approximately 279 billion records.

Having such a large dataset, 309 billion records between both dataset results, totaling 398.03 TB.

We explored the feasibility of using open source, community-supported database engines better suited to handle our specific requirements. This exploration aimed to identify a database engine capable of effectively managing the large scale of our datasets.

We focused on evaluating the feasibility of distributed database systems, comparing RDBMS and NoSQL databases.

4.2 Database survey and feasibility objective

In this section, we present our analysis of three prominent databases for managing and querying up to 309 billion records. Our performance assessment included the MySQL RDBMS, Apache Cassandra NoSQL large table, and Elasticsearch NoSQL document store.

The databases for our study were selected for their extensive open source community base. Our assessment took into account data structures while simultaneously weighing the practicality and efficacy of temporal ETL operations in addition to the latency of data retrieval. Preliminary tests were performed using the atSNP test infrastructure, depicted in Figure 4.1.

Providing 398.03 TB of SNP affinity scores with motif data with queryable parameters in real time requires analysis of the data structures to optimize searchability. Furthermore, data loading 398.03 TB of records can cause additional issues, such as throughput, as loading times become a bottleneck.

Our first configuration, as seen in Figure 4.1, used three production servers augmented with additional storage: 7200 rpm 4 TB hard drives. Two systems operated with three HDD and one system operated with five drives. We used this configuration because these were not running production compute tasks.

Machine	RAM (GB)	#Processors	Total HDD (TB)
atsnp-db1	24	4	12
atsnp-db2	24	4	12
atsnp-db3	32	8	20
TOTAL	80	16	44

TABLE 4.1: Our first generation test Elasticsearch cluster composition

Our survey concentrated on evaluating the functional and operational viability of database engines, with a specific focus on comparing MySQL RDBMS, with two distinct

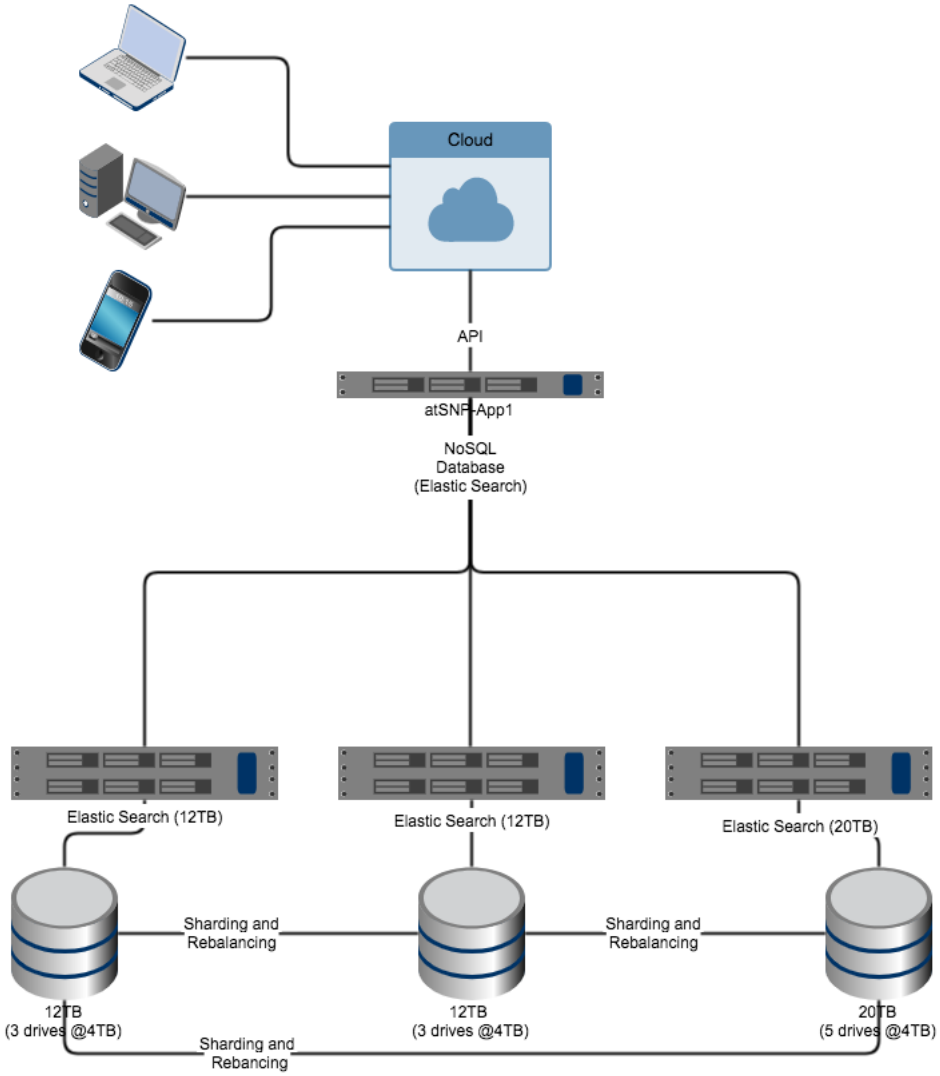


FIGURE 4.1: The test cluster

types of NoSQL databases: one based on a wide-column model and the other on a document store model. Each database node was tested with the default minimal installation configurations a RPM-based Linux system. The test setup included three nodes, as detailed in Table 4.1. The key parameters assessed in our evaluation included ETL speeds, search retrieval times (targeted within a 5-second window), estimated system administration effort for ongoing maintenance, scalability, and failure recovery capabilities.

To effectively evaluate our test cluster ETL process, we selected a data set size that was large enough to meaningfully assess ETL times but small enough to be manageable. For this purpose, a dataset comprising 1 million records was used for testing. This quantity was arbitrary, but intuitively seemed adequate to meet our ETL requirements while also allowing for effective SNP-specific and range queries against the database.

In the following sections, we present the results of our database survey. The results involved loading, storing, and retrieving genomic datasets using three widely supported and using databases: Apache Cassandra, MySQL, and Elasticsearch.

4.2.1 Apache Cassandra

Apache Cassandra is an open-source distributed NoSQL database management system that is notable for its scalability and performance in managing large volumes of data across multiple servers. This distributed NoSQL architecture uses a wide-column NoSQL database engine. The wide-column NoSQL database architecture has been shown to perform better for ETL and retrieval than other databases [99].

Apache Cassandra is recognized for its scalability and performance, yet it encounters significant challenges with full-table searches, especially for range queries. This database system, designed for rapid writes and accessing data via specific keys, faces difficulties with queries that require scanning extensive sections of a table. Performing full table scans in Cassandra means reading every row to locate needed information, a process that can lead to problematic response times. Such queries often become inefficient and consume substantial resources, primarily due to the absence of traditional indexing found in conventional relational databases.

We validated our first test with Apache Cassandra version 2.4. Our results showed that Apache Cassandra had the best ETL speeds of any of our evaluated databases. Our data were able to enter ETL into the test database at a rate of 14,664.2 records per second and as seen in Table 4.2. We compare our results with the literature and find them to be similar to the University of Toronto NoSQL benchmarks, suggesting that our work was in line with others' results for database ETL performance characteristics for data import [100].

In addition to ETL, we evaluated a few other factors to select a successful database for our atSNP search platform. These factors included data set scalability for data loading, storage, and retrieval. For the first two, Apache Cassandra performed well. However, data retrieval caused problems for this database choice.

A notable limitation of Apache Cassandra lies in the architecture choices made between hash table speed in key-value matches against sequential range matches, also known as range queries. To execute range queries, which involve finding all values within a defined upper and lower limit, Cassandra must perform a full table scan for each range

and subrange query. Consequently, response times can vary significantly, ranging from a second to several hundred seconds. This variability had a substantial impact on the user experience of the atSNP search.

The nature of the queries played a crucial role, especially for range queries. The resulting query-time variability caused response times that exceeded acceptable limits for our users. Thus, due to these extended query durations, Apache Cassandra was ruled out as a viable option for our needs based on this artifact.

4.2.2 MySQL

MySQL is a widely-used open-source RDBMS that functions as a server providing access to databases. It is built on the Structured Query Language (SQL) which is used to add, remove, and modify information in the database. MySQL is known to be reliable, robust, and easy to use. MySQL has been in active development for almost 30 years and has a robust development and contribution community.

One of the core strengths of MySQL is its high performance for data processing and retrieval, which it achieves through a unique storage engine framework that allows system administrators to configure the MySQL database server for performance based on specific applications. Examples of storage engines include MyISAM and InnoDB. Our evaluation focused on the MyISAM high performance storage engine, MyISAM.

We tested MySQL with MyISAM to quantify ETL times for our data. As our database increased in size, both disk and memory usage increased linearly and resulted in a significant decrease in ETL after ingesting one million records. We reached the point where the database engine became completely unresponsive. Attempts to address unresponsiveness by forcing a database engine restart were futile, and our database engine memory footprint consumed all available memory. In debugging the situation, we were able to determine that the system memory usage was being consumed by database index tables. Once the MySQL database entered the consume-all-memory state, any additional ETL operations caused the database to become unresponsive.

Throughout the MySQL ETL phase, our average input was 1023 records per second. We were able to verify that these are expected result by comparing work to work done by kvs.io [101].

4.2.3 Elasticsearch

Elasticsearch is an open source search and analytic engine built on Apache Lucene [102]. This database provides powerful search and indexing features and advanced analysis capabilities. It excels at managing and retrieving large volumes of data and it is particularly adept at performing fast range queries, a capability that makes it the preferred choice for various data-intensive applications.

At the heart of Elasticsearch's efficiency for range queries is a sharded multi-file index, derived from Lucene, which enables it to quickly locate and retrieve the range of data requested. This indexing mechanism is optimized for speed, allowing Elasticsearch to perform range queries much faster than other database systems.

Elasticsearch also has impressive data load times. The engine is designed to ingest and index data swiftly, which means new data can be made available for search and analysis almost immediately after it is entered into the system. This quick data ingestion is facilitated by Elasticsearch's distributed nature, where data is spread across multiple nodes, enabling parallel processing and, consequently, faster indexing.

Elasticsearch's architecture also contributes to its quick data load times. It breaks the data down into data shards, more manageable pieces of a larger data set that can be processed independently and in parallel. This sharding mechanism, combined with Elasticsearch's ability to run on multiple nodes, ensures that the system can handle large volumes of data without a significant drop in performance.

The relevance of Elasticsearch's quick range queries and data load times is especially significant in temporal log-analysis systems, where vast amounts of log data need to be ingested and queried rapidly for monitoring and troubleshooting purposes. The temporal search of range queries makes Elasticsearch especially interesting for searching segments of DNA as our use case will require searches of DNA string sub-sections locations.

We expected that the Elasticsearch database ETL would perform significantly faster than the other options, as it is optimized for log ingestion (timestamp range query data). Our system experienced ETL at an average rate of 11,944.5 records per second. Although performance times are significantly faster than MySQL (Table 4.2), we found it surprising that the resulting ETL performed worse than Apache Cassandra. To evaluate Elasticsearch lookups of address range queries, we loaded 1,012,032 samples ($\approx$ 1 million records) to prove the feasibility of the range query. The initial tests query times are less than 5 ms.

Our final choice for the atSNP search database engine is based on our objectives for our feasibility evaluation matrix (Table 4.2).

Database	Loading (rec/sec)	Loading (days)	Rebalance data	Range query support	Team knowledge
Cassandra	14,664	29.2	Yes	No	No
MySQL	1023	418.6	No	Yes	Yes
Elasticsearch	11,944	35.9	Yes	Yes	No

TABLE 4.2: atSNP evaluation matrix

4.3 atSNP Search results

atSNP analysis identified and quantified the best DNA sequence matches to the TF for PWM's with both the reference and the SNP alleles. atSNP accomplishes this by analyzing a small genomic region surrounding the SNP site, extending up to +/- 30 base pairs and encompassing sub-sequences that span the SNP position. For each SNP, atSNP evaluates the statistical significance by comparing the match scores of each allele and calculating the significance of the difference in scores between the best matches for the reference and SNP alleles (Figure 4.2).

The process of discovering the best matches is carried out separately for the reference and SNP alleles. Consequently, the genomic sub-sequences that provide the best match for each allele may differ. This variation in match quality between the reference and SNP alleles provides insights into the potential *in-silica* impact of the SNPs on the binding ability of TF to DNA, a key aspect in understanding genetic regulation.

Furthermore, atSNP Search offers a composite logo plot (Figure 4.3) for a more intuitive visualization of the quality of the match between the reference and SNP allele matches to the PWM, and the consequential influences of SNPs on these matches. This visualization aids in the easier interpretation and understanding of the SNPs impact on TFDNA interactions.

Addressing the affinity scores of SNPs necessitates the handling of large volumes of data to determine probabilistic significance. Consequently, generating and analyzing these SNP affinity scores required significant compute resources. Hence, our desire to disseminating the SNP affinity scores to the scientific community. This dissemination would mitigate the redundant effort of other groups engaged in identifying optimal SNP affinity

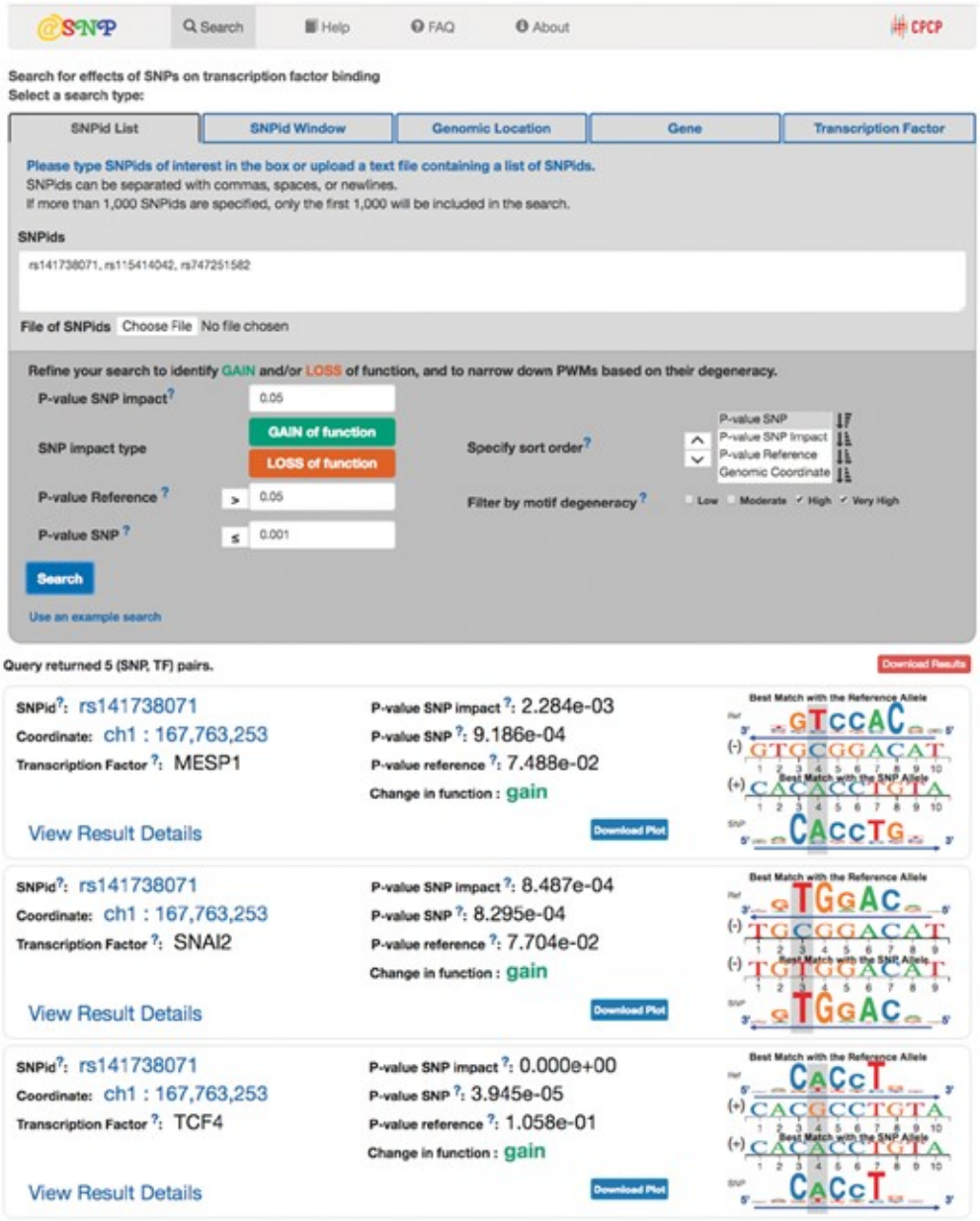


FIGURE 4.2: atSNP search web site example

scores. Our approach was based on a run-once, share-with-many philosophy for PWM motif data [103].

We encountered challenges related to data set size, including the costs associated with hosting and storage. To balance cost considerations with impact, a threshold strategy was used, using a cutoff value of P of 0.05 for SNP impact. Implementing this cutoff effectively reduced the size of the data set and, consequently, the search size to 11.97% of its original volume, equating to 37 billion SNP-PWM record pairs for the final atSNP Search platform. This refinement enables the atSNP Search platform to traverse 37 billion records instead

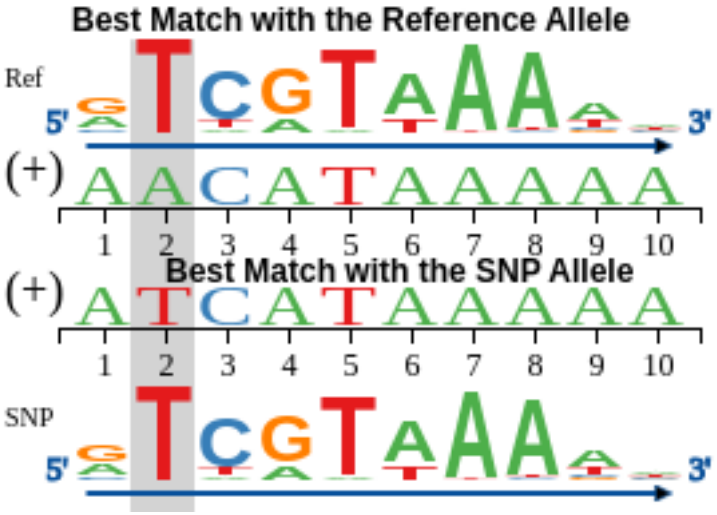


FIGURE 4.3: Example logo plot

of the initial 309 billion SNP-PWM evaluation.

The initial generation of the PWM affinity scores was carried out using the atSNP [2] R package [104]. To process the complete data from JASPAR and ENCODE, a distributed divide-and-conquer methodology was applied. The combined effort of HTCondor involving 132,946,852 SNPs and 2,270 PWMs, yielded SNP-PWM combination records and necessitated 115,500 CPU hours.

Previous research involving genomic databases had limitations specific to its functionality and usage. Our atSNP database was developed to address the need for a comprehensive, queryable resource consolidating all possible SNPs from a given organism in a single location and is searchable on the Web, obviating the requirement of additional external references.

Our final implementation of the atSNP database encompassed a database engine employing data sharding, which involves the partitioning and distribution of data for queries across extensive datasets. This architectural choice facilitated efficient query processing over the large-scale, especially with vast SNP-PWM data points.

4.4 Discussion

In this section, we discuss the atSNP database survey and our contributions.

4.4.1 Survey

At the beginning of our investigation, we recognized the need to compare various database systems to determine the most suitable for our specific requirements. During our preliminary research, we identified a gap in the existing literature regarding comprehensive comparisons of our shortlisted databases. This led us to conduct a comparative evaluation between MySQL, Cassandra, and Elasticsearch, particularly focusing on their capabilities to meet our specific functional needs for building and deploying a database capable of managing the anticipated data volumes for the atSNP data search engine.

In formulating our evaluation criteria, we factored data search functionality into our requirements. We anticipated the need for searches involving range queries across genome sections, based on parameters such as the P-value, genomic location, and chromosome locations. Furthermore, we anticipated requirements for direct searches of specific genomic sections and SNP segments. Thus, we recognized the need to integrate multiple search functionality, including key lookups, range queries, and sorted lists, to meet our end-users' expectations.

Our testing indicated that our initially preferred database, Cassandra, encountered significant challenges in providing efficient search functionality for a range of values. Upon further investigation, we discovered that the range query implementation in Cassandra's lacked efficiency. Specifically, range queries required each database node to scan all its data for any rows that met the range criteria, regardless of indexing [105]. This limitation in range query performance ultimately rendered Cassandra unsuitable for our atSNP genomic database, particularly given our requirement for efficient range value searches and indexed range queries.

Moreover, our selection of MyISAM, the then default storage engine for MySQL as detailed in Figure 4.4, revealed that while it performed adequately for range queries, it fell short in other search and ETL functions. This finding further informed our decision-making process in the selection of the database for the atSNP Search platform.

Elasticsearch demonstrated consistent performance across all dimensions of our testing, without particularly excelling in any specific task. Its generalized functionality emerged as a robust option, leading to its selection as the preferred system for the needs of our work. This choice was influenced by its overall balanced performance profile, which aligned well with the diverse requirements of our testing criteria.

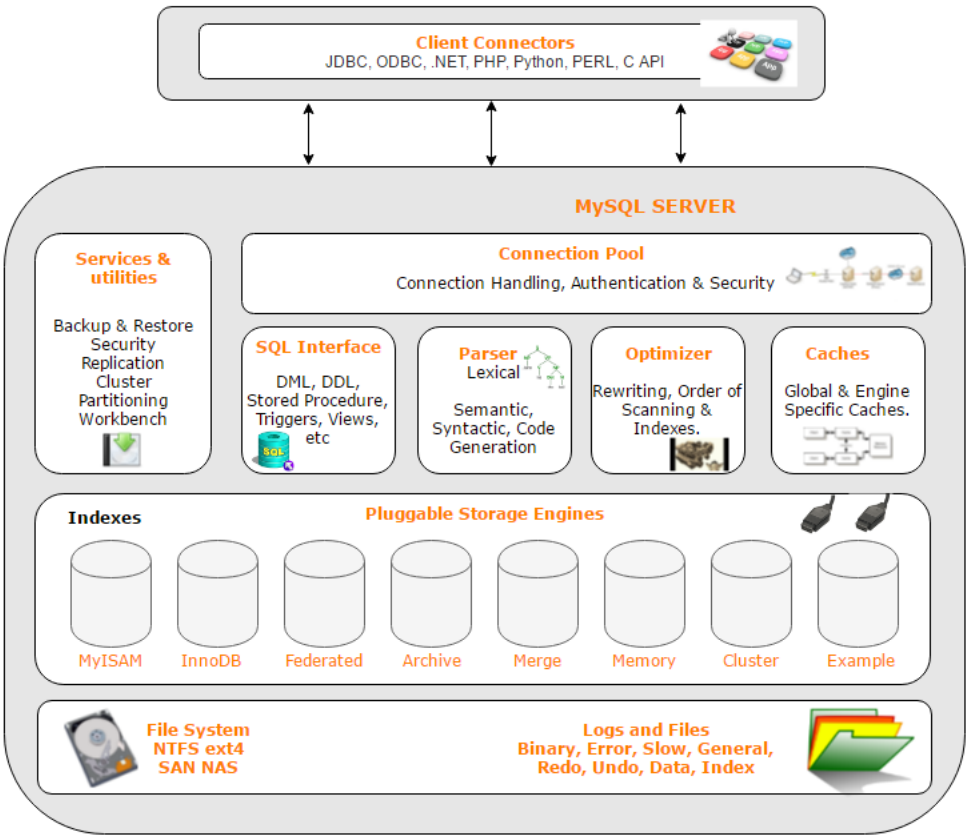


FIGURE 4.4: Internal MySQL database architecture with storage engines example [106]

Our test setup for the atSNP system revealed that the primary factors that influenced our test scenarios were the ETL process and the response times of the queries. Consequently, our ETL survey aimed to juxtapose these database systems. Our initial survey was limited by time constraints and was not intended to be exhaustive. However, a review of existing literature, as detailed in Chapter 3, underscored the necessity for a thorough ETL evaluation. Although numerous studies have compared the performance of these databases, we noted a lack of literature on ETL using HTCondor.

4.4.2 Composite Logo Plots

Composite logo plots provide a visualization of the region around the SNP and the sequence matches to the PWM. This logo plot helps with the interpretation of the PWM based on the corresponding SNP Allele with the SNP sequence as seen in Figure 4.5.

The composite logo plot PWM and corresponding p-values were generated by the initial run of the JASPAR and ENCODE datasets with the atSNP R package (consuming 115,000 CPU hours). In generating the p-values we initially believed that the Composite

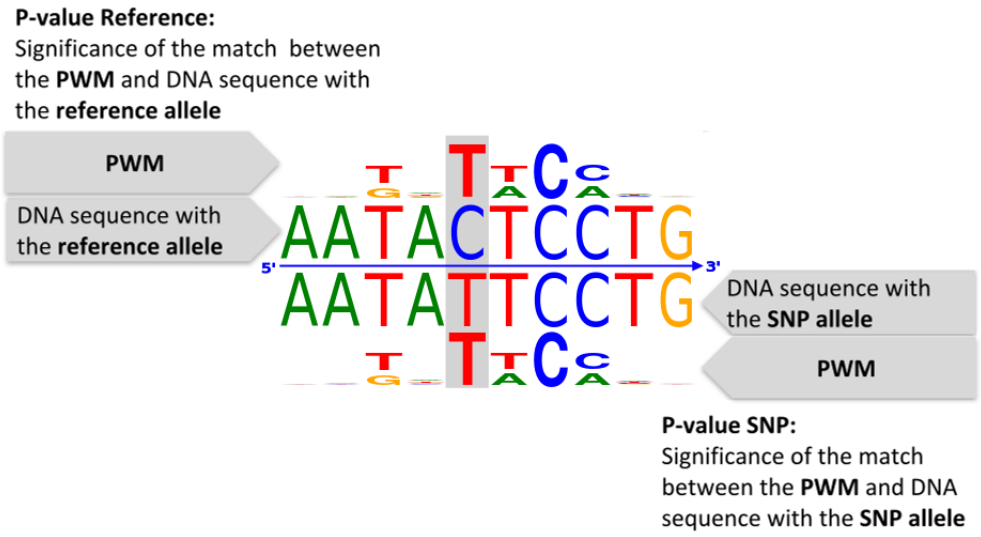


FIGURE 4.5: Example Composite Logo Plot

Logo Plot could be generated and stored as static images, as originally developed in the atSNP R package. However, when considering the number of images we would need to store (37 billion), storing and retrieving these images would have become cost prohibitive. Additionally, similar work was already being done by cloud vendors using blob storage infrastructures such as [107]. Therefore, we attempted to address the data storage need and provide dynamic generation of the Logo Plot.

Our implementation of dynamically generated Composite Logo Plots helped address our disk space consumption for pre-generated logo plots requirement. We addressed the disk space consumption issue through the use of *d3.js* [108] and Scalable Vector Graphics (SVG) [109]. Our novel logo plot generation technique moved the need of pre-computing the Composite Logo Plots beforehand and storing these images by moving the computing to the endpoint. At the endpoint, the web browser is able to render the images by using the existing atsnp datum's and calculate each logo plot based on the individual datum PWM. The JavaScript libraries and display logic that we developed to produce the logo plots is available here [110].

4.4.3 ETL with HTCondor

We required a method to quickly load 37 TB of data into a database. We developed custom Python code to evaluate each combination and verify if a given record and the corresponding P-value score were above a predetermined threshold. The parallel independent characteristics of this task and the data provided an ideal use case to be run in a

distributed computational cluster environment. After we refined our process, we used HTCondor [3] to distribute the ETL pipeline processes.

When searching the literature, we could not find any instances of using HTCondor to populate NoSQL databases. Our original plan was to ETL data from 309 billion records. We tried to ETL the data using one computer but realized that the time would be excessive. As a result, we used a divide-and-conquer methodology that is typically used in a dataset analysis.

Our custom Python ETL script involved preprocessing the data by sampling smaller subsets. We split the data set into smaller files for faster transfer to individual HTCondor compute nodes. These smaller partitions reduced latency during data transfer.

However, our HTCondor cluster is a shared resource, so a four-day delay for the entire ETL process was impractical. Therefore, we limited the execution time of our Python ETL code itself to four hours. This script focused on loading data into an Elasticsearch database.

Following the ETL process, a separate script parsed the data attributes. This script transformed the data into JavaScript Object Notation (JSON) format for efficient insertion into the Elasticsearch database using its official API [111].

We initially assumed our Python code would run smoothly on HTCondor because it executed locally without problems. However, submitting the tasks to HTCondor resulted in numerous failures.

This led to a time-consuming troubleshooting process. We had to manually identify the failed tasks and then rerun the code specifically on the problematic subset of atSNP data outside of the HTCondor cluster scheduler. After numerous attempts and failures we realized this event was more frequent than anticipated and prompted a further investigation.

Our work repurposed a widely adopted strategy for telemetry and time series applications that involves the concurrent transmission of multiple data streams to a database, known as the multi-node push method [112]. We adapt this method for our purposes, employing HTCondor as an intermediary for ETL operations. This method proved to be effective, and upon reviewing the existing literature, we identified our use of HTCondor in tandem with our script for database ETL purposes as a new contribution to the field.

In the initial test in the test cluster with three nodes, each node could maintain six

concurrent ETL scripts per node (corresponding to the number of CPUs per host) without experiencing a "Bulk queue full" error. Since the cluster was built with fifteen data search nodes and three management aggregation nodes, an assignment of the ETL task was submitted to HTCondor using these parameters.

4.5 Conclusions

Database	JASPAR	ENCODE	hg version	# initial SNPs	Pre-computed data
atSNP Search	✓	✓	hg38	133M	✓
SNP2TFBS	✓		hg19	85M	✓
Raven	✓		hg17	30K	✓
OncoCis	✓		hg19	N/A	

TABLE 4.3: Comparison of motif-based regulatory SNP discovery databases

The Affinity Testing SNP (atSNP) Search platform was designed to address the challenges posed to genomic researchers. This system addressed the challenges of searching a vast domain space for TF from SNP-PWM by providing a database to search the tens of billions of SNP.

We set out to satisfy our motivation for the atSNP Search web resource platform. Our work compared the resources of existing regulatory motif based TF SNP regulatory databases and found that the existing resources were insufficient (Table 4.3). Therefore, we were motivated by the following:

- providing genomic researchers atSNP statistics of the entire human genome through a web interface
- a more comprehensive motif-based discovery of regulatory variants beyond the current state-of-the-art
- statistical quantification of SNPacPWM motif matches and changes in motif matches and
- a convenient graphical depiction of potential TF on SNP-PWM via motif logo plots

Our case study demonstrated the feasibility of using NoSQL database engines for large-scale user searchable genomic SNP databases. Our successful implementation is proof that the systems were feasible and we achieved our goal to provide an atSNP Search

platform a database engine. We achieved the objectives described, and our final cluster architecture can be seen in Figure 4.6

During our development, additional factors that we were constrained by included financial, personnel time, speed to discovery, and operational domain knowledge. The operational domain knowledge was an even larger contributing factor when we took into account our needs for: administrators support-ability, the physical data-center footprint, and the network speed.

The implementation time frame of our work was constrained and we encountered a series of unexpected challenges. However, during the development process, we faced limitations due to range query performance issues. This led to the addition of search functionality as a selection criterion in our feasibility study of database infrastructures.

To overcome our unforeseen issues, new techniques were developed. For example, we introduced a new ETL method for distributed NoSQL databases using HTCondor. Additionally, we were the first to develop a motif logo plot in a JavaScript library.

The deployment of the atSNP Search platform, whose architecture can be seen in Figure 4.6, successfully provides a genomic researchers a motif database resource for TF in the human genomes from SNPPWM. Despite limitations imposed by financial constraints, we developed a scalable effective system for searching and viewing atSNP data set. Our core contributions from this chapter are as follows.

- a genomic search tool that provides the knowledge of human SNPs on TF-DNA interactions based on PWM
- a comparison of MySQL, Apache Cassandra and Elasticsearch for use as a genomic SNP-PWM motif databases
- a big data loading method to extract transform and load genomic data using HTCondor [3] for ingestion into a database
- a generation visualization library producing composite motif logo plots

To this end, we showcased our ability to execute novel research while also producing a functional operational resource.

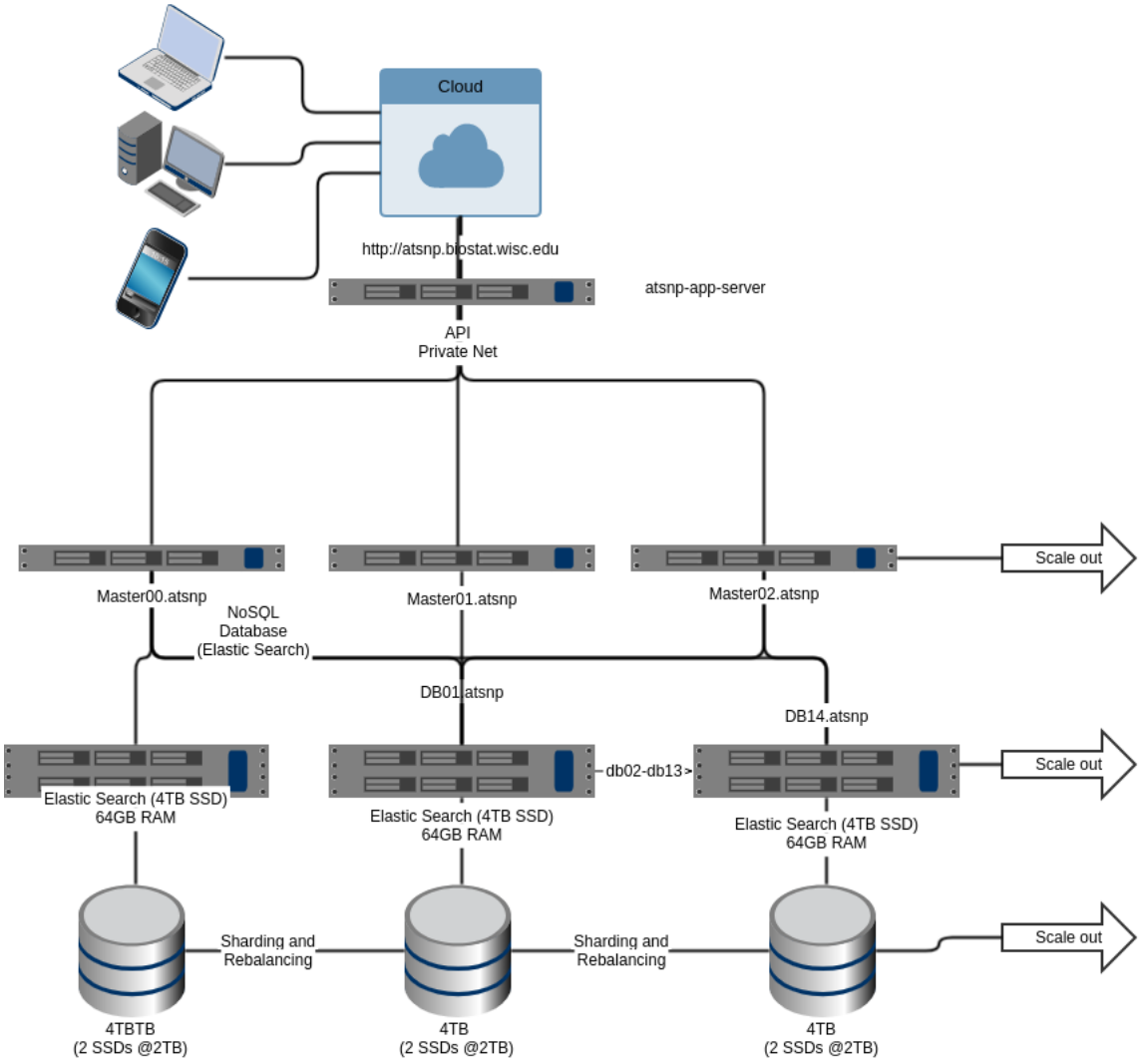


FIGURE 4.6: atSNP search infrastructure

4.6 Failures new motivation

As time progressed and after the atSNP Search platform was put into production for active search, the failure conditions began to affect the system. These failures cause a rebalance of the storage shards of the data. Unfortunately, the shard rebalance process caused additional failures by write amplify during the copy and rebalancing of the data on the cluster. At this point, the atSNP Search platform was experiencing a text book "batch-correlated disk failures" [113] [114]. We attributed these failures to using one HDDM for HDD storage and the purchase of these drives all at the same time. In experiencing such a failure, data slowly corrupted and was no longer available from the search results (original data was stored in a secondary location for future re-import). What we experienced is typically

considered a catastrophic failure and one that we did not expect. This caused us to ask additional questions.

We had considered node failure rate estimates within the cluster on a per node basis and had calculated node availability based on system component failure statistics, specifically: RAM, power supply and disk failures. However, our initial estimates proved to be naive as these estimates failed to take into account batch-failure events. We built a computational cluster that was composed of eighteen nodes and is considerably smaller than other computational clusters such as Biostat; however, our HDD failure rates for our data storage disk drives was high by all available metrics (>30% failure rate within a year).

In practice, batch-correlated disk failures are rare events. These events can significantly challenge a cluster's resiliency and lead to unexpected data loss and failures. Our cluster experienced numerous disk failure events that were followed by cluster rebalance events. Thus, the cluster rebalancing caused additional disk failure events, which lead to a cascading failure event.

We were motivated to explore existing research to gain insight into the prevention and management of cascading failures.

We embarked on an in-depth exploration of HDD failures. Thus, our investigation aimed to understand the underlying causes and implications of such cascading failure events in complex computing environments. The forthcoming chapter will focus on analyzing HDD failures, incorporating a predictive element to these events. This analysis aims to improve our understanding of HDD failures and contribute to the development of strategies to predict and mitigate such incidents in complex computing systems.

Chapter 5

Hard Disk Drive Faults and Failures and Cluster Task Failures; Background and Related Work

In this chapter, we present an overview on HDD concepts and S.M.A.R.T. attributes. Next, we present some data mining techniques, followed by exploring machine learning algorithms and time series concepts.

The last area of focus is the prediction of the failure of computation cluster tasks. We focus on data collection and state-of-the-art system monitoring applications currently in use. Beyond computational cluster and system monitoring applications, we explore current works in anomaly detection as it relates to computational system faults.

This section of our work was motivated by our prior work in atSNP search and specifically that our monitoring system missed cascading failure event. Realizing fault and failure events can be problematic and cause cluster failures. We explore background work in this field to better understand existing work and shape our own.

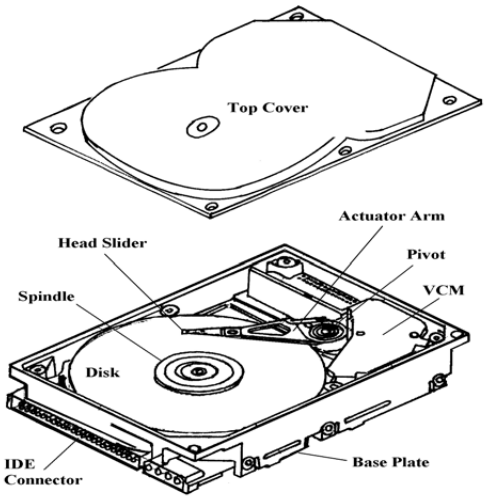
5.1 Hard disk drives faults and failures

The hard disk industry has been around for more than half a century, with the first HDD manufactured in 1956 by IBM. This drive had only 5 *MB* of capacity. Today, anyone has access to a HDD on their computer, but a few decades ago, a HDD was a gigantic device that was the size of a washing machine. The ever-growing demand for digital data storage

has fueled rapid advances in the field in recent years. The ability to meet this demand at a relatively low cost makes the HDD the undisputed candidate for online storage [115].

A HDD is a magnetic data storage device that uses one or more rotating disks (platters) coated with magnetic material, as seen in Figure 5.1. The components of a HDD can be classified into four categories: magnetic, mechanical, electromechanical, and electronic. The disks are paired with magnetic heads that can read and write information on the surfaces of the disk. Data is written and read from the HDD in chunks of data or data blocks, and each block is assigned to a specific addressable location on the HDD. These blocks are the smallest unit of storage on any given HDD. HDDs can be connected to systems via *PATA* (Parallel Advanced Technology Attachment), *SATA* (Serial Advanced Technology Attachment), *USB* (Universal Serial Bus) or *SAS* (Serial Attached SCSI) cables [115]. Figure 5.1 illustrates some HDD essential components.

Magnetic HDDs are a critical component for large-scale data systems due to their cost effectiveness. Even though the Cost Per Gigabyte (CPG) of non-magnetic SSD continues to fall, HDD CPG remains highly economical due to a confluence of factors around the component technologies employed. Thus, HDD continues to be used as a core component in large-scale storage systems. As a core component, its faults and failures can greatly affect the usability and performance of the system [116].



center the figure

FIGURE 5.1: *HDD* components. [115]

5.1.1 Disk Failures

Failures can be categorized into two main groups, predictable and unpredictable [117]. Unpredictable failures, such as electronic and some mechanical problems, occur quickly

without any chance of control from the user. For example, a power surge may cause chip or circuit damage. Predictable failures are characterized by degradation over time. All mechanical components undergo degradation over their useful life. Therefore, attributes can be monitored, making it possible to use predictive failure analysis to predict a failure before it fails [117]. Our work was partially inspired by the work of Velasco *et al.* [93] who showed that MLM could be applied to peripheral components in optical networks. Since both optical networks and HDDs transmit data across a bus, we show that this work can be applied to HDDs.

Understanding a failure is dependent on what constitutes the failure. Schroeder *et al.* [118] argues that HDD vendors use different definitions of what a fault is than customers. A disk misbehavior may consist of a read operation taking longer to complete than usual. For vendors, this may not be alarming because a HDD's internal performance threshold has not been crossed. Furthering this case, Elerath and Shah found [119] "The drive manufacturer would count only 36 of all the drives returned from customers in their failure rate calculations." Elerath and Shah study is even more revealing since they studied the server class disk drives, which are typically reserved for servers and theoretically have higher standards. These server class drives often become members of a Redundant Array of Independent Disks (RAID) that provided data protections through redundancy. Validating Elerath and Shah, Pinheiro *et al.* [120] found, drive manufacturers often quote yearly failure rates below 2%, while user studies have seen rates as high as 6%.

The study by Schroder *et al.* also states that even if two HDD are of the same model, they can differ in their behavior because disks are manufactured using processes and parts that can change. A simple change in a drive's firmware or in a hardware component, or even in the assembly line on which a drive was manufactured, can change the failure behavior of a disk.

According to the Backblaze Company, a disk is considered failed when [121]:

"it is removed from a Storage Pod and replaced because it has 1) totally stopped working, or 2) because it has shown evidence of failing soon. A drive is considered to have stopped working when the drive appears physically dead (e.g. won't power up), doesn't respond to console commands or the RAID system tells us that the drive can't be read or written."

Backblaze conducted a study to understand the failure rates by HDD age [122]. Backblaze theorized defects come from three factors: factory defects, resulting in infant mortality ; random failures; and parts that wear out resulting in failures after much use. Figure 5.2 shows how these factors are expected to contribute to a failure rate and the resulting factors produce a failure rate bathtub-curve. In comparing the theory bathtub curve to the data Backblaze found the bathtub to be leaking, as the left side of the bathtub curve (decreasing failure rate) and was much lower and more consistent with a constant failure rate, Figure 5.3.

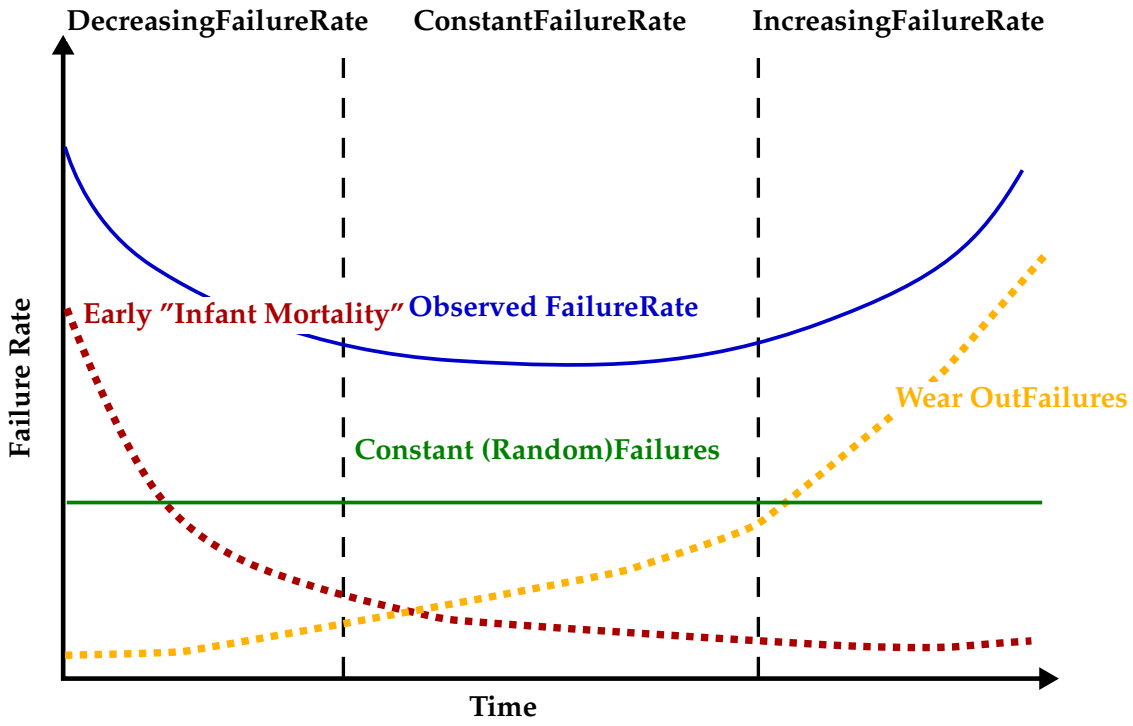


FIGURE 5.2: Bathtub curve [123]

5.1.2 *S.M.A.R.T.* Attributes

S.M.A.R.T. emerged from the need to protect critical information stored on disk drives. As system storage capacity requirements increased, the industry identified the importance of creating an early warning system that would allow enough lead time to back up data if failure was imminent, preventing catastrophic data loss [117].

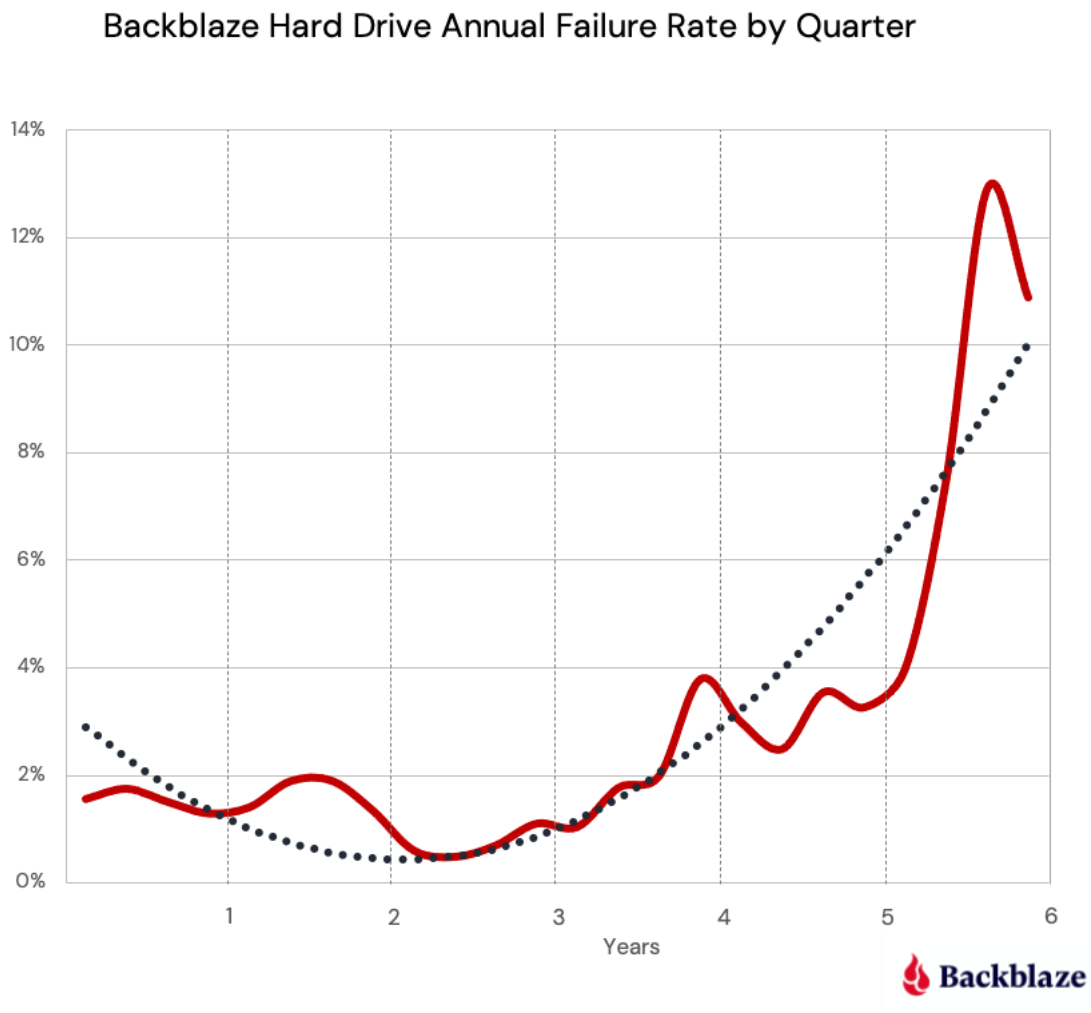


FIGURE 5.3: Backblaze leaking bathtub curve (red line = failure rate, dotted line = trend-line) [122]

S.M.A.R.T. includes a series of attributes chosen specifically for each drive model. This individualism is important because HDD architectures vary from model to model. Attributes and thresholds that detect failure for one model may not be effective for another model.

Table 5.1 presents some of the S.M.A.R.T. attributes and their meaning.

TABLE 5.1: *S.M.A.R.T.* Attributes

ID	Attribute Name	Description
smart_1	Read Error Rate	Rate of hardware read errors that occurred when reading data from a disk surface.
smart_2	Throughput Performance	Throughput performance of a hard disk drive.

TABLE 5.1: *S.M.A.R.T.* Attributes (Cont.)

ID	Attribute Name	Description
smart_3	Spin-Up Time	Average time of spindle spin up (from zero RPM to fully operational [milliseconds]).
smart_4	Start/Stop Count	A tally of spindle start/stop cycles.
smart_5	Reallocated Sectors Count	Count of reallocated sectors.
smart_7	Seek Error Rate	Rate of seek errors of the magnetic heads.
smart_8	Seek Time Performance	Average performance of seek operations of the magnetic heads.
smart_9	Power-On Hours	Count of hours in power-on state.
smart_10	Spin Retry Count	A total count of the spin start attempts to reach the fully operational speed.
smart_11	Recalibration Retries	A count that recalibration was requested.
smart_12	Power Cycle Count	A count of full hard disk power on/off cycles.
smart_184	End-to-End error	A count of parity errors which occur in the data path to the media via the drive's cache RAM.
smart_187	Reported Uncorrectable Errors	The count of errors that could not be recovered using hardware ECC
smart_188	Command Timeout	The count of aborted operations due to HDD timeout.
smart_189	High Fly Writes	This attribute indicates the count of rewritten or reallocated information over the lifetime of the drive.
smart_190	Temperature Difference	Value is equal to (100-temp. C), allowing manufacturer to set a minimum threshold which corresponds to a maximum temperature.
smart_191	G-sense Error Rate	The count of errors resulting from externally induced shock and vibration.
smart_192	Power-off Retract Count	Number of power-off or emergency retract cycles.
smart_193	Load Cycle Count	Count of load/unload cycles into head landing zone position.
smart_194	Temperature	Indicates the device temperature.
smart_195	Hardware ECC Recovered	
smart_196	Reallocation Event Count	A count of attempts to transfer data from reallocated sectors to a spare area. Both successful and unsuccessful attempts are counted.

TABLE 5.1: *S.M.A.R.T.* Attributes (Cont.)

ID	Attribute Name	Description
smart_197	Current Pending Sector Count	Count of "unstable" sectors (waiting to be remapped, because of unrecoverable read errors).
smart_198	Uncorrectable Sector Count	The total count of uncorrectable errors when reading/writing a sector.
smart_199	UltraDMA CRC Error Count	The count of errors in data transfer via the interface cable as determined by ICRC (Interface Cyclic Redundancy Check).
smart_200	Multi-Zone Error Rate	The count of errors found when writing a sector.
smart_201	Soft Read Error Rate	Count indicates the number of uncorrectable software read errors.
smart_223	Load/Unload Retry Count	Count of times head changes position.
smart_240	Head Flying Hours	Time spent during the positioning of the drive heads.

Some variables are considered critical by the literature in predicting failure events; these variables include 5, 12, 187, 188, 189, 190, 198, 199 and 200 [124]. Our work attempts to validate the existing literature and to pay attention to other variables that are not observed.

Many S.M.A.R.T. attributes are normalized by the HDDM from their raw values. There is no standard for how manufacturers convert raw attribute values to normalized ones: it can be a linear, exponential, logarithmic, or any other range normalization. The range is normally 0-100 and for some attributes 0-255. Generally, S.M.A.R.T. attributes with higher values are always better (except for the temperature in some HDDM). As a result, it is difficult to have a clear overview of a HDD behavior on a cloud storage system, since they usually use dozens of different disk models.

5.1.3 Data Mining

The technological boom has created a lot of information in the digital world. These data are a great source of knowledge extraction for all companies. The constant need to interpret data and discover relevant information has caused data analysis to develop rapidly in recent years.

Fayyad *et al.* [125] described the necessary steps to extract relevant information from databases. The *KDD* (Knowledge Discovery in Databases) process is a set of continuous activities that share the knowledge discovered from data. According to Fayyad *et al.* this set is composed of five steps: data selection, preprocessing and data cleaning, processing of data, data mining, interpretation, and evaluation of results (Figure 5.4).

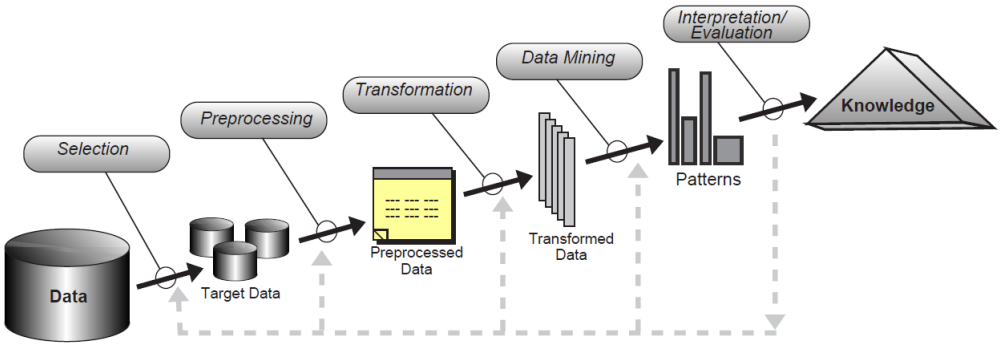


FIGURE 5.4: An overview of the KDD steps [125]

5.1.4 Machine Learning

Machine learning's principal objective is to understand the way data is related. It is based on algorithms that can learn models from data and make predictions on unseen data. Machine learning tasks can be classified as supervised, unsupervised, or semi-supervised. In supervised learning, the goal is to train the machine using data that is already labeled in order to learn the distribution of the data to be able to predict future events. In this case, the labeled values correspond to what is called the target variable. In unsupervised learning, examples do not have a target variable associated, so the objective is to group similar observations without knowing what is represented by each group. Semi-supervised learning is a task that uses both labeled and unlabeled cases.

Supervised learning tasks can be split into two categories of algorithms: classification and regression. Classification is a form of data analysis that extracts models that describe important data classes. Such models, called classifiers, predict categorical class labels. The classification model is built from the analysis of the training data set and is used to predict the class label for observations that are not categorized. Regression is used to predict numeric or continuous values. Regression is a statistical methodology that is most often used for numeric prediction, so it is used to predict missing or unavailable numerical data values rather than (discrete) class labels [126].

The work of Flup *et al.* [127] demonstrated an ML approach to predict failure events based on the computational system logs.

5.1.5 Classification Algorithms

In the development of this work, two machine learning algorithms were used for the classification task Random Forests and Support Vector Machine (SVM). In the following, the algorithms are described in more detail and in Table 5.2 showing the mathematical principles of various SVM kernels.

The goal of a classification task is to obtain a good approximation of the unknown function that maps predictor variables to the target value. The unknown function can be defined as $Y = f(X_1, X_2, ..., X_p)$, where Y is the target variable, $X_1, X_2, ..., X_p$ are features, and $f()$ is the unknown function we want to approximate. This approximation is obtained using a training data set $D = \left\{ \langle x_i, y_i \rangle \right\}_{i=1}^{n}$

5.1.5.1 Random Forest

The Random Forest algorithm was first introduced by Breiman [128] and is defined as an ensemble method. An ensemble is a set of multiple models, in this case, a set of decision trees. As the name suggests, this algorithm creates a forest with a large number of decision trees, where each considers a distinct random subset of features when forming the decision nodes while accessing a subset of the training data. Each classifier tree is a predictor component.

In classification, Random Forest constructs its decision by counting the votes of the predictor components in each class and then selects the winning class by checking the number of votes accumulated. The process of this algorithm is represented in Figure 5.5, the first phase consists of training each decision tree with data subsets from the training set. The test cases are then classified by majority vote.

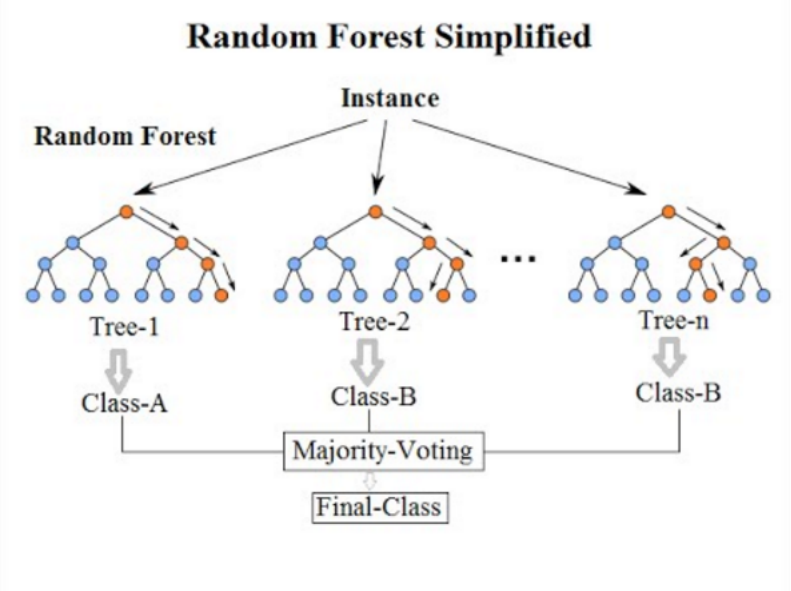


FIGURE 5.5: Random Forest example [129]

5.1.5.2 Support Vector Machine

Support Vector Machine SVM was originally proposed by Boser *et al.* [130] in 1992. His method tries to find the largest margin that separates different classes of data. The objective of the SVM is to construct an optimal hyperplane that can separate different classes of data. In Figure 5.6, it is possible to see how SVM works. There are several straight lines that can be drawn to separate the data. The support vectors are data points that are closer to the hyperplanes and they serve to choose the best one, represented by the filled points.

center the figure

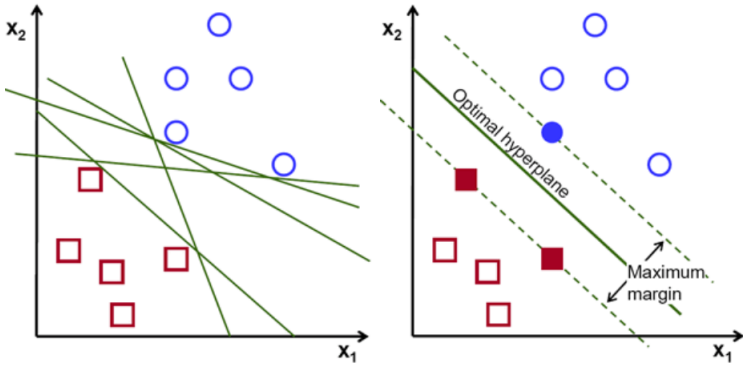


FIGURE 5.6: SVM example [131]

Data can not always be separable in a linear way. In these cases, the SVM maps the data to a space of higher dimension. At this point, the concepts of soft margin and kernel trick are introduced. The main idea of a soft margin is to allow some examples to be placed on the wrong side of the dividing hyperplane. The kernel transforms non-separable data

to separable data by adding more dimensions. Nonlinear kernel functions were proposed by Boser *et al.* [132] so SVM could be applied to data that could not be divided by linear hyperplanes. Table 5.2 provides some of the kernel functions.

TABLE 5.2: SVM Kernel Types

Kernel Type	Formula
Polynomial kernel	$(x_i, x_j) = (x_i * x_j + r)^p, r \geq 0$
RBF kernel	$(x_i, x_j) = exp(-\gamma \|\|x_i - x_j\|\|^2), \gamma > 0$
Sigmoid kernel	$(x_i, x_j) = tanh(\eta x_i * x_j + v)$

5.1.6 Time Series

In almost every scientific field, measurements are performed over time [133]. The purpose of time-series models is to extract the most meaningful knowledge from the shape of the temporal data. A time series is a collection of observations obtained chronologically. Time series can be regular if there is an equally spaced interval of time between the observations, and irregular if the opposite occurs.

The values are typically measured at equal time intervals (e.g.,every minute, hour, or day). This type of data can be characterized in four different movements [126]:

- **Trend or long-term movements**: These indicate the general direction in which a time series graph is moving over time.
- **Cyclic movements**: Are the long-term oscillations about a trend line or curve.
- **Seasonal variations**: Are nearly identical patterns that a time series appears to follow during seasons of successive time.
- **Random movements**: As the name refers to, these are random movements with no pattern associated.

Figure 5.7 depicts the four different types of movements.

These movements referred to above can also be grouped into two types of data, stationary and non-stationary. In stationary data, the time series values do not depend on the time that the observations were collected, therefore it will not have predictable patterns in the long term. Non-stationary data typically have some kind of trend or seasonality over time [135]. In Figure 5.8 it is possible to better see the differences between these two types.

center the figure

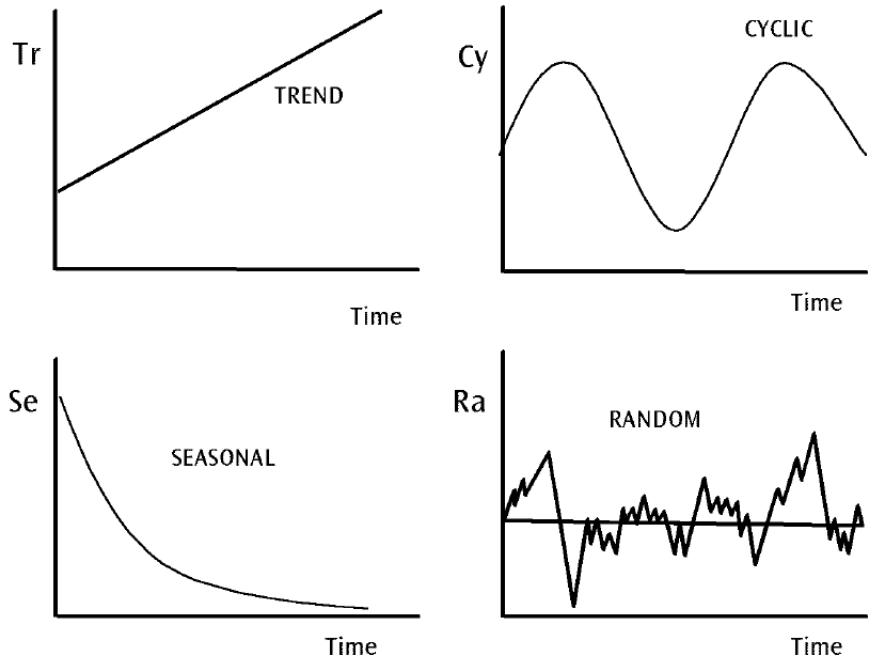


FIGURE 5.7: Time Series Types [134]

5.1.7 Vector Auto Regression

Lütkepohl *et al.* [135] say that if time series observations are available for a variable of interest and the past observations contain information about the future development of a feature, it is worth using the feature as a forecast.

The VAR model expresses each variable as a linear function of its own past values, the past values of all other variables considered, and a serially uncorrelated error term [137]. Each variable has an equation describing its progression over time. This equation includes the lagged (past) values of the variable, the lagged values of the other variables in the model, and an error term.

To better understand the VAR, we simplified to a two-variable or bivariate autoregression [138]. Where the α's and δ's are the coefficients of the linear projection of y_t onto a constant and past values of y_y and x_t, and the lag length m is sufficiently large to ensure that u_t is a white noise error term. Although it is not essential that the lag lengths for y and x are equal, we follow typical practice by assuming that they are identical. We can visualize the bivariate autoregression in the Equation 5.1.

center the figure

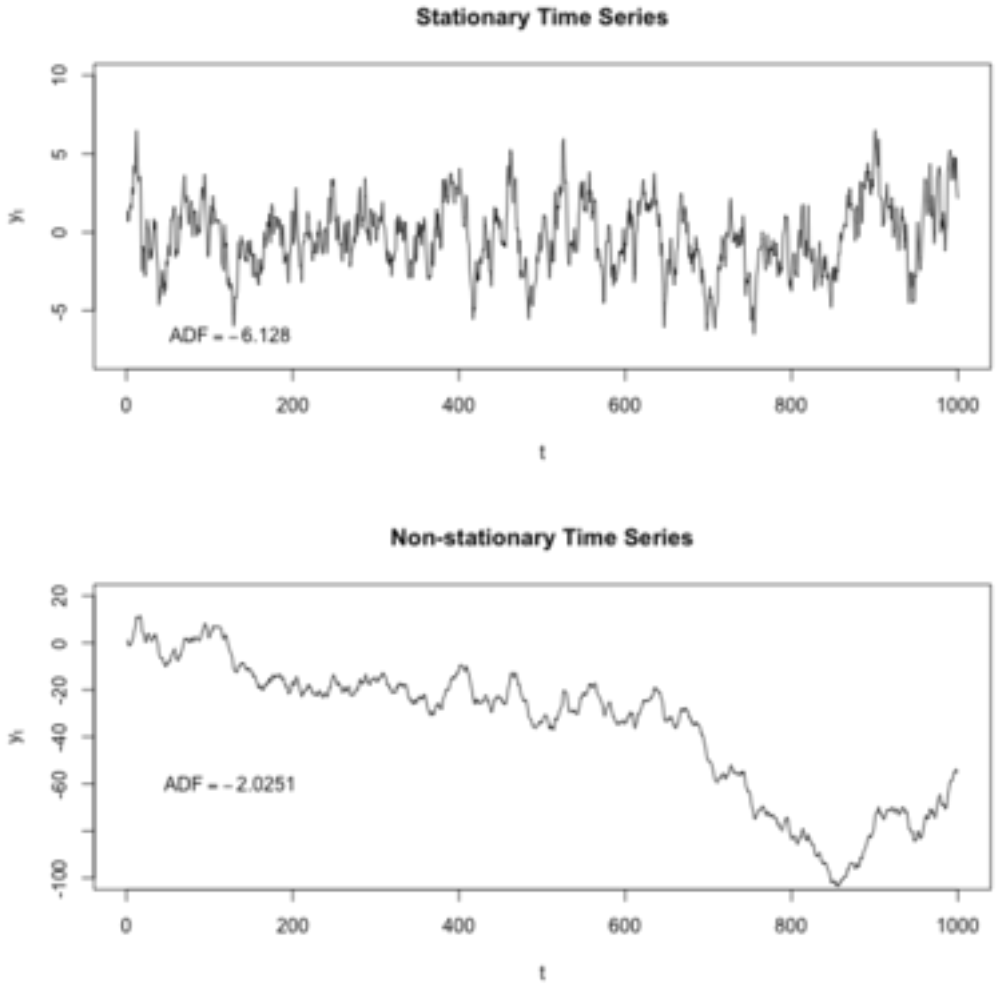


FIGURE 5.8: Stationary and non-stationary data [136]

$$y_t = \alpha_0 + \sum_{l=1}^{m} \alpha_l y_{t-l} + \sum_{l=1}^{m} \delta_l x_{t-l} + u_t \tag{5.1}$$

5.1.8 Imbalanced domain learning

This project faces an imbalanced domain learning problem. This occurs whenever the user has an interest in cases that are rare in the training set. This can create several obstacles in the learning methods that are applied. The models created by standard learning algorithms tend to be biased towards the majority class, and because of that, the evaluation metrics will not capture the completeness of the models for relevant cases.

Han *et al.* [126] describe this imbalance: "given two-class data, the data are class-imbalanced if the main class of interest (the positive class) is represented by only a few tuples, while the majority of tuples represent the negative class."

Therefore, it is important to pay close attention to this type of data and take the necessary steps to prevent getting wrong information from the data mining processes that are used.

Two of the methods utilized to handle imbalanced data are oversampling and undersampling. Oversampling works by resampling positive tuples so that the training set contains an equal number of positive and negative tuples. Undersampling works by decreasing the number of negative tuples. It randomly eliminates tuples from the majority (negative) class until there is an equal number of positive and negative tuples [126].

In Figure 5.9 we can see a clear example of these two sampling methods:

center the figure

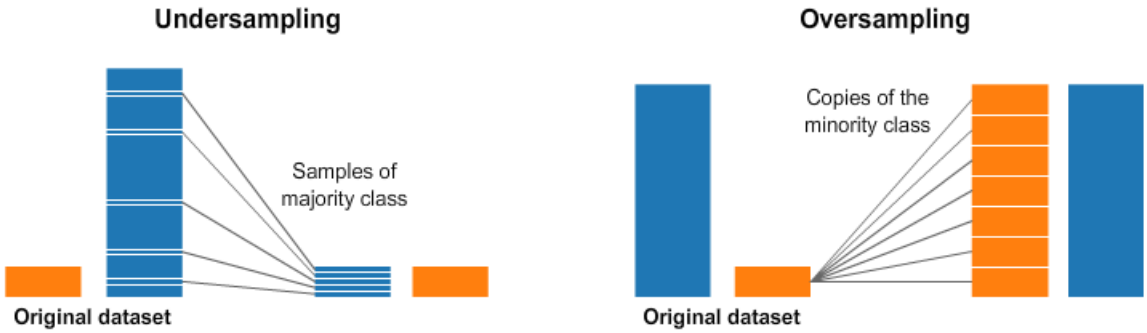


FIGURE 5.9: Undersampling and Oversampling examples. [139]

5.1.9 Evaluation Metrics

In machine learning, the terms of positive tuples (tuples of the class of interest) and negative tuples (all the other tuples) are normally used. Four more terms are used, and they are the base-line of many evaluation metrics used. In the following, each of the terms is explained [126]:

- **True positives (TP)**: Positive tuples that were correctly labeled by the classifier.
- **True negatives (TN)**: Negative tuples that were correctly labeled by the classifier.
- **False positives (FP)**: Negative tuples that were incorrectly labeled as positive.
- **False negatives (FN)**: Positive tuples that were incorrectly labeled as negative.

With these four definitions, we can build a confusion matrix. It is a square matrix with as many rows and columns as there are classes of the data. Each row represents the actual class of the observation, while each column represents the predicted class. This matrix serves as a source of information for most of the metrics used. An example of a confusion matrix can be seen in Figure 5.10.

center the figure

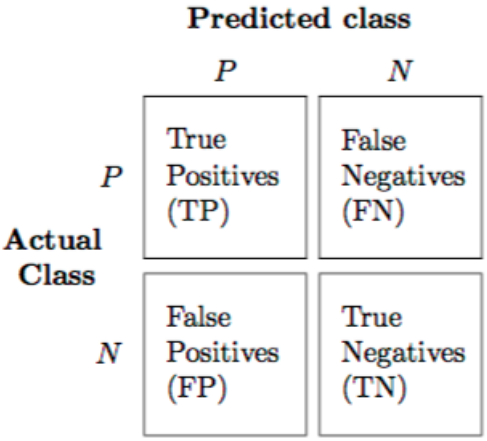


FIGURE 5.10: Confusion Matrix. [140]

In Figure 5.11 we can see the metrics that are the most used to evaluate learning models.

center the figure

Metric	Formula
True positive rate, recall	$\frac{TP}{TP+FN}$
False positive rate	$\frac{FP}{FP+TN}$
Precision	$\frac{TP}{TP+FP}$
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
F-measure	$\frac{2 \cdot precision \cdot recall}{precision + recall}$

FIGURE 5.11: Evaluation Metrics. [141]

5.1.10 Cascading Failures - COME BACK IF TIME PERMITS

The literature has numerous examples [142] [143] [144] [145] [146] of network-centric cascading failures. Each of these examples includes statistical modeling and RCA from base components up to the management layer.

Wang *et al.* [143] [145] proposed a network cascading failure mitigation strategy that involves allowing overloaded edges to redistribute some of their load to neighboring edges before they completely fail. The goal was to improve the robustness of the network against cascading failures. Their findings suggest that the mitigation strategy can be a simple and useful tool to improve the reliability and security of complex infrastructure networks.

5.1.11 Backblaze dataset and related work

Much existing work on hard drive failure prediction uses the Backblaze dataset. This data set aggregates S.M.A.R.T. variables [124] [147] on more than 100,000 HDD daily.

Aussel *et al.* [124] say that the existing predictive MLM do not perform sufficiently well in the Backblaze dataset due to the extremely unbalanced ratio of 5000: 1 between healthy and failing HDDs. For that reason, they selected MLM for classification, like SVM, Random Forests and Gradient Boosting Trees. They achieved results of 95 precision and 67 recall with the Random Forests MLM and 94 precision and 67 recall with Gradient Boosting Trees. The SVM performance had a precision below 1%.

Wang *et al.* [147] argue that reactive fault-tolerant measures, like RAID's and Error Correction Codes (ECC) are not enough to mitigate or eliminate the negative effects of the HDD's failures. Proactive measures are more efficient because they predict failures in advance. However, the built-in prediction MLM that the HDDM used have weak predictive power. To overcome these issues, Wang *et al.* proposed a neural network architecture called Attention-augMENted Deep architEctuRe (Amender) [147]. Amender used a layered neural network that was composed of a feature integration layer (combined S.M.A.R.T. attributes into a single representation); a temporal dependency extraction layer (a recurrent neural network (RNN), specifically a gated recurrent unit (GRU) to analyze how S.M.A.R.T. values change over time); an attention layer (focused on the most important days in the S.M.A.R.T. data sequence for predicting failures); and finally a classification layer (takes the output from the attention layer and predicts the health status of the hard drive and the likelihood of failure).

After analyzing the results, Wang *et al.* [147] concluded that different S.M.A.R.T. attributes have different abilities to indicate failures. Compared to recurring neural networks (RNNs), the architecture improved by 8.3 in failure prediction and 90.2 in health status assessment.

Shen *et al.* [148] proposed a Random Forest prediction model for failure prediction for HDD's. They show that most statistical approaches, machine learning, and deep learning technologies are good at identifying failures that occur more frequently but perform poorly when faced with less known behavior. They used a clustering-based undersampling method, so the data imbalance problem was solved and the quality of training improved. The results show that the Random Forest model can achieve a Failure Detection Rate (FDR) of over 97.67 with a False Alarm Rate (FAR) of 0.017%.

Li *et al.* [149] propose two predictive MLMs based on Decision Trees (DT) and Gradient Boosted Regression Trees (GBRT) and apply them to two different real-world datasets (one with 121,698 and the other with 39,091 hard disks). In data preparation and preprocessing, they use quantile functions to select the more key features on healthy drives and failed drives. Li's results are referenced in Table 5.3.

Model	dataset	FDR	FAR
DT	Big	85.1	0.07
GBRT	Big	79.7	0.02
DT	Small	96.0	0.12
GBRT	Small	86.0	0.02%

TABLE 5.3: Li's result from the Decision Trees (DT) and Gradient Boosted Regression Trees (GBRT)

Zhao *et al.* [150] believe that much of the previous research in the area failed to consider the characteristics of the observed features over time and tend to make predictions based on individual or a set of attributes. They also believe that attribute values observed over time are not independent and that a sequence of observed values with certain patterns might be a good indicator of whether a drive may fail soon. Therefore, they considered the observations from the disks as a time series and applied a hidden Markov model and a hidden semi-Markov model to build a predictive MLMs that could label disks as healthy or pre-failing. Although their FDR results are not high (up to 46 for single attributes and 52 for multiple attributes), they achieve a FAR of zero in both cases.

5.2 Computational cluster task failures

Clusters use middleware that provides an interface (command line, API, or portal) for submitting single or batch tasks. When submitting a task, you benefit from having knowledge of the resource needs of the given task. Examples of each task's requirements may

include expected execution time, expected memory usage, temporary data storage, and number of cores. After all these constraints are taken into account, a submission is submitted to the cluster and distributed through a cluster queuing management system. These submissions correspond to the cluster task manager. The central task manager manages the hosts on which the submitted tasks run and the number of hosts, from one to millions of these tasks. Unfortunately, submitted tasks can fail to complete. These failure rates are an impediment to optimal utilization of cluster resources and ultimately impact the response time for results and the experience of the submitter.

Task failures are a problem on both HPC-HTC systems. As the following Table 5.4 shows, task failure rates are two different clusters. In the Table 5.4 UW-Biostat refers to the BCG cluster at the University of Wisconsin - Madison and Argonne is the Argonne National Laboratory, Argonne Leadership Computing Facility Theta [151] cluster.

Group	Tasks	% of tasks	Completion status
UW-Biostat	106,695	82.2%	success
UW-Biostat	23,159	17.8%	failed
Argonne	39,508	62.8%	success
Argonne	23,351	37.2%	failed

TABLE 5.4: UW Madison BCG HTC vs Argonne Theta HPC cluster, Theta includes time limited tasks

Computational cluster task schedulers, such as ones used in HPC/HTC systems, distribute tasks based on available HPC/HTC resources. These task schedulers match user requested resources to expected available resources. If a task's resource request is not sufficient to complete the task, then a task will wait in a queue until resource allocations are sufficient on schedulable cluster resources. The goal is to efficiently allocate resources based on task-matched requested resources and optimize on requested resource task demand. Re-balancing algorithms also appear in HPC and HTC systems use rebalancing to handle node failures.

An HPC/HTC scheduler has two areas of optimization:

1. optimizing resource use based on measurement process needs: IO, CPU, network, inter-task communications

2. cluster node temporal availability

Monitoring of the performance metrics system and on the job is essential to evaluate resource allocation and utilization. Long-term trends based on system analysis compared to base lines are widely used as justification for adding or reducing the number of resources allocated over time. However, optimizing resources is difficult without access to fine-grained resource job trends. Resource usage statistics and trend tools may lose fidelity over time due to the number of performance metrics and their stored monitoring data. In addition, the amount of data collected for the metrics grows as a function of the size of the cluster. A common method to string these collected metrics is to use of Round-Robin Databases (RRDs) [152] results in degrading the signal value over time.

Numerous deterministic and heuristic tools, such as cacti [153], ganglia [154], nagios [155], and zabbix [156], to evaluate system utilization and track performance metrics. These tools attempt to provide graphical representations of system performance metrics using the network standard Simple Network Monitoring Protocol (SNMP) [157] to cluster administrators who need to interpret the metrics to better administer the cluster.

Tracking metrics can address application bottlenecks such as application starvation or over usage of memory, CPU , or IO, by allowing critical understanding of resource allocation starvation and optimization. These tools provide a general idea of the performance of the system. They are useful for understanding cluster performance; however, they do not provide the granular process level data which is useful for predicting failures at the job level. Furthermore, according to Haider *et al.*, understanding and adapting to fault conditions can lead to improved dependability in HPC and cloud environments [158].

Larger scale system monitoring of HPC and the Open Science Grid [42] have used systems such as OVIS [159] or TACC Stats [160]. For example, OVIS uses a Bayesian inference scheme to dynamically infer models for the normal behavior of a system and to determine bounds on the probability of values evinced in the system. OVIS addresses hardware-related failure issues and system-level performance analysis on systems based on Mean Time To Failure (MTTF) analysis.

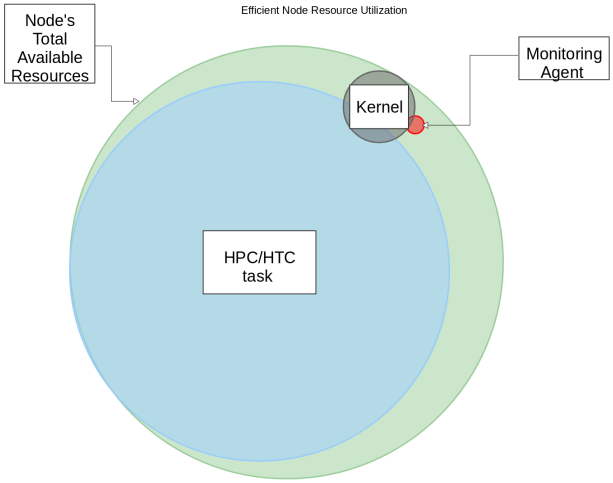


FIGURE 5.12: Efficient job run state on cluster node

Efficiently utilizing computing resources within a cluster requires understanding the resource needs of individual tasks, rather than just the overall system usage. This necessitates measuring resource consumption on a per-task basis, as shown in Figure 5.12.

Research in fault prediction further emphasizes the importance of task-level resource measurements. Guan *et al.* [161] proposed a Bayesian and decision tree approach for cloud environments, demonstrating the potential of decision trees in this area [1]. This concept was further explored by Gao *et al.* [162] who investigated deep neural networks for early faults.

Although these studies offer valuable information, it is important to note that more research has focused specifically on cloud computing environments [163] [164].

The work of Ibidunmoye *et al.* [165] showcased that machine learning techniques could also be used to address performance bottlenecks in cloud systems. Although cloud systems are similar to HPC and HTC systems; however, Ibidunmoye's work never explored performance bottlenecks and failures and faults in HPC or HTC systems. For outage prediction, the paper "Outage Prediction and Diagnosis for Cloud Service Systems" [166] showcased a method for predicting outages in complex cloud environments utilizing a similar approach to Guan *et al.* [161] with both methods utilizing Bayesian networks as the predictive framework.

Modeling resource utilization as a means to make optimization predictions is needed in both HPC and HTC. Even the newest-of-breed techniques, such as the use of Kubernetes [167] for cloud orchestration, only consider CPU in scheduling decisions. This is too simplistic for true optimization. Newer models and software implementations are needed

to more effectively understand the usage and suggest future resource scheduling. Chang *et al.* [168] suggested using multiple variables, such as memory and disk access, and created a dynamic algorithm for containerized network operations centers, achieving better resource allocation through comprehensive monitoring and user-defined algorithms. Wei *et al.* [169] demonstrated a technique to allocate virtual machine (VM) resources based on CPU and memory, albeit limited.

Other work in the area has been motivated by optimization based on energy savings. Pinheiro *et al.* [170] notes that it is key to examine resource reconfiguration and keep a load stable (relatively unchanged), as throughput loss can be resource intensive. Due to heterogeneous hardware and unpredictable loads in shared environments [171], cloud systems may need predictive scheduling and optimization more than traditional, homogeneous HPC clusters.

Bhavani *et al.* [172] stated that refusing to optimize in cloud systems is to ignore the promise of the cloud as an elastic resource, capable of adjusting to computing loads in ways previously limited by dedicated hardware. The work of Bhavani *et al.* is extendable to heterogeneous HPC systems and in places where nodes are purchased over time [173]. Furthermore, their method also works well when users are able to bring their own nodes to a cluster. Mateescu *et al.* [174] describe a technique where scheduling is done in part based on the timing demands of the tasks and can use a combinations of HPC and Cloud as a combined workflow, either managing at the node level (physical machine) or at the VM level.

Although prior research [172][175][176][177] has explored various scheduling and optimization techniques, what distinguishes Rodrigo *et al.* approach is its use of machine learning to recommend real-time workload optimizations applicable to various system sizes and a wide range of applications. However, a key limitation of this work is its disregard for predicting task failures across the cluster [178].

Rodrigo *et al.* attempted to address temporal and locality resource scheduling. This approach is interesting, as it attempts to match multi-dimentional task workflows to resources for data locality and increased throughput. However, the resource locality effect is muted for clusters that do not have local resources (e.g.nodes without local disk for scratch space), which affect scheduled tasks in a homogeneous HPC-HTC environment. However, this work disregards predicting job failures across clusters.

Juve *et al.* [179] tackles the complex issue of resource loss in cluster computing by profiling tasks and considering how the loss affects overall resource utilization over time. All submitted tasks are subject to "spatio-temporal phenomena" [180], meaning their resource usage and execution times are influenced by both spatial factors (task location) and temporal factors (task submission time and queue backlog).

Despite notable efforts to improve job and resource schedulers, job failure rates are still too high according to a survey of the current state of the practice in this field conducted by Jauk *et al.* [181]. Furthermore, Jauk *et al.* argues that "regarding job failures, most works in the literature report that these are due to memory, disk, GPU usage or application-level, for example, users requesting more resources than the job actually needs" [181].

5.2.1 Anomaly detection in recent works

A more recent work in failure prediction and HPC has emerged with Das *et al.* on Doomsday [182]. Doomsday aggregates system events and system logs applying a technique to predict which cluster system poses a challenge. The actual doomsday work was specific to HPE and HPE Cray systems, which limited the generalizability. Even with limitations, Doomsday was expanded by Bautista and Shasta [183] who built on the log aggregation strategy with a deterministic algorithm.

The work of Aksar *et al.* [184] attempts to address anomaly detection in HPC systems. Aksar *et al.* project, Prodigy, attempts to address anomalies in systems through an explainability model. Specifically, Prodigy's "primary goal is to identify whether any compute node within a system displays anomalous behavior that leads to performance variations. We are particularly interested in detecting anomalies that cause performance variability without resulting in program errors or premature termination, as such anomalies tend to be more difficult to detect." [184]. Although the end results may be similar, our work focuses on failures and faults and removing these nodes from clusters before task event impacts.

All this research attempts to address failures or anomalies in many ways. However, stepping outside the computer science domain, we find a failure prediction associated with specifically manufactured components. Specifically, Aljaž Ferencek *et al.* [185] describe a method that uses machine learning to predict failures in components for product failure prediction models. This specific work is interesting because it showcases models

for this type of failure classification and ”The results suggest show that the best performing models are Random Forest and AdaBoost and Linear Regression models”. However, unfortunately the paper’s contributions were limited by ”even in the beginning of the data preparation process, we realized that most of the attributes were unusable due to too many missing and erroneous data entries”.

Chapter 6

Hard Disk Drive Faults and Failures

We present our Hard Disk Drive faults and failures body of our work, discussing our use of machine learning to predict HDD failures. We attempt to answer the question: is it possible to provide explainability for HDD faults and failures and can we predict failures before they occur?

As the primary data storage devices in computational systems, HDDs are integral to the functionality and efficiency of computing operations. Their role as the main storage component in computational clusters underscores the importance of HDD performance within the overall architecture of computational systems. The influence of HDDs on data transfer speeds, throughput, and access reliability is significant, regardless of their specific location within a computational architecture.

As a result of our work in atSNP, we experienced a rapid cascading system failure event. These failures were caused by a rapid succession of multiple disk failures, which caused additional stress on other cluster components. The additional stress in rebalancing the clusters data causes additional failures.

As highlighted in previous chapters, cascading failures can lead to data loss and significantly degrade overall system performance. Although such failures are relatively rare in computing environments, their impact is disruptive. Experiencing a cascading failure event showcases the importance of understanding the initial trigger that sets off such a chain reaction.

Access to a relevant data set is essential to analyze HDD failures comprehensively. In our approach to understanding HDD failures, we utilize the Backblaze data set. These data sets are rich in features and contain the necessary parameters that we anticipated to be critical in predicting HDD failures. Through these data, our objective is to develop

a more nuanced understanding of the failure mechanisms of HDDs, which is needed to improve the reliability and resilience of computational systems.

Although we appreciate the access and community support that Backblaze offers the community in providing its data set to us, this data set is not without challenges. Careful consideration is needed be taken while working with this data, especially since the data is in "RAW" format and is taken directly from the HDD firmware. Figure 6.1 shows the type of data being provided. With this type of RAW data come challenges, such as missing data. We addressed these data concerns and discuss our methods for data cleansing in Table 6.1. One feature of this data set is the metadata per drive included, as referenced in Table 6.2.

Figure 6.1 shows the type of data being provided.

TABLE 6.1: Backblaze data considerations

Name	Description
Blank Fields	The daily snapshots record the SMART stats information reported by the drive. Since most drives do not report values for all SMART stats, there are blank fields in every record. Also, different drives may report different stats based on their model and/or manufacturer.
Inconsistent Fields	Reported stats for the same S.M.A.R.T. stats can vary in meaning based on the drive manufacturer and the drive model. Make sure you are comparing apples-to-apples as drive manufacturers don't generally disclose what their specific numbers mean.
Out-of-Bounds Values	The values in the files are the values reported by the drives. Sometimes, those values are out of whack. For example, in a few cases, the RAW value of SMART 9 (Drive life in hours) reported a value that would make a drive 10+ years old, which was not possible. In other words, its a good idea to have bounds checks when you process the data.
The # of Drives Change	When a drive fails, the "Failure" field is set to "1" on the day it fails. The next day, the drive is removed from the list and is no longer counted, reducing the overall number of drives. On the other hand, new drives are added on a regular basis increasing the overall number of drives. In other words, count the number of drives each day.

TABLE 6.2: Metadata and drive identifiers Backblaze provides in each dataset

Name	Description
Date	The day when the snapshot was taken in yyyy-mm-dd format.
Serial Number	The manufacturer-assigned serial number of the drive.
Model	The manufacturer-assigned model number of the drive.
Capacity	The drive capacity in bytes.
Failure	Contains a 0 if the drive is OK. Contains a 1 if this is the last day the drive was operational before failing.

	date	serial_number	model	capacity_bytes	failure	smart_1_normalized	smart_1_raw	smart_2_normalized	smart_2_raw	smart_3_normalized	smart_3_raw	smart_4_normalized	smart_4_raw
0	2019-10-01	Z305B2QN	ST4000DM000	4000787030016	0	115.0	97236416.0	NaN	NaN	91.0	0.0	100.0	13.0
1	2019-10-01	ZJV0XJQ4	ST12000NM0007	12000138625024	0	67.0	4665536.0	NaN	NaN	96.0	0.0	100.0	3.0
2	2019-10-01	ZJV0XJQ3	ST12000NM0007	12000138625024	0	80.0	92892872.0	NaN	NaN	99.0	0.0	100.0	1.0
3	2019-10-01	ZJV0XJQ0	ST12000NM0007	12000138625024	0	84.0	231702544.0	NaN	NaN	93.0	0.0	100.0	6.0
4	2019-10-01	PL1331LAHG1S4H	HGST HMS5C4040ALE640	4000787030016	0	100.0	0.0	134.0	103.0	100.0	436.0	100.0	9.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
115254	2019-10-01	ZA10MCEQ	ST8000DM002	8001563222016	0	77.0	51786816.0	NaN	NaN	94.0	0.0	100.0	3.0
115255	2019-10-01	ZCH0CRTK	ST12000NM0007	12000138625024	0	73.0	20128440.0	NaN	NaN	97.0	0.0	100.0	3.0
115256	2019-10-01	AAGA7W2H	HGST HUH721212ALN604	12000138625024	0	100.0	0.0	132.0	96.0	100.0	0.0	100.0	1.0
115257	2019-10-01	PL1331LAHGD9NH	HGST HMS5C4040BLE640	4000787030016	0	100.0	0.0	134.0	100.0	100.0	459.0	100.0	5.0
115258	2019-10-01	ZJV5JLF1	ST12000NM0007	12000138625024	0	84.0	244009373.0	NaN	NaN	99.0	0.0	100.0	1.0

115259 rows × 131 columns

FIGURE 6.1: A dataset sample example

6.1 Dataset exploration

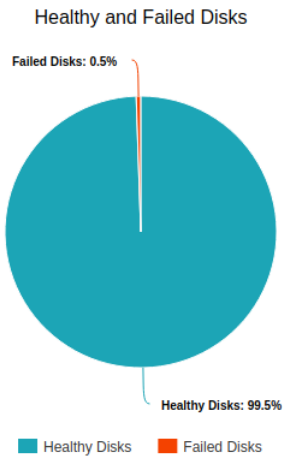


FIGURE 6.2: Diagram of Failed and Healthy Disks

The data set we used for our study includes data from 125,731 hard disk drives, with a date from October 1 to December 31, 2019. This data set was obtained from Backblaze [35]

and is presented in Comma Separated Value format (CSV). During the observed period, of the total of 127,731 hard disk drives, there were 678 instances of drive failure. This translates to a failure rate of 0.54% for the quarter ending December 31, 2019, a rate shown in Figure 6.2.

An added layer of complexity in the analysis of this data set arises from the fact that different hard disk drive manufacturers employ different S.M.A.R.T. attributes. Table 6.3 provides a detailed breakdown of the S.M.A.R.T. attributes used by each vendor.

In addressing HDD storage failure rates, it is important to understand the inventory of hard disk drives in production. These HDD and their in-use frequency can affect our prediction abilities. Furthermore, each drive vendor populates different S.M.A.R.T. attributes (Table 6.3). In Table 6.4, we show the number of hard disks with their corresponding frequency in production.

TABLE 6.3: **S.M.A.R.T.** Attributes by vendor.

S.M.A.R.T. Vendors	Toshiba	Hitachi	Seagate	WDC
smart_1	✓	✓	✓	✓
smart_2	✓	✓	✓	
smart_3	✓	✓	✓	✓
smart_4	✓	✓	✓	✓
smart_5	✓	✓	✓	✓
smart_7	✓	✓	✓	✓
smart_8	✓	✓	✓	
smart_9	✓	✓	✓	✓
smart_10	✓	✓	✓	✓
smart_11			✓	✓
smart_12	✓	✓	✓	✓
smart_18			✓	
smart_22		✓		
smart_23	✓			
smart_24	✓			
smart_183			✓	
smart_184			✓	
smart_187			✓	

TABLE 6.3: **S.M.A.R.T.** Attributes by vendor. (cont)

S.M.A.R.T. Vendors	Toshiba	Hitachi	Seagate	WDC
smart_188			✓	
smart_189			✓	
smart_190			✓	
smart_191	✓		✓	✓
smart_192	✓	✓	✓	✓
smart_193	✓	✓	✓	✓
smart_194	✓	✓	✓	✓
smart_195			✓	
smart_196	✓	✓	✓	✓
smart_197	✓	✓	✓	✓
smart_198	✓	✓	✓	✓
smart_199	✓	✓	✓	✓
smart_200			✓	✓
smart_220	✓			
smart_222	✓			
smart_223	✓	✓	✓	
smart_224	✓			
smart_225			✓	
smart_226	✓			
smart_240	✓		✓	✓
smart_241		✓	✓	
smart_242		✓	✓	
smart_254			✓	
Total	**26**	**21**	**34**	**19**

TABLE 6.4: All disk vendors and models with their respective frequencies in Backblaze's data center provided our 3 month window

Vendor	Disk model	Number of Disks
HGST	HMS5C4040BLE640	12758
HGST	HUH721212ALN604	10866

TABLE 6.4: All disk vendors and models with their respective frequencies in Backblaze's data center provided our 3 month window (Cont.)

Vendor	Disk model	Number of Disks
HGST	HMS5C4040ALE640	2833
HGST	HUH721212ALE600	1561
HGST	HUH728080ALE600	1002
HGST	HUS726040ALE610	28
HGST	HDS5C4040ALE630	26
HGST	HUH721010ALE600	20
HGST	HMS5C4040BLE641	1
HGST	HDS5C4040ALE630	2
Seagate	ST12000NM0007	37442
Seagate	ST4000DM000	19330
Seagate	ST8000NM0055	14502
Seagate	ST8000DM002	9844
Seagate	ST12000NM0008	7226
Seagate	ST10000NM0086	1205
Seagate	ST6000DX000	887
Seagate	ST500LM012 HN	501
Seagate	ST500LM030	259
Seagate	BarraCuda ZA250CM10002	157
Seagate	Generic SSD	107
Seagate	ST16000NM001G	40
Seagate	ST4000DM005	39
Seagate	ST500LM021	33
Seagate	ST8000DM005	25
Seagate	BarraCuda ZA500CM10002	18
Seagate	ST12000NM0117	15
Seagate	ST6000DM001	4
Seagate	BarraCuda ZA2000CM10002	4
Seagate	ST8000DM004	3
Seagate	ST1000LM024 HN	1

TABLE 6.4: All disk vendors and models with their respective frequencies in Backblaze's data center provided our 3 month window (Cont.)

Vendor	Disk model	Number of Disks
Seagate	ST6000DM004	1
TOSHIBA	MG07ACA14TA	3627
TOSHIBA	MQ01ABF050	475
TOSHIBA	MQ01ABF050M	425
TOSHIBA	MD04ABA400V	99
TOSHIBA	HDWF180	20
TOSHIBA	HDWE160	4
WDC	WD5000LPVX	214
WDC	WD5000LPCX	54
WDC	WD5000BPKT	10
WDC	WD60EFRX	3

The data were fragmented into 92 separate daily records. This fragmentation required us to aggregate individual daily data sets into a cohesive whole for a comprehensive analysis.

With such a small number of failed HDD, a huge disparity between the two classes of disk categories (failed and working) must be overcome. To illustrate the imbalance and provide a clearer understanding of the data distribution, we provide a visualization of this disparity in Figure 6.2.

Backblaze's unique position in the marketplace as a data storage provider contributes an additional layer of complexity to our analysis. The company operates numerous storage systems, each comprising a diverse array of HDDs from various manufacturers and HDDMs. Table 6.5 lists the HDDs along with their corresponding failure numbers. This table serves as a reference to understand the variety of storage devices included in our study and their distribution between different manufacturers.

TABLE 6.5: *HDDM* and their numbers available in the Backblaze Storage dataset with failed drive count.

Types of HDDM	# of Disks	# of failed Disks
HGST HDS5C4040ALE630	26	0
HGST HMS5C4040ALE640	2833	4
HGST HMS5C4040BLE640	12758	12
HGST HMS5C4040BLE641	1	0
HGST HUH721010ALE600	20	0
HGST HUH721212ALE600	1561	1
HGST HUH721212ALN604	10866	6
HGST HUH728080ALE600	1002	2
HGST HUS726040ALE610	28	0
Hitachi HDS5C4040ALE630	2	0
ST10000NM0086	1205	5
ST1000LM024 HN	1	0
ST12000NM0007	37442	364
ST12000NM0008	7226	9
ST12000NM0117	15	0
ST16000NM001G	40	0
ST4000DM000	19330	119
ST4000DM005	39	0
ST500LM012 HN	501	13
ST500LM021	33	0

Since not all vendors employ identical variables for their HDD, careful consideration is required to determine which variables to be retained for analysis within our data set. This selection process is essential to ensure consistency and reliability in our findings.

6.1.1 Cleaning the Data

To analyze this data set, we identified the day with the highest number of failures. We then collected data on the failed disks, tracing their observations from the first day to the selected day. This approach enabled the construction of a time series data set that described the HDD behavior over time.

To balance the dataset, we employed an under-sampling method. This method involved selecting healthy disks that remained operational up to the identified day and gathering their respective observations for the same period. It was crucial to ensure that these healthy disks were from the same HDDM as the failed disks. This consistency was necessary for accurate comparisons, as different HDDMs use distinct variables. As a result, we created two separate datasets: one for healthy and another for failed HDD. Figure 6.3 provides a visual representation of this process, with blue indicating healthy disks

and red indicating failed disks. Our data set had a HDD models with no failed drives over the three-month period; we excluded these HDD models.

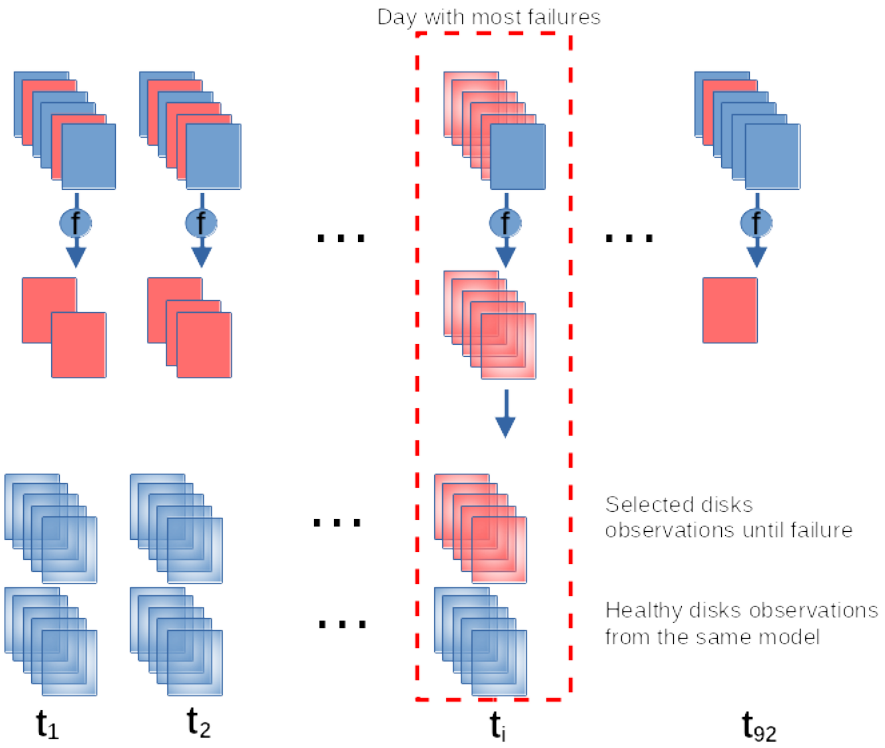


FIGURE 6.3: Pre-Processing Diagram

6.1.2 Methodology

We conducted a temporal analysis is executed of both datasets (Healthy Disks and Failed Disks) for a better visualization of the oscillations in the variable's values and to explore the differences between the healthy and failing disks. We calculated the Euclidean distances of the attribute values vs. time between the failed disks and the healthy ones for every feature. With this distance, a better numerical perception was achieved and helped in verifying our model's high-ranking contribution features. Both processes will help extract information from the data and turn the decision-making, before applying the learning algorithms, more efficient and accurate. The temporal analysis is performed by plotting the attributes of interest for failing disks and healthy disks.

The Euclidean distance is calculated for each variable. This distance helps to understand how dispersed the values are between a healthy disk and a disk that ends up failing. To better clarify this process, we compared the smart_1 vs. time of a healthy disk, against

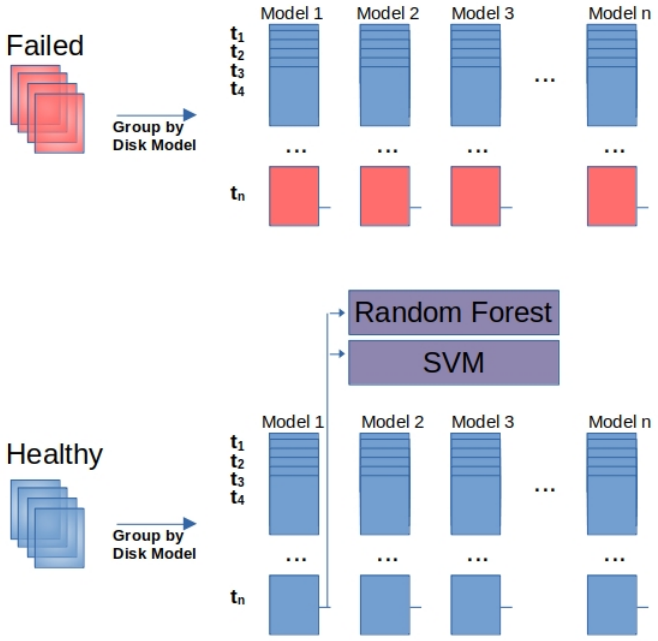


FIGURE 6.4: Data set assembly for our HDDM classification model

the smart_1 vs. time of a failed disk. We verified that the comparison between the disks is always done with the same HDD as seen in Table 6.6.

TABLE 6.6: Euclidean distances between the healthy and failed disks

	EUCLIDEAN DISTANCES BETWEEN THE HEALTHY AND FAILED DISKS					
	smart_1_normalized	**smart_1_raw**	**smart_7_normalized**	**smart_7_raw**	**smart_9_normalized**	**smart_9_raw**
ST12000NM0007	2.896801	3.141699	N/A	0.186010	3.000000	0.056366
ST12000NM0007	2.054655	2.665402	1.092906	0.532815	**3.041381**	0.064692
ST12000NM0007	1.964036	3.185758	2.403701	0.704135	3.000000	0.040648
ST12000NM0007	3.832814	2.756828	**4.358899**	1.431158	3.041381	0.066226
ST12000NM0007	1.991425	2.650050	4.845187	**5.122032**	2.645751	0.053915
ST12000NM0007	2.565644	**3.179320**	1.471768	0.574069	2.449490	0.046023
ST12000NM0007	2.885591	2.882519	**3.122499**	0.871523	3.000000	0.056832
ST12000NM0007	3.051245	2.747076	3.752777	0.313711	3.000000	0.050737
ST12000NM0007	1.454620	1.372290	2.373880	0.160891	N/A	0.107380
ST4000DM000	2.293516	3.111510	2.719062	**3.947537**	2.236068	0.050780
ST4000DM000	2.394414	2.934578	N/A	0.299613	**3.000000**	0.031744
ST8000NM0055	2.323827	3.176403	**4.716991**	0.275336	3.041381	0.056363
ST8000NM0055	**3.533003**	3.427996	3.464102	0.530212	3.041381	0.057715
ST12000NM0008	1.980921	**2.321437**	0.139754	0.056393	N/A	0.005388
TOSHIBA MG07ACA14TA	N/A	N/A	N/A	N/A	0.881917	0.039677
TOSHIBA MQ01ABF050	N/A	N/A	N/A	N/A	N/A	0.135676

To apply a VAR model, it was necessary to divide the two data sets, creating sub-data sets that we grouped by serial_number. Thus, the sub-data sets would only contain observations over time of a given disk. In this way, it became possible to identical models of each disk and make a comparison between the healthy and the failing drives, see Figure 6.5

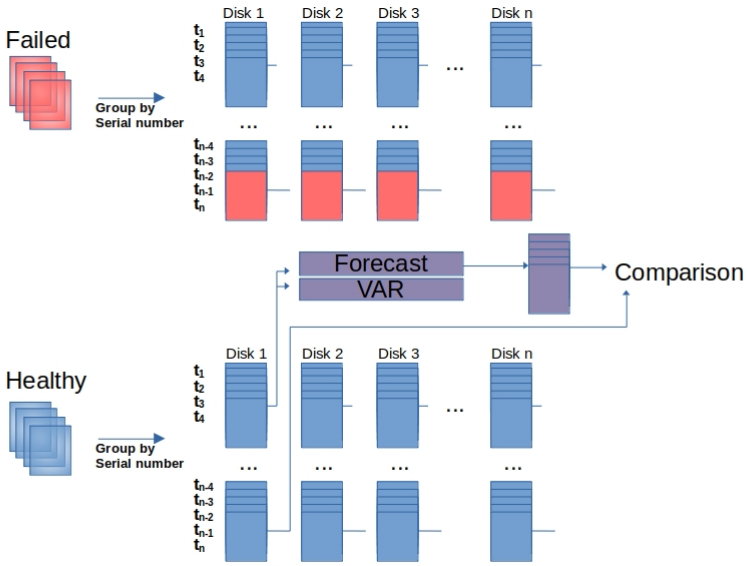


FIGURE 6.5: Data set assembly for our HDDM VAR model

The observations from the last five days of each disk were removed so that we could forecast and compare with the real values. The drive information that was added by Backblaze is removed, except the Date values, so that the resulting data set only contains S.M.A.R.T. variables, over time. This measure is taken because the VAR MLM performs operations only on numeric variables and would not extract any information from the textual variables that were added to describe the disk (e.g. Serial_number).

Before the execution of the algorithms, we divided the data sets (healthy and failed) into sub-data sets once more, but this time in sub-data sets grouped by HDDM. This is done because we no longer need to have a temporal view of this data, so more than one disk can be placed in the new data subset as seen in Figure 6.4. Then it is necessary to add the class variable to all observations . The disks that fail will have the class equal to 1 and those that remain healthy will have the class equal to 0. In these algorithms, only the S.M.A.R.T. attributes remain in the data frame, the rest of the variables are eliminated for the same reasons referenced in the VAR model.

All features were normalized to values between 0 and 1, since the learning algorithms had difficulties performing the operations on non-integer values. This normalization was made after the disks were divided by HDDM, so the values range were not mixed up. The validation method used a train-test-split ratio of 80 to 20.

6.2 Classification Algorithms

The classification MLM were applied to the sub-data sets created (12 data frames distinguished by HDDM). Both classification algorithms, SVM and Random Forest, were executed using default parameters.

Table 6.7 presents the metric results of HDD Seagate ST12000NM0007 model along with the respective Confusion Matrix. The tables present precision, recall, f1-score, and accuracy. It is also possible to observe the support of each class, corresponding to how many observations are labeled for each class. In the confusion matrices the predicted cases for each classification algorithm are presented, so it is possible to evaluate the respective performance. All the results presented are obtained from the test set.

TABLE 6.7: Metrics Results for HDDM ST12000NM0007

	SVM				RANDOM FOREST			
	precision	**recall**	**f1-score**	**support**	**precision**	**recall**	**f1-score**	**support**
Healthy	1.00	0.99	0.99	92	0.99	0.99	0.99	92
Failure	0.99	1.00	0.99	83	0.99	0.99	0.99	83
accuracy			0.99	175			0.99	175
macro avg	0.99	0.99	0.99	175	0.99	0.99	0.99	175
weighted avg	0.99	0.99	0.99	175	0.99	0.99	0.99	175

TABLE 6.8: Confusion Matrix *HDDM* ST12000NM0007

		SVM		RANDOM FOREST	
		Predicted Class		**Predicted Class**	
		Healthy	*Failure*	*Healthy*	*Failure*
Actual Class	*Healthy*	91	1	91	1
	Failure	0	83	1	82

TABLE 6.9: Metrics Results for HDDM ST4000DM000

	SVM				RANDOM FOREST			
	precision	**recall**	**f1-score**	**support**	**precision**	**recall**	**f1-score**	**support**
Healthy	0.83	1.00	0.91	20	0.95	1.00	0.98	20
Failure	1.00	0.80	0.89	20	1.00	0.95	0.97	20
accuracy			0.90	40			0.97	40
macro avg	0.92	0.90	0.90	40	0.98	0.97	0.97	40
weighted avg	0.92	0.90	0.90	40	0.98	0.97	0.97	40

TABLE 6.10: Confusion Matrix HDDM ST4000DM000

		SVM		RANDOM FOREST	
		Healthy	*Failure*	*Healthy*	*Failure*
Actual Class	*Healthy*	20	0	20	0
	Failure	4	16	1	19

TABLE 6.11: Metrics Results for HDDM ST8000NM0055

	SVM				RANDOM FOREST			
	precision	**recall**	**f1-score**	**support**	**precision**	**recall**	**f1-score**	**support**
Healthy	0.95	1.00	0.98	20	1.00	1.00	1.00	20
Failure	1.00	0.95	0.98	21	1.00	1.00	1.00	21
accuracy			0.98	41			1.00	41
macro avg	0.98	0.98	0.98	41	1.00	1.00	1.00	41
weighted avg	0.98	0.98	0.98	41	1.00	1.00	1.00	41

TABLE 6.12: Confusion Matrix HDDM ST8000NM0055

		SVM		RANDOM FOREST	
		Predicted Class		**Predicted Class**	
		Healthy	*Failure*	*Healthy*	*Failure*
Actual Class	*Healthy*	20	0	20	0
	Failure	1	20	0	21

TABLE 6.13: Metrics Results for HDDM ST12000NM0008

	SVM				RANDOM FOREST			
	precision	**recall**	**f1-score**	**support**	**precision**	**recall**	**f1-score**	**support**
Healthy	0.40	0.40	0.40	5	0.50	0.40	0.44	5
Failure	0.25	0.25	0.25	4	0.40	0.50	0.44	4
accuracy			0.33	9			0.44	9
macro avg	0.33	0.33	0.33	9	0.45	0.45	0.44	9
weighted avg	0.33	0.33	0.33	9	0.46	0.44	0.44	9

TABLE 6.14: Confusion Matrix HDDM ST12000NM0008

		SVM		RANDOM FOREST	
		Predicted Class		**Predicted Class**	
		Healthy	*Failure*	*Healthy*	*Failure*
Actual Class	*Healthy*	2	3	2	3
	Failure	3	1	2	2

In Table 6.7 we observe that the ST12000NM0007 HDDM metrics values are very close to 100%, demonstrating that our methodology is accurate. It is important to note that the ST12000NM0008, TOSHIBA MQ01ABF050 and TOSHIBA MG07ACA14TA HDDM, do not have a favorable support (i.e. exceptionally few observations and information from past behaviors) for the execution of algorithms, and therefore their results are not the most

TABLE 6.15: Metrics Results for HDDM TOSHIBA MQ01ABF050

	SVM				RANDOM FOREST			
	precision	**recall**	**f1-score**	**support**	**precision**	**recall**	**f1-score**	**support**
Healthy	0.80	0.80	0.80	10	0.91	1.00	0.95	10
Failure	0.82	0.82	0.82	11	1.00	0.91	0.95	11
accuracy			0.81	21			0.95	21
macro avg	0.81	0.81	0.81	21	0.95	0.95	0.95	21
weighted avg	0.81	0.81	0.81	21	0.96	0.95	0.95	21

TABLE 6.16: Confusion Matrix HDDM TOSHIBA MQ01ABF050

		SVM		RANDOM FOREST	
		Predicted Class		**Predicted Class**	
		Healthy	*Failure*	*Healthy*	*Failure*
Actual Class	*Healthy*	8	2	10	0
	Failure	2	9	1	10

promising. In the future work section, we discuss some points that could improve these results are discussed.

Note the importance that the Random Forest model gives to variables in its decision making and in the tree's creation. With this, it is easier to understand which features are more important in helping the algorithm to predict if the disk will fail or if it will remain healthy. Table 6.17 shows the importance ranking given to the six different HDDM's.

TABLE 6.17: Random Forest Features Importance for each *HDDM*

ST12000NM0007	
S.M.A.R.T. Variable	importance
7 normalized	0.228980
193 raw	0.187561
3 normalized	0.165387
9 normalized	0.058750
9 raw	0.057293
241 raw	0.051673
7 raw	0.041517
240 raw	0.032224
12 raw	0.026624
242 raw	0.026087

TABLE 6.18:
ST12000NM0007 RF Importance

ST4000DM000	
S.M.A.R.T. Variable	importance
193 raw	0.156771
183 raw	0.127605
3 normalized	0.099825
190 normalized	0.083486
183 normalized	0.080752
194 raw	0.063729
194 normalized	0.061899
190 raw	0.056995
7 normalized	0.046583
240 raw	0.045435

TABLE 6.19:
ST4000DM000 RF Importance

ST8000NM0055	
S.M.A.R.T. Variable	importance
195 normalized	0.207715
1 normalized	0.182538
193 normalized	0.126374
191 raw	0.066237
7 normalized	0.060627
191 normalized	0.053196
192 raw	0.043330
190 raw	0.022044
194 normalized	0.021385

TABLE 6.20:
ST8000NM0055 RF Importance

ST12000NM0008	
S.M.A.R.T. Variable	importance
190 raw	0.165062
194 normalized	0.146380
190 normalized	0.139104
194 raw	0.130639
192 raw	0.061040
1 normalized	0.050796
240 raw	0.038349
7 raw	0.037081
9 raw	0.036924

TABLE 6.21:
ST12000NM0008 RF Importance

TOSHIBA MQ01ABF050	
S.M.A.R.T. Variable	importance
191 raw	0.400617
194 raw	0.286671
9 raw	0.134172
222 raw	0.125146
222 normalized	0.028102
9 normalized	0.025292

TABLE 6.22:
TOSHIBA MQ01ABF050
RF Importance

TOSHIBA MG07ACA14TA	
S.M.A.R.T. Variable	importance
226 raw	0.278620
222 raw	0.193245
9 raw	0.181570
220 raw	0.145886
194 raw	0.104060
193 raw	0.096619

TABLE 6.23:
TOSHIBA MG07ACA14TA
RF Importance

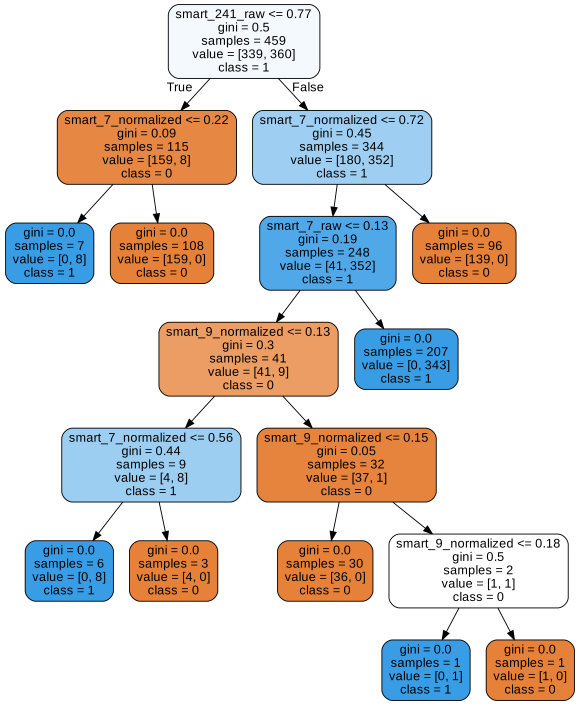


FIGURE 6.6: A Decision Tree from the Random Forest of the ST12000NM0007 HDDM

Figure 6.6 shows one of the trees created after the Random Forest algorithm was executed on the ST12000NM0007 data set. As normal, the variables presented in the decision tree are shown in the importance table, showing that the algorithm uses them to classify the observations. In Table 6.24 the variables presented in the decision tree are described. Note that this description was made after the variables were already normalized between 0 and 1. We use six digits of accuracy, which is the default output format of our model. Our importance values are based on the model output and are not normalized as we did to the values when we built the model.

TABLE 6.24: Decision Tree S.M.A.R.T. variable description

	7_normalized	7_raw	9_normalized	9_raw	241_raw
count	699	699	699	699	699
mean	0.705797	0.476673	0.270684	0.739938	0.769213
std	0.184539	0.306533	0.301359	0.301370	0.266498
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.600000	0.178866	0.100000	0.737158	0.804327
50%	0.685714	0.476555	0.150000	0.847509	0.853286
75%	0.900000	0.772911	0.263158	0.939120	0.925315
max	1.000000	1.000000	1.000000	1.000000	1.000000

6.3 VAR Model

We created 32 sub-data sets to apply the time series MLM, grouping the sub-data sets by serial_number. The algorithm could be applied individually and to calculate the respective correlation matrices and predicting the variables behavior over time.

We show correlation matrices for four different disk's HDDM. These matrices are presented in Tables 6.25 - 6.32.

These correlations are important. The features are monitored together, not just those that present critical values because their thresholds were exceeded. S.M.A.R.T. attributes 9 and 240 are present in all disks (healthy and failed), with correlations close to 1. This happens because both are hour counters, with 9 being the number of hours in power-state and 240 the time during the positioning of the drive heads.

It is possible to verify that higher correlations between variables (9-193) and (240-193) also happen more frequently in the disks that fail from ST12000NM0007 HDDM. The high correlation between these variables may alert us that a more careful observation of the disks should be made. However, there are also healthy disks that show high correlations between these variables, but there is no guarantee that the disk will remain healthy in the future, so these disks may even already show some type of anomaly.

Even though there are not enough disks to draw a strong conclusion, the ST4000DM000 HDDM shows correlations between the variables (9−7) and (240−7) for the disks that fail, and the ST12000NM0008 HDDM with correlations in the variables (1−193), (1−194) and (193−194) also in the failing disks.

TABLE 6.25: Correlation Matrix for failed disk from HDDM ST12000NM0007

Failed Disk ST12000NM0007 (ZCH0C5JJ)	**smart_1_raw**	**smart_7_raw**	**smart_9_raw**	**smart_193_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	0.253239	0.201611	-0.120629	-0.190959	0.187426
smart_7_raw	0.253239	1.000000	**0.852643**	**0.696433**	0.055252	**0.849855**
smart_9_raw	0.201611	**0.852643**	1.000000	**0.923364**	0.039857	**0.998544**
smart_193_raw	-0.120629	**0.696433**	**0.923364**	1.000000	-0.040797	**0.920185**
smart_194_raw	-0.190959	0.055252	0.039857	-0.040797	1.000000	0.068931
smart_240_raw	0.187426	**0.849855**	**0.998544**	**0.920185**	0.068931	1.000000

TABLE 6.26: Correlation Matrix for healthy disk from HDDM ST12000NM0007

Healthy Disk ST12000NM0007 (ZCH06YQ3)	**smart_1_raw**	**smart_7_raw**	**smart_9_raw**	**smart_193_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	0.166218	-0.085673	-0.141543	0.260900	-0.001269
smart_7_raw	0.166218	1.000000	0.351067	-0.135600	-0.120242	0.392243
smart_9_raw	-0.085673	0.351067	1.000000	0.254303	-0.457013	**0.992656**
smart_193_raw	-0.141543	-0.135600	0.254303	1.000000	-0.546709	0.235824
smart_194_raw	0.260900	-0.120242	-0.457013	-0.546709	1.000000	-0.406834
smart_240_raw	-0.001269	0.392243	**0.992656**	0.235824	-0.406834	1.000000

TABLE 6.27: Correlation Matrix for failed disk from HDDM ST4000DM000

Failed Disk ST4000DM000 (Z302T88S)	**smart_1_raw**	**smart_7_raw**	**smart_9_raw**	**smart_193_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	0.557678	0.641922	0.054301	0.027109	0.612161
smart_7_raw	0.557678	1.000000	**0.933525**	-0.456721	0.200886	**0.942728**
smart_9_raw	0.641922	**0.933525**	1.000000	-0.262591	0.029598	**0.998036**
smart_193_raw	0.054301	-0.456721	-0.262591	1.000000	-0.332683	-0.271897
smart_194_raw	0.027109	0.200886	0.029598	-0.332683	1.000000	0.036335
smart_240_raw	0.612161	**0.942728**	**0.998036**	-0.271897	0.036335	1.000000

TABLE 6.28: Correlation Matrix for healthy disk from HDDM ST4000DM000

Healthy Disk ST4000DM000 (Z302DJZ6)	**smart_1_raw**	**smart_7_raw**	**smart_9_raw**	**smart_193_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	-0.105579	-0.208478	-0.153385	**0.724605**	-0.208246
smart_7_raw	-0.105579	1.000000	0.458922	-0.102037	-0.192726	0.471757
smart_9_raw	-0.208478	0.458922	1.000000	0.569640	-0.315389	**0.998916**
smart_193_raw	-0.153385	-0.102037	0.569640	1.000000	-0.432104	0.559510
smart_194_raw	**0.724605**	-0.192726	-0.315389	-0.432104	1.000000	-0.336655
smart_240_raw	-0.208246	0.471757	**0.998916**	0.559510	-0.336655	1.000000

TABLE 6.29: Correlation Matrix for failed disk from HDDM ST8000NM0055

Failed Disk ST8000NM0055 (ZA1819DM)	*smart_1_raw*	**smart_7_raw**	**smart_9_raw**	**smart_193_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	0.257158	-0.061240	-0.286411	-0.268057	-0.022319
smart_7_raw	0.257158	1.000000	**0.830599**	-0.177065	-0.081025	**0.844100**
smart_9_raw	-0.061240	**0.830599**	1.000000	0.161528	0.070777	**0.995699**
smart_193_raw	-0.286411	-0.177065	0.161528	1.000000	0.370282	0.189769
smart_194_raw	-0.268057	-0.081025	0.070777	0.370282	1.000000	0.100202
smart_240_raw	-0.022319	**0.844100**	**0.995699**	0.189769	0.100202	1.000000

TABLE 6.30: Correlation Matrix for healthy disk from HDDM ST8000NM0055

Healthy Disk ST8000NM0055 (ZA18BTFV)	*smart_1_raw*	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	-0.455708	-0.264859	0.321201	0.271168	-0.249651
smart_7_raw	-0.455708	1.000000	**0.918986**	0.235787	-0.325022	**0.921027**
smart_9_raw	-0.264859	**0.918986**	1.000000	0.528043	-0.273634	**0.999511**
smart_193_raw	0.321201	0.235787	0.528043	1.000000	-0.302633	0.529167
smart_194_raw	0.271168	-0.325022	-0.273634	-0.302633	1.000000	-0.267527
smart_240_raw	-0.249651	**0.921027**	**0.999511**	0.529167	-0.267527	1.000000

TABLE 6.31: Correlation Matrix for failed HDDM ST12000NM0008

Failed Disk ST12000NM0008 (ZHZ3MSH6)	*smart_1_raw*	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	0.061050	0.395718	**0.741040**	**0.703804**	0.072431
smart_7_raw	0.061050	1.000000	**0.922226**	-0.374584	-0.446844	**0.961005**
smart_9_raw	0.395718	**0.922226**	1.000000	-0.097896	-0.156114	**0.927476**
smart_193_raw	**0.741040**	-0.374584	-0.097896	1.000000	**0.863018**	-0.457007
smart_194_raw	**0.703804**	-0.446844	-0.156114	**0.863018**	1.000000	-0.442773
smart_240_raw	0.072431	**0.961005**	**0.927476**	-0.457007	-0.442773	1.000000

TABLE 6.32: Correlation Matrix for healthy disk from HDDM ST12000NM0008

Healthy Disk ST12000NM0008 (ZHZ3PT1S)	*smart_1_raw*	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	-0.106104	-0.175308	-0.074616	0.041358	-0.206763
smart_7_raw	-0.106104	1.000000	**0.976768**	-0.217435	0.040426	**0.982719**
smart_9_raw	-0.175308	**0.976768**	1.000000	-0.054606	0.107858	**0.960382**
smart_193_raw	-0.074616	-0.217435	-0.054606	1.000000	0.578504	-0.306347
smart_194_raw	0.041358	0.040426	0.107858	0.578504	1.000000	-0.096563
smart_240_raw	-0.206763	**0.982719**	**0.960382**	-0.306347	-0.096563	1.000000

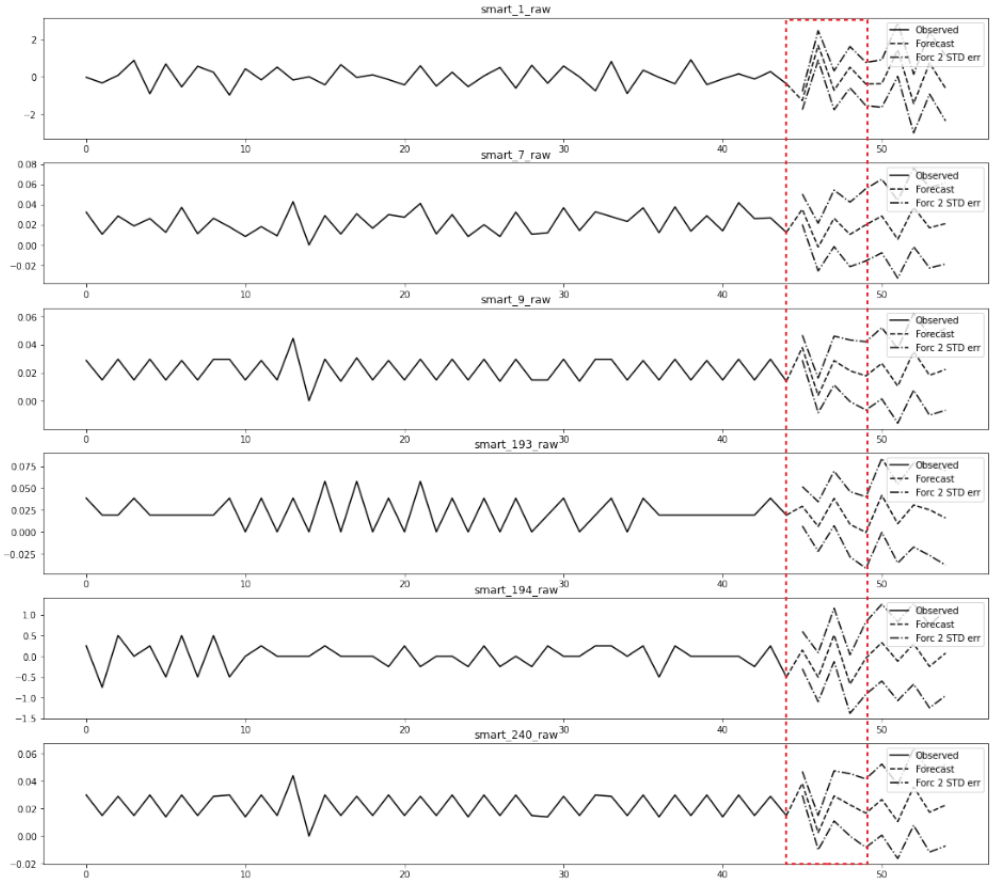


FIGURE 6.7: Forecast for the first healthy disk

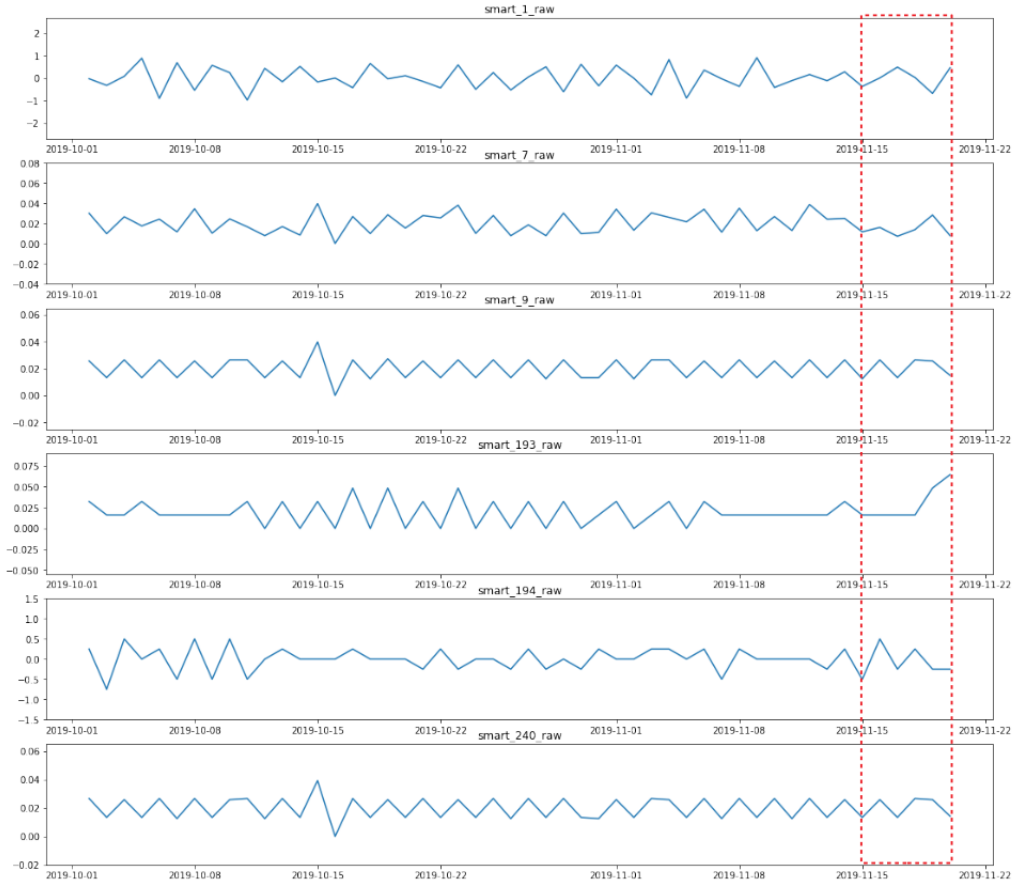


FIGURE 6.8: Real Values for the first healthy disk

Our forecasted values (in the red dashed lines) in the Figure 6.7 include a two times standard deviation from the predicted values at 95% confidence level. The top line is the upper-bound standard deviation and the bottom line s the lower bound standard deviation that we expect the predicted value to lie within. We can see the actual values in Figure 6.8 and notice some of the S.M.A.R.T. predicted values are more closely predicted, like S.M.A.R.T. 7, 9 and 240; however, other values are less obviously predictive like S.M.A.R.T. 1 and 192 raw. These predictive values follows with our decision tree attributes importance ranking. This also implies that these attributes are able to be predicted and are predictive for failures.

Finally, the forecasting was done to predict the disks behavior over the time. In Figure 6.7, inside the red dashed line, a five−day forecast can be observed. We forecasted each variable from a disk that fails and a disk considered healthy. In Figure 6.8 inside this dashed line it is possible to see how the disks performed in the last five days before being selected. The X axis represents a time scale, with daily periodicity and the Y axis represents the values for each variable. It is important to notice that these five days were removed from the datasets at the beginning of the learning HDDM, so now that they could be compared with the respective forecasted values.

6.4 Discussion

Working with imbalanced data reduces the effectiveness of prediction MLM. Because of this, it was necessary to take an overly cautious approach to the data, so little information about the disks that could be useful, was not lost during the process.

One of the biggest beliefs in this project was that the pre-processing and statistical analysis methods used to create the sub datasets were fundamental in the learning process of the data. All prior studies made on this subject analyzed the disks all together without splitting them by HDDM. This means that the variables standard values, the thresholds and even the features normalization process are not distinguished between them.

Although, most sub datasets created during the project, did not have the ideal number of observations, the metrics values for the classification MLM are quite promising, showing values close to 100%, for the precision, recall and f-score of the evaluated HDDM.

The VAR MLM application, allowed to trace temporal correlation between the features, showing that variables 9, 240 and 193 are related over time. The forecasting showed

a low forecasting error and may be an interesting method to predict the variables behavior for storage companies.

We note that variables 7, 9 and 240 are often present in the results. Therefore, we would expect in the future to monitored the variables more carefully and combine them with ones that already labeled critical in the literature.

Over the course of our work, some obstacles appeared and had to be overcome. The lack of perception about the variables, the difference between vendors references values and the number of missing values presented in the dataset, made the decision making difficult. Often our methods had to be redone from scratch. Since the VAR is an algorithm that works with mathematical matrices, it proved to be a MLM with high complexity that requires a lot of attention in the type of data that we used.

6.5 Conclusions

The objective of this chapter was to expand on the current state-of-the-art. Our work, set out to measure if we could predict HDD's failures and anomalies using S.M.A.R.T. data. Over the time-frame we sampled, we believe we met our objective. We do note that with a three month data set study period may be too short. We believe a larger dataset, over a longer period of time, would help validate our results; we expect to do this in future work.

We recognized from the outset that imbalanced data could hinder the effectiveness of machine learning models (MLMs) for prediction. To address this challenge, we segmented our data based on HDDM. This approach ensured that even small but potentially valuable information about the disks wouldn't be overshadowed by more frequent data points during the machine learning process.

Our work used pre-processing and statistical analysis to create data subsets. It was this pre-processing and segmentation that was fundamental for applying MLMs to learn from our processed data. Our literature review showed almost all studies on our subject used all HDDM together to build a MLM.

A core contribution of our work was the segmentation based on HDDM, which deviated from prior work that analyzed all the disks together and sometimes broken apart by manufacturer. Limited prior art has attempted to splitting the HDD by model and manufacturer and for this work an only attempted to use the split to showcase HDDM failure rates. Other prior art did break out the variables, standard values, the thresholds

but never normalized the data so their work appeared unable to distinguished significant features. No prior art applied a MLM at the individual manufacturer and model level as our work had done.

When we created the data subsets for our work, unfortunately we had limited numbers of observations for failed HDD. We would have liked to have more failed HDD for the metrics values for the classification models. Our classification models showed promise in producing values really close to 100%, for the precision, recall and f-score of certain HDD models. This was exciting and shows that our method of breaking out by HDDM is promising.

When we applied VAR models it allowed us to trace temporal correlation matrices between the features. This application of the VAR models showed that variables 9, 240 and 193 are related over time.

S.M.A.R.T. Variables, like smart_7, smart_9, and smart_193 as described in Table 5.1, have a high importance in almost all disk MLM. As mentioned, some the S.M.A.R.T. variables are considered critical in the literature and are the ones that the storage systems follow the most. We note that variables 7, 9 and 240 are present often in our results. Therefore, these variables should be monitored together with the ones that are considered already considered critical in the literature. As previously noted, S.M.A.R.T. attributes 7,9 and 240 all have a temporal components. These temporal components also correlate with the ”bathtub” curve of HDD failures by age as seen in Figure 5.3.

Throughout our work, some obstacles appeared and had to be overcome. Working with such a large and imbalanced dataset was a challenge.

We have included additional tables and graphics of our other evaluated HDDM in Appendix D.

Chapter 7

Computational Cluster Task Failures

In this chapter, we discuss user submissions to computational clusters. Specifically, we examine user submitted computational tasks and profile them for a domain specific use case. Our goal is to understand the characteristics of the tasks submitted to the computational cluster submitted task characteristics and apply machine learning to predict if a task will failure. The data for our use case was provided by the University of Wisconsin - Madison, BCG.

Clusters of machines use a middle-ware to provide an interface (command line, API, or portal) for submitting single or batch tasks.

To submit a task it is recommended that a submitter has knowledge of a given tasks' needs.

The tasks that we used in our analyses originate from professors, students, staff at the University of Wisconsin-Madison in the department of BMI. These tasks where submitted between October 5, 2016 and September 24, 2018. All submitted tasks are used by biostatisticians or medical informaticians. The vast majority of the tasks required the cluster to have HIPPA [186] regulatory compliance or specific dataset security restrictions. These restrictions inhibit the tasks from utilizing general cloud compute infrastructures or other academic focused computational infrastructures such as the OSG [42]. Therefore, the tasks run within the cluster have unique and domain specific attributes to machine learning and medical data analysis.

We used the HTCondor class-Ad history to provide logs of the historical information about the tasks and their sub-tasks. For our work, we analyze the history 17,282 cluster submissions, producing 129,854 tasks. This data was from the BCG computational cluster running HTCondor.

We are motivated in this work to better understand cluster submitted task failure patterns so we can help our users have less failed tasks and secondarily improve cluster success rates. We attempt to detect patterns that could be used to proactively identify areas that cause a task to fail. We expect patterns to emerge that will provide clues into cluster adjustments for reducing failures and optimizing resource usage.

7.1 Dataset exploration

To understand why our tasks succeeded or failed, we need classifiers. We use HTCondor class-ad attribute *JobStatus* as the classifier of successful and failed tasks. We found a commonality among the tasks that failed, indicated by a HTCondor class ad *JobStatus=3*, while tasks that succeeded had a *JobStatus=4*. Beyond tasks we also identified two primary submissions categories, those consisting of a single task and those comprising multiple tasks. Combining task success and failures with submission categories results in the following.

We categorized all tasks into five groups:

1. tasks submissions of single tasks that finished successfully
2. tasks submissions of single tasks that failed
3. tasks with multiple task submission for which all tasks finished successfully
4. tasks with multiple task submission for which all tasks failed
5. tasks with multiple task submission for which some of the tasks succeeded and some failed

While single task task submissions are common, these submissions only run one task, which makes finding characteristics patterns difficult to find for this single task. Therefore, we concentrated on the multiple submission tasks, these tasks share common executable, executable parameters and the same resource requirements.

Users are allowed to submit tasks to the cluster from all cluster nodes. We combine the task submissions from each cluster node into one dataset which represents all cluster submissions from all nodes in the cluster. The task submission data was generated through the use of *condor_history* with the *–json* option which produced the Condor history data in JSON format. Using the HTCondor history data partitioned by individual class-Ad and generating a separate file for each Condor submission node, we combine

the data and import this data into a NoSQL database. MongoDB [187] was chosen as the data store for exploring and analyzing the data due to its simple import procedures and well established user base. In Table 7.1, we show a breakdown of the tasks by completion state. Table 7.2 displays a breakdown of exit status events from the clusters failed tasks.

Number	Percent	Description
106695	82.2%	Complete successful, no errors
23159	17.8%	Task Error Issues
129854	100%	All submitted task

TABLE 7.1: Full cluster submitted Task Breakdown

Number	Percent	Event
13,070	10.1%	User removed before scheduled to cluster
9,183	7.1%	User defined task attribute expression error
434	0.33%	Failed to initialize user log
229	0.17%	Out-of-memory event
160	0.11%	Other
83	0.06%	No such file or directory
23,159	17.8%	Total Task Error Issues

TABLE 7.2: Submitted tasks with errors breakdown

The category, "tasks with multiple task submission for which some of the tasks succeeded and some failed", is especially interesting to us. We personally experienced these failed tasks first hand in our ETL for atSNP search. It also was the cluster submission category with the most number of tasks. With only 491 submissions, this cluster submission category had a majority of the cluster tasks at 69.4%. We can compare this category group (as highlighted in green) to other category groups in Table 7.3.

Table 7.4 breakdowns this category further and we see in this group: 14.3% failed and 85.7% succeeded tasks.

To analyze the data, we clustered the using the HTCondor class-Ad *ClusterId* and combine it with the submission host part of the *GlobalJobId* field and use this as the unique submission cluster key. Other class-Add attributes can be seen in Table 7.5.

Our cluster had 21,152 unique cluster submissions and 20,518 submissions where only one task was submitted by the submission. A breakdown of cluster submission by task results is found in Table 7.6 and we plot our data based on tasks-per-submission in Figure 7.1. Of the remaining 1,860 cluster submissions, 491 of these have greater than five

Usage statistic	Cumulative	Average	Max	Min	Description
CPU	978,394.3	7.53	11,975.6	0	Cumulative CPU time per task
System CPU	43,848.3	.38	101.9	0	Cumulative CPU time in system/kernel time
User CPU	671,066.3	5.8	5,466.4	0	Cumulative CPU time in User space time
Suspension	40.8	.0003	3.6	0	Cumulative time a task is suspended/held

∗Note: System and User CPU time are a subset of the total due to a change in the way HTCondor collected dataset over the sample period.

TABLE 7.3: Cluster task usage in hours

Number	Percent	Description
77,183	85.7%	Complete successful, no errors
12,879	14.3%	Task error, issues
90,062	100%	All multi-task submissions

TABLE 7.4: Multi-task submitted with failed task breakdown

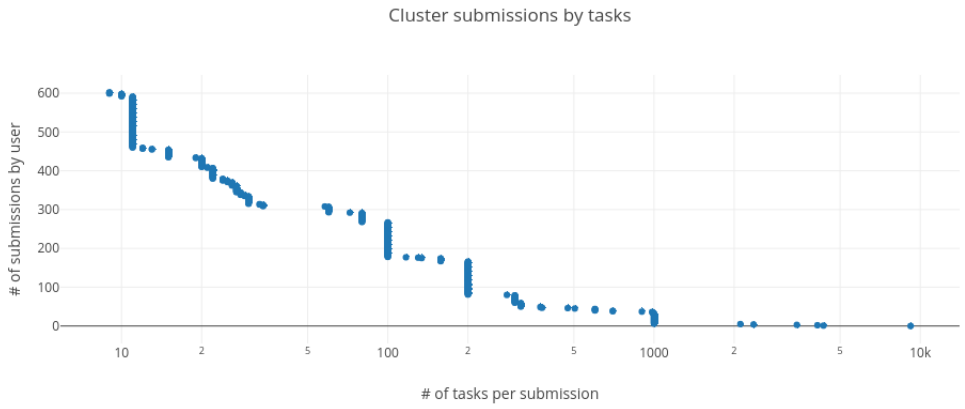


FIGURE 7.1: Number for tasks per cluster submission

tasks submitted per cluster submission and have both failed and successful completed task. Only 491 cluster submissions produced 69.4% of our cluster scheduled tasks. This group had at least one failed task when other tasks in the same submission succeeded. The task category group is the largest by cluster task numbers. Having previously experienced a submission with succeeded and failed tasks in our atSNP search ETL work. We want to know why do tasks from the same submission both succeed and fail?

We observe, in Table 7.6 that our green highlighted cluster submission category group averaged 256.4 tasks per submission. We focus this analysis on the submissions of ten

Attribute	Type	Description
Args	string	Arguments passed to the job
ClusterId	Int	A group of tasks submitted together, each with unique task identifier but shared cluster identifier
Cmd	string	The path a file name of the task being run
CommittedTime	Time(sec)	Wall time allocated on a machine to the job
CommittedSlotTime	Time(sec)	CommittedTime * SlotWeight
CumulativeSlotTime	Time(sec)	RemoteWallClockTime * slotweight
DiskUsage	Int	Amount of space used in the exec directory by the job
HoldReason	String	message to why the task is in a held state
HoldReasonCode	Int	int representing the task hold reason
ImageSize	Int	Max observed virtual image size of the job
JobStatus	Int	The current status of the job: 1=idle, 2=Running, 3=Removed, 4=Completed, 5=Held, 6=Transferring Output
Owner	String	user submitting the job
ProcId	Int	Id of a task within a submission
RemoteWallClockTime	Time(sec)	Cumulative number of seconds of user CPU time the task used on remote machine

TABLE 7.5: HTCondor Class Ad attributes examples

or more tasks with a combination of successful and failed tasks for multi-tasked submissions. Submissions with less then ten can be run easily as individual tasks on individual machines; however, at five tasks we experienced it is easier to run these tasks with a computational cluster scheduler if available. Additionally, the number of task submissions in the *one* $>X<$ *ten* group is only 0.2% of the total cluster task submissions.

The task resource usage by submission grouping is described in Table 7.3.

# submission	% Submission	# Task	% Total Task	Submission Type	Description
14,337	67.7%	14,337	11.0%	Single	Success
6,181	29.2%	6,181	4.8%	Single	Failed
35	0.2%	148	0.1%	1>#tasks <10	Both Success & Failed
351	1.7%	57,843	44.6%	>10 Multi-task	All Successful
75	0.4%	6,996	5.4%	>10 Multi-task	All Failed
173	0.8%	44,349	34.2%	>10 Multi-task	Both Success & Failed
21,152	**100%**	**129,854**	**100%**	**Total**	**All Task Status**

TABLE 7.6: Cluster submissions breakdown

7.2 Methodology

The data contains some evaluated equations where numeric values would be expected. This data required us to evaluate the encountered text data and equate it from other class-ad attributes. Not all tasks have all the required data and we address missing data through interpolating from other existing data.

HTCondor evaluates text to determine matching attributes. Since the data does not evaluate to numeric but instead maintains the original text, a step needed to be added to address these concerns. Class-Ad attributes affected by text matching criteria included: *CmdHash*, *TransferInput*, *RequestDisk*, *ImageSize*, *RequestMemory*. To address data issues with text and numeric values in the same field, such as *RequestDisk RequestMemory*, we evaluated the text to the actual numeric values based on the evaluated text of the class-ad attribute value, specifically we evaluated the **/EVAL/** value criteria.

Address any multi-modal, multi-class issues within our dataset we convert this data by integrating numerical, binomial, and categorical data through a bucket normalization process [188]. Thus, we organize commonality of data points through attribute value proximity and evaluate the remaining data by the corresponding attribute. Bucketing class-Ad attributes allows for temporal and high cardinally values to be grouped by nearest values, similarly to our work with HDD 6.1.2. Each class-Ad attribute bucket we implemented varied based on the type of data, for each type of data we used a different bucketing techniques with the following categories: numeric, epoctime, taskruntime, and dates. The implemented bucketing algorithm is as follows: numeric in buckets starting at zero and incremented 2,622,144 per bucket; epoctimes buckets started at the minimum

epoctime and had a maximum epoctime with 200 buckets equally spaced; taskruntimes started at the minimum time and incremented by 3,600 sec (1 min), and dates was done by individual day.

Some class-ad attributes provided no value since the data was empty. For all empty class-ad attributes, we dropped the corresponding column from our dataset. Remaining class-ad attributed which we could not convert to Numeric, Boolean's, Strings or bucket classifiers where removed from the data dataset. Class-ad attributes who's value was empty were also removed from the dataset.

After the cleaning process, we partitioned our data based on each task's class-ad *JobStatus* field with value three (3) being a failed task and four (4) being a successful task. Next, we apply a random forest learning model [189] to produce a Principal Component Analysis (PCA) of these tasks. The python library, scikit-learn [190], is used to product a list of feature importance in determining failures and successes.

We randomly selected our tree depth and the number of estimators. Our final results ran two thousand five hundred random forest iterations with nine cross validations with a max tree depth of eight and sixty one estimators.

For our distribution function we utilized a multi-nomial distribution when building our model. Lastly, we ignored columns when building our model, namely the columns: *RemoveReason*, *CompletionDate*, *TerminationPending*, *NumJobCompletions*, *LastVacateTime*. Additionally, we removed any class-ad attributes which contained all exact or null values.

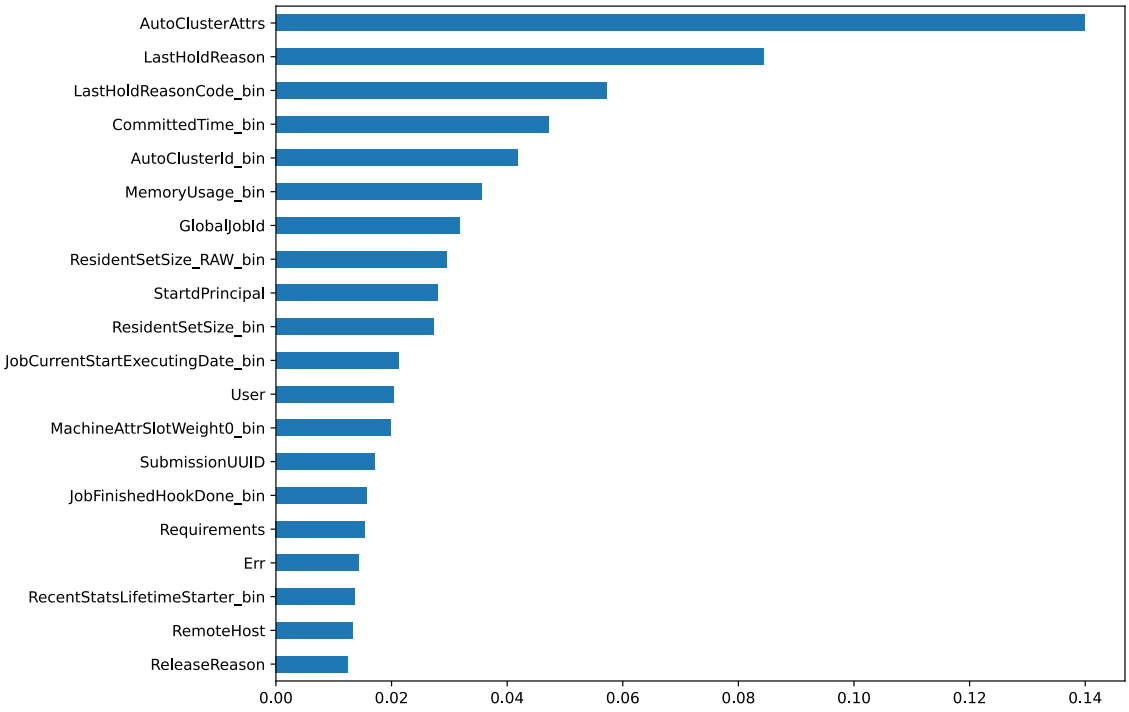


FIGURE 7.2: Task Failure Class-Ad Variable Importance

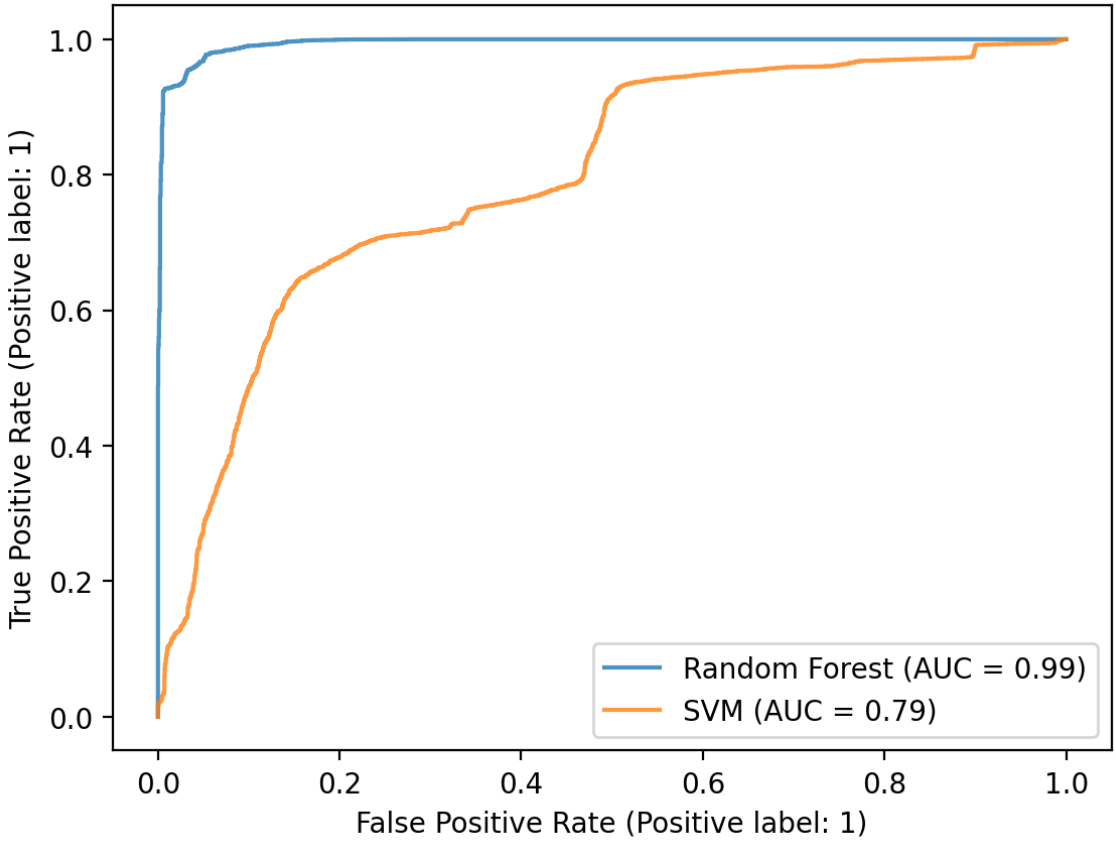


FIGURE 7.3: AUC curve

7.3 Discussion

Task failures in computational clusters lead to resource wastage and often user dissatisfaction. Our work examined a medium sized heterogeneous cluster at the University of Wisconsin - Madison, BMI BCG (Section 2.3.4). We witnessed a significant portion of CPU time was allocated to tasks that ultimately failed. The cost implications of such failures become more pronounced, as each task failure represents a loss in cluster time. Our MLM has demonstrated efficacy in identifying variables that correlate with task success or failure with a degree of confidence.

Our model Random Forest model produced an Accuracy of 94.4%, Precision of 95% and Recall of 96.8%. Our resulting Random Forest can be seen in Figure 7.4. We list our class-ad feature importance in Table 7.7 and in graph format in Figure 7.2

For our SVM model our Accuracy was only 69.8% , Precision was 77.9% and Recall was 94.1%.

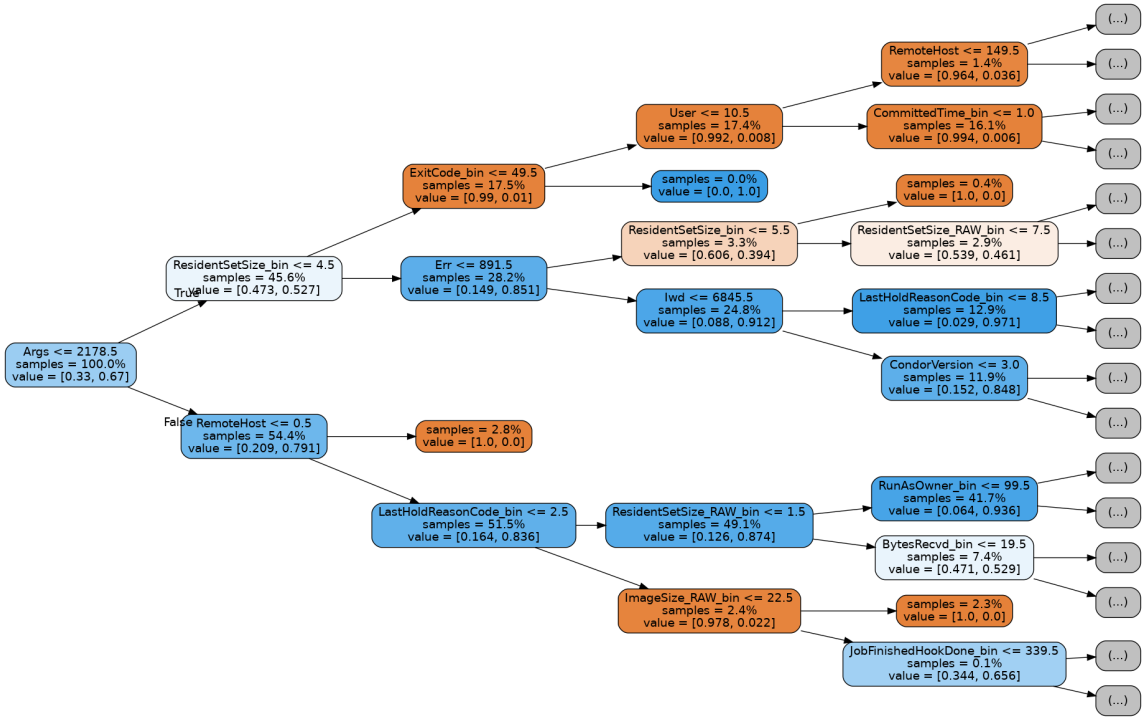


FIGURE 7.4: Random Forest class-ad Submission success, failure prediction

The class-ad variable *AutoClusterAttrs* importance in determining a tasks success or failure implies that the cluster schedule auto matches if a task does not specific an attribute, the cluster will attempt to automatically assign a class-ad attribute value to this. Interesting in that the task auto cluster attribute is the single largest contributor to cluster

Class-ad	Feature Importance
AutoClusterAttrs	0.139877
LastHoldReason	0.084341
LastHoldReasonCode_bin	0.057216
CommittedTime_bin	0.047076
AutoClusterId_bin	0.041881
MemoryUsage_bin	0.035518
GlobalJobId	0.031709
ResidentSetSize_RAW_bin	0.029498
StartdPrincipal	0.027925
ResidentSetSize_bin	0.027252
JobCurrentStartExecutingDate_bin	0.021230
User	0.020295
MachineAttrSlotWeight0_bin	0.019838
SubmissionUUID	0.017029
JobFinishedHookDone_bin	0.015647
Requirements	0.015402
Err	0.014286
RecentStatsLifetimeStarter_bin	0.013677
RemoteHost	0.013308
ReleaseReason	0.012465

TABLE 7.7: List of class-ad feature importance in predicting task failures

		SVM		RANDOM FOREST	
		Predicted Class		Predicted Class	
		Success	*Failed*	*Success*	*Failed*
Actual Class	*Success*	1539	1834	3019	354
	Failed	410	6486	221	6675

TABLE 7.8: Confusion Matrix task submission failure

task failures implies a potential cluster configuration issue. The second most important class-ad attribute is the *LastHoldReason*, this makes sense if a task already failed once, the chance of it failing again is higher.

The class-ad attribute *StartdPrincipal* showed to be important in distinguishing between failed and succeeded task. *StartdPrincipal* value is directly associated with the host system the submission was made from and runs HTCondor scheduler daemon. An explanation to the importance of this variable revolves around access to systems within the cluster. Specifically, students are limited to a subset of cluster computers from which to submit task. The results also show an important variable *CommittedSlotTime* which describes the amount of time a process spends running on a system.

As discussed *CommittedSlotTime* HTCondor attribute is also an important contributor

to our MLM. The *CommittedSlotTime* attribute significance would imply a user submission to the cluster may not be scheduled to run or ran for a very limited time on the cluster. A possible explanations for the significance of the *CommittedSlotTime* attribute include user task requirements misalignment (which may be resulting in zero scheduled time *CommittedSlotTime*) or rapid task failure due to the task not being ready to run on the cluster.

An interesting attribute is *RemoteHost*. The fact that is shows up suggests that the task running host is important to the success of failure of the task. I also suggests that there is a problem with the monitoring system and not detecting issues with systems. In the cluster, each host is system monitored by a monitoring application. The monitoring system detects if a host exhibits problematic behavior, flags it as unhealth it is removes it from the cluster. However, unseen soft faults may still occur on cluster hosts which is clearly impacting task success rates. Therefore, this explanation for the *RemoteHost* importance factor is: a failing node which has not failed.

A node continuously shares to the schedule that it has resources for tasks but due to a soft fault condition the task fails. Additionally, if we combine the *RemoteHost* soft failure condition with *CommittedSlotTime* attribute contribution we can see a situation where a task fails quickly and another task is rescheduled to the failing node. Following, this task fails again and the cycle repeats. The cluster administrators have even stated that this event condition has occurred numerous times. Now that we see the fail-reschedule to same node-fail condition in the data, further exploration is needed to determine if it can be used as predictor of a computational node failure.

Chapter 8

Conclusions

Our work addressed computational cluster operational issues. These issues are some of the numerous types of issues experienced by computational systems engineer and distributed computational systems engineers over their careers.

8.1 atSNP Search

Our atSNP search showcased an ability to engineer a 37 billion record SNP-PWM Motif database which was the most comprehensive motif-SNP database available at the time. This work required a scalable database with query response times similar to other sites with much smaller datasets. This led us to options and conclude that, of the databases evaluated, a document store NoSQL database (Elasticsearch) was our best option. In building atSNP search, we acquired drives at the same time (same batch) and from the same manufacturer (Micron). Unfortunately for atSNP search, one of the drives failed and the Elasticsearch database attempted to proactively repair the cluster, which resulted in other drives failing.

Having experienced, with atSNP, a cascading failure event and subsequent data loss, we explored an additional lines of inquiry. These lines of inquiry pointed toward two primary contributors, hard disk drives and the cluster scheduler, which caused the cascading failure event.

8.2 Hard Disk Drive Faults and Failures

Exploring hard disk drive failures, we applied machine learning to understand and predict HDD failures. Using S.M.A.R.T. attributes, our results showcased MLMs Random Forest and Support Vector Machines (SVM) algorithms can identify failure events.

We found that the S.M.A.R.T. attributes attributes 7 (Seek Error Rate), 9 (Power-On Time Count), 193 (Load/Unload Cycle Count),240 (Head Flying Hours) had the most predictive power in predicting hard disk failures. Interestingly, the highest predictive power attributes are also correlated with the operational age of the drive. Specifically the S.M.A.R.T. attributes 9 (Power-On Time Count) and 240 (Head Flying Hours) are directly correlated to the operational age of a drive. It is known that time plays in role in HDD failure events but prior work left these attributed unverified. Our work proves S.M.A.R.T. attributes 9 and 240 are a strong indicator of a failure event. Additionally, our work does not verify Blackblaze's published S.M.A.R.T. attributes importance list for predicting hard disk drive failures [122].

Taking the drives and segmenting them by HDDM had stronger alignments of the S.M.A.R.T. attributes to the existing HDD failure data 'bathtub curve' [123].

8.3 Computational Cluster Task Failures

Task profiling HTCondor cluster tasks allows cluster administrators to examine failed task features, beyond application specific code errors. We expect our HTC feature examination work to extend be useful for HPC cluster. HPC clusters have similar hardware profiles, thus our HTCondor failed task features importance ranking is applicable other clusters. Our HTC cluster allowed us to gain an understanding of a per task failure profile. These per task failure profiles are difficult to obtain on HPC systems since these tasks have interdependence's on each scheduled task. Importantly, we notice that task failures are major issue in HPC, Argonne's Theta 5.4 system averaged 37.2% of all submission tasks fail during the same time frame as our analysis which is similar to our studied cluster.

When we explored the HTCondor BCG dataset, we noticed *StartdPrinciple* class-ad attribute is a top contributor to computational cluster task failures. Intuitively, this makes sense as user parameterization of task requirements is prone to user error. Additionally, we expect to refine our research to isolate failed tasks which can not be attributed to user error. We already have a MLM which accurately predicts task failures generally. We also

note that our many predictive attributes for MLM are correlated to the submission system. Therefore, we expect to reduce our dataset for training our next MLM by segmenting our failed tasks by submission system.

Computational cluster task profiling on the cluster level provides a viewpoint beyond traditional task trace logs. Our work demonstrated the use of MLM as a method for explaining task failures within a computational cluster. HTCondor allowed us to explore details of task submissions at a granular level and enabled us to produce a feature vector of contributing factors for cluster task failures.

While our task failure prediction work showcases a need for better failure detection in computational clusters. Our model Random Forest model produced an Accuracy of 94.4%, Precision of 95% and Recall of 96.8%. This predictive power is useful for understanding and adjusting the computational cluster as needed. Specifically, the class-ad attribute *RemoteHost* is worrisome and implies an underlying issue with some computational hosts not being detected by system monitoring.

While our work did not definitively produce all answers to all task cluster failures, we identified contributing factors which warrant further exploration. Furthermore, we have an explainable model which can explain many of these task failures.

Chapter 9

Future work and Final thoughts

Our work is a beginning towards early detection of failures and faults. We laid the groundwork towards a probabilistic system to early detect faults and failures in both hardware systems and software applications. We expect to expand our line of inquiry towards the future as failures continue to impact systems. Furthermore, as systems evolve, we believe our work can be adapted to accommodate progress.

9.1 atSNP Search

Our atSNP Search work exposed a opportunity in failure prediction. However, we still need to operationally address the data loss issue. While the data loss issue is more of an house keeping then research topic, we still need to address the atSNP Search platform condition. In other words, we NEVER want to experience a "Batch-Correlated Disk Failures" again. Therefore, we expect future work in applying our Hard Disk failure prediction utilizing S.M.A.R.T. attributes to help predict failures within the atSNP Search cluster.

Further work for atSNP involves expanding the search scope to include additional species beyond homo sapiens. We have had numerous inquires to increasing the number and types of organisms available. Therefore, we plan on reviewing these requests and addressing the inquires in order of number of requests.

The last area of future work is to expand telemetry monitoring. Temporal insights into the performance characteristics of systems and applications running on computational clusters is core to our work. While kubernetes based cloud services have been utilizing telemetry metric monitoring for some time, telemetry data use is relatively rare in HPC/HTC environments and is often considered "operations" which is not available

to researchers [191]. We believe that applying telemetry metric knowledge to the field of computational clusters will have benefits and provide additional data points for applied fault detection.

9.2 Hard Disk Drive failures, faults and misbehavior's

While our initial 3 months worth of data is enough to draw conclusions. An expanded time series dataset will provide more data points to sample from and enable additional validation of our conclusions.

Expanding our time series dataset will also provide more data and we hope to find an additional cascading failure event. Encountering such an event would be fantastic and allow us to build a model based on additional events. We expect an outcome will provide predictive analytics to help prevent catastrophic data storage failures in the future.

Our future research on HDD failures can be summarized by the following:

- Using a longer time scale, the disk information will also be bigger, and the learning MLM will prove to show a greater performance and better results for all disk HDDM. This is because the processes executed by the algorithms will have a greater support and show that the methodology used in this work can also be used in datasets with a higher number of observations.
- studying and analyze the variable's normalization, we expect this to improve the state of the art and help future works to better understand the data.
- Using parallel methodology carried out in this project, a class variable that defines the disks lifetime. This classification can be done through variable 9, that counts the number of hours that the *HDD* was powered on.
- Apply MLMs built in this work to a more recent Backblaze disks observations.

Beyond HDD failures and faults, we expect our work to generalize to other complex system components in other domains beyond computing. Thus, we expect our work could be applied in other industries were failures and faults are costly or unacceptable. To that end, we see additional applications for RCA (Section 2.2.7) and FMEA (Section 2.2.6). Currently, much of the FMEA analysis is conducted through human intervention and diagnostics. We expect our work, when applied to other components besides HDD, could

simplify the RCA and FMEA process. A longer term goal is to eventually eliminating the need of a FMEA and RCA along with using a language model for the FMEA text generation.

9.3 Computational Cluster task failure prediction

Computational cluster task scheduling has improved considerably; however, task failures remain an open question. As such, we expect our research to eventually address user scheduling issues and the scheduler problems known as "the black hole" [192] problem. Our methods could be further refined and extended through analysis of other clusters and task profiles. In particular, we plan to apply our analysis techniques to private clouds. We hope to better inform the task schedulers for better service delivery and optimization of resources.

9.4 Final Thoughts

This work has taking me on a journey, both physically and mentally. I have experienced the full series of researcher emotions; complete with the intrigue of new knowledge, successes, disappointments, and of course failures. Through serendipity our PhD journey ends with failures and fault prediction. The topic of failures within any research seem foredoomed. Its human nature to want to talk about successes but failure discussions are often taboo. In many ways our work showcases that a discussion of failures can be had openly, without fear or stigma that a failure is personally attributable.

This journey was originally focused on answering a completely different research question. Our original goal was to expand the parallelization of Linear Mixed effect's Models for GWAS (Section 2.1.3.2) genomic analysis. The goal was to utilize General Purpose Graphical Processing Unit (GPGPU)s and distributed computing for GWAS analysis. To this end, we developed code for 18 months and our GPGPUs code almost ready for performance analysis. Our work had a generous $25,000 donation of cloud compute credits from Microsoft to help move our GWAS work forward. Unfortunately, this work, including the methods we were employing, was published in Nature [193] by another research team. Around this time, our atSNP Search platform was experiencing numerous disk failures. These two situations, occurring in concurrence, forced hard reflection on my

research direction. I personally began to feel like a failure because everything was going wrong and quickly.

I took some time to reflect and I realized failures and faults happen. Our atSNP Search platform failures lead me to look at disk failures and faults and failures more broadly as a research topic. In fact, I had learned lessons from the disk failures so it seemed natural to explore this area.

While part of our original motivation of our work was driven by the atSNP Search platform. Our experience demonstrated the pressing need to address the technological challenges in cluster management. By focusing on the atSNP search database and addressing "batch-correlated disk failures" [113], we led an effort to improve hard disk drive failure prediction. We also learned that improving compute task schedulers in computational clusters could lead to improved resource reliability.

In conclusion, our PhD thesis has addressed the challenges of constructing and maintaining a big data database for genomics research, utilizing machine learning to predict hard disk drive failures, and focused on understanding and forecasting compute task failures in medium-sized computational clusters. Our work has produced numerous novel contributions to the field of computer science and advanced genomic data management and research.

Appendix A

AtSNP Search Journal Papers

The following paper was accepted for publication by the journal of Bioinformatics with supplementary data.

Bioinformatics, 35(15), 2019, 2657–2659
doi: 10.1093/bioinformatics/bty1010
Advance Access Publication Date: 10 December 2018
Applications Note

Genome analysis

atSNP Search: a web resource for statistically evaluating influence of human genetic variation on transcription factor binding

Sunyoung Shin[1,*], Rebecca Hudson[2], Christopher Harrison[2], Mark Craven[2,3] and Sündüz Keleş[2,4,*]

[1]Department of Mathematical Sciences, University of Texas at Dallas, Richardson, TX 75080, USA, [2]Department of Biostatistics and Medical Informatics, [3]Department of Computer Sciences and [4]Department of Statistics, University of Wisconsin-Madison, Madison, WI 53706, USA

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on August 5, 2018; revised on November 13, 2018; editorial decision on December 4, 2018; accepted on December 6, 2018

Abstract

Summary: Understanding the regulatory roles of non-coding genetic variants has become a central goal for interpreting results of genome-wide association studies. The regulatory significance of the variants may be interrogated by assessing their influence on transcription factor binding. We have developed atSNP Search, a comprehensive web database for evaluating motif matches to the human genome with both reference and variant alleles and assessing the overall significance of the variant alterations on the motif matches. Convenient search features, comprehensive search outputs and a useful help menu are key components of atSNP Search. atSNP Search enables convenient interpretation of regulatory variants by statistical significance testing and composite logo plots, which are graphical representations of motif matches with the reference and variant alleles. Existing motif-based regulatory variant discovery tools only consider a limited pool of variants due to storage or other limitations. In contrast, atSNP Search users can test more than 37 billion variant-motif pairs with marginal significance in motif matches or match alteration. Computational evidence from atSNP Search, when combined with experimental validation, may help with the discovery of underlying disease mechanisms.
Availability and implementation: atSNP Search is freely available at http://atsnp.biostat.wisc.edu.
Contact: sunyoung.shin@utdallas.edu or keles@stat.wisc.edu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Genome-wide association studies have provided overwhelming evidence that a large number of potential causative genetic variants reside in non-coding regions such as intronic or intergenic regions (Nishizaki and Boyle, 2017). These variants might be involved in a variety of regulatory mechanisms such as transcription factor binding, histone modifications or alternative splicing, and play a key role in disease-specific regulatory networks. Significant efforts have been made to identify such regulatory variants especially in the human genome. Comprehensive web resources such as Haploreg and regulomeDB explore functional annotations in human genome at potential regulatory variants (Boyle *et al.*, 2012; Ward and Kellis, 2016). While epigenomic annotations depend on the availability of experimental data in the relevant experimental systems, annotating and nominating genetic variants for their potential regulatory effect on transcription factor binding can be achieved by *in silico* calculations (Zuo *et al.*, 2015). In disrupting or enhancing transcription factor binding to DNA, regulatory variants might modulate expression levels of disease genes. While the web resources statistically quantify the transcription factor motif matches with both reference and variant alleles, they only nominate a set of regulatory variants

passing stringent thresholds and/or do not consider creation of new binding sites by variants.

We built the atSNP Search web resource motivated by the need for (i) a more comprehensive motif-based discovery of regulatory variants; (ii) statistically well-justified quantification of motif matches and changes in motif matches and (iii) their convenient graphical depiction. All the single nucleotide polymorphisms (SNPs) in dbSNP build 144 for human genome assembly 38 (Sherry *et al.*, 2001) were initially examined against motifs from JASPAR (Mathelier *et al.*, 2014) and ENCODE (Kheradpour and Kellis, 2014) libraries by atSNP (Zuo *et al.*, 2015) testing. atSNP Search currently encompasses the results for all SNP-motif combinations having a *P*-value ≤ 0.05 for either motif matches with the reference or the SNP allele or variant-led changes in motif matches. These statistical quantifications are supplemented with composite sequence logo plots that depict the motif matches with both alleles. None of the existing web databases or software packages provide such automated visual depiction despite the fact that rapid and automated access to sequence logos enables visual exploration of impact of variants on the motif matches. SNP2TFBS, Raven and OncoCis are existing motif-based web databases that nominate regulatory variants with numerical and graphical summaries (Andersen *et al.*, 2008; Kumar *et al.*, 2017; Perera *et al.*, 2014). The Supplementary Material provides a point-by-point comparison of the atSNP Search to these resources.

2 Database contents

The current version of the atSNP Search provides statistical testing results on 37 141 563 102 variant-motif pairs with *P*-values <0.05 in either motif matches or the change in motif matches. SNPs with multiple alleles are accommodated by comparing each of the alleles with the corresponding reference allele. The atSNP Search motif collection consists of 205 JASPAR motifs and 2065 ENCODE motifs, which represent 792 unique transcription factors with motif lengths between 5 and 30.

3 Search features

The atSNP Search input form has search options with the following queries: (i) a set of SNP rs numbers (RSIDs), (ii) an RSID with a choice of window size around the variant for an extended search region, (iii) genomic coordinates (iv) a gene symbol with a choice of window size for an extended search region around the gene and (v) a transcription factor (Fig. 1). While the first four search types take genomic regions as input, the transcription factor query performs genome-wide searches, and identifies all variants that alter motif matches for the query transcription factor. Prior to submitting a search request, users can specify *P*-value thresholds for the changes in motif matches, and motif matches with the SNP and reference alleles. atSNP Search also allows the user to designate multi-level sorting based on the *P*-values and the genomic coordinates to an output table. Both the user-defined *P*-value cutoffs and sorting options make the atSNP Search more flexible and convenient compared to existing resources. Further, users may restrict searches to variants that either enhance or disrupt binding, and filter out records based on information contents of the motifs.

The atSNP Search output table provides summary statistics and composite logo plots for variant-motif pairs that meet the search criteria (Fig. 1). Each SNP ID is linked to its dbSNP webpage (Sherry *et al.*, 2001) and the UCSC Genome Browser webpage (Casper *et al.*, 2017) to display the genomic region around the variant position. Users can also access the webpages for transcription factors curated in

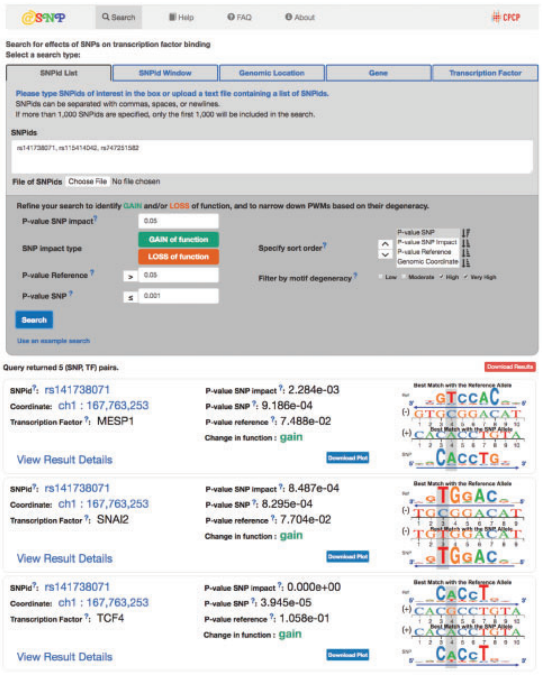


Fig. 1. The atSNP Search input form and output table. atSNP Search evaluates both strands of the reference and variant alleles around each SNP location given a motif. The composite sequence logo plot depicts the best matches of both alleles to the motif along with the strand information

Factorbook (Wang *et al.*, 2013). Testing results for each variant-motif pair are summarized with three *P*-values for motif matches with the reference and SNP alleles (*P-value Reference*, *P-value SNP*) and the changes in motif matches (*P-value SNP Impact*) along with the direction of the change. Corresponding composite logo plots display sequence logos aligned to best motif matches with the reference and SNP alleles, and visually reveal the direction of the change in motif match. Composite logo plots are especially useful to validate or reject matches to nearly degenerate motifs. Both the tables and logo plots are downloadable in csv and png formats, respectively. atSNP Search further provides detail pages displaying all statistical results of variant-motif pairs, and hypertext links for JASPAR motifs (Mathelier *et al.*, 2014).

In Supplementary Material, we provide a general purpose use-case for the atSNP Search using variants in the UK Biobank Axiom Array (Allen *et al.*, 2012). This application showcases how such an analysis can identify main transcription factors affected by regulatory SNPs.

Acknowledgements

We are grateful to Matt Ziegler for his guidance on designing the usability test of atSNP Search and Danny Panyard, Kyle Hewitth and Makoto Ohash for participating in the usability test and providing feedback. We also thank the Keleş Research Group for their continuous feedback on atSNP Search.

Funding

This work was supported by National Institutes of Health BD2K grant [U54 AI117924]; National Institutes of Health/National Human Genome Research Institute R01 grant [HG003747]; and U01 grant [HG007019].

Conflict of Interest: none declared.

References

Allen,N. *et al.* (2012) UK Biobank: current status and what it means for epidemiology. *Health Policy Technol.*, **1**, 123–126.

Andersen,M.C. *et al.* (2008) In silico detection of sequence variations modifying transcriptional regulation. *PLoS Comput. Biol.*, **4**, e5.

Boyle,A.P. *et al.* (2012) Annotation of functional variation in personal genomes using RegulomeDB. *Genome Res.*, **22**, 1790–1797.

Casper,J. *et al.* (2017) The UCSC Genome Browser database: 2018 update. *Nucleic Acids Res.*, **46**, D762–D769.

Kheradpour,P. and Kellis,M. (2014) Systematic discovery and characterization of regulatory motifs in ENCODE TF binding experiments. *Nucleic Acids Res.*, **42**, 2976–2987.

Kumar,S. *et al.* (2017) SNP2TFBS - a database of regulatory SNPs affecting predicted transcription factor binding site affinity. *Nucleic Acids Res.*, **45**, D139–D144.

Mathelier,A. *et al.* (2014) JASPAR 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. *Nucleic Acids Res.*, **42**, D142–D147.

Nishizaki,S.S. and Boyle,A.P. (2017) Mining the unknown: assigning function to noncoding single nucleotide polymorphisms. *Trends Genet.*, **33**, 34–45.

Perera,D. *et al.* (2014) OncoCis: annotation of cis-regulatory mutations in cancer. *Genome Biol.*, **15**, 485.

Sherry,S.T. *et al.* (2001) dbSNP: the NCBI database of genetic variation. *Nucleic Acids Res.*, **29**, 308–311.

Wang,J. *et al.* (2013) Factorbook. org: a Wiki-based database for transcription factor-binding data generated by the ENCODE consortium. *Nucleic Acids Res.*, **41**, D171–D176.

Ward,L.D. and Kellis,M. (2016) HaploReg v4: systematic mining of putative causal variants, cell types, regulators and target genes for human complex traits and disease. *Nucleic Acids Res.*, **44**, D877–D881.

Zuo,C. *et al.* (2015) atSNP: transcription factor binding affinity testing for regulatory SNP detection. *Bioinformatics*, **31**, 3353–3355.

Bioinformatics
doi.10.1093/bioinformatics/bty1010
Advance Access Publication Date: 8 December 2018
Applications Note

OXFORD

Genome Analysis

Supplement Material for "atSNP Search: a web resource for statistically evaluating influence of human genetic variation on transcription factor binding"

Sunyoung Shin, Rebecca Hudson, Christopher Harrison, Mark Craven, and Sündüz Keleş

1 Comparison of the atSNP Search with existing resources

Several motif-based web resources are currently available to quantify the regulatory impacts of human SNPs, among which the atSNP Search is one of the most comprehensive and up-to-date tools (Supplementary Table 1). In generation of the atSNP Search contents, we matched all 132,946,852 SNPs in dbSNP 144 on hg 38 to 2,270 motifs in total. The initial SNP set of SNP2TFBS is a SNP catalogue from 1000 Genomes project, which contains approximately 64% of the initial SNP set of the atSNP Search. The final database of SNP2TFBS itself contains a much smaller subset of variant-motif pairs that survive at p-value cutoff of 3×10^{-6}. While the atSNP Search, SNP2TFBS, and Raven harbor pre-computed results on web servers and return immediate search results, OncoCis implements motif searches on the fly using the Possum tool (Haverty *et al*., 2004).

Table 1. Comparison of motif-based regulatory SNP discovery tools

Tools	JASPAR	ENCODE	hg version	# initial SNPs	Pre-computed data
atSNP Search	✓	✓	hg38	133M	✓
SNP2TFBS	✓		hg19	85M	✓
Raven	✓		hg17	30K	✓
OncoCis	✓		hg19	NA	

Tools	Statistical significance	User-defined thresholds	Genome-wide search given a motif	Graphics
atSNP Search	✓	✓	✓	✓
SNP2TFBS	✓			✓
Raven				✓
OncoCis				

	Annotation
atSNP Search	UCSC Genome Browser hyperlink
SNP2TFBS	RefSeq gene
Raven	phastCons score
OncoCis	Gene expression, phastCons score, Histon ChIP/DNase-seq peak UCSC Genome Browser/DGIdb hyperlink FANTOM5 enhancer/promoter TSS prediction

SNP2TFBS (Kumar *et al*., 2016), Raven (Andersen *et al*., 2008), OncoCis (Perera *et al*., 2014)

2 Additional database contents

The atSNP (Zuo *et al*., 2015) testing framework estimates a background distribution to use as the null distribution for evaluating motif matches. We paid special attention to the GC content in these evaluations because GC content has been found to diversify mutation rates, as evidenced by their explanatory power in human genome variability (Hellmann *et al*., 2005). Specifically, we computed the GC content for all the 201-base-long windows centered at the SNP positions on reference alleles and classified each variant location into one of the two GC classes depicted with a mixture of two normal distributions (Supplementary Figure 1). Then the first order Markov models were fitted separately to the two sub-populations in order to impose adjacent base dependencies. Next, for every SNP-motif pair, we identified the best motif matches in the 61-base DNA sequence, centered at the variant location with both the reference and SNP alleles and quantified both the significance of the motif matches and the change in the motif matches using a likelihood-based approach.

atSNP Search utilizes the p-value of the log rank statistic evaluated at the best motif matches with both reference and SNP alleles, which is named *p-value SNP impact*, as the key sequence-based measure of detecting regulatory variants. Statistical evidence for the regulatory roles of variants is further assessed with three additional statistical hypothesis

tests on the match alteration: (1) log likelihood ratio evaluated at the best matches of both alleles, (2) log likelihood ratio evaluated at the matches of both alleles at the best match position of the reference allele, (3) log likelihood ratio evaluated at the matches of both alleles at the best match position of the SNP allele. The p-values from these calculations are named *p-value Difference*, *p-value Condition Ref*, and *p-value Condition SNP*, respectively. The scores and p-values are reported in the detail page. Users can quickly retrieve each detail page using the intuitive URL, which is a combination of the motif ID, RSID, and variant nucleotide, e.g., http://atsnp.biostat.wisc.edu/detail/motifID_RSID_ N. This feature enables programmatic access to atSNP Search results. For studying human variants other than SNPs or non-human genetic variations, we suggest the `R` package atSNP, which is publicly available at https://github.com/keleslab/atSNP.

3 Heterogeneity of GC content around genomic locations of variants

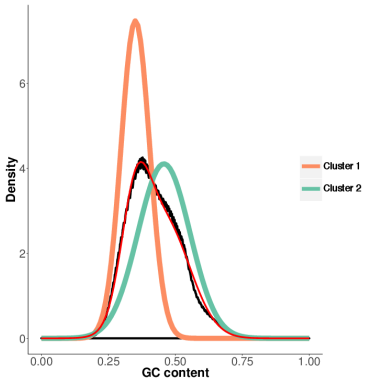


Fig. 1. Grouping of SNPs based on their local neighborhood GC content with a mixture modeling framework. The black curve denotes the observed GC content, whereas the red curve is the fitted probability density function of the mixture of two normal distributions. The curves labeled as Cluster 1 and 2 denote the two identified components.

Table 2. Estimated stationary distributions and transition matrices of the two SNP groups based on their GC contents.

Cluster 1					Cluster 2			
A	C	G	T		A	C	G	T
0.34	0.16	0.16	0.34		0.26	0.24	0.24	0.26
	A	C	G	T		A	C	G	T
A	0.37	0.15	0.18	0.30		0.28	0.20	0.30	0.22
C	0.41	0.19	0.03	0.37		0.32	0.29	0.08	0.31
G	0.34	0.17	0.19	0.30		0.26	0.24	0.29	0.21
T	0.27	0.16	0.20	0.37		0.17	0.24	0.30	0.28

Supplementary Figure 1 displays the distribution of GC content in the local neighbourhood of SNPs, i.e., a 201-base-long windows centered at the SNPs, and the two mixture components that are identified. The two groups have significantly different transition patterns, and in the stationary state, the second cluster has higher GC content than the first cluster (Supplementary Table 2).

4 The atSNP Search infrastructure

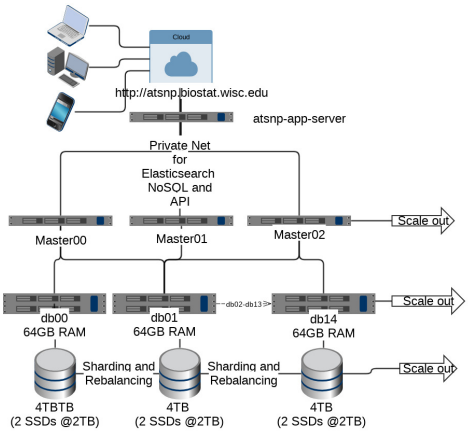


Fig. 2. The atSNP Search design.

atSNP Search is written with Django, a high-level Python Web framework that encourages rapid development and pragmatic design (Forcier *et al.*, 2008). atSNP Search contents were first generated in RData format using UW Madison HTCondor, an open-source high-throughput computing software framework for coarse-grained distributed parallelization of computationally intensive tasks (Thain *et al.*, 2005). "Years of compute hours" on the entire task are roughly 13 years (113,500 HTC hours) on a single core CPU machine with at least 7GB of disk space and 10GB of memory. Records on variant-motif pairs with marginal significance in motif matches or alteration were provided as input to the atSNP Search server in JSON format. Custom Python scripts for ETL (Extract, Transform and Load) were utilized for data loading (Harrison *et al.*, 2018). Elasticsearch, a NoSQL database, runs on the atSNP Search server, utilizing a distributed scale-out system architecture for large workloads (Gormley and Tong, 2015). It accomplishes the task of search and retrieval by distributing requests for searches among the scaled computing resources. As requirements for storage and performance increase with user demand, we can scale out by adding more machines. A restAPI (Masse, 2011) handles communication between the search page and the Elasticsearch data store. The complete atSNP Search infrastructure is illustrated in Supplementary Figure 2. Composite sequence logos are generated on the fly using D3.js, which is a JavaScript library for dynamic and interactive data visualizations on web (Bostock *et al*, 2011).

5 atSNP query response time

Table 3. Response time for SNPid List and SNPid Window Searches (in seconds).

SNPid List			SNPid Window		
# of	p-value		Window	p-value	
SNP IDs	0.05	0.01	size	0.05	0.01
15	13.7-19.6	14.7-32.1	100	2.7-9.6	2.7-3.3
50	9.2-40.1	3.5-27.0	1K	2.7-9.6	2.7-3.3
100	29.7-32.5	13.6-14.4	10K	40.5-53.4	12.3-24.1
500	API timeout	API timeout	100K	API timeout	API timeout

Table 4. Response time for Genomic Location and Gene Searches (in seconds).

Genomic Location			Gene		
Loscation	p-value		Window	p-value	
size	0.05	0.01	size	0.05	0.01
1K	3.9-12.5	3.1-5.2	100	4.9-11.5	2.2-2.7
10K	22.8-51.1	7.9-32.3	1K	9.5-29.3	2.6-7.3
50K	39.4-55.9	36.1-37.4	5K	46.7-61.2	24.5-33.2
100K	API timeout	API timeout	10K	API timeout	API timeout

Table 5. Response time for Transcription Factor Search (in seconds).

Library	Transcription	p-value	
	Factor	0.05	0.0005
JASPAR	ZNF263	2.3-4.7	2.5-2.6
	CTCF	3.8-5.1	2.6-3.9
ENCODE	AFP	2.7-3.3	1.7-5.4
	GATA	40.3-47.7	27.9-43.6

Supplementary Tables 3-5 report response time for the five search types under various combinations of query parameters and significance levels. We performed two experimental runs under one combination of each search type at a time and recorded both response times. Our empirical studies suggest that, overall, both query type and size of query results determine the response time albeit some exceptions exist. The four types in Supplementary Tables 3-4 search through all variant-pairs which meet the user-defined criteria within a collection of SNPs or a specified genomic range. Transcription factor search, which needs no access to genomic coordinates, returns thousands to hundreds of millions of variant-motif pairs within a minute.

6 Enrichment analysis of acute myeloid leukemia SNPs

UK Biobank genotyped 820,967 SNPs using the Affymetrix Axiom arrays, a subset of which are annotated with disease genes in Online Mendelian Inheritance in Man (OMIM) database (Hamosh *et al.*, 2005). We used atSNP Search to assess whether the 1,475 acute myeloid leukemia (AML) SNPs in the UK Biobank are enriched for impact on a set of transcription factors. To assess enrichment, we utilized the 21,529 non-AML cancer SNPs in the biobank as the background set of SNPs. Binding enhancement or disruption of a transcription factor by a SNP are assumed to occur when

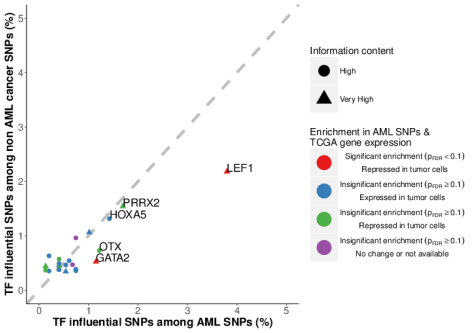


Fig. 3. Proportions of SNPs impacting binding of the 23 transcription factors among non-AML and AML SNP sets. For each transcription factor, the proportion of SNPs with significant *P-value SNP Impact* (Bonferroni correction at level 0.05) for the AML SNPs was compared to that for the background set of SNPs from non-AML cancers.

the SNP significantly impacts matches of at least one motif corresponding to the transcription factor at the significance level of 0.05 after Bonferroni multiple testing correction. Using atSNP Search queries to conduct this analysis results in 13,578 non-AML cancer SNPs and 906 AML SNPs as impacting at least one of the 102 transcription factors that have motifs with high information content (median IC ≥ 1.1). For each transcription factor, we evaluated whether the proportion of SNPs with significant impact differed between the two SNP sets after constructing a contingency table. Supplementary Figure 3 summarizes the results on 23 transcription factors, the contingency tables of which have all expected cell frequencies larger than or equal to 5. We found the proportion of SNPs impacting binding of LEF1 and GATA2 significantly differ between the two groups at a false discovery rate of 0.1.

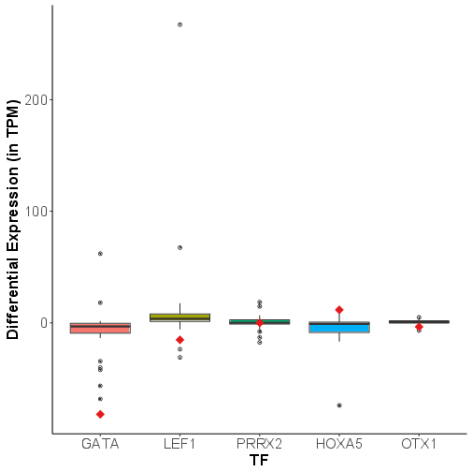


Fig. 4. Differential expression level distribution of the 31 TCGA cancer types for the five transcription factors from GEPIA. Red rhombi indicate differential expression levels in AML tumor samples.

We next asked whether expression of these transcription factors across cancer types are supportive of this finding. Specifically, we computed differences in their median expression levels in TPM (Transcripts Per Million) between AML tumor samples and matched normal tissues with the GEPIA web server (Tang *et al*., 2017). GATA2 is repressed in AML compared to matched controls by 82.04 TPM, and LEF1 is repressed by 15.23 TPM (Supplementary Figure 4). Both differential repression levels in AML are identified as outliers with respect to their distributions in all 31 TCGA cancer types, thus both transcription factors are considered having AML-specific expressional differences. Furthermore, recent research on GATA2 (Hsu *et al*., 2013; Johnson *et al*., 2012) showed that mutations of a GATA2 intronic binding site cause a primary immunodeficiency (MonoMAC) associated with myelodysplastic syndrome that progresses to AML. PRRX2 and HOXA5, which are affected by a larger proportion of AML SNPs compared to GATA2, exhibit less specificity to AML compared to the rest of cancer types. OTX1 is more repressed in AML tumor; however, overall differential OTX1 expression levels are marginal, thus its AML-specificity may not be appreciable.

References

Andersen, M. C., Engström, P. G., Lithwick, S., Arenillas, D., Eriksson, P., Lenhard, B., Wasserman, W. W., & Odeberg, J. (2008). In silico detection of sequence variations modifying transcriptional regulation. *PLoS computational biology*, **4(1)**, e5.

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3 data-driven documents. *IEEE transactions on visualization and computer graphics*, **17(12)**, 2301-2309.

Forcier, J., Bissex, P., & Chun, W. J. (2008). Python web development with Django. *Addison-Wesley Professional*.

Gormley, C., & Tong, Z. (2015). Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine. *O'Reilly Media, Inc*.

Hamosh, A., Scott, A. F., Amberger, J. S., Bocchini, C. A., & McKusick, V. A. (2005). Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic acids research*, **33** (suppl_1), D514-D517.

Harrison, C., Keleş, S., Hudson, R., Shin, S., & Dutra, I. (2018). atSNPInfrastructure, a case study for searching billions of records while providing significant cost savings over cloud providers. In *Proceedings of the 32nd IEEE International Parallel and Distributed Processing Symposium Workshops*, pp. 497-506.

Haverty, P. M., Hansen, U., & Weng, Z. (2004) Computational inference of transcriptional regulatory networks from expression profiling and transcription factor binding site identification. *Nucleic acids research*, **32**, 179-188.

Hellmann, I., Prüfer, K., Ji, H., Zody, M. C., Pääbo, S., & Ptak, S. E. (2005). Why do human diversity levels vary at a megabase scale?. *Genome research*, **15** (9), 1222-1231.

Hsu, A.P., Johnson, K.D., Falcone, E.L., Sanalkumar, R., Sanchez, L., Hickstein, D.D., Cuellar-Rodriguez, J., Lemieux, J.E., Zerbe, C.S., Bresnick, E.H., & Holland, S.M. (2013). GATA2 haploinsufficiency caused by mutations in a conserved intronic element leads to MonoMAC syndrome. *Blood*, **121** (19), 3830-3837, S1-S7.

Johnson, K.D., Hsu, A.P., Ryu, M.J., Wang, J., Gao, X., Boyer, M.E., Liu, Y., Lee, Y., Calvo, K.R., Keles, S., Zhang, J., Holland, S. M., & Bresnick E. H. (2012). Cis-element mutated in GATA2- dependent immunodeficiency governs hematopoiesis and vascular integrity. *The Journal of clinical investigation*, **122** (10), 3692-3704.

Kumar, S., Ambrosini, G., & Bucher, P. (2016). SNP2TFBS-a database of regulatory SNPs affecting predicted transcription factor binding site affinity. *Nucleic acids research*, **45(D1)**, D139-D144.

Masse, M. (2011). REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. *O'Reilly Media, Inc*.

Perera, D., Chacon, D., Thoms, J. A., Poulos, R. C., Shlien, A., Beck, D., Campbell, P. J., Pimanda, J. E. & Wong, J. W. (2014). OncoCis: annotation of cis-regulatory mutations in cancer. *Genome biology*, **15(10)**, 485.

Tang, Z., Li, C., Kang, B., Gao, G., Li, C., & Zhang, Z. (2017). GEPIA: a web server for cancer and normal gene expression profiling and interactive analyses. *Nucleic acids research*, **45** (W1), W98-W102.

Thain, D., Tannenbaum, T., & Livny, M. (2005). Distributed computing in practice: the Condor experience. *Concurrency and computation: practice and experience*, **17(2-4)**, 323-356.

Zuo, C., Shin, S., & Keleş, S. (2015). atSNP: transcription factor binding affinity testing for regulatory SNP detection. *Bioinformatics*, **31(20)**, 3353-3355.

Appendix B

AtSNP infrastructure Conference Paper

The following paper was accepted for conference proceeding publication as a workshop paper at IEEE International Parallel and Distributed Processing Symposium (IPDPS) in 2018.

atSNPInfrastructure, a case study for searching billions of records while providing significant cost savings over cloud providers

Christopher Harrison [*†], Sündüz Keleş[*], Rebecca Hudson[*], Sunyoung Shin[*] and Inês Dutra[†]
[*] Department of Biostatistics and Medical Informatics
School of Medicine and Public Health
University of Wisconsin - Madison
Madison, Wisconsin USA
[†] Departamento de Ciência de Computadores
Faculdade de Ciências
Universidade do Porto
Porto, Portugal

***Abstract*—We explore the feasibility of a database storage engine housing up to 307 billion genetic Single Nucleotide Polymorphisms (SNP) for online access. We evaluate database storage engines and implement a solution utilizing factors such as dataset size, information gain, cost and hardware constraints. Our solution provides a full feature functional model for scalable storage and query-ability for researchers exploring the SNP's in the human genome. We address the scalability problem by building physical infrastructure and comparing final costs to a major cloud provider.**

***Index Terms*—NoSQL, Billion Records, SNP, Big Data, Data Reduction, Distributed Computing, PWM, Edge Computing, Economical Computing, Genomics, Cassandra, MySQL, Elasticsearch**

Introduction

Genome-wide association studies (GWAS) have been important to identify Single Nucleotide Polymorphisms (SNPs). In particular, regulatory SNPs (rSNPs) are the ones that affect gene regulation by changing transcription factor (TF) binding affinities to genomic sequences. These changes are crucial for understanding disease mechanisms. **A**ffinity **T**esting SNP (atSNP) [1] is a powerful and efficient R package which implements an importance sampling algorithm coupled with a first-order Markov model for the background nucleotide sequences to test the significance of affinity scores and SNP-driven changes in these scores. A standard in silico approach for identifying rSNPs is by evaluating how the NP-driven nucleotide change impacts binding affinity of TFs to the region surrounding the SNP. More specifically, the DNA sequences around each SNP are scored against a library of TF motifs with both the reference an the SNP alleles using Position Weight Matrices (PWMs) of the motifs. The process is very time-consuming and usually needs to handle a large amount of input data and generates huge output files. atSNP is considered the state-of-the-art tool for handling such large-scale task. The output files generated by atSNP can be inspected and analyzed for SNP-motif interactions. However, given the size of the files, it can be unfeasible to browse and visualize the motif information, given current tools. We test and analyze some solutions for this problem, using hardware and software available in the literature. We evaluate the feasibility of three major open source databases for: data loading, storage and search; then implement a solution based on our feasibility testing. Furthermore, we showcase that our implementation: database choice, required system hardware and network provide an extremely cost effective option even against a major cloud provider.

This paper is organized as follows. In Section I we discuss the problem in more detail and in Section II we discuss solutions in the literature. In Section III we present our data requirements, hardware, software and evaluation method used in our analysis. Section IV presents results. Finally, we close in Section V with concluding remarks.

I. Background

Large scale real time searching is difficult for any significantly large dataset. Genetics and genomics datasets and searching domains are significantly larger as small

variations within a DNA sequence can produce numerous phenotypic variations [2]. As such, DNA analysis is still a domain ruled by statistically probabilities based on a specific subregion within the DNA strand. The subregions known as Single Nucleotide Polymorphisms [3] (SNPs) are produced using a DNA sequencing technique commonly known as the shotgun approach to DNA sequencing [4]. While the merits of the approach are known and solidly accepted, the probability of any given SNP occurring within a Genome-Wide Association Study (GWAS) is lesser known. The atSNP R package [1] attempts to address the probabilistic significance of affinity scores in SNPs.

atSNP search provides statistical evaluation of impact of SNPs on transcription factor-DNA interactions based on PWMs. The current version is built on SNPs from the dbSNP [5] Build 144 for the human genome assembly GRCh38/hg38 [6]. atSNP search uses in silico calculations based on the statistical method developed by Zuo *et al*. [1]. atSNP first-time users can start scanning vast amounts of SNP-PWM combinations for potential gain and loss of function with two key features:

- p-values quantifying the impact of SNP on the PWM matches (p-value SNP impact)
- composite logo plots.

atSNP identifies and quantifies best DNA sequence matches to the transcription factor PWMs with both the reference and the SNP alleles in a small window around the SNP location (up to +/- 30 base pairs and considering sub-sequences spanning the SNP position). It evaluates statistical significance of the match scores with each allele and calculates statistical significance of the score difference between the best matches with the reference and SNP alleles.

The discovery of best matches with the reference and SNP alleles are conducted separately. As a result, genomic sub-sequences providing the best match with each allele may be different. The change in goodness of match between the reference and SNP alleles indicates in silico impact of SNPs on transcription factor binding ability to DNA. atSNP search also provides a composite logo plot for easy visualization of the quality of the reference and SNP allele matches to the PWM and the influence of SNPs on the change in match.

Addressing GWAS SNP affinity scores requires addressing magnitudes of data to determine probabilistic significance. After the generation and analysis of the SNP affinity scores, sharing the affinity scores with the community reduces the need to "reinvent the wheel" for any group determining best SNP affinity scores complete with Position Weight Matrix [7] (PWM) motif data [1]. Run once and share with many allows for efficient computational resource allocations for future projects.

The initial generation of the affinity score PWMs was done *in silico* with the atSNP [1] R [8] package. To complete the full data of JASPAR [9] and ENCODE [10] a distributed divide and conquer method was applied. This research was performed using the compute resources and assistance of the UW-Madison Center For High Throughput Computing (CHTC) in the Department of Computer Sciences. The CHTC is supported by UW-Madison, the Advanced Computing Initiative, the Wisconsin Alumni Research Foundation, the Wisconsin Institutes for Discovery, and the National Science Foundation, and is an active member of the Open Science Grid, which is supported by the National Science Foundation and the U.S. Department of Energy's Office of Science. Using HTCondor [11] and the CHTC 132,946,852 SNPs combined with 2,270 PWMs generated 3.02×10^{12} SNP-PWM combinations records and required 115,500 CPU hours of computational power. All the records weights are based on the p-values of:

The complete details for the data generated are described in the atSNP method [1].

Our dataset will search based on p-value. Our p-values are represented as floating point numbers and to search our p-values we will need to utilize range queries. Range queries are data searches of number fields within a numeric range, less then or great then are using for range queries.

II. Related Work

Numerous examples of SNP databases [12] exist across the web with examples ranging from the dbSNP [5] to dbGAP [13]. While these data are provided through online web accessible database, they are a magnitude smaller when compared with the atSNP-generated data. For example, the Genome Variation Map database [14] recently published their current implementation which houses a total of about 4.9 billion variations using a MySQL database [15]. dbSNP [5] also utilizes a Relational Database Management System (RDBMS) [16], an entity relational database [17], which consists of well over 100 relational tables.

Example databases housing hundreds of billion records are found utilizing NoSQL infrastructure. Netflix, for example, has a NoSQL Cassandra [18] deployment consisting of 2,500 nodes hosting 420 TB of data. The Netflix database handles over a trillion requests per day [19]. LinkedIn uses Voldemort [20], a key/value

TABLE I
atSNP p-values

atSNP p-values:	
p-value SNP Impact	A significant p-value, e.g. 0.0001, statistically supports the potential gain or loss of function of the genomic region with the SNP in terms of transcription factor binding.
p-value Reference	p-value for score with the reference allele (Log Likelihood Reference). A significant p-value indicates that the match to the PWM with the reference allele is statistically supported.
p-value SNP	p-value for scores with the SNP allele (Log Likelihood SNP). A significant p-value indicates that the match to the PWM with the SNP allele is statistically supported.
p-value Difference	p-value for the difference in scores with the reference and the SNP alleles (Log Likelihood Ratio). This is essentially the result from a likelihood ratio test. While we report this for completeness, the final atSNP results are based on p-value SNP impact which quantifies the significance of the Log Rank Ratio.
p-value Condition Ref	Conditional p-value for scores on the reference allele based on Log Enhance Odds.
p-value Condition SNP	Conditional p-value for scores on the SNP allele based on Log Reduce Odds.

store for their critical infrastructure. Google, often credited with starting the big data movement, created their own NoSQL engine BigTable [21]. Therefore, many examples of successful NoSQL database implementations exist and they have proven the design for scale [22]. It has been demonstrated [23] that NoSQL databases have been successfully deployed with hundreds of millions or billions of records and "can result in a dramatic performance difference against RDBMS while dealing with hundreds of millions or billions of records" [23].

Each of the NoSQL databases used by Netflix, Google, and LinkedIn; utilize a different data sharding algorithm. While the algorithmic implementation for each NoSQL sharded database varies, each share a divide and conquer approach through their distributed data storage engine [24].

III. Design

The publicly searchable repository for SNP affinity scores with PWM motifs (over 132,946,852 SNPs) combined with 2,270 PWM generated 3.02×10^{12}records. Each record consumes 1,416 bytes (B) of data. The specific data, generated by atSNP [1], used in this work are related with the human genome. The TF libraries used by atSNP came from the JASPAR core motif library [9] and from the ENCODE project [12]. After our analysis the JASPAR [9] dataset comprised 2.81×10^{10} records totaling 36.18 TeraBytes (TB), while the ENCODE [12] dataset comprises 2.79×10^{11} records totaling 361.84 TB. The total dataset is 3.02×10^{12} records and consumes 398.03 TB of data.

Providing 398.03 TB of SNP affinity scores with motif data online with real time queryable parameters requires advanced database techniques. Additionally, data loading and searching for 398.03 TB at 3.02×10^{12} records can cause additional issues as loading time and throughput start to become a bottleneck. As an example, using a gigabit ethernet (gigE) network interface to transfer 1 TB of data takes around two and a half hours. We calculuate transferring over our gigE network the complete 398.03 TB dataset will take 45 days.

In addition to the affinity score records, the corresponding motif representation for a given SNP-PWM in graphical representation is a multi-petabyte data storage problem. The graphical representation for a given SNP motif saved as a Portable Network Graphic (PNG) consumes approximately 250 KiloBytes (KB) per image. The full SNP database representation stored in PNG format as a graphical motif requires 3.7 PetaBytes (PB) of data storage.

Single computational systems supporting 398.03 TB of online searching with 3.7 PB of additional storage for images simply do not exist.

The problem domain decomposition starts with the space defined by the SNP-PWM affinity scores. Our full compressed JASPAR [9] dataset consuming 1.8 TB seems realistic for ETL (Extraction, Transformation and Loading) into a traditional relational database (e.g. MySQL, Oracle or Postgres). The problem expands considerably when considering the 19.8x compression ratio for each dataset. The 1.8 TB dataset, containing 2.81×10^{11} records, expands to 18.27 TB which can stretch the boundaries of current RDBMS. An additional 19TB of compressed files from the ENCODE [12] motif library contains 2.79×10^{12} records, which is 181.43 TB

after decompression. Individual RDBMS systems, with only one host, housing dataset of 307 billion records and store 398.03TB of data likely not feasible but we will test RDBMS vs NoSQL databases for feasibility. We consider a data model while evaluating feasible and functionality with temporal ETL requirements and data result query time. We hypothesise the final implementation to house the SNP-PWM data will consist of a database engine utilizing data sharding or the partitioning/breaking apart data for distributed queries against the large datasets.

Performance comparisons between RDBMS and NoSQL database [25] show inconclusive results [26]. For our specifical use case, searching billions of records, scalability to store billions of records was our primary concern. In Puangsaijai2017 [27], we note that MariaDB [28], a MySQL forked project which is feature complete with MySQL [15], shows ETL insert times to be comparable between NoSQL database Redis [29] to be a >20x faster with pipelines over MariaDB [28]. Interesting [27] also shows select operations to be comparable across many simple and complex queries with the exception of 2 simple equals queries. More interesting are the tested range queries (greater then and less then) [27] shows the NoSQL database to out perform MariaDB by 2x.

We utilized a custom ETL script developed using python for data loading [30]. The code was custom developed for reading the SNP-PWM files and translating them to the specific database ETL format. For our feasible testing we used one system to run our ETL pipeline code. After we refined our process, we utilized HTCondor [11] to distribute our ETL pipeline processes.

A. Hardware cost evaluation and failure estimating

Hardware and human resources costs were given consideration. We evaluated the cost per hardware node and compared a new machine vs a used machines. These comparisons included similarly specifications, cpu, memory, remote management, power supplies and OS disk drives, etc. Additionally, we considered node failure rates within the cluster and calcuate node availality based on system component failure statists, specifically: RAM, Power supply and Disk failures. Using the failure statistical model we estimate required write endurance per drive accounting for node availality hardware and cluster re-balancing. Provided by our failure statistical model, we consider the cost per TB for our cluster, amortizing the cost over 5 years. Comparing new nodes vs used nodes vs Cloud hosting including: addressing the Total Cost of Ownership (TOC), data center floor space cost, power/cooling and system administration time we demonstrate cost efficiency.

TABLE II
OUR FIRST GENERATION ELASTICSEARCH CLUSTER COMPOSITION

Machine	RAM (GB)	#Processors	Total HDD (TB)
atsnp-db1	24	4	12
atsnp-db2	24	4	12
atsnp-db3	32	8	18
TOTAL	80	16	42

We evaluated the performance of MySQL, Cassandra (a NOSQL database) and ElasticSearch as database engines. In order to run these database engines, we used two computational infrastructures: (1) a customized in-house cluster and a (2) cloud provider. We tested different configurations of clusters varying memory sizes and disk sizes as well as number of disks. We ran the experiments using a trial and refine approach while following agile methodologies. We tested each database for feasibility, scalability and features which could support our data search needs. Each database tested was tested with default configurations using minimal install of Scientific Linux 6, our test cluster was comprised of 3 nodes described in table II. Feasible testing evaluation items included: ETL speeds, search retrieval times(within 5 seconds), required on going system administrator time to maintain, scalability and failure recovery. For testing, we evaluated a database populated with $\tilde{1}$ million records.

IV. RESULTS AND EVALUATION

A. Cassandra

Our first test considered Cassandra [18], a Big Table NoSQL database engine. Our data ETL times to database import were 14,664.2 records per second, which is similar to the University of Toronto NoSQL benchmarks suggest [31]. Our data requires multiple range queries which are not easily indexed within Cassandra. The range queries in the implementation of Cassandra's instance of NoSQL required, every node to query all data housed on the node for any row meeting the range query parameter without indexes [32]. The lack of range queries and on the performance penalty for evaluating a data subset proved a fatal flaw in our first method of genomic SNP data at scale using a NoSQL implementation. Our requirement for multiple range query through the atSNP dataset using application and used defined searches meant that Cassandra was not feasible for long term use.

B. MySQL

Using our data with MySQL, after the first million records our database was responsive (results on the order of milliseconds). As our database started to grow, efficiency started to drop. Once our database was consuming all available ram for indexes any additional record required the use of system swap space which caused response times and ETL times to be unworkable. Based on our ETL estimates approximately 1000 records a second, which is similar to the 1023 records/sec demonstrated in [33]. Therefore, if we continued ETL with database transactions enabled by default to ETL our data would have required 1:

$$\frac{3.02 \times 10^{12}}{(\frac{1000\ records}{second} * 60\ seconds * 60\ minutes * 24\ hours)} = 3,553.2\ days \quad (1)$$

10 years to ETL our data makes the use of a MySQL database engine unfeasible.

Consideration for using the storage engines innodb and NDB with clusters were not even considered, given the limitations of MySQL [34]. Factors against distributing the load across multiple instances of MySQL using partition tables (one per chromosome) in a MySQL cluster were overridden because of a concern for the long term support-ability of the database service and system administration time overhead required. Specifically, concerns raised included: status checking, time to maintain and migrate data when needed, updating the application logic after a partition table migration and system monitoring.

C. Elasticsearch

For the third test case, we used a distributed NoSQL since we are validating the hypothesis of using shared NoSQL database for genomic SNP data as a feasible option for large scale storage and query-ability. This test uses a shared NoSQL database engine, but optimized for range queries. Specifically, we consider the Elasticsearch database engine for its focus and utilization of range queries for system log searches. To evaluate Elasticsearch lookups of address range queries, we loaded 1,012,032 samples ($\approx$ 1 million records to prove range query feasibility. Our initial tests passed with query times less than 5 ms.

Database ETL issues arose in the ETL phase for our Elasticsearch database. Our ETL times averaged 11,944.5 records per second or 1,040,000,000 records per day which is significantly faster than the MySQL approach but not sufficient for our time-line. Given our ETL rates the ETL process would have completed in 295 days to complete our ETL of the entire 3.02×10^{12} records. Additionally, our query search speed degraded significantly while concurrently inserting ETL data; therefore, additional I/O capacity was recommended. Our first configuration utilized 3 prior used servers augmented with additional storage: 7200 rpm 4 TB hard drives. 2 systems operated with 3 HDD and 1 system operated with 5 drives. Having accomplished a main feasibility objective, we were satisfied with the Elasticsearch option.

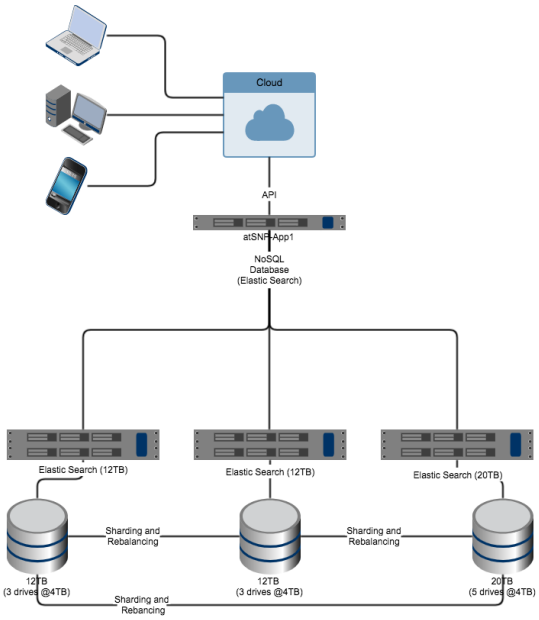


Fig. 1. Our test cluster with final selection

TABLE III
OUR EVALUATION MATRIX

Database	ETL rec/sec	>5sec search	system admin time	scalability	failure auto recovery
Cassandra	14,664	no	ok	yes	yes
MySQL	1023	yes	no	no	no
Elastic-Search	11,944	yes	ok	yes	yes

D. Refinements to reduce cost

The atSNP output data was computed with high precision p-values for a given SNP evaluation. Addressing each record and reducing the precision of the p-value

$$\frac{\#drives * 540MB}{MBnode * \#nodes}$$

Fig. 2. Datasearch for cluster based on SSD speed

$$\frac{nodes * 10}{1}$$

Fig. 3. cluster excess capacity accounting for dram node failure rate

calculation, we were able to achieve a significant reduction in the record size from 1,416 to 300 bytes of data. The reduction in data record size allowed us to reduce the size of the data storage impact to 92.1 TB. Our data size reduction however did not translate into an overall database size reduction. Elasticsearch, like many sharded NoSQL database engines, utilized internal data partitioning to spread the dataset over the entire cluster. The database internal partition schema provides cluster redundancy in the case of individual system outages. Our dataset total records equated 92.1 TB of data. However, the default Elasticsearch replica factor of 3x ballooned our database sizing requirements back to 276.3 TB. Thus, any space saved in record size reduction was consumed by the cluster redundancy schema.

Further consideration was done for the three key p-values in atsnp search results are p-value SNP impact, p-value reference, and p-value SNP. In a typical application, users would want to identify SNP-TF combinations for which p-value SNP impact is significant (e.g., p-value SNP impact <= 0.05) along with significance of either the p-value reference or the p-value SNP. To enable this in the most liberal way, we included all SNP-TF combinations with a p-value smaller than 0.05 for at least one of the three p-values.

Our final p-value cutoff value of 0.05 enables querying of these statistical evaluations which exhibit a significant score change or a significant match with either the reference or the SNP allele. The resulting 37 billion SNP-PWM records total consumed 18.4 TB. With 18.4 TB we could utilize 2 data replicas for a total consumed disk space of 60TB.

See cost breakdown in the Hardware Cost table IV:

To address our ETL time for our dataset, we utilized HTCondor for additional throughput with the ETL scripts. Distributing the ETL tasks in HTCondor proved to be effective, however our cluster experienced overload conditions. Elasticsearch was a feasible database engine for our atSNP data; however, our ETL times were still beyond functional for our needs. Our initial experiments

TABLE IV
HARDWARE COST

Type	SSD (TB)	Node Cost ($)	Node cost/yr ($)
new	1	5441	1088.2
new	2	5731	1146.2
new	4	6251	1250.2
used	1	1010	202
used	2	1300	260
used	4	1820	364

TABLE V
RUNNING SYSTEMS COST

node $/yr	pwr/clg ($)	adm ($)	monthly ($)	yearly ($)
1088.2	206.64	1343	219.82	2637.84
1146.2	206.64	1343	224.65	2695.84
1250.2	206.64	1343	233.32	2799.84
202.0	206.64	1343	145.97	1751.64
260.0	206.64	1343	150.81	1809.64
364.0	206.64	1343	159.47	1913.64

# nodes	GB	Cost/GB ($)	monthly ($)	yearly ($)
15	15360	0.216	3297.30	39567.60
15	30720	0.109	3369.75	40437.00
15	61440	0.057	3499.80	41997.60
15	15360	0.143	2189.55	26274.60
15	30720	0.078	2262.15	27145.80
15	61440	0.039	2392.05	28704.60
Hosted	Hosted	$0.135	varies	varies

TABLE VI
HOSTED VS CLOUD

showed that our first generation cluster running nodes in Taable II not sufficient to query and search the large database at hand. ETL on our first generation test cluster, we realized a node could maintain 6 ETL scripts per node without experiencing a ‘"Bulk Queue Full‘" error exception. Addition considerations were needed to account for query response times. We knew the maximum read speed per SSD is between 530-540MB/sec [35], so a complete scan all data per node can be represented by the data scan equation in Figure 2. After our refinements we can estimate the number of nodes our cluster requires, we started with the equation in figure 2 for the raw Disk I/O.

For each node, we followed Elasticsearch heap size recommended to keep below 32GB [36]. Additionally, following Elasticsearch "The standard recommendation is to give 50% of the available memory to Elasticsearch heap, while leaving the other 50% free" [36] this also providing the Operating system with free memory for data page tables. Thus, we concluded our optimized

RAM per node at 64 GB of RAM: 30GB for Heap, 30GB system paging, 4GB Kernel/OS.

We evaluated physical chassis and compared the cost of buying new vs usedIV. The node cost hardware for new $5,151.23 was based on a Dell quote from dell.com for an R630 with 64 GB RAM, 2 x 8 core Intel Xeon E5-2623 v4 2.6GHz, 10 MB Cache, 8.00 GT/s QPI, Turbo, HT, 4C/8T. The cost per hardware node for used $720 was based on a similar but one generation older dell model, a R620, similarly configured with 64 GB of RAM, 2x8 core Intel Xeon E5-2650 2.0 GHz, 20 MB Cache, 8.00 GT/s QPI, Turbo, HT, 4C/8T. For both new and used we did not select any extra extended warranties.

Our SSDs selection was a bit more complexity. Our initial write endurance concerns where displaced after examining the application data requirements, specifically the static nature after the initial ETL stage.

Calculating the failure rates for a cluster node allows us to model our cluster re-balancing. We modeled node failure rates based on numerous factors, specifically: Dynamic Random Access Memory (DRAM), Disk Drive (Disk), Power Supply (PS), CPU, Motherboards (MB) with MB including PCI Bus, SCSI Backplane, NIC, etc. For DRAM our model used hard error rates between 1.7% to 2.3% [37] per month. Disk failure rates varied greatly between Hard Disk Drives (HDD) and Solid State Disk (SSD), within SSD's there was additional variations for where bo, power supplies, and , so write endurance numbers based on manufactures specifications and combining them with per month failure rates allows us to select the we base our cost structure on the SSD class enterprise vs desktop to purchase.

of drive throughput and endurance since our cluster auto rebalances when a node fails. We use the equation 3 to determine

Since our application main purpose is a storage and search engine; therefore, write endurance requirements where considerably less then . The data per node based on ETL was decided were based on crucial MX300 family of SSD, to simplify we compared only 1 and 2 TB SSDs. Each node provided 4 disk trays per server 2 for OS drives and 2 for data. To reduced costs by shopping for the best SSD pricing and we found the best cost for a crucial 300MX SSD 1 TB SSD at $289.99 [38] and 2 TB SSD $549.99 [39]. Addressing power consumption and power cost/cooling costs over 95 days time period a similar system consumed 192.1 KWh or 2.022 KWh/day/node, including cooling, we doubled this cost to 4.044 KWh/day/node at a current KWh price about $0.14KWh [40]. Systems administration time cost

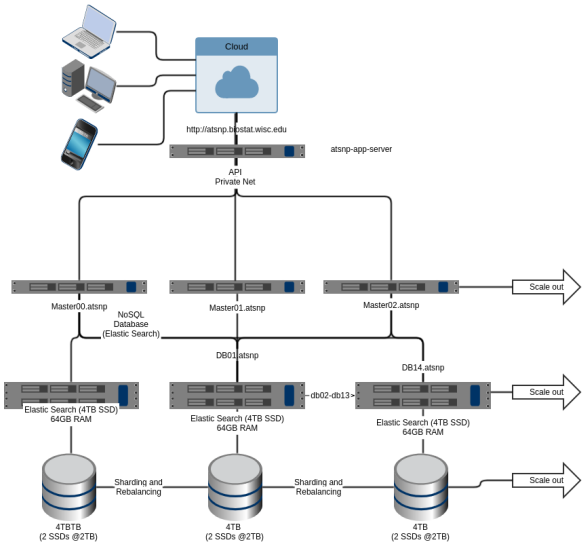


Fig. 4. Final atSNPElasticsearch cluster

was based off the IBM TOC [41] for Application server support and administration at $1,343/year for 5 years.

Based on 15 data nodes and the cost per gigabyte equation, and using TableIV-D, we selected 2x 2TB SSD's for the data storage on each node which would give us 60TB of usable search space across 15 data nodes. Data storage per node also required knowing the maximum read bandwidth per SSD is 540MB/sec with 90,000 Input/Outputs Per Second (IOPS)/second [35]. We dismissed Hard disk drives as an option due to the required speed to completely scan the data [42].

Reducing storage through intelligent runtime

Further reducing scope involved a move away from graphical representation of motif SNP-PWM. As previously stated to store the graphic files for each affinity scored SNP would have required 3.7 PB of storage. Having taken into consideration cost, we reduced our scope to only using 11% of the total records. Therefore, we were able to reduce the graphical storage to 11% of the 3.7PB or 370TB. 370TB of storage for graphics is cost prohibitive as data storage prices, while always decreasing [43], is still at $369 for a 10TB drive. The cost of storage is well known to be more than just the cost per TB [44] but for simplicity a price per TB of raw storage with no redundancy at $39/TB would still cost $14,430.

Since cost is always an issue and the storage or PNG files for the PWM motifs are computer generated

anyway, we decided against storage of the PWM motifs. Instead we chose to build software to generate the a Scalable Vector Graphic (SVG). The software module utilizes javascript and the d3.js [45] library, which uses the PWM data, for generating the Motif at runtime.

To stay consistent and allow us to render the proper PWM motif for a given SNP-PWM record, we chose to store the PWM motif data with each corresponding record. Adjusting our approach from a centralized storage of PNG motif PWM files to distributed edge computing render model, provided savings. The dynamic motif generation through edge computing in the client web browser's javascript engine allowed us to eliminate all motif image storage. Thus, a data storage savings for the application saved an additional 370TB when using p-value cutoff 0.05 which resulted in a 89% reduction in total storage required.

V. Conclusion

Our use case study has demonstrated the feasibility of using NoSQL database engines for large scale real-time searchable genomic SNP databases. The use of elastics search and the successful implementation is proof that our systems was feasible. Admittedly, our evaluation of database engines is not a conclusive finding. Our goal was to prove database engine feasibility and not be conclusive or exhaustive study case. Factors which impeded a more through database engine study included: cost, personal, speed to implement, and domain knowledge. Additional factors which played a roll in our final implementation included: support-ability, system administrator time to implement, datacenter rackspace and networking.

Our time-frame for implement was limited and numerous unforeseeable issues arose. At first we expected to build a cluster based on 14 nodes as described in Cassandra for Sysadmins slide 6 [46]. However, we realized our quick turn-a-round times to a fully functional web resource was limited by at first range queries, then scalability (MySQL), until we agreed to utilize Elasticsearch through our feasibility testing. Our successful deployment exhibits 11% of the human genome SNP's with PWM. Barring financial resources constraints, we believe our presented solution would continue to scale and fully support the complete 307 billion records.

Our cost structure, based on purchasing and deploying equipment, was validated as a highly competitive cost effective option. For our do it ourselves Elasticsearch infrastructure, our cost saving showed a significant savings. When comparing to Amazon's Elasticsearch implementation, our 2x cost savings proved significant at $0.095 per GB cost savings.

Acknowledgments

This work is supported by:

- NIH Big Data to Knowledge (BD2K) Initiative under Award Number U54 AI117924
- Center for Predictive Computational Phenotyping (CPCP)
- University of Wisconsin - Madison

References

[1] C. Zuo, S. Shin, and S. Kele, "atsnp: transcription factor binding affinity testing for regulatory snp detection," *Bioinformatics*, vol. 31, no. 20, pp. 3353–3355, 2015. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btv328

[2] M. Kreitman, "Nucleotide polymorphism at the alcohol dehydrogenase locus of drosophila melanogaster," *Nature*, vol. 304, pp. 412 EP –, Aug 1983. [Online]. Available: http://dx.doi.org/10.1038/304412a0

[3] I. C. Gray, D. A. Campbell, and N. K. Spurr, "Single nucleotide polymorphisms as tools in human genetics," *Human Molecular Genetics*, vol. 9, no. 16, pp. 2403–2408, 2000. [Online]. Available: +http://dx.doi.org/10.1093/hmg/9.16.2403

[4] D. Altshuler, V. J. Pollara, C. R. Cowles, W. J. Van Etten, J. Baldwin, L. Linton, and E. S. Lander, "An snp map of the human genome generated by reduced representation shotgun sequencing," *Nature*, vol. 407, pp. 513 EP –, Sep 2000. [Online]. Available: http://dx.doi.org/10.1038/35035083

[5] "dbsnp short genetic variations," https://www.ncbi.nlm.nih.gov/SNP/, accessed: 2018-01-03.

[6] "Genome reference consortium human build 38," https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.26/, accessed: 2016-05-05.

[7] G. D. Stormo, T. D. Schneider, L. Gold, and A. Ehrenfeucht, "Use of the 'perceptron' algorithm to distinguish translational initiation sites in e. coli," *Nucleic Acids Research*, vol. 10, no. 9, pp. 2997–3011, 1982. [Online]. Available: +http://dx.doi.org/10.1093/nar/10.9.2997

[8] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2013, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org/

[9] A. Mathelier, X. Zhao, A. W. Zhang, F. Parcy, R. Worsley-Hunt, D. J. Arenillas, S. Buchman, C.-y. Chen, A. Chou, H. Ienasescu, J. Lim, C. Shyr, G. Tan, M. Zhou, B. Lenhard, A. Sandelin, and W. W. Wasserman, "Jaspar 2014: an extensively expanded and updated open-access database of transcription factor binding profiles," *Nucleic Acids Research*, vol. 42, no. D1, pp. D142–D147, 2014. [Online]. Available: +http://dx.doi.org/10.1093/nar/gkt997

[10] T. E. P. Consortium, "An integrated encyclopedia of dna elements in the human genome," *Nature*, vol. 489, no. 7414, pp. 57–74, Sep 2012, 22955616[pmid]. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3439153/

[11] J. Basney and M. Livny, *Deploying a High Throughput Computing Cluster*, R. Buyya, Ed. Prentice Hall PTR, 1999.

[12] S. D. Mooney, V. G. Krishnan, and U. S. Evani, "Bioinformatic tools for identifying disease gene and snp candidates," *Methods Mol Biol*, vol. 628, pp. 307–319, 2010, 20238089[pmid]. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3957484/

[13] M. D. Mailman, M. Feolo, Y. Jin, M. Kimura, K. Tryka, R. Bagoutdinov, L. Hao, A. Kiang, J. Paschall, L. Phan, N. Popova, S. Pretel, L. Ziyabari, Y. Shao, Z. Y. Wang, K. Sirotkin, M. Ward, M. Kholodov, K. Zbicz, J. Beck, M. Kimelman, S. Shevelev, D. Preuss, E. Yaschenko, A. Graeff, J. Ostell, and S. T. Sherry, "The ncbi dbgap database of genotypes and phenotypes," *Nat Genet*, vol. 39, no. 10, pp. 1181–1186, Oct 2007, 17898773[pmid]. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2031016/

[14] S. Song, D. Tian, C. Li, B. Tang, L. Dong, J. Xiao, Y. Bao, W. Zhao, H. He, and Z. Zhang, "Genome variation map: a data repository of genome variations in big data center," *Nucleic Acids Research*, vol. 46, no. D1, pp. D944–D949, 2018. [Online]. Available: +http://dx.doi.org/10.1093/nar/gkx986

[15] "Mysql server," https://www.mysql.com, accessed: 2018-01-28.

[16] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, "Access path selection in a relational database management system," in *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '79. New York, NY, USA: ACM, 1979, pp. 23–34. [Online]. Available: http://doi.acm.org/10.1145/582095.582099

[17] P. P.-S. Chen, "The entity-relationship model—toward a unified view of data," *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9–36, march 1976.

[18] A. S. Foundation, "Cassandra." [Online]. Available: http://cassandra.apache.org/

[19] "7 reasons why netflix uses cassandra databases," https://www.jcount.com/7-reasons-netflix-uses-cassandra-databases/, accessed: 2017-09-20.

[20] P. Voldemort, "Voldemort." [Online]. Available: http://www.project-voldemort.com/voldemort/

[21] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," in *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 06)*. Seattle, WA: USENIX Association, 2006. [Online]. Available: https://www.usenix.org/conference/osdi-06/bigtable-distributed-storage-system-structured-data

[22] "Relational databases are not designed for scale," http://www.marklogic.com/blog/relational-databases-scale/, accessed: 2018-01-15.

[23] G. Vaish, *Getting Started with Nosql*. Packt Publishing, 2013. [Online]. Available: https://books.google.com/books?id=oPiT-V2eYTsC

[24] R. Cattell, "Scalable sql and nosql data stores," *SIGMOD Rec.*, vol. 39, no. 4, pp. 12–27, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1978915.1978919

[25] J. Klein, I. Gorton, N. Ernst, P. Donohoe, K. Pham, and C. Matser, "Performance evaluation of nosql databases: A case study," in *Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems*, ser. PABS '15. New York, NY, USA: ACM, 2015, pp. 5–10. [Online]. Available: http://doi.acm.org/10.1145/2694730.2694731

[26] Y. Li and S. Manoharan, "A performance comparison of sql and nosql databases," in *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Aug 2013, pp. 15–19.

[27] W. Puangsaijai and S. Puntheeranurak, "A comparative study of relational database and key-value database for big data applications," in *2017 International Electrical Engineering Congress (iEECON)*, 2017, pp. 1–4.

[28] "Mariadb server," https://mariadb.com/, accessed: 2018-01-28.

[29] "Redis," https://redis.io/, accessed: 2018-01-15.

[30] R. Hudson, "ss utility scripts." [Online]. Available: https://github.com/RebeccaHudson/ss_utility_scripts

[31] "Apache cassandra nosql performance benchmarks," https://academy.datastax.com/planet-cassandra/nosql-performance-benchmarks, accessed: 2017-01-20.

[32] P. Pirzadeh, J. Tatemura, O. Po, and H. Hacıgümüş, "Performance evaluation of range queries in key value stores," *Journal of Grid Computing*, vol. 10, no. 1, pp. 109–132, Mar 2012. [Online]. Available: https://doi.org/10.1007/s10723-012-9214-7

[33] "Improve mysql insert performance," https://kvz.io/blog/2009/03/31/improve-mysql-insert-performance/, accessed: 2017-06-15.

[34] "Differences between the ndb and innodb storage engines," https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster-ndb-innodb-engines.html, accessed: 2017-06-15.

[35] "Crucial mx300 solid state drive," http://www.crucial.com/usa/en/storage-ssd-mx300, accessed: 2017-06-18.

[36] "Elasticsearch, heap sizing and swapping," https://www.elastic.co/guide/en/elasticsearch/guide/current/heap-sizing.html, accessed: 2017-09-28.

[37] B. Schroeder, E. Pinheiro, and W.-D. Weber, "Dram errors in the wild: A large-scale field study," in *SIGMETRICS*, 2009.

[38] "Crucial mx300 2.5" 1tb sata iii 3d nand internal solid state drive (ssd) ct1050mx300ssd1," https://www.newegg.com/Product/Product.aspx?item=N82E16820156152, accessed: 2017-06-10.

[39] "Crucial mx300 2tb sata 2.5" 7mm (with 9.5mm adapter) internal ssd," http://www.crucial.com/usa/en/ct2050mx300ssd1, accessed: 2017-06-10.

[40] "Commercial and industrial (c&i) electric rates," https://www.mge.com/customer-service/business/elec-rates-comm/, accessed: 2017-06-30.

[41] "Tco for application servers: Comparing linux with windows and solaris," http://www-03.ibm.com/linux/whitepapers/robertFrancesGroupLinuxTCOAnalysis05.pdf, accessed: 2017-06-30.

[42] "Comparing iops for ssds and hdds," http://www.tvtechnology.com/expertise/0003/comparing-iops-for-ssds-and-hdds/276487, accessed: 2017-03-29.

[43] "Hard drive cost per gigabyte," https://www.backblaze.com/blog/hard-drive-cost-per-gigabyte/, accessed: 2017-09-15.

[44] "The true "cost" of enterprise storage - understanding storage management," https://www.zadarastorage.com/blog/industry-insights/cost-of-enterprise-storage-understanding-storage-management/, accessed: 2017-12-29.

[45] Mike Bostock, "D3: Data-driven documents - d3.js." [Online]. Available: https://d3js.org

[46] "Cassandra 101 for sysem administrators," https://www.slideshare.net/nmilford/cassandra-for-sysadmins, accessed: 2017-02-02.

Appendix C

Predicting Hard Disk Drive faults, failures and associated misbehaviors

The following paper was accepted for conference proceeding publication as a workshop paper at IEEE International Parallel and Distributed Processing Symposium (IPDPS) 2023.

Predicting Hard Disk Drive faults, failures and associated misbehavior's

Christopher Harrison
Department of Computer Science
Universidade do Porto
Porto, Portugal
Kairos Technologies
Madison, WI, USA
chris@kairostechnologies.us

Henish Balu
Porto Digital Association
Porto, Portugal
henish.balu@portodigital.pt

Inês Dutra
Department of Computer Science
Universidade do Porto
Porto, Portugal
ines@dcc.fc.up.pt

***Abstract*—Magnetic hard disk drives continue to be heavily used to store global information. However, due to the physical characteristics these components fatigue and fail, sometimes in unexpected ways. A failing hard disk can cause problems to a group of hard disks and result in sub-optimal performance which impacts cloud providers. To address failures, redundancies are put in place, but these redundancies have a high cost. Utilizing Machine learning we identify predictive failure features within a hard disk vendor's Hard Disk Drive Model line which can be used as an early failure prediction method which may be used to reduce redundancies in cloud storage infrastructures.**

***Index Terms*—cloud fault detection, hard disk errors**

I. Introduction

Magnetic Hard Disk Drives (HDD) store 59% of all information produced worldwide [1]. The boom in cloud computing, online services and big data applications, have moved data storage resources away from individual computers into large capacity data centers. Hyperscale computing systems', like the ones used by cloud providers, are increasingly prevalent with 49% of the world's data projected to reside in public clouds and most of this data will be stored on Magnetic HDD[1].

All data center systems and public clouds must provide high quality reliable services for their customers. These customers are highly dependent on the providers' storage systems. Many service providers have Service Level Agreements (SLA), which provide a guarantee of service availability or monetary losses occur. To promise SLA's, providers implement numerous levels of redundancies that insure against service disruption failures and faults. However, Magnetic HDD, like all physically moving components, suffers from component fatigue which degrades performance over time and eventually cause component failures.

Hard drives are reported to be the components that most need to be replaced in storage systems. *HDD* failures cause service delays and sometimes data loss, costing companies millions of dollars per year. According to research [2], the average cost of data center down time is $9,000/minute.

The motivation of our work is to reduce the impact of faults and failure occurring to storage systems so proactive maintenance and avoid fatal impact events. To do so we consider several machine learning methods that predict failures and faults by using data from a common *HDD* telemetry reporting technique known as *S.M.A.R.T.* (self-monitoring, analysis, and reporting technology). Our results show promise for data center operators and cloud providers.

Our contribution partitions the dataset by Hard Disk Drive Models (HDDM), instead of the common all in one bucket approach, to develop Machine Learning Models (MLM) which produce highly accurate MLM for HDDM. Additionally, our work found 4 additional SMART attribute predictors which have not been mentioned previously in the literature. We organized this paper as follows: background and related work, methods and results, conclusions, and future work.

II. Background

Magnetic hard disk drives *HDD* are a critical component for large scale data systems, due to their cost effectiveness. Even though the Cost Per Gigabyte *CPG* of non-magnetic (aka Solid-State Disks) *SSD* continues to fall, *HDD CPG* remain highly economical due to a confluence of factors around the component technologies

employed. Thus, *HDD* continues to be used as a core component in large scale storage systems. As a core component their faults, failures and greatly affect a systems usability and performance[3].

A. Disk Failures

Understanding a failure is dependent on what constitutes the failure. Schroeder *et al.* [4] argues that *HDD* vendors use different definitions of what is a fault than *HDD* costumers. A disk misbehavior may consist in a reading operation that takes longer to execute than usual. For vendors, that value may not be an alarm to be considered because the threshold is not being passed. A disk manufacturer once published that 43% of all disks returned by customers, because they found that the disks had some problem, were considered healthy by the vendors. While drive manufacturers often quote yearly failure rates below 2%, user studies have seen rates as high as 6% [5].

Failures can be categorized in two major groups, predictable and unpredictable [6]. Unpredictable failures, such as electronic and some mechanical problems, occur quickly without any chance of control from the user. For example, a power surge may cause chip or circuit damage on the hard disk. On the other hand, predictable failures are characterized by degradation of an attribute over time. Any mechanical component suffers degradation over their lifetime. Therefore, attributes can be monitored, making it possible for predictive-failure analysis for the user to control the *HDD* components more carefully before they fail [6].

The study made by Schroeder *et al.* [4] also refers that even if the *HDD* is from the same HDDM, they can differ on their behavior, because disks are manufactured using processes and parts that may change. A simple change in a drive's firmware or in a hardware component, or even in the assembly line on which a drive was manufactured, can change the failure behavior of a disk.

According to the Backblaze Company, a disk is considered failed when [7]:

"it is removed from a Storage Pod and replaced because it has 1) totally stopped working, or 2) because it has shown evidence of failing soon. A drive is considered to have stopped working when the drive appears physically dead (e.g., will not power up), do not respond to console commands or the RAID system tells us that the drive cannot be read or written."

B. S.M.A.R.T. Attributes

S.M.A.R.T. emerged from the need to protect critical information stored on disk drives. As system storage capacity requirements increased, the industry identified the importance of creating an early warning system that would allow enough lead time to back up data if failure were imminent, preventing catastrophic data loss [6]. However, these attributes have failed to live up to their intended design goals

S.M.A.R.T. includes a series of attributes, chosen specifically for each drive HDDM. This differentiation is important because *HDD* architectures vary from HDDM to HDDM. Attributes and thresholds that detect failure for one HDDM may not be functional for another HDDM or another vendor. We show important *S.M.A.R.T* attributes in tableI.

The literature has flagged *S.M.A.R.T.* variables 5, 12, 187, 188, 189, 190, 198, 199 and 200 as important for associations with failure events[8] and their descriptions are here in Table I. In this study, pay attention to all variables including those already flagged as important to find new associations.

TABLE I
S.M.A.R.T. ATTRIBUTES

ID	Attribute Name	Description
5	Reallocated Sectors Count	Count of reallocated sectors.
7	Seek Error Rate	Rate of seek errors of the magnetic heads.
9	Power-On Hours	Count of hours in power-on state.
12	Power Cycle Count	A count of full hard disk power on/off cycles.
187	Reported Uncorrectable Errors	The count of errors that could not be recovered using hardware ECC
188	Command Timeout	The count of aborted operations due to HDD timeout.
189	High Fly Writes	This attribute indicates the count of rewritten or reallocated information over the lifetime of the drive.
190	Temperature Difference	Value is equal to (100-temp. °C), allowing manufacturer to set a minimum threshold which corresponds to a maximum temperature.
193	Load Cycle Count	Count of load/unload cycles into head landing zone position.
194	Temperature	Indicates the device temperature.
198	Uncorrectable Sector Count	The total count of uncorrectable errors when reading/writing a sector.
199	UltraDMA CRC Error Count	The count of errors in data transfer via the interface cable as determined by ICRC (Interface Cyclic Redundancy Check).
200	Multi-Zone Error Rate	The count of errors found when writing a sector.
240	Head Flying Hours	Time spent during the positioning of the drive heads.

Numeric attributes that are normalized by the vendors, higher values are always better (except for temperature in some manufactures). The range is usually 0-100 or 0-255. There is no standard on how manufacturers convert the raw values to the normalized ones: it can be a linear, exponential, logarithmic or any other range normalization. That said, it is difficult to quickly perceive the disks behavior on a cloud storage system, since they usually use dozens of different HDDM.

C. Machine Learning

In the development of this work, two machine learning algorithms were used for the classification task such as Random Forests and *SVM* (Support Vector Machine). We chose these methods for their robustness

The classification tasks goal is to obtain an approximate for the unknown function that maps predictor variables toward the target value. The unknown function can be defined as $Y = f(X_1, X_2, ..., X_p)$, where Y is the target variable, $X_1, X_2, ..., X_p$ are features and $f()$ is the unknown function we want to approximate. We approximate the unknown function by using a training dataset $D = \{\langle x_i, y_i \rangle\}_{i=1}^{n}$

III. Related Work

Most existing work about hard drive failure prediction uses the Backblaze dataset, that gathers more than 100 thousand hard disk drives and reports their respective *S.M.A.R.T.* variables daily [8], [9].

Aussel *et al.* [8] say that the existing predictive MLM do not perform sufficiently well on the Backblaze dataset due to the extremely high unbalanced ratio of 5000:1 between healthy and failure disks, and the lack of control on the environment. For that reason, they selected MLM for classification, like *SVM*, Random Forests and Gradient Boosting Trees. They achieved results of 95% precision and 67% recall with the Random Forests MLM and 94% precision and 67% recall with Gradient Boosting Trees. The SVM got a precision below 1%.

Wang *et al.* [9] defend that the reactive fault-tolerant measures, like *RAID*'s (Redundant Array of Inexpensive Drives) and *ECC* (error correction codes) are not enough to mitigate or eliminate the negative effects of the *HDD*'s failures. Proactive measures are more efficient because they will predict failures in advance. However, the built-in prediction MLM that the *HDD*'s manufactures are using, have quite a weak prediction power, with only 4% of failure prediction rate. To overcome the issues and obtain results, they proposed a deep architecture called Amender (for Attention-augMENted Deep architEctuRe) composed of a feature integration layer, a temporal dependency extraction layer, an attention layer, and a classification layer.

After analyzing the results, they concluded that different *S.M.A.R.T.* attributes have different abilities to indicate failures. Compared with Recurrent Neural Networks (RNNs) the architecture proposed improves 8.3% on failure-prediction and 90.2% in the health status assessment. This will also help find the causes of *HDD* failures.

Shen *et al.* [10] propose a Random Forest predicting model capable of differentiating failure prediction for *HDD*'s. They show that most of the statistical approaches, machine learning, and deep learning technologies are good at identifying failures that occur more frequently but perform poorly when they face a less known behavior. A clustering-based under-sampling method is used, so the data imbalance problem is mitigated, and the quality of training set is improved. The results show that the Random Forest model can achieve a *FDR* (Failure Detection Rate) of over 97.67% with a *FAR* (False Alarm Rate) of 0.017%.

Li *et al.* [11] propose two prediction MLM based on Decision Trees and Gradient Boosted Regression Trees in two different real-world datasets (one with 121,698 and other with 39,091 hard disks). In data preparation and pre-processing, they use quantile functions, to select the more key features on healthy and drives that fail.

- Bigger dataset: The Decision Trees model, helps in improving the hard drive failure prediction with a 93% *FDR* and a *FAR* under 0.01%. The Gradient Boosted Regression Trees also contribute to evaluating the health degree level and the results show a 90% *FDR* and a 0% *FAR*.
- Smaller dataset: Both MLM show steady prediction performance, with failure detection rates of 80% to 96% and low false alarm rates of 0.006% to 0.31%.

They also mention an interesting point by using a metric that calculates the expected number of data loss events per petabyte used by year in these companies.

Zhao *et al.* [12] believe that many works done in the area fail to consider the characteristics of the observed features, over time, and tend to make the predictions based on individual or a set of attributes. They also believe that it is reasonable that attribute values observed over time are not independent, and a sequence of observed values with certain patterns may be a good indicator on whether a drive may fail soon. Therefore, they consider the observations from the disks as a time series and apply a Hidden Markov Model and a Hidden

semi-Markov model to build prediction MLMs that could label disks as healthy or pre-failing. Although their *FDR* results are not high (up until 46% for single attributes and 52% for multiple attributes), they achieve a *FAR* of 0% in both cases.

IV. Dataset and Methodology

The data from the fourth quarter of 2019 will be used. 92 datasets from the period of 01-October to 31-December were downloaded from the Backblaze data center. After a brief analysis of all files, 125,731 different disks were identified during the three months.

Of the 125,731 disks, only 678 failed showing a failure rate below 0.54%. This demonstrates a huge disparity between the two classes that categorize the disks. In Figure 1 it is possible to have a better visualization of the data distribution.

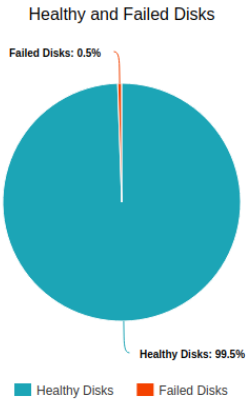


Fig. 1. Diagram of Failed and Healthy Disks

The company has in its storage systems, dozens of different *HDD*'s HDDM and from different vendors too. In Table II, it is possible to see types of disks available on this dataset.

a) : Not all vendors use the same variables, so it is important to note which ones will be kept in the dataset.

A. Cleaning the Data

As the objective is to analyze the dataset in a temporal way, a function was executed to determine the day with most failures. After selecting it, the failed disks were gathered, and the respective observations were collected from day 1 to the selected day. With this, it is possible to obtain a dataset in the form of a time series, to better analyze the behavior of the disks over the days. From now on, the under-sampling method is used to select

TABLE II
A sample of HDDM and their numbers available in the Backblaze Storage dataset.

Types of HDDM	# of Disks
DELLBOSS VD	60
HGST HDS5C4040ALE630	26
HGST HMS5C4040ALE640	2833
HGST HMS5C4040BLE640	12758
HGST HMS5C4040BLE641	1
HGST HUH721010ALE600	20
HGST HUH721212ALE600	1561
HGST HUH721212ALN604	10866
HGST HUH728080ALE600	1002
HGST HUS726040ALE610	28
Hitachi HDS5C4040ALE630	2
ST10000NM0086	1205
ST1000LM024 HN	1
ST12000NM0007	37442
ST12000NM0008	7226
ST12000NM0117	15
ST16000NM001G	40
ST4000DM000	19330
ST4000DM005	39
ST500LM012 HN	501
ST500LM021	33

the disks that were healthy until the day and collect the respective observations over the period. It is important to note that the healthy disks must be from the same HDDM as the disks that failed, so that the comparison would be correctly made, because as stated by Backblaze, different HDDM make use of different variables. Therefore, 2 datasets are created: Healthy Disks and Failed Disks. In Figure 2 it is possible to have a better visualization of this process where blue corresponds to healthy disks and red to the failed disks.

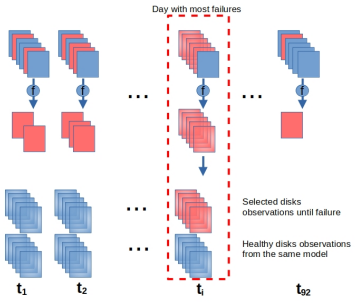


Fig. 2. Pre-Processing Diagram Example.

We remove HDDMs with less than 100 disks and HDDMs that never failed over the 3 months, would also be removed. This measure was taken because the

observations of these disks will never take our class of interest into account, so it is not worth analyzing them.

B. Methodology

A temporal analysis is executed over both datasets (Healthy Disks and Failed Disks) for a better visualization of the oscillations in the variable's values, so the differences between the healthy and failing disks. Also, the Euclidean distances, between the failed disks and the healthy ones, are calculated for every feature. With this, it is possible to obtain a better numerical perception. Both processes will help extracting information from the data and turning the decision making, before applying the learning algorithms, more efficient and accurate.

The temporal analysis is performed by plotting the variables of interest for failing disks and healthy disks. The two graphs will be placed side by side for the comparison to be made.

The Euclidean distance is calculated for each variable. This distance helps understand how dispersed the values are, of the same variables, between a healthy disk and a disk that ends up failing. To better clarify this process, the objective is to compare the smart_1, over time, of a healthy disk, with the smart_1 over time of a disk that will fail. To ensure that the execution of the algorithms is well done, the comparison between the disks is always done with the same HDDM.

To apply a Vector Auto Regression (VAR) model, it was necessary to divide the two datasets created into sub datasets that were grouped by serial_number. Thus, sub datasets would only contain observations over time of a given disk. By this way it is possible to apply the HDDM to each disk and make a comparison between the healthy and the failing ones.

The observations from the last 5 days of each disk were removed so when the forecast was executed, the prediction could be compared with the real values. The drive information that was added by Backblaze is removed, except the Date values, so that the table is only constituted with *S.M.A.R.T.* variables, over time. This measure is taken, because the *VAR* MLM only performs operations on numeric variables and would not extract any information from the variables that were added only to describe the disk (Ex: Serial_number, HDDM).

The first thing to do before the execution of the algorithms is to divide the datasets (Healthy and Failed) into sub datasets once more, but this time, in sub datasets grouped by HDDM. This can be done because there is no need to have a temporal view of the data, so more than 1 disk can be placed on the new sub dataset. Then it is necessary to add the class variable to all observations. The disks that fail will have the class equal to 1 and those that remain healthy will have the class equal to 0. In these algorithms, only the *S.M.A.R.T.* attributes remain in the data frame, the rest of the variables are eliminated for the same reasons referenced in the VAR model.

All features were normalized to values between 0 and 1, since the learning algorithms had difficulties performing the operations on standard values. This normalization was made after the disks were divided by HDDM, so the values range were not mixed up. The validation method utilized was the train-test-split with an 80 to 20 ratio. Both classification algorithms, *SVM* and Random Forest, were executed using default parameters.

V. Classification Algorithms

After all steps were executed, the classification MLM were applied to the sub datasets created (12 data frames distinguished by HDDM).

From Table III the metric results are presented along with the respective Confusion Matrix. The tables present metrics like precision, recall, f1-score, and accuracy. It is also possible to observe each class's support, corresponding to how many observations are labeled for each class. In the confusion matrices the predicted cases for each classification algorithm are presented so it is possible to evaluate the respective performance. All the presented results are obtained from the test set.

TABLE III
METRICS RESULTS FOR HDDM ST12000NM0007

	SVM				RANDOM FOREST			
	precision	recall	f1-score	support	precision	recall	t tps://www.overleaf.com/project/63d7500e20d2c150f0a88e7c—f1-score	support
Healthy	1.00	0.99	0.99	92	0.99	0.99	0.99	92
Failure	0.99	1.00	0.99	83	0.99	0.99	0.99	83
accuracy			0.99	175			0.99	175
macro avg	0.99	0.99	0.99	175	0.99	0.99	0.99	175
weighted avg	0.99	0.99	0.99	175	0.99	0.99	0.99	175

TABLE IV
CONFUSION MATRIX HDDM ST12000NM0007

		SVM Predicted Class		**RANDOM FOREST Predicted Class**	
		Healthy	*Failure*	*Healthy*	*Failure*
Actual	*Healthy*	91	1	91	1
	Failure	0	83	1	82

TABLE V
METRICS RESULTS FOR HDDM ST4000DM000

	SVM				RANDOM FOREST			
	precision	recall	f1-score	support	precision	recall	f1-score	support
Healthy	0.83	1.00	0.91	20	0.95	1.00	0.98	20
Failure	1.00	0.80	0.89	20	1.00	0.95	0.97	20
accuracy			0.90	40			0.97	40
macro avg	0.92	0.90	0.90	40	0.98	0.97	0.97	40
weighted avg	0.92	0.90	0.90	40	0.98	0.97	0.97	40

TABLE VI
CONFUSION MATRIX HDDM ST4000DM000

		SVM Predicted Class		RANDOM FOREST Predicted Class	
		Healthy	*Failure*	*Healthy*	*Failure*
Actual	*Healthy*	20	0	20	0
	Failure	4	16	1	19

TABLE VII
METRICS RESULTS FOR HDDM ST8000NM0055

	SVM				RANDOM FOREST			
	precision	recall	f1-score	support	precision	recall	f1-score	support
Healthy	0.95	1.00	0.98	20	1.00	1.00	1.00	20
Failure	1.00	0.95	0.98	21	1.00	1.00	1.00	21
accuracy			0.98	41			1.00	41
macro avg	0.98	0.98	0.98	41	1.00	1.00	1.00	41
weighted avg	0.98	0.98	0.98	41	1.00	1.00	1.00	41

TABLE VIII
CONFUSION MATRIX HDDM ST8000NM0055

		SVM Predicted Class		RANDOM FOREST Predicted Class	
		Healthy	*Failure*	*Healthy*	*Failure*
Actual	*Healthy*	20	0	20	0
	Failure	1	20	0	21

TABLE IX
METRICS RESULTS FOR HDDM ST12000NM0008

	SVM				RANDOM FOREST			
	precision	recall	f1-score	support	precision	recall	f1-score	support
Healthy	0.40	0.40	0.40	5	0.50	0.40	0.44	5
Failure	0.25	0.25	0.25	4	0.40	0.50	0.44	4
accuracy			0.33	9			0.44	9
macro avg	0.33	0.33	0.33	9	0.45	0.45	0.44	9
weighted avg	0.33	0.33	0.33	9	0.46	0.44	0.44	9

TABLE X
CONFUSION MATRIX HDDM ST12000NM0008

		SVM Predicted Class		RANDOM FOREST Predicted Class	
		Healthy	*Failure*	*Healthy*	*Failure*
Actual	*Healthy*	2	3	2	3
	Failure	3	1	2	2

It is possible to observe in Table III that the ST12000NM0007 HDDM shows metrics values really close to 100%, this can demonstrate that the methodology used may prove to be quite accurate. It is important to note that the ST12000NM0008, TOSHIBA MQ01ABF050 and TOSHIBA MG07ACA14TA HDDM, do not have a favorable support (exceptionally sparse number of observations and past behaviors information) for the algorithms execution, and therefore their results are not the most promising. In the future work section, some points that could improve these results are discussed.

It is essential to have a perception of the importance that the Random Forest model gives to variables in its decision making and in the tree's creation. With this, it is easier to understand which features weight more in helping the algorithm to predict if the disk will fail or remain healthy. Table XIII shows the importance ranking given to the 6 different HDDM.

TABLE XI
METRICS RESULTS FOR HDDM TOSHIBA MQ01ABF050

	SVM				RANDOM FOREST			
	precision	recall	f1-score	support	precision	recall	f1-score	support
Healthy	0.80	0.80	0.80	10	0.91	1.00	0.95	10
Failure	0.82	0.82	0.82	11	1.00	0.91	0.95	11
accuracy			0.81	21			0.95	21
macro avg	0.81	0.81	0.81	21	0.95	0.95	0.95	21
weighted avg	0.81	0.81	0.81	21	0.96	0.95	0.95	21

TABLE XII
CONFUSION MATRIX HDDM TOSHIBA MQ01ABF050

		SVM Predicted Class		RANDOM FOREST Predicted Class	
		Healthy	*Failure*	*Healthy*	*Failure*
Actual	*Healthy*	8	2	10	0
	Failure	2	9	1	10

TABLE XIII
RANDOM FOREST FEATURES IMPORTANCE FOR EACH HDDM

ST12000NM0007	importance
smart_7_normalized	0.228980
smart_193_raw	0.187561
smart_3_normalized	0.165387
smart_9_normalized	0.058750
smart_9_raw	0.057293
smart_241_raw	0.051673
smart_7_raw	0.041517
smart_240_raw	0.032224
smart_12_raw	0.026624
smart_242_raw	0.026087

ST4000DM000	importance
smart_193_raw	0.156771
smart_183_raw	0.127605
smart_3_normalized	0.099825
smart_190_normalized	0.083486
smart_183_normalized	0.080752
smart_194_raw	0.063729
smart_194_normalized	0.061899
smart_190_raw	0.056995
smart_7_normalized	0.046583
smart_240_raw	0.045435

ST8000NM0055	importance
smart_195_normalized	0.207715
smart_1_normalized	0.182538
smart_193_normalized	0.126374
smart_193_raw	0.075515
smart_191_raw	0.066237
smart_7_normalized	0.060627
smart_191_normalized	0.053196
smart_192_raw	0.043330
smart_190_raw	0.022044
smart_194_normalized	0.021385

ST12000NM0008	importance
smart_190_raw	0.165062
smart_194_normalized	0.146380
smart_190_normalized	0.139104
smart_194_raw	0.130639
smart_192_raw	0.061040
smart_1_raw	0.059038
smart_1_normalized	0.050796
smart_240_raw	0.038349
smart_7_raw	0.037081
smart_9_raw	0.036924

TOSHIBA MQ01ABF050	importance
smart_191_raw	0.400617
smart_194_raw	0.286671
smart_9_raw	0.134172
smart_222_raw	0.125146
smart_222_normalized	0.028102
smart_9_normalized	0.025292

TOSHIBA MG07ACA14TA	importance
smart_226_raw	0.278620
smart_222_raw	0.193245
smart_9_raw	0.181570
smart_220_raw	0.145886
smart_194_raw	0.104060
smart_193_raw	0.096619

S.M.A.R.T. Variables, like smart_7, smart_9, and smart_193 (described in Table I) have a high importance in almost all disk MLM, so monitoring them more often, is probably a good approach in the future. As mentioned, the *S.M.A.R.T.* variables are considered critical by the literature, are the ones that the storage systems follow the most. So giving more attention to these features could also help preventing some misbehavior.

Figure 3 shows one of the trees created after the Random Forest algorithm was executed in the ST12000NM0007 dataset. As is normal, the variables presented in the Decision Tree are shown in the importance table, showing that the algorithm uses them to classify the observations. In Table XIV the variables presented on the Decision Tree are described. It is important to note that this description was made after the variables were already normalized between 0 and 1.

TABLE XIV
DECISION TREE VARIABLES DESCRIPTION

	smart_7_normalized	smart_7_raw	smart_9_normalized	smart_9_raw	smart_241_raw
count	699.000000	699.000000	699.000000	699.000000	699.000000
mean	0.705797	0.476673	0.270684	0.739938	0.769213
std	0.184539	0.306533	0.301359	0.301370	0.266498
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.600000	0.178866	0.100000	0.737158	0.804327
50%	0.685714	0.476555	0.150000	0.847509	0.853286
75%	0.900000	0.772911	0.263158	0.939120	0.925315
max	1.000000	1.000000	1.000000	1.000000	1.000000

VI. VAR MODEL

As stated in the methodology, 32 subdatasets were created to apply this time series MLM, all of them grouped by serial_number, so the algorithm could be applied individually and could calculate the respective correlation matrices and a possible prediction of the variables behavior over time.

We show correlation matrices for 4 different disks HDDM (one healthy and one failed) starting with HDDM: ST12000NM0007 in Tables XV and XVI, HDDM ST4000DM000 in Tables XVII and XVIII, HDDM ST8000NM0055 in Tables XIX and XX, lastly HDDM ST12000NM0008 in Tables XXI and XXII.

TABLE XV
CORRELATION MATRIX FOR FAILED DISK FROM HDDM ST12000NM0007

Failed Disk ST12000NM0007 (ZCH0C5JJ)	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	0.253239	0.201611	-0.120629	-0.190959	0.187426
smart_7_raw	0.253239	1.000000	**0.852643**	**0.696433**	0.055252	**0.849855**
smart_9_raw	0.201611	**0.852643**	1.000000	**0.923364**	0.039857	**0.998544**
smart_193_raw	-0.120629	**0.696433**	**0.923364**	1.000000	-0.040797	**0.920185**
smart_194_raw	-0.190959	0.055252	0.039857	-0.040797	1.000000	0.068931
smart_240_raw	0.187426	**0.849855**	**0.998544**	**0.920185**	0.068931	1.000000

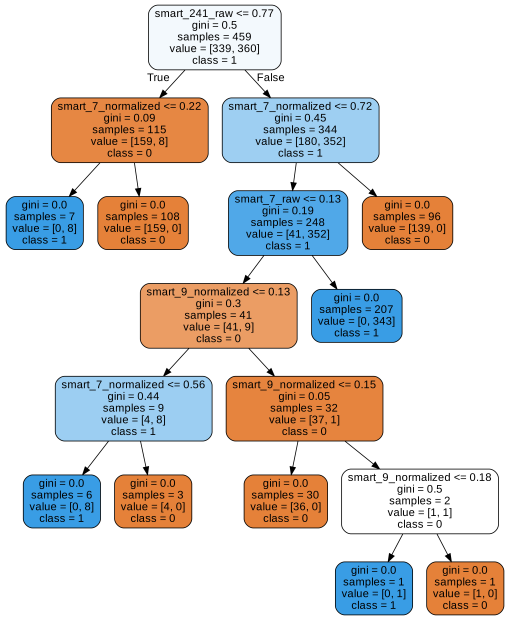


Fig. 3. A Decision Tree from the Random Forest of the ST12000NM0007 HDDM

TABLE XVI
CORRELATION MATRIX FOR HEALTHY DISK FROM HDDM ST12000NM0007

Healthy Disk ST12000NM0007 (ZCH06YQ3)	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	0.166218	-0.085673	-0.141543	0.260900	-0.001269
smart_7_raw	0.166218	1.000000	0.351067	-0.135600	-0.120242	0.392243
smart_9_raw	-0.085673	0.351067	1.000000	0.254303	-0.457013	**0.992656**
smart_193_raw	-0.141543	-0.135600	0.254303	1.000000	-0.546709	0.235824
smart_194_raw	0.260900	-0.120242	-0.457013	-0.546709	1.000000	-0.406834
smart_240_raw	-0.001269	0.392243	**0.992656**	0.235824	-0.406834	1.000000

TABLE XVII
CORRELATION MATRIX FOR FAILED DISK FROM HDDM ST4000DM000

Failed Disk ST4000DM000 (Z302T88S)	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	0.557678	0.641922	0.054301	0.027109	0.612161
smart_7_raw	0.557678	1.000000	**0.933525**	-0.456721	0.200886	**0.942728**
smart_9_raw	0.641922	**0.933525**	1.000000	-0.262591	0.029598	**0.998036**
smart_193_raw	0.054301	-0.456721	-0.262591	1.000000	-0.332683	-0.271897
smart_194_raw	0.027109	0.200886	0.029598	-0.332683	1.000000	0.036335
smart_240_raw	0.612161	**0.942728**	**0.998036**	-0.271897	0.036335	1.000000

TABLE XVIII
CORRELATION MATRIX FOR HEALTHY DISK FROM HDDM ST4000DM000

Healthy Disk ST4000DM000 (Z302DJZ6)	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	-0.105579	-0.208478	-0.153385	**0.724605**	-0.208246
smart_7_raw	-0.105579	1.000000	0.458922	-0.102037	-0.192726	0.471757
smart_9_raw	-0.208478	0.458922	1.000000	0.569640	-0.315389	**0.998916**
smart_193_raw	-0.153385	-0.102037	0.569640	1.000000	-0.432104	0.559510
smart_194_raw	**0.724605**	-0.192726	-0.315389	-0.432104	1.000000	-0.336655
smart_240_raw	-0.208246	0.471757	**0.998916**	0.559510	-0.336655	1.000000

These correlations are be important, so the features

TABLE XIX
CORRELATION MATRIX FOR FAILED DISK FROM HDDM ST8000NM0055

Failed Disk ST8000NM0055 (ZA1819DM)	*smart_1_raw*	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	0.257158	-0.061240	-0.286411	-0.268057	-0.022319
smart_7_raw	0.257158	1.000000	**0.830599**	-0.177065	-0.081025	**0.844100**
smart_9_raw	-0.061240	**0.830599**	1.000000	0.161528	0.070777	**0.995699**
smart_193_raw	-0.286411	-0.177065	0.161528	1.000000	0.370282	0.189769
smart_194_raw	-0.268057	-0.081025	0.070777	0.370282	1.000000	0.100202
smart_240_raw	-0.022319	**0.844100**	**0.995699**	0.189769	0.100202	1.000000

TABLE XX
CORRELATION MATRIX FOR HEALTHY DISK FROM HDDM ST8000NM0055

Healthy Disk ST8000NM0055 (ZA18BTFV)	*smart_1_raw*	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	-0.455708	-0.264859	0.321201	0.271168	-0.249651
smart_7_raw	-0.455708	1.000000	**0.918986**	0.235787	-0.325022	**0.921027**
smart_9_raw	-0.264859	**0.918986**	1.000000	0.528043	-0.273634	**0.999511**
smart_193_raw	0.321201	0.235787	0.528043	1.000000	-0.302633	0.529167
smart_194_raw	0.271168	-0.325022	-0.273634	-0.302633	1.000000	-0.267527
smart_240_raw	-0.249651	**0.921027**	**0.999511**	0.529167	-0.267527	1.000000

TABLE XXI
CORRELATION MATRIX FOR FAILED DISK FROM MODEL ST12000NM0008

Failed Disk ST12000NM0008 (ZHZ3MSH6)	*smart_1_raw*	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	0.061050	0.395718	**0.741040**	**0.703804**	0.072431
smart_7_raw	0.061050	1.000000	**0.922226**	-0.374584	-0.446844	**0.961005**
smart_9_raw	0.395718	**0.922226**	1.000000	-0.097896	-0.156114	**0.927476**
smart_193_raw	**0.741040**	-0.374584	-0.097896	1.000000	**0.863018**	-0.457007
smart_194_raw	**0.703804**	-0.446844	-0.156114	**0.863018**	1.000000	-0.442773
smart_240_raw	0.072431	**0.961005**	**0.927476**	-0.457007	-0.442773	1.000000

TABLE XXII
CORRELATION MATRIX FOR HEALTHY DISK FROM HDDM ST12000NM0008

Healthy Disk ST12000NM0008 (ZHZ3PT1S)	*smart_1_raw*	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	-0.106104	-0.175308	-0.074616	0.041358	-0.206763
smart_7_raw	-0.106104	1.000000	**0.976768**	-0.217435	0.040426	**0.982719**
smart_9_raw	-0.175308	**0.976768**	1.000000	-0.054606	0.107858	**0.960382**
smart_193_raw	-0.074616	-0.217435	-0.054606	1.000000	0.578504	-0.306347
smart_194_raw	0.041358	0.040426	0.107858	0.578504	1.000000	-0.096563
smart_240_raw	-0.206763	**0.982719**	**0.960382**	-0.306347	-0.096563	1.000000

are monitored together and not just those that present critical values because their thresholds were exceeded. Variables 9 and 240 show in all disks (healthy and failed), correlations close to 1. This is happening because both are hour counters, with 9 being the number of hours in power-state and 240 the time during the positioning of the drive heads.

It is possible to verify that higher correlations between variables (9-193) and (240-193) also happen more frequently in the disks that fail from ST12000NM0007 HDDM. The high correlation between these variables may alert us that a more careful observation of the disks should be made. However, there are also healthy disks that show high correlations between these variables, but there is no guarantee that the disk will remain healthy in the future, so these disks may even already show some type of anomaly.

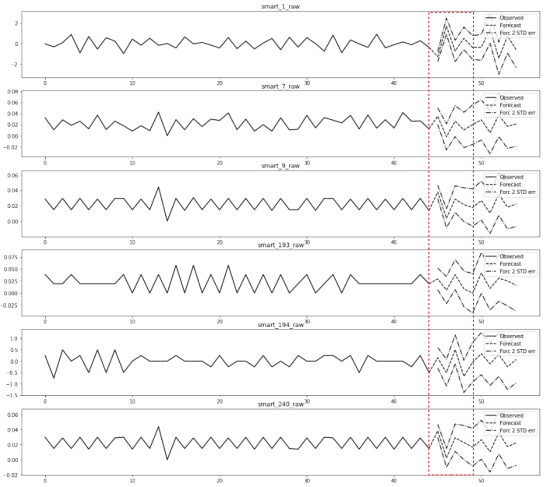


Fig. 4. Forecast for the first healthy disk with a FPE (failure prediction error) of $4x10^{-22}$

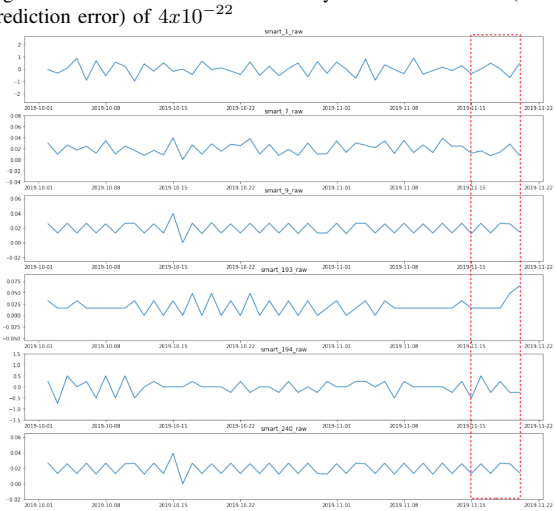


Fig. 5. Real Values for the first healthy disk

Even though there are not enough disks to draw a strong conclusion, the ST4000DM000 HDDM shows correlations between the variables (9-7) and (240-7) for the disks that fail, and the ST12000NM0008 HDDM with correlations in the variables (1-193), (1-194) and (193-194) also in the failing disks.

Finally, the forecasting was done to predict the disks behavior over the time. In Figure 4 , inside the red dashed line, a 5-day forecast can be observed, for each variable, from a disk that fails, and a disk considered healthy. In Figure 5 inside the dashed line, it is possible to see how the disks performed in the last 5 days before being selected. The X axis represents a time scale, with

daily periodicity and the Y axis represents the values for each variable. It is important to notice that these 5 days were removed from the datasets at the beginning of the learning HDDM, so now that they could be compared with the respective forecasted values.

VII. Conclusions and Future work

In short, satisfactory results were obtained, and the predicted results are close to the real values.

Working with imbalanced data reduces the effectiveness of prediction MLM. Because of this, it was necessary to take an overly cautious approach to the data, so little information about the disks that could be useful, was not lost during the process.

One of the biggest beliefs in this project was that the pre-processing and statistical analysis methods used to create the sub datasets were fundamental in the learning process of the data. All studies made on the subject analyze the disks together, without splitting them by HDDM and vendors. This means that the variables standard values, the thresholds and even the features normalization process are not distinguished between them.

Although, most sub datasets created during the project, did not have the ideal number of observations, the metrics values for the classification MLM are quite promising, showing values really close to 100%, for the precision, recall and f-score of certain disk HDDM.

The *VAR* MLM application, allowed to trace temporal correlation matrices between the features, showing that variables 9, 240 and 193 are related over time. The forecasting performed fulfilled the expectation, showing a low forecasting error and may be an interesting method to predict the variables behavior in storage companies.

It should also be noticed that variables 7, 9 and 240 are present quite often in the results, and therefore, they must also be monitored carefully, together with the ones that are critical by the literature.

During the project, some obstacles appeared and had to be overcome. Working with such a large and imbalanced dataset was undoubtedly a great challenge and helped to clarify the reality that in data science, the work is done with data that is not perfect to apply learning MLM. The lack of perception about the variables, the difference between vendors references values and the number of missing values presented in the dataset, made the decision making difficult, and many times, some methods had to be redone from scratch. Since the *VAR* is an algorithm that works with mathematical matrices, it proved to be a MLM with high complexity and that requires a lot of attention in the type of data that is used.

A. Future work

Although the objectives have been achieved, we believe that some measures can be taken to improve the results obtained and their reliability in future works:

- If a bigger time scale is used, the disk information will also be bigger, and the learning MLM will prove to show a greater performance and better results for all disk HDDM. This is because the processes executed by the algorithms will have a greater support and show that the methodology used in this work can also be used in datasets with a higher number of observations.
- A further study and analysis on the variable's normalization should be done, to improve the state of the art and help future works to better understand the data.
- Use, in parallel with the methodology carried out in this project, a class variable that defines the disks lifetime. This classification can be done through variable 9, that counts the number of hours that the *HDD* was powered on.
- Apply the MLMs built in this work to more recent Backblaze disks observations.

References

[1] D. Reinsel, J. Gantz, and J. Rydning, "Data age 2025: the digitization of the world from edge to core," *Seagate*, 2018.

[2] P. Institute, "Cost of data center outages," Traverse City, Michigan 49686, USA, 2016.

[3] S. Antomarioni, F. E. Ciarapica, and M. Bevilacqua, "Data-driven approach to predict the sequence of component failures: a framework and a case study on a process industry," *International Journal of Quality & Reliability Management*, 2022.

[4] S. Bianca and A. G. Garth, "Understanding disk failure rates: What does an mttf of 1000000 hours mean to you?" *Trans. Storage*, vol. 3, no. 3, p. 8, 2007.

[5] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *Proceedings of the Conference on File and Storage Technologies (FAST)*, 2007.

[6] S. P. Marketing, "Get smart for reliability," Technical report, Seagate Technology Paper, Tech. Rep., 1999.

[7] B. Beach, "Hard drive smart stats," Sep 2020. [Online]. Available: https://www.backblaze.com/blog/hard-drive-smart-stats/

[8] N. Aussel, S. Jaulin, G. Gandon, Y. Petetin, E. Fazli, and S. Chabridon, "Predictive models of hard drive failures based on operational data," in *ICMLA 2017 : 16th IEEE International Conference On Machine Learning And Applications*. Cancun, Mexico: IEEE Computer Society, 2017, pp. 619 – 625. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01703140

[9] J. Wang, W. Bao, L. Zheng, X. Zhu, and P. S. Yu, "An attention-augmented deep architecture for hard drive status monitoring in large-scale storage systems," *ACM Trans. Storage*, vol. 15, no. 3, Aug. 2019. [Online]. Available: https://doi.org/10.1145/3340290

[10] J. Shen, J. Wan, S.-J. Lim, and L. Yu, "Random-forest-based failure prediction for hard disk drives," *International Journal of Distributed Sensor Networks*, vol. 14, no. 11, p. 1550147718806480, 2018. [Online]. Available: https://doi.org/10.1177/1550147718806480

[11] J. Li, R. J. Stones, G. Wang, X. Liu, Z. Li, and M. Xu, "Hard drive failure prediction using decision trees," *Reliability Engineering & System Safety*, vol. 164, pp. 55–65, 2017.

[12] Y. Zhao, X. Liu, S. Gan, and W. Zheng, "Predicting disk failures with hmm-and hsmm-based approaches," in *Advances in Data Mining. Applications and Theoretical Aspects: 10th Industrial Conference, ICDM 2010, Berlin, Germany, July 12-14, 2010. Proceedings 10*, vol. 6171. Springer, 2010, pp. 390–404.

Appendix D

Hard Disk failure prediction tables and results

We have included our full HDD dataset by vendor and model to showcase our work in HDD failure prediction. The following graphs are additional charts of all our vendors and models we ran our MLM on.

TABLE D.1: Failed Disks Dataset Description

Disk	Stat	Smart_1	Smart_3	Smart_7	Smart_9	Smart_193	Smart_194
Disk1	count	46.0			46.0		46.0
	mean	79.6			80.6		37.1
	std	4.2			0.5		0.9
	min	67.0			80.0		35.0
	max	84.0			81.0		39.0
Disk2	count	46.0		46.0	46.0		46.0
	mean	79.6		83.1	90.3		23.9
	std	4.7		0.8	0.5		0.8
	min	67.0		82.0	90.0		22.0
	max	84.0		84.0	91.0		25.0
Disk3	count	46.0		46.0	46.0		46.0
	mean	80.4		87.3	87.5		24.4
	std	5.7		0.5	0.5		3.7
	min	69.0		87.0	87.0		20.0
	max	100.0		88.0	88.0		30.0

TABLE D.1: Failed Disks Dataset Description (cont)

Disk	Stat	Smart_1	Smart_3	Smart_7	Smart_9	Smart_193	Smart_194
Disk4	count	46.0		46.0	46.0		46.0
	mean	79.2		89.5	80.6		35.3
	std	4.4		0.5	0.5		1.1
	min	64.0		89.0	80.0		33.0
	max	84.0		90.0	81.0		38.0
Disk5	count	45.0		45.0	45.0		45.0
	mean	80.3		80.6	83.5		21.9
	std	3.8		8.5	0.5		1.0
	min	65.0		60.0	83.0		20.0
	max	84.0		90.0	84.0		24.0
Disk6	count	46.0	46.0	46.0	46.0		46.0
	mean	79.5	96.4	78.7	99.5		27.8
	std	4.4	1.5	4.6	0.5		0.8
	min	68.0	95.0	63.0	99.0		26.0
	max	84.0	98.0	83.0	100.0		29.0
Disk7	count	46.0		46.0	46.0		46.0
	mean	79.8		87.8	82.6		30.0
	std	3.9		0.4	0.5		0.9
	min	68.0		87.0	82.0		28.0
	max	84.0		88.0	83.0		32.0
Disk8	count	45.0		45.0	45.0		45.0
	mean	79.4		79.9	82.6		25.4
	std	5.6		1.7	0.5		0.7
	min	67.0		76.0	82.0		24.0
	max	100.0		82.0	83.0		27.0
Disk9	count	46.0		46.0	46.0		46.0
	mean	79.3		78.7	99.6		22.8
	std	4.4		4.8	0.5		1.1
	min	69.0		63.0	99.0		21.0
	max	84.0		83.0	100.0		25.0
Disk10	count	45.0		45.0	45.0		45.0

TABLE D.1: Failed Disks Dataset Description (cont)

Disk	Stat	Smart_1	Smart_3	Smart_7	Smart_9	Smart_193	Smart_194
	mean	115.2		74.8	53.4		17.1
	std	4.0		4.9	0.5		0.4
	min	102.0		63.0	53.0		16.0
	max	120.0		90.0	54.0		18.0
Disk11	count	45.0			45.0		45.0
	mean	114.4			63.2		33.6
	std	4.8			0.4		1.0
	min	102.0			62.0		32.0
	max	120.0			64.0		35.0
Disk12	count	45.0		45.0	45.0	45.0	45.0
	mean	80.1		88.3	78.7	88.3	33.6
	std	3.5		0.7	0.4	0.4	0.6
	min	69.0		87.0	78.0	88.0	33.0
	max	84.0		89.0	79.0	89.0	35.0
Disk13	count	46.0		46.0	46.0	46.0	46.0
	mean	79.8		90.7	77.5	96.2	36.0
	std	3.7		0.5	0.5	0.4	0.9
	min	70.0		90.0	77.0	96.0	35.0
	max	84.0		91.0	78.0	97.0	38.0
Disk14	count	17.0		17.0			17.0
	mean	80.4		71.6			30.6
	std	3.4		5.3			2.4
	min	73.0		63.0			25.0
	max	83.0		77.0			32.0
Disk15	count				17.0		
	mean				100.0		
	std				0.0		
	min				100.0		
	max				100.0		

TABLE D.2: Healthy Disks Dataset Description

Disk	Stat	Smart_1	Smart_3	Smart_7	Smart_9	Smart_193	Smart_194
Disk1	count	45.0	45.0		45.0		45.0
	mean	80.4	88.8		85.9		26.9
	std	4.7	1.0		0.5		1.2
	min	65.0	88.0		85.0		25.0
	max	100.0	90.0		87.0		29.0
Disk2	count	45.0	45.0	45.0	45.0		45.0
	mean	80.2	91.3	80.8	82.9		26.5
	std	3.9	1.4	1.3	0.4		1.0
	min	65.0	90.0	78.0	82.0		22.0
	max	84.0	93.0	83.0	84.0		28.0
Disk3	count	45.0		45.0	45.0		45.0
	mean	81.3		84.4	82.9		37.7
	std	5.0		1.1	0.5		1.3
	min	72.0		83.0	82.0		35.0
	max	100.0		86.0	84.0		40.0
Disk4	count	46.0		46.0	46.0		46.0
	mean	81.7		89.1	85.2		36.5
	std	4.9		0.3	0.4		1.1
	min	73.0		89.0	85.0		35.0
	max	100.0		90.0	86.0		39.0
Disk5	count	46.0	46.0	46.0	46.0		46.0
	mean	79.9	89.6	88.3	82.6		30.6
	std	3.5	0.5	0.5	0.5		1.5
	min	68.0	89.0	88.0	82.0		28.0
	max	84.0	90.0	89.0	83.0		33.0
Disk6	count	45.0		45.0	45.0		45.0
	mean	80.0		86.2	82.6		26.4
	std	4.0		0.6	0.5		1.3
	min	67.0		85.0	82.0		24.0
	max	84.0		87.0	83.0		29.0
Disk7	count	45.0	45.0	45.0	45.0		45.0

TABLE D.2: Healthy Disks Dataset Description (cont)

Disk	Stat	Smart_1	Smart_3	Smart_7	Smart_9	Smart_193	Smart_194
	mean	79.4	89.8	85.8	82.3		30.7
	std	3.8	0.4	0.6	0.5		0.8
	min	69.0	89.0	85.0	82.0		29.0
	max	84.0	90.0	87.0	83.0		32.0
Disk8	count	46.0	46.0	46.0	46.0		46.0
	mean	79.4	89.6	88.2	85.2		32.8
	std	3.6	0.9	0.4	0.4		2.6
	min	68.0	89.0	88.0	85.0		27.0
	max	84.0	91.0	89.0	86.0		37.0
Disk9	count	10.0	10.0	10.0			10.0
	mean	80.5	97.8	71.8			23.3
	std	1.8	1.8	10.1			0.5
	min	78.0	93.0	65.0			23.0
	max	83.0	99.0	100.0			24.0
Disk10	count	45.0		45.0	45.0		45.0
	mean	115.3		74.1	59.3		28.5
	std	4.3		3.9	0.5		1.7
	min	102.0		62.0	59.0		26.0
	max	120.0		78.0	60.0		33.0
Disk11	count	45.0		45.0	45.0		45.0
	mean	115.9		88.7	60.6		21.6
	std	3.3		0.4	0.5		0.5
	min	108.0		88.0	60.0		20.0
	max	120.0		89.0	61.0		22.0
Disk12	count	46.0		46.0	46.0	46.0	46.0
	mean	80.8		92.1	78.3	95.7	46.3
	std	2.5		0.3	0.5	0.5	0.8
	min	76.0		92.0	78.0	95.0	45.0
	max	84.0		93.0	79.0	96.0	48.0
Disk13	count	46.0		46.0	46.0		46.0
	mean	80.2		92.4	78.3		39.4

TABLE D.2: Healthy Disks Dataset Description (cont)

Disk	Stat	Smart_1	Smart_3	Smart_7	Smart_9	Smart_193	Smart_194
	std	3.4		0.5	0.5		0.8
	min	71.0		92.0	78.0		38.0
	max	84.0		93.0	79.0		41.0
Disk14	count	17.0		17.0			17.0
	mean	80.2		71.8			24.6
	std	3.6		5.1			1.5
	min	74.0		63.0			21.0
	max	84.0		77.0			26.0
Disk15	count				17.0		
	mean				100.0		
	std				0.0		
	min				100.0		
	max				100.0		

Temporal Analysis

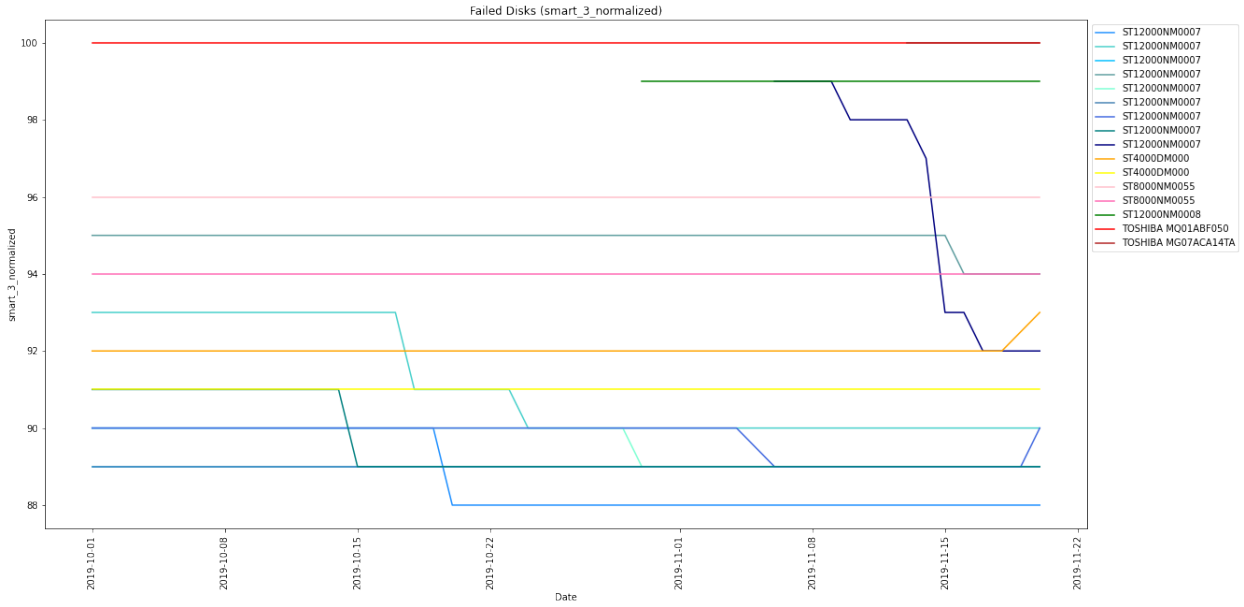


FIGURE D.1: Smart_3 for the failed disks along time

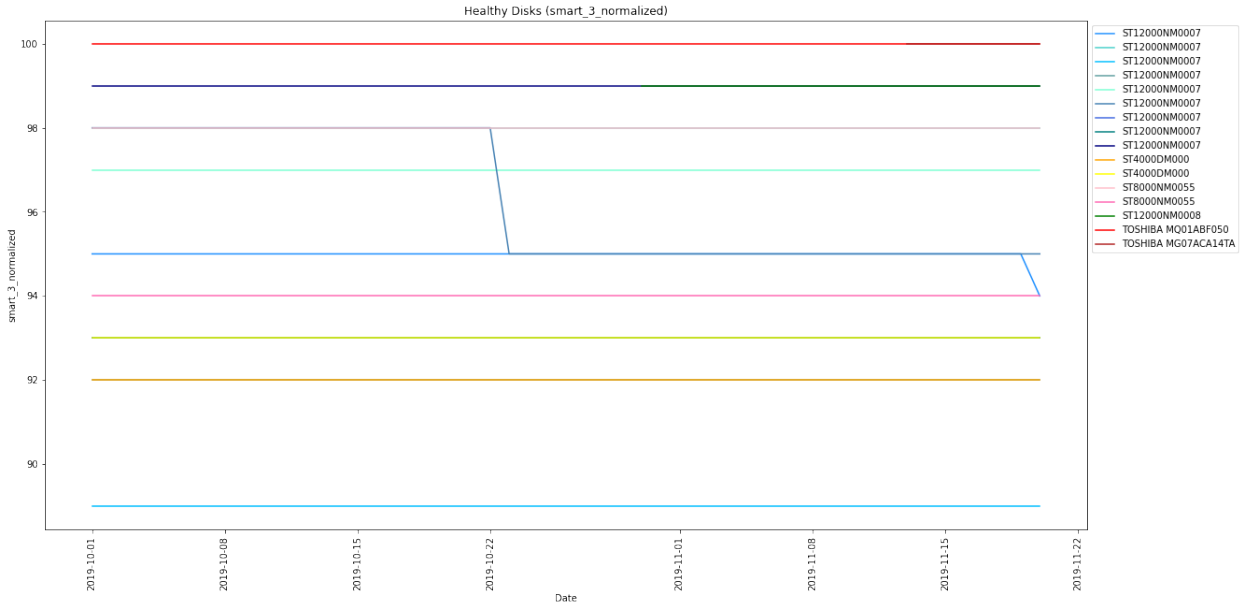


FIGURE D.2: Smart_3 for the healthy disks along time

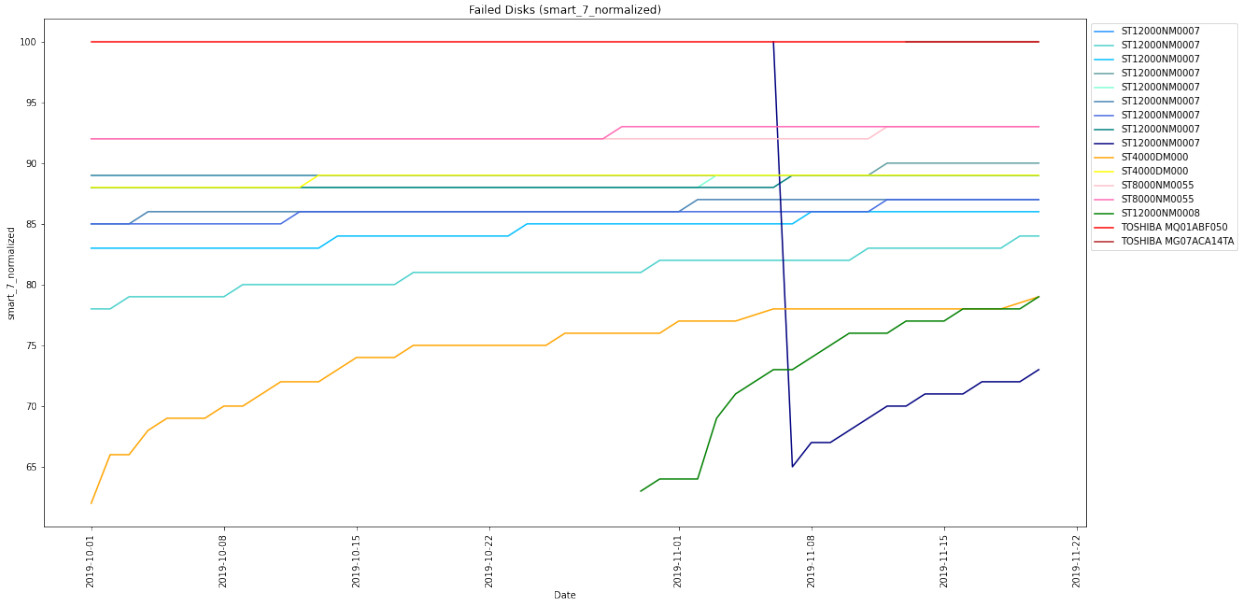


FIGURE D.3: Smart_7 for the failed disks along time

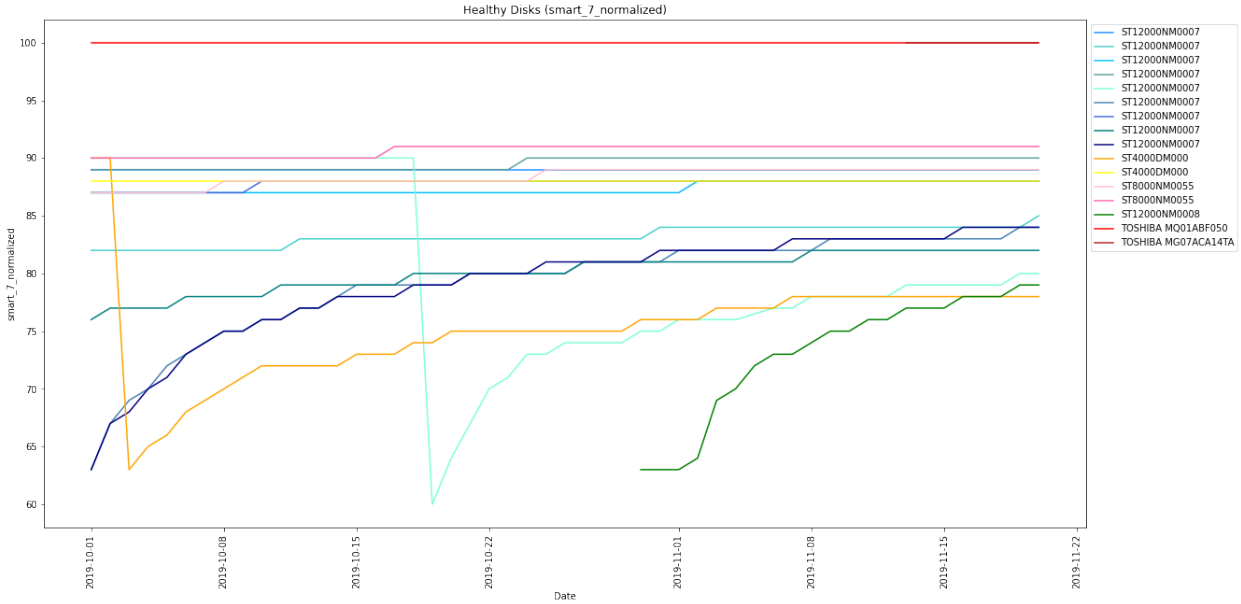


FIGURE D.4: Smart_7 for the healthy disks along time

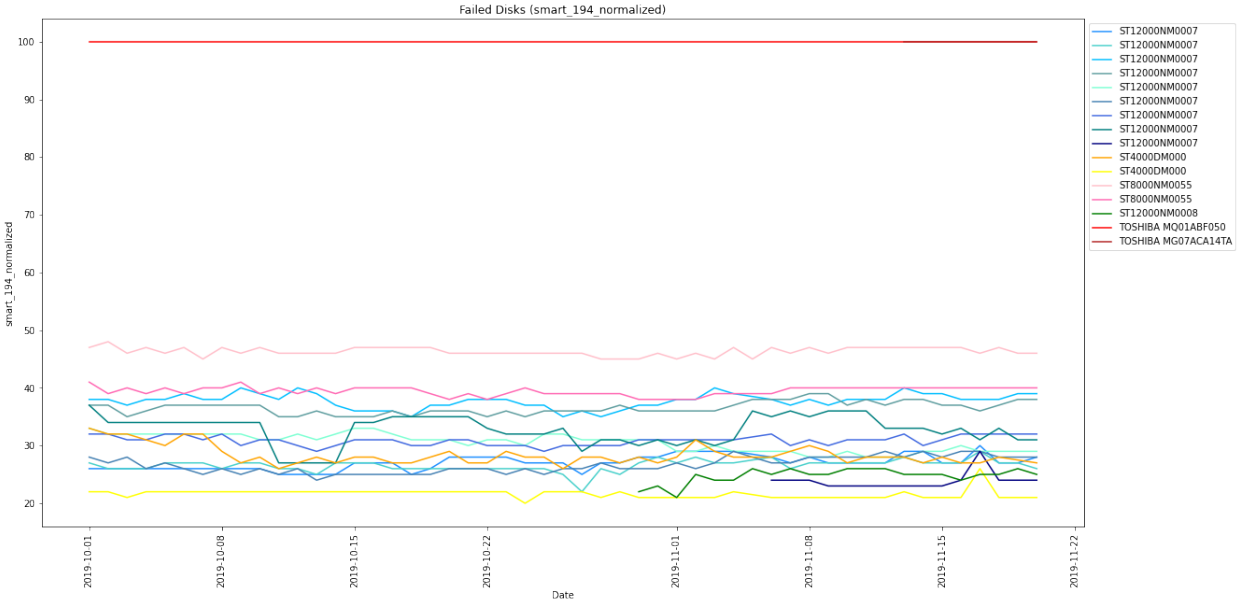


FIGURE D.5: Smart_194 for the failed disks along time

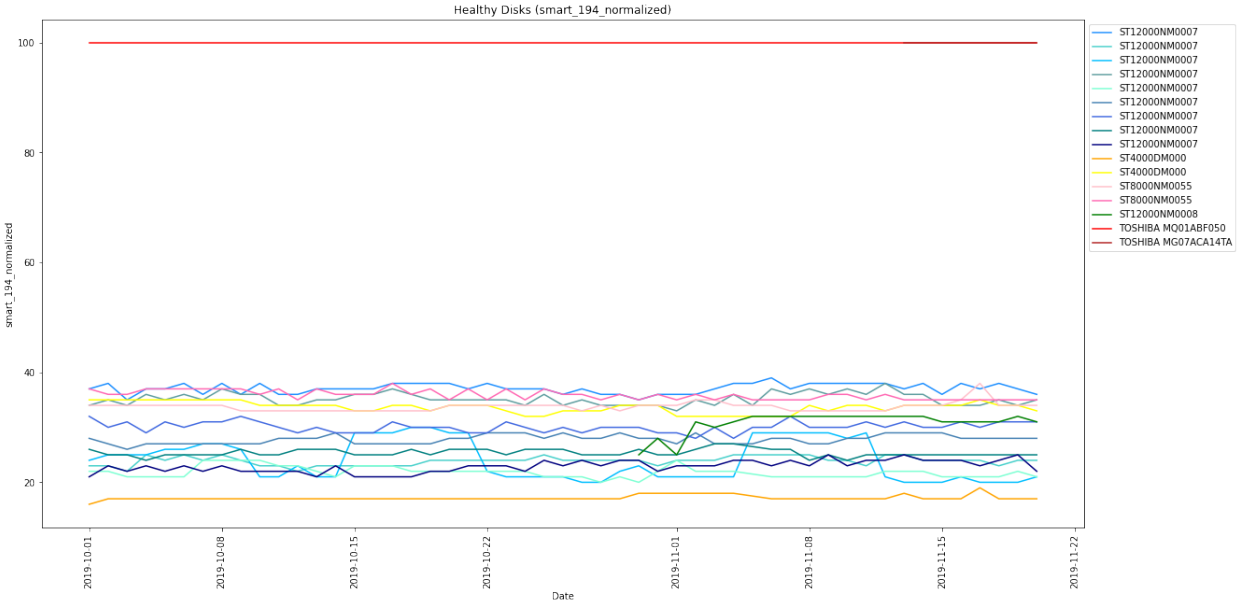


FIGURE D.6: Smart_194 for the healthy disks along time

Correlation Matrices

TABLE D.3: Correlation Matrix for failed disk from model ST12000NM0007 - ZCH0A7G6

Failed Disk ST12000NM0007 Serial # ZCH0A7G6	**smart_1_raw**	**smart_7_raw**	**smart_9_raw**	**smart_193_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	0.326879	0.470757	0.396524	0.429924	0.456820
smart_7_raw	0.326879	1.000000	0.539748	0.128281	0.585176	0.519329
smart_9_raw	0.470757	0.539748	1.000000	-0.020046	0.266706	**0.997736**
smart_193_raw	0.396524	0.128281	-0.020046	1.000000	0.335702	-0.079894
smart_194_raw	0.429924	0.585176	0.266706	0.335702	1.000000	0.228991
smart_240_raw	0.456820	0.519329	**0.997736**	-0.079894	0.228991	1.000000

TABLE D.4: Correlation Matrix for failed disk from model ST12000NM0007 - ZCH0A7G6

Failed Disk ST12000NM0007 Serial # ZCH0A7G6	**smart_1_raw**	**smart_7_raw**	**smart_9_raw**	**smart_193_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	0.326879	0.470757	0.396524	0.429924	0.456820
smart_7_raw	0.326879	1.000000	0.539748	0.128281	0.585176	0.519329
smart_9_raw	0.470757	0.539748	1.000000	-0.020046	0.266706	**0.997736**
smart_193_raw	0.396524	0.128281	-0.020046	1.000000	0.335702	-0.079894
smart_194_raw	0.429924	0.585176	0.266706	0.335702	1.000000	0.228991
smart_240_raw	0.456820	0.519329	**0.997736**	-0.079894	0.228991	1.000000

TABLE D.5: Correlation Matrix for healthy disk from model ST12000NM0007 - ZCH056VR

Healthy Disk ST12000NM0007 Serial # ZCH056VR	**smart_1_raw**	**smart_7_raw**	**smart_9_raw**	**smart_193_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	-0.291492	0.040868	0.198374	-0.004008	0.046690
smart_7_raw	-0.291492	1.000000	**0.722485**	0.455088	-0.064878	**0.713665**
smart_9_raw	0.040868	**0.722485**	1.000000	0.642411	-0.326361	**0.999289**
smart_193_raw	0.198374	0.455088	0.642411	1.000000	-0.510889	0.642498
smart_194_raw	-0.004008	-0.064878	-0.326361	-0.510889	1.000000	-0.313906
smart_240_raw	0.046690	**0.713665**	**0.999289**	0.642498	-0.313906	1.000000

TABLE D.6: Correlation Matrix for failed disk from model ST12000NM0007 - ZCH0AL23

Failed Disk ST12000NM0007 Serial # ZCH0AL23	**smart_1_raw**	**smart_7_raw**	**smart_9_raw**	**smart_192_raw**	**smart_193_raw**	**smart_240_raw**
smart_1_raw	1.000000	**0.743022**	0.503125	0.451996	0.344509	0.513814
smart_7_raw	**0.743022**	1.000000	0.500417	0.405775	0.167038	0.486951
smart_9_raw	0.503125	0.500417	1.000000	0.601211	**0.839095**	**0.999076**
smart_192_raw	0.451996	0.405775	0.601211	1.000000	**0.759134**	0.614697
smart_193_raw	0.344509	0.167038	**0.839095**	**0.759134**	1.000000	**0.851500**
smart_240_raw	0.513814	0.486951	**0.999076**	0.614697	**0.851500**	1.000000

TABLE D.7: Correlation Matrix for healthy disk from model ST12000NM0007 - ZJV10J45

Healthy Disk ST12000NM0007 Serial # ZJV10J45	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	-0.155619	-0.166672	-0.016085	-0.659190	-0.170055
smart_7_raw	-0.155619	1.000000	0.173853	-0.020315	0.416513	0.191309
smart_9_raw	-0.166672	0.173853	1.000000	0.575493	-0.062259	**0.991522**
smart_193_raw	-0.016085	-0.020315	0.575493	1.000000	-0.311858	0.543125
smart_194_raw	-0.659190	0.416513	-0.062259	-0.311858	1.000000	-0.079599
smart_240_raw	-0.170055	0.191309	**0.991522**	0.543125	-0.079599	1.000000

TABLE D.8: Correlation Matrix for failed disk from model ST12000NM0007 - ZJV03NQB

Failed Disk ST12000NM0007 Serial # ZJV03NQB	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	0.544369	0.353490	-0.074066	0.373296	0.375813
smart_7_raw	0.544369	1.000000	-0.305087	0.011450	0.420312	-0.316823
smart_9_raw	0.353490	-0.305087	1.000000	0.018962	0.042215	**0.993542**
smart_193_raw	-0.074066	0.011450	0.018962	1.000000	0.047066	-0.027482
smart_194_raw	0.373296	0.420312	0.042215	0.047066	1.000000	0.067022
smart_240_raw	0.375813	-0.316823	**0.993542**	-0.027482	0.067022	1.000000

TABLE D.9: Correlation Matrix for healthy disk from model ST12000NM0007 - ZCH06HY1

Healthy Disk ST12000NM0007 Serial # ZCH06HY1	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	**-0.721362**	0.020591	**-0.832200**	0.202707	0.005923
smart_7_raw	**-0.721362**	1.000000	0.047877	**0.809806**	-0.043604	0.053896
smart_9_raw	0.020591	0.047877	1.000000	-0.087280	-0.063281	**0.991122**
smart_193_raw	**-0.832200**	**0.809806**	-0.087280	1.000000	-0.129605	-0.072844
smart_194_raw	0.202707	-0.043604	-0.063281	-0.129605	1.000000	0.016449
smart_240_raw	0.005923	0.053896	**0.991122**	-0.072844	0.016449	1.000000

TABLE D.10: Correlation Matrix for failed disk from model ST12000NM0007 - ZCH097GA

Failed Disk ST12000NM0007 Serial # ZCH097GA	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	-0.090013	-0.559926	-0.414314	-0.284392	-0.554120
smart_7_raw	-0.090013	1.000000	0.665603	0.419819	**0.818138**	0.673308
smart_9_raw	-0.559926	0.665603	1.000000	**0.842269**	0.640534	**0.999752**
smart_193_raw	-0.414314	0.419819	**0.842269**	1.000000	0.207130	**0.839416**
smart_194_raw	-0.284392	**0.818138**	0.640534	0.207130	1.000000	0.649345
smart_240_raw	-0.554120	0.673308	**0.999752**	**0.839416**	0.649345	1.000000

TABLE D.11: Correlation Matrix for healthy disk from model ST12000NM0007 - ZCH0BCML

Healthy Disk ST12000NM0007 Serial # ZCH0BCML	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	0.265662	0.288319	-0.222821	-0.149011	0.285033
smart_7_raw	0.265662	1.000000	-0.147119	-0.247221	**-0.699470**	-0.170810
smart_9_raw	0.288319	-0.147119	1.000000	**0.694975**	0.378101	**0.997797**
smart_193_raw	-0.222821	-0.247221	**0.694975**	1.000000	0.449099	0.677031
smart_194_raw	-0.149011	**-0.699470**	0.378101	0.449099	1.000000	0.412304
smart_240_raw	0.285033	-0.170810	**0.997797**	0.677031	0.412304	1.000000

TABLE D.12: Correlation Matrix for failed disk from model ST12000NM0007 - ZJV00F20

Failed Disk ST12000NM0007 Serial # ZJV00F20	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	-0.626566	-0.172211	-0.556303	0.240412	-0.109197
smart_7_raw	-0.626566	1.000000	0.494762	0.598639	0.003462	0.468865
smart_9_raw	-0.172211	0.494762	1.000000	0.531663	-0.138053	**0.996179**
smart_193_raw	-0.556303	0.598639	0.531663	1.000000	-0.299230	0.488389
smart_194_raw	0.240412	0.003462	-0.138053	-0.299230	1.000000	-0.085365
smart_240_raw	-0.109197	0.468865	**0.996179**	0.488389	-0.085365	1.000000

TABLE D.13: Correlation Matrix for healthy disk from model ST12000NM0007 - ZJV501TY

Healthy Disk ST12000NM0007 Serial # ZJV501TY	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	-0.221391	-0.503270	-0.396752	0.235408	-0.513900
smart_7_raw	-0.221391	1.000000	**0.885034**	0.088399	-0.281677	**0.896764**
smart_9_raw	-0.503270	**0.885034**	1.000000	0.032943	-0.346043	**0.998117**
smart_193_raw	-0.396752	0.088399	0.032943	1.000000	-0.007978	0.048129
smart_194_raw	0.235408	-0.281677	-0.346043	-0.007978	1.000000	-0.362542
smart_240_raw	-0.513900	**0.896764**	**0.998117**	0.048129	-0.362542	1.000000

TABLE D.14: Correlation Matrix for failed disk from model ST12000NM0007 - ZJV00C88

Failed Disk ST12000NM0007 Serial # ZJV00C88	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	0.550420	**0.768246**	**0.703953**	-0.156656	**0.764068**
smart_7_raw	0.550420	1.000000	**0.923780**	**0.888632**	-0.435427	**0.921210**
smart_9_raw	**0.768246**	**0.923780**	1.000000	**0.933976**	-0.430801	**0.998644**
smart_193_raw	**0.703953**	**0.888632**	**0.933976**	1.000000	-0.407332	**0.916139**
smart_194_raw	-0.156656	-0.435427	-0.430801	-0.407332	1.000000	-0.425609
smart_240_raw	**0.764068**	**0.921210**	**0.998644**	**0.916139**	-0.425609	1.000000

TABLE D.15: Correlation Matrix for healthy disk from model ST12000NM0007 - ZCH0CDWV

Healthy Disk ST12000NM0007 Serial # ZCH0CDWV	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	-0.581867	-0.192267	0.126343	0.257818	-0.175353
smart_7_raw	-0.581867	1.000000	0.184252	0.041591	-0.142339	0.168532
smart_9_raw	-0.192267	0.184252	1.000000	**0.805019**	-0.560167	**0.998472**
smart_193_raw	0.126343	0.041591	**0.805019**	1.000000	-0.517995	**0.793278**
smart_194_raw	0.257818	-0.142339	-0.560167	-0.517995	1.000000	-0.547501
smart_240_raw	-0.175353	0.168532	**0.998472**	**0.793278**	-0.547501	1.000000

TABLE D.16: Correlation Matrix for failed disk from model ST12000NM0007 - ZJV03JDV

Failed Disk ST12000NM0007 Serial # ZJV03JDV	smart_1_raw	smart_7_raw	smart_9_raw	smart_193_raw	smart_194_raw	smart_240_raw
smart_1_raw	1.000000	-0.037589	-0.556252	-0.159486	-0.407908	-0.525013
smart_7_raw	-0.037589	1.000000	0.283011	0.049724	0.168751	0.261249
smart_9_raw	-0.556252	0.283011	1.000000	0.495007	0.422434	**0.998160**
smart_193_raw	-0.159486	0.049724	0.495007	1.000000	0.202369	0.487922
smart_194_raw	-0.407908	0.168751	0.422434	0.202369	1.000000	0.410003
smart_240_raw	-0.525013	0.261249	**0.998160**	0.487922	0.410003	1.000000

TABLE D.17: Correlation Matrix for healthy disk from model ST12000NM0007

Healthy Disk ST12000NM0007 Serial # ZCH0D2V0	**smart_1_raw**	**smart_7_raw**	**smart_9_raw**	**smart_193_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	-0.041900	0.071323	0.507615	-0.437712	0.077877
smart_7_raw	-0.041900	1.000000	0.645153	0.122852	-0.205947	**0.696068**
smart_9_raw	0.071323	0.645153	1.000000	0.618001	-0.115715	**0.994133**
smart_193_raw	0.507615	0.122852	0.618001	1.000000	-0.289708	0.625184
smart_194_raw	-0.437712	-0.205947	-0.115715	-0.289708	1.000000	-0.140796
smart_240_raw	0.077877	**0.696068**	**0.994133**	0.625184	-0.140796	1.000000

TABLE D.18: Correlation Matrix for failed disk from model ST4000DM000

Failed Disk ST4000DM000 Serial # S301P6Y6	**smart_1_raw**	**smart_7_raw**	**smart_9_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	0.453119	0.366908	0.018715	0.365931
smart_7_raw	0.453119	1.000000	0.568157	-0.161762	0.558772
smart_9_raw	0.366908	0.568157	1.000000	-0.230670	**0.999374**
smart_194_raw	0.018715	-0.161762	-0.230670	1.000000	-0.234252
smart_240_raw	0.365931	0.558772	**0.999374**	-0.234252	1.000000

TABLE D.19: Correlation Matrix for healthy disk from model ST4000DM000

Healthy Disk ST4000DM000 Serial # Z305D2CY	**smart_1_raw**	**smart_7_raw**	**smart_9_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	-0.285876	0.242102	-0.167771	0.237615
smart_7_raw	-0.285876	1.000000	0.379360	-0.150866	0.374310
smart_9_raw	0.242102	0.379360	1.000000	-0.228496	**0.999596**
smart_194_raw	-0.167771	-0.150866	-0.228496	1.000000	-0.233356
smart_240_raw	0.237615	0.374310	**0.999596**	-0.233356	1.000000

TABLE D.20: Correlation Matrix for failed disk from model ST8000NM0055

Failed Disk ST8000NM0055 Serial # ZA17ZNQ9	***smart_1_raw***	**smart_7_raw**	**smart_9_raw**	**smart_193_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	0.141209	0.133120	0.681647	0.148984	0.146289
smart_7_raw	0.141209	1.000000	**0.974040**	0.146295	0.473009	**0.972681**
smart_9_raw	0.133120	**0.974040**	1.000000	0.256401	0.485871	**0.999378**
smart_193_raw	0.681647	0.146295	0.256401	1.000000	0.421099	0.266986
smart_194_raw	0.148984	0.473009	0.485871	0.421099	1.000000	0.501445
smart_240_raw	0.146289	**0.972681**	**0.999378**	0.266986	0.501445	1.000000

TABLE D.21: Correlation Matrix for healthy disk from model ST8000NM0055

Healthy Disk ST8000NM0055 Serial # ZA16YG7B	***smart_1_raw***	**smart_7_raw**	**smart_9_raw**	**smart_193_raw**	**smart_194_raw**	**smart_240_raw**
smart_1_raw	1.000000	-0.207226	-0.305566	-0.400377	-0.211327	-0.321603
smart_7_raw	-0.207226	1.000000	**0.907206**	-0.212340	-0.541860	**0.901361**
smart_9_raw	-0.305566	**0.907206**	1.000000	-0.216698	-0.650402	**0.996615**
smart_193_raw	-0.400377	-0.212340	-0.216698	1.000000	-0.038489	-0.259011
smart_194_raw	-0.211327	-0.541860	-0.650402	-0.038489	1.000000	-0.612920
smart_240_raw	-0.321603	**0.901361**	**0.996615**	-0.259011	-0.612920	1.000000

Appendix E

Computational Cluster Batch Task Profiling with Machine Learning for Failure Prediction

The following paper was submitted to arXiv and is awaiting acceptance to a peer-review publication. This paper has been cited by a peer-reviewed publication.

Bioinformatics Computational Cluster Batch Task Profiling with Machine Learning for Failure Prediction

Christopher Harrison*† Christine R. Kirkpatrick‡† and Inês Dutra†
* Department of Biostatistics and Medical Informatics
School of Medicine and Public Health
University of Wisconsin - Madison
Madison, Wisconsin USA
† Departamento de Ciência de Computadores
Faculdade de Ciências
Universidade do Porto
Porto, Portugal
‡ San Diego Supercomputer Center
University of California San Diego
La Jolla, California USA

***Abstract*—Motivation: Traditional computational cluster schedulers are based on user inputs and run time needs request for memory and CPU, not IO. Heavily IO bound task run times, like ones seen in many big data and bioinformatics problems, are dependent on the IO subsystems scheduling and are problematic for cluster resource scheduling. The problematic rescheduling of IO intensive and errant tasks is a lost resource. Understanding the conditions in both successful and failed tasks and differentiating them could provide knowledge to enhancing cluster scheduling and intelligent resource optimization.**

Results: We analyze a production computational cluster contributing 6.7 thousand CPU hours to research over two years. Through this analysis we develop a machine learning task profiling agent for clusters that attempts to predict failures between identically provision requested tasks.

***Index Terms*—Cluster scheduling, Cluster task failure analysis, Computing clusters analysis, HPC, HTC, Machine learning cluster fault prediction, HTCondor cluster**

I. Introduction

Resource allocation is a well studied problem. Solutions have been sought for different kinds of applications and platforms, ranging from multi-core centralized memory machines to distributed memory machines, passing by many-core and GPGPUs. The objective is always the same: to better allocate resources (CPU, memory and storage) in order to improve some performance metrics (response time, throughput, speedup, efficiency, Quality of Service (QoS), among others). Due to specific characteristics of applications and platforms, it is not possible to devise one solution that fits all. It is also not possible to devise solutions that simultaneously will improve all metrics. With the ever growing use of cloud computing platforms and their virtual machines, new solutions have been studied. Monitoring tools have been built that can help notify system administrators to address jobs which need to be rescheduled, restart, create, or destroy and these tools extend to virtual machines as a way to also keep these systems running. These tools also help operators justify the need for additional compute resources.

Cloud computing platforms rely on clusters of machines to run users jobs and their middleware provides an interface (command line, API or portal) to help submitting single or batch jobs. Usually, users need to specify their job's requirements such as expected execution time, expected memory usage, storage, number of cores, among others. However, the fact is that since the creation of the first parallel and distributed batch platforms, most submitted jobs fail to complete for several different reasons. The high rate of failure is an impediment to the good use of resources. Statistical analysis of trace logs were performed by Prodan and Fahringer [1], and by Christodoulopoulos *et al.*, among

others, where conventional algorithmic solutions have been used to improve resource allocation and prevent failures. More sophisticated solutions, based on machine learning techniques, have also been used to identify patterns of success and failure and to optimize resource usage [2]. We study traces of a two-year period of jobs submitted using the Condor HTC tool with the objective of understanding patterns of success and failure. These jobs were all submitted by biostatisticians or medical informatics groups. We analyzed all jobs and identified two kinds of submissions: jobs with a single task submission and jobs with multiple tasks submissions. While single task job submissions are very common, they have very specific characteristics that make it difficult to find patterns. Therefore, we concentrated on the multiple tasks submissions jobs, since they tend to share common characteristics such as the same command and possibly the same requirements. We categorized these jobs in five mainstream groups:

1) jobs submissions of single tasks which finished successfully
2) jobs submissions of single tasks which failed
3) jobs with multiple task submission for which all tasks finished successfully
4) jobs with multiple task submission for which all tasks failed
5) jobs with multiple task submission for which some of the tasks succeeded and some failed

With the last group, jobs with multiple task submission for which some of the tasks succeeded and some failed, we applied a machine learning technique to each job in order to identify patterns of success and failure among their tasks. The average number of tasks per job of this kind is 182.3. Results show that an average of 14.3% fail and 85.7% succeed. The jobs that fail all have in common a HTCondor class ad 'JobStatus=3' whereas Jobs that succeed all have in common a HTCondor class ad 'JobStatus=4'. By detecting such patterns, we expect to use them to detect future new tasks likely to fail that enter the system, and adjust or correct them in order to minimize failures and optimize resource usage.

II. Background

Computational Cluster task scheduling, such as ones used in HPC/HTC systems, distribute tasks based on available HPC/HTC resources. The task schedulers match user requested resources per task to the expected resource available. If a task resource request is not sufficient to complete the task then additional task resource allocations are made based on the scheduled cluster host resource. The goal is to efficiently allocate resources based on task matched resource requests and optimize based on requested demand. As such, an HPC/HTC scheduler has two optimization areas which are:

1) granular measurements of process needs: IO, CPU, network, system wait times, etc.
2) enhancing the end user's task estimating needs based on a best guess.

System and task monitoring of performance metrics are essential for evaluating resource allocation and utilization. These long term trends based on system analysis compared to base lines are widely used as a justification for additional resources. However, optimization of resources is difficult without granular resource task trends. Resource usage statistics and trends tools are designed to lose fidelity over time due to the amounts of data generated by performance metrics. Additionally, the amount of data collected for metrics grows as a function to cluster size. While HPC resource monitoring platforms exist; such as ganglia [3], the value of their longitudinal data decreases over time due to the application storage formats of their metrics. Specifically RRD's (Round-Robin Databases) use results in degrading signal value over time.

We monitor cluster resource utilization with granular metrics for scheduling and running tasks. Our problem is to optimize resources utilization through modeling of failed tasks through monitoring and adaptively responding to task resource needs based on available cluster resources. Similarly to Juve *et al.* [4], we profile our cluster tasks while addressing resource loss as a function of time and impact to total resource utilization. Specifically, we focus on cluster resource loss as a function of failed tasks which have identical attributes to other successful tasks, also known as a same cluster submission job. Our problem is complicated due to temporal computational cluster resources utilization needs which fluctuates based on scheduled running tasks, service outages, unplanned outages, network anomalies, etc. All submitted tasks are impacted by Spatio-temporal phenomena[5] with time impacting usage, and queued demand impacting temporal run time and scheduling.

Understanding and tracking performance metrics over time with consistent time series granularity is resource intensive. At scale, systems designed for simple collection and analysis of groups of systems cannot handle the influx of 10 to 100 of thousands records a second while being real time search-able. Inefficient resource utilization equals wasted resources; therefore, gauging

improvements to application and system performance will provide a baseline for resource optimization. Cluster tasks requiring a reschedule are wasted resources due to the cost of compute time on a cluster. Thus a need for better failure prediction and cluster rescheduling after task failure provides a pathway towards resource optimization.

Human best guess estimates can often be wrong as can be seen in baseball [6]. Individual users can misinterpret the resources required for a given task and request more or less resources which leads to a misallocation of cluster resources and non optimal cluster utilization outcomes.

III. Related Work

Task profiling and system performance metrics are essential for effectively evaluating task allocations and cluster resource utilization. Long term trends and individual analysis compared to base lines are widely used as a justification for additional resources. When evaluating HPC system utilization a baseline analysis and long term trends are needed to justify additional resource allocations. However, optimization of resources is difficult without granular resource trends. Resource usage statistics and trends lose fidelity over time due to the amounts of data generated by performance metrics and grows as a function to cluster size. While HPC resource monitoring platforms exist such as ganglia [3], the value of their longitudinal data decreases over time due to the application storage formats of their metrics. Specifically RRD's (Round-Robin Databases) [7] use results in degrading signal value over time.

Numerous tools, such as cacti [8], ganglia [3], nagios [9], and zabbix [10], exist to evaluate system utilization and track performance metrics. These tools attempt to prove graphical representations of system performance metrics and utilize the network standard Simple Network Monitoring Protocol (SNMP) [11]. Tracking metrics can address application bottlenecks such as application starvation or usage of memory, CPU, or IO by critically understanding resource allocation starvation and optimization at starvation time.

OVIS [12] has attempted to address scheduling of resources based on predictive failure analysis but not on the user requested resource allocation. Additionally, OVIS's scope was limited to resource allocation improvements to address task scheduling around node failures not optimizing cluster resources. While a given node's failure will impact scheduling to that node and is impacted by the Mean Time To Failure (MTTF), the resource scheduling to a given node is a function of the given node's failure probability. Thus, OVIS attempts to address the issue of resource failures within a cluster by working around hardware failures at the node level.

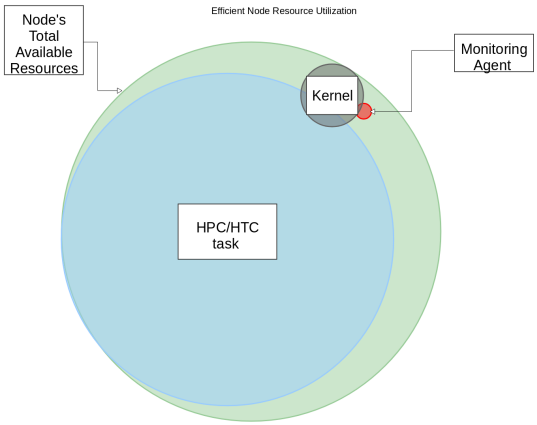


Fig. 1: Efficient task run state on cluster node

Larger scale system monitoring of HPC and the Open Science Grid [13] have used such systems as OVIS [12] or TACC Stats [14]. OVIS uses a Bayesian inference scheme to dynamically infer models for the normal behavior of a system and to determine bounds on the probability of values evinced in the system. OVIS addresses hardware related failure issues and system level performance analysis on systems based on MTTF analysis of a given system.

Understanding system resource usage across a cluster, based on a per scheduled task basis instead of an entire system, requires an understanding of task profiling. Given tasks profiles for CPU, memory, and disk have been demonstrated [15] as a method for "choosing the most suitable set of computers for the deployment of the tasks" [15]. As such, computing resource usage measurements are required to effectively and efficiently utilize computing resources, Fig. 1.

Modeling resource utilization as a means for optimization predictions is needed in both HPC and cloud (as HTC) contexts. Even newest of breed techniques, such as the use of Kubernetes for cloud orchestration, only consider CPU in scheduling decisions. This is too simplistic for true optimization. Newer models and software implementations are needed to more effectively understand usage and suggest future scheduling of resources. Chang *et al.* suggested using multiple variables, such as memory and disk access, and creates a dynamic algorithm that can adjust to tasks in situ [16]. Wei *et al.* demonstrated a

technique for allocating Virtual Machine (VM) resources based on CPU and memory [17], albeit limited. Along with various approaches, systems and models schedule using different parameters and thereby optimize usage based on various priorities, such as deadline, cost minimization or maximization, such as for cloud providers, to adhere to Service Level Agreements, and to reduce power consumption [18]. Additional work in this area has been motivated by optimization based on energy savings, where Pinheiro notes that it is key to examine resource reconfiguration and keeping a load stable (relatively unchanged) as the throughput loss can be resource intensive [19]. Because of the heterogeneous hardware and unpredictable loads in shared environments [20], cloud systems perhaps, need predictive scheduling and optimization more than traditional, homogeneous HPC clusters. To disregard optimization in cloud systems is to ignore the promise of the cloud as an elastic resource, made to adapt to computing loads in ways previously limited by dedicated hardware [18]. This work can be extended to heterogeneous HPC systems, as can occur in condo storage models, where nodes are purchased over time [21], as well as where users are able to bring their own nodes to the cluster. Mateescu describes a technique where scheduling is done in part based on the timing demands of the tasks and can utilize combinations of HPC and Cloud as a combined workflow, either managing at the node level (physical machine) or at the VM level [22].

Recently, Rodrigo *et al.* [23] proposed a workflow aware scheduling algorithm specific to slurmc̃iteyoo2003slurm which attempts to address temporal and locality resource scheduling. This approach is interesting as it attempts to match pipelines, beyond individual tasks, to resources for data locality and increased throughput. However, the resource locality effect is muted for clusters that do not have local resources (e.g., nodes without local disk for scratch space), which affect scheduled tasks in a homogeneous HPC/HTC environment. Considering previous work in scheduling and optimization [18], what is unique about this method is the use of machine learning to suggest optimization for real-time workloads, that works across various size systems, under a wide range of applications.

IV. Methods

Following procedures originally used by Juve *et al.* [4] we collect task statistics and compare our statistics against the task's requested cluster resources. Using Juve *et al.* method we use the metric data polled directly from the kernel through the procfs file system. We also note that only a subset of the proc file system, of which the kernel provides a non-intrusive task metric to gain individual process statistics about performance and run time state in real time, is used. A list of available procfs metric's is defined in I. The defined list is meant to be a demonstration to the richness to which the kernel procfs application level statistics can be distilled.

Procfs provides kernel counters on a per task basis. However, these task based kernel counters have grown organically over the years and have resulted in a less than ideal task counter framework. Specifically, the procfs counter files within the procfs file system are a mixture of file formats and none are consistent.

A. Computational cluster

1) University of Wisconsin - Madison, Biostatistics Computing Group: The cluster is a departmental level cluster comprised of 240 64 bit multicore x86 systems running a Redhat Package Management (RPM) [25] based Linux variant, Scientific Linux 7 [26]. The cluster is a heterogeneous mixture of hardware vendors and CPU brands with approximately 2500 CPU cores and 11.8 TB of total memory across the cluster. Hardware manufacturers used in the cluster include: Dell, Cisco, HPC and Supermicro with x86 CPUs.

The cluster is comprised of shared use interactive machines and dedicated compute machines. Both types of machines are part of a cluster which uses HTCondor [27] as the cluster resource manager.

Profiling statistics from a system are generated through the kernel. The Linux kernel procfs contains task level runtime parameters which live reflect the task runtime within the OS scheduler. Additionally, any task's sub task or children are also defined in the procfs, as the Parent Process Id (PPID). Any subtask task which is created by fork the system will register the child process Id (pid) under the parent as defined in the procfs path $/proc/\$ppid/task/\$ppid/children$. The procfs we describe is used on the cluster installed Linux variant Scientific Linux 7 [26] running a Linux kernel 3.10.x.

The focus of the tasks we used in our analysis are all in support of the Department of Biostatistics and Medical Informatics at the University of Wisconsin - Madison. These tasks where submitted between epoch times (1475644627 - 1537752190) or between October 5, 2016 to September 24, 2018. All submitted tasks are used by biostatisticians or medical informaticians and require compliance or have dataset restrictions which inhibit them from running on across infrastructures such

file	type	description
children	list numbers	Child tasks process id (pid) from the this task id
cmdline	text	full command line used to execute this task with options
cwd	simlink	symbolic link to the current working directory of the process
environ	text	initial environment that was set when the currently executing program was started via execve
fd	directory	a subdirectory containing one entry for each file which the process has open, named by its file descriptor, and which is a symbolic link to the actual file
fdinfo	directory	a subdirectory containing one entry for each file which the process has open, named by its file descriptor
io	text	contains I/O statistics for the process • rchar: characters read • wchar: characters written • syscr: read syscalls • read_bytes: bytes read • write_bytes: bytes written • cancelled_write_bytes: The big inaccuracy here is truncate.
mounts	text	lists all the filesystems currently mounted in the process's mount namespace
oom_score	int	displays the current score that the kernel gives to this process for the purpose of selecting a process for the OOM-killer
pagemap	text	the mapping of each of the process's virtual pages into physical page frames or swap area
smaps	text	shows memory consumption for each of the process‘'s mappings
stat	text	Status information about the process
statm	text	Provides information about memory usage, measured in pages
status	text	Provides much of the information in /proc/[pid]/stat and /proc/[pid]/statm in a format that's easier for humans to parse
syscall	text	exposes the system call number and argument regis- ters for the system call currently being executed by the process, followed by the values of the stack pointer and pro- gram counter registers
wchan	text	symbolic name corresponding to the location in the kernel where the process is sleeping

TABLE I: procfs task files [24]

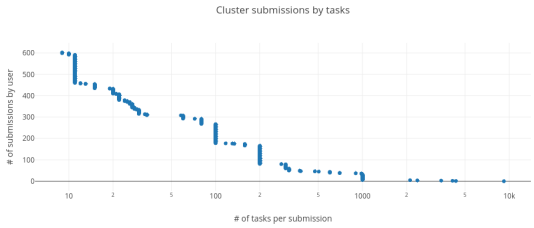


Fig. 2: Number for tasks per cluster submission

number	percent	description
106695	82.2%	Complete successful, no errors
23159	17.8%	Task Error Issues
129854	100%	All submitted tasks

TABLE II: Submitted Task Breakdown

as the OSG[13] or other open resources. Using the HTCondor class-Ad history we analyze 17,282 cluster submissions producing 129,854 tasks in the BCG HTCondor[27] computational cluster. Table II shows a complete breakdown of the total tasks by completion state. Additionally, in Table III, we can see the full exit status of tasks which did not complete successfully.

2) Data exploration: We produced our data as an aggregate from all cluster nodes which are allowed to submit to the cluster. The data was generated through the use of condor_history with the –json option which produced the Condor history data in JSON format. Using the preformatted JSON HTCondor history data partitioned by individual class ad and generating a unique file for each Condor submission node, we aggregate the data by importing the data into a NoSQL database. MongoDB [28] was chosen as the data store for exploring and analyzing the data due to the simplistic import nature and well established user base with a feature rich NoSQL language.

To analyze the data we used the MongoDB internal

number	percent	event
13070	10.1%	User removed before scheduled to cluster
9183	7.1%	User defined task attribute expression error
434	0.33%	Failed to initialize user log
229	0.17%	Out-of-memory event
160	0.11%	Other
83	0.06%	No such file or directory
23159	17.8%	Total Task Error Issues

TABLE III: Submitted Tasks with errors breakdown

# submission	% submission	# tasks	% total tasks	submission type	description
10758	62.2%	10758	8.3%	single	successfully completed
4285	24.8%	4285	3.3%	single	failed
1860	10.8%	93908	72.3%	multi	combo success and failed
341	2.0%	16356	12.6%	multi	all successfully completed
38	0.2%	4547	3.5%	multi	all failed
17282	100%	129854	100%		total cluster submissions

TABLE IV: Cluster submissions breakdown

language and were able to aggregate data clusters using the condor class-Add 'ClusterId' as the unique cluster key. The cluster task resource breakdown by task and totals are defined in V. For our final dataset, we apply a split frame approach to randomly assign our data to different buckets for use in the applied machine learning. Specifically, we split frame our data based on the ratios 60%, 30%, 10%.

3) Feature selection: To understand why our tasks either succeeded or failed by submission we applied random forest [29], a well known machine learning technique, as a classifier estimator. We utilize the machine learning platform H2O [30] to build our models. In choosing the Random Forest algorithm, we address the classification and regression predictions by using the average prediction trees to make the final prediction based on class or numerical values. Thus, we address our multi-modal multi-class dataset without excessive normalization's so we can integrate: numerical, binomial, and categorical data.

V. Experiment

Our cluster had a total of 15,043 unique cluster submissions where only one task was submitted to the cluster by the job submission. A breakdown of cluster submission by task result(s) is found in IV and we plot our data based on tasks per submission. Of the remaining 1,860 cluster submissions, 491 of these have greater then 5 tasks submitted per cluster submission and have both failed and successful completed tasks. We set the cutoff at 5 tasks per submission so we have enough data points to apply a machine learning algorithm, random forest, for feature selection. We focus this analysis on the submissions of 5 or more tasks with a combination of successful and failed tasks for multi-tasked submissions, the total number of tasks submitted with 5 or greater is a subset from the last line in IV. The total number of remaining tasks is 90,062 of the 93,908 or 69.4% of the total cluster tasks. We use 90,062 as our base data from which to train, validate, and test.

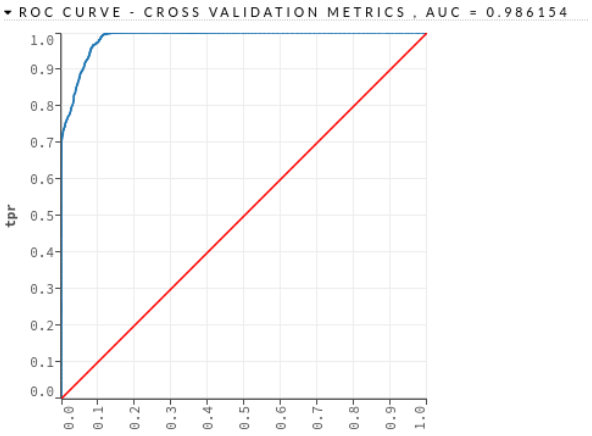


Fig. 3: ROC curve, cross validation

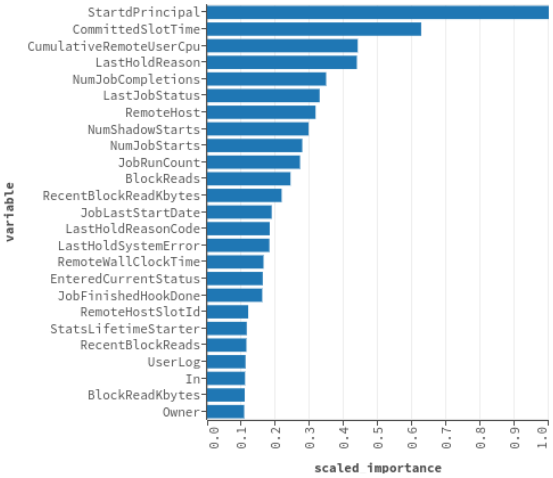


Fig. 4: Variable Importance

A. Prediction

To build our Random Forest we select 50 trees, each with a max depth of 50. Number of variables randomly sampled as candidates at each split was set at 5 (**mtries**) from the active predictors requests within H2o.ai [31].

Usage statistic	Cumulative total	Average	Max	Min	Description
CPU	3522219511	27124.45	43112204	0	Cumulative CPU time per task, min
System CPU	157853822	1358.85	366756	0	Cumulative CPU time in system/kernel time, min
User CPU	2415838569	20796.25	19679010	0	Cumulative CPU time in User space time, min
Suspension	146800	1.13	12895	0	Cumulative time a task is suspended/held, min
BytesSent	36400253248	295044.68	1475104768	0	

Note: System and User CPU time are a subset of the total due to a HTCondor version change over the course of the collected dataset.

TABLE V: Cluster task usage in minutes

Additionally, we chose to build histograms for numerical columns and chose to use histogram bins at 1000. Addressing cross validation of our random forest trees we chose a cross validate fold number of 5, with a random fold assignment. The number of categorical and top level histogram bins was set at 1024. For our distribution function we utilized a multinomial distribution when building our model. Lastly, we ignored columns when building our model, namely the columns: *id*, *AutoClusterId*, *CommittedTime*, *CompletionDate*, *ExitCode*, *LastVacateTime*, *RemoteSysCpu*, *RemoteUserCpu*, *RemoveReason*. On our initial feature selections tests, these columns confounded feature selection which heavily skew our initial results.

Using our random forest model, we were able to obtain a 99% precision for failed tasks and a 94% precision for successful tasks. Of these tasks we had an error rate of 11.6% for failed tasks and a 0.5% error rate for successful tasks. For recall during our cross validation we saw a 88% recall of failed tasks and 1.0 for successful tasks. The total error rate was 4.6%. We were able to predict the relative importance of each attribute within the HTCondor class-Ad with scaled importance. Deterministically, our class-Ad attributes for predicting failure vs. success is ranked by contribution according to 4: *StartdPrincipal* (12.1%), *CommittedSlotTime* (7.6%), *CumulativeRemoteUserCPU* (5.3%), *LastHoldReason* (5.3%), *NumJobCompletions* (4.2%), *LastJobStatus* (4%), *RemoteHost* (3.8%), *NumShadowStarts* (3.6%), *NumJobStarts* (3.4%), *JobRunCount* (3.3%), *BlockReads* (2.9%), *RecentBlockReadKbytes* (2.6%), *JobLastStartDate* (2.3%). *CommittedSlotTime* (second most valuable feature) most likely would be zero for tasks which where scheduled to a cluster node but where unable to access the users data (network or file system issue on a host). Thus, our proposed model may still have internal feature selection confounding issues.

VI. Conclusions

Computational cluster analysis and failure prediction is based on temporal events with identically provision requested tasks is a significant concern. 72% of all muli-task clusters submissions had at least one task fail on our production cluster. Understanding why identically provision requested tasks succeed or fail on a heterogeneous cluster has real world benefits for both our cluster resource scheduling and to the end users. We address the failure question through a machine learning feature selection process. The feature selection is based on the random forest algorithm and is unique due to the significant number of production task submission we use as data for our machine learning inputs. Our results showcase an overall accurate model (**95.4%**) but lagged in prediction for failed jobs at 88%. Our use of a feature selection method provides a baseline for other production computational cluster analysis. As with all real world example cases, normal production clusters idiosyncrasies likely impacted the specific results of our model.

VII. Future Work

This method could be further refined and extended through the analysis of other clusters and job profiles and address job/task confounding. In particular, we plan to apply this technique to the analysis of other private cloud (OpenStack), to inform better service delivery and optimization of resources. With these resource recommendations, further work should be done to implement their use to combine cloud (HTC) and dynamic HPC scheduling slots. Additionally, the linux procfs is a rich source of task resource usage data which should be explored further for resource usage scheduling optimization. Lastly, further refinements to the feature selection process and combining it with other temporal data should result in a ‘job cluster effect‘ which will provide a more generalizable model for cluster task scheduling.

REFERENCES

[1] R. Prodan and T. Fahringer, "Overhead analysis of scientific workflows in grid environments," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 3, pp. 378–393, March 2008.

[2] D. Zeinalipour-Yazti, K. Neocleous, C. Georgiou, and M. D. Dikaiakos, "Identifying failures in grids through monitoring and ranking," in *2008 Seventh IEEE International Symposium on Network Computing and Applications*, July 2008, pp. 291–298.

[3] M. Massie, B. Li, B. Nicholes, V. Vuksan, R. Alexander, J. Buchbinder, F. Costa, A. Dean, D. Josephsen, P. Phaal, and D. Pocock, *Monitoring with Ganglia*, 1st ed. OŔeilly Media, Inc., 2012.

[4] G. Juve, B. Tovar, R. F. d. Silva, D. Król, D. Thain, E. Deelman, W. Allcock, and M. Livny, "Practical resource monitoring for robust high throughput computing," in *2015 IEEE International Conference on Cluster Computing*, Sept 2015, pp. 650–657.

[5] S. Yanchuk and G. Giacomelli, "Spatio-temporal phenomena in complex systems with time delays," *Journal of Physics A: Mathematical and Theoretical*, vol. 50, no. 10, p. 103001, 2017. [Online]. Available: http://stacks.iop.org/1751-8121/50/i=10/a=103001

[6] A. Houser, "Which baseball statistic is the most important when determining team success?" *The Park Place Economist*, vol. 13, pp. 29–36, 2005.

[7] D. Carder, "Rrdtool scalability," http://net.doit.wisc.edu/~dwcarder/rrdcache/, 2007, online; Accessed 9-Sept-2018.

[8] I. The Cacti Group, "Cacti Monitoring Tool," https://www.cacti.net/, 2018, online; Accessed: 24-Sept-2018.

[9] W. Barth, *Nagios: System and Network Monitoring*. San Francisco, CA, USA: No Starch Press, 2006.

[10] Zabbix LLC, "Zabbix monitoring system," https://www.zabbix.com/, 2018, online; Accessed 24-Sept-2018.

[11] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "Simple network management protocol (snmp)," Internet Requests for Comments, Internet Engineering Task Force, RFC 1157, May 1990. [Online]. Available: http://www.ietf.org/rfc/rfc1157.txt?number=1157

[12] J. M. Brandt, A. C. Gentile, D. J. Hale, and P. P. Pebay, "Ovis: a tool for intelligent, real-time monitoring of computational clusters," in *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, April 2006, pp. 8 pp.–.

[13] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, F. Würthwein, I. Foster, R. Gardner, M. Wilde, A. Blatecky, J. McGee, and R. Quick, "The open science grid," *Journal of Physics: Conference Series*, vol. 78, no. 1, p. 012057, 2007. [Online]. Available: http://stacks.iop.org/1742-6596/78/i=1/a=012057

[14] T. Evans, W. L. Barth, J. C. Browne, R. L. DeLeon, T. R. Furlani, S. M. Gallo, M. D. Jones, and A. K. Patra, "Comprehensive resource use monitoring for hpc systems with tacc stats," in *2014 First International Workshop on HPC User Support Tools*, Nov 2014, pp. 13–21.

[15] S. Seneviratne and D. C. Levy, "Task profiling model for load profile prediction," *Future Gener. Comput. Syst.*, vol. 27, no. 3, pp. 245–255, Mar. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.future.2010.09.004

[16] C. Chang, S. Yang, E. Yeh, P. Lin, and J. Jeng, "A kubernetes-based monitoring platform for dynamic cloud resource provisioning," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[17] L. Wei, C. H. Foh, B. He, and J. Cai, "Towards efficient resource allocation for heterogeneous workloads in iaas clouds," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 264–275, Jan 2018.

[18] H. S. Guruprasad and b. B H, "Resource provisioning techniques in cloud computing environment: A survey," *International Journal of Research in Computer and Communication Technology*, vol. 3, pp. 395–401, 03 2014.

[19] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," 2001.

[20] A. Homer, J. Sharp, L. Brader, M. Narumoto, and T. Swanson, *Cloud Design Patterns: Prescriptive Architecture Guidance for Cloud Applications*. Microsoft patterns & practices, 2014.

[21] San Diego Supercomputer Center, UC San Diego, "Tscc purchase program," http://www.sdsc.edu/services/hpc/tscc-purchase.html, 2018, online; Accessed 3-Dec-2018.

[22] G. Mateescu, W. Gentzsch, and C. J. Ribbens, "Hybrid computing—where hpc meets grid and cloud computing," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 440 – 453, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X1000213X

[23] G. P. Rodrigo, E. Elmroth, P.-O. Östberg, and L. Ramakrishnan, "Enabling workflow-aware scheduling on hpc systems," in *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '17. New York, NY, USA: ACM, 2017, pp. 3–14. [Online]. Available: http://doi.acm.org/10.1145/3078597.3078604

[24] Linux Foundation, "process information pseudo-file system," in *Linux Man Pages*, 2017, online; Accessed 25-Sept-2018.

[25] E. Troan, M. Ewing, and Red Hat, "Redhat package management," http://rpm.org/, 2018, online; Accessed: 24-Sept-2018.

[26] C. Sieh, "Scientific linux," https://www.scientificlinux.org, 2018, [Online; Accessed 24-Sept-2018].

[27] M. Livny and R. Raman, "High-throughput resource management," in *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds. Morgan Kaufmann, 1998.

[28] I. MongoDB, "MongoDB," https://www.mongodb.com/, 2018, online; Accessed: 14-Dec-2018.

[29] G. Louppe, "Understanding random forests: From theory to practice," Ph.D. dissertation, University of Liege, Belgium, 10 2014, arXiv:1407.7502.

[30] D. Cook, *Practical machine learning with H2O: powerful, scalable techniques for deep learning and AI.* " O'Reilly Media, Inc.", 2016.

[31] H2o Core Team, *H2o: A Language and Environment for Machine Learning*, Vienna, Austria, 2013. [Online]. Available: http://www.R-project.org/

Bibliography

[1] Sunyoung Shin, Rebecca Hudson, Christopher Harrison, Mark Craven, and Sündüz Keleş. atSNP Search: a web resource for statistically evaluating influence of human genetic variation on transcription factor binding. *Bioinformatics*, 35(15):2657–2659, 12 2018. [Cited on pages 1, 2, and 5.]

[2] Chandler Zuo, Sunyoung Shin, and Sndz Kele. atsnp: transcription factor binding affinity testing for regulatory snp detection. *Bioinformatics*, 31(20):3353–3355, 2015. [Cited on pages 2, 10, 11, 29, 37, and 47.]

[3] Jim Basney and Miron Livny. *Deploying a High Throughput Computing Cluster*. Prentice Hall PTR, 1999. [Cited on pages 3, 38, 51, and 53.]

[4] NIGEL CHAFFEY. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. and Walter, P. Molecular biology of the cell. 4th edn. *Annals of Botany*, 91(3):401–401, 02 2003. [Cited on page 6.]

[5] Geoffrey M Cooper. *The Cell: A Molecular Approach. 2nd edition*. Sinauer Associates, 2000. [Cited on page 6.]

[6] D. G. Feitelson and M. Treinin. The blueprint for life? *Computer*, 35(7):34–40, 2002. [Cited on page 6.]

[7] J. D. WATSON and F. H. C. CRICK. Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, Apr 1953. [Cited on page 6.]

[8] A. Siewierska-Grska, A. Sitek, E. dziska, G. Bartosz, and D. Strapagiel. Association of five snps with human hair colour in the polish population. *HOMO*, 68(2):134–144, 2017. [Cited on page 8.]

[9] Olivia S. Meyer, Maja M. B. Lunn, Sara L. Garcia, Anne B. Kjrbye, Niels Morling, Claus Brsting, and Jeppe D. Andersen. Association between brown eye colour in

rs12913832:gg individuals and snps in tyr, tyrp1, and slc24a4. *PLOS ONE*, 15(9):1–15, 09 2020. [Cited on page 8.]

[10] Marc A Schaub, Alan P Boyle, Anshul Kundaje, Serafim Batzoglou, and Michael Snyder. Linking disease associations with regulatory information in the human genome. *Genome research*, 22(9):1748–1759, 2012. [Cited on page 10.]

[11] Volker Matys, Olga V Kel-Margoulis, Ellen Fricke, Ines Liebich, Sigrid Land, A Barre-Dirrie, Ingmar Reuter, D Chekmenev, Mathias Krull, Klaus Hornischer, et al. Transfac® and its module transcompel®: transcriptional gene regulation in eukaryotes. *Nucleic acids research*, 34(suppl_1):D108–D110, 2006. [Cited on page 12.]

[12] Anthony Mathelier, Xiaobei Zhao, Allen W. Zhang, Franois Parcy, Rebecca Worsley-Hunt, David J. Arenillas, Sorana Buchman, Chih-yu Chen, Alice Chou, Hans Ienasescu, Jonathan Lim, Casper Shyr, Ge Tan, Michelle Zhou, Boris Lenhard, Albin Sandelin, and Wyeth W. Wasserman. Jaspar 2014: an extensively expanded and updated open-access database of transcription factor binding profiles. *Nucleic Acids Research*, 42(D1):D142–D147, 2014. [Cited on page 12.]

[13] Peter J. A. Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, and Peter M. Rice. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, 38(6):1767–1771, 12 2009. [Cited on page 13.]

[14] Illumina Cambridge Ltd. (ILLUMINA). Illumina sequencing of escherichia coli str. k-12 substr. mg1655 genomic paired-end library. https://www.ncbi.nlm.nih.gov/sra/SRR001666, 2008. Access: 12-Aug-2022,Name: 500bp-insert library,Instrument: Illumina Genome Analyzer, Strategy: WGS, Source: GENOMIC, Selection: RANDOM,Layout: PAIRED. [Cited on pages xiii and 14.]

[15] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A Albers, Eric Banks, Mark A DePristo, Robert E Handsaker, Gerton Lunter, Gabor T Marth, Stephen T Sherry, et al. The variant call format and vcftools. *Bioinformatics*, 27(15):2156–2158, 2011. [Cited on page 13.]

[16] David E Reich, Michele Cargill, Stacey Bolk, James Ireland, Pardis C Sabeti, Daniel J Richter, Thomas Lavery, Rose Kouyoumjian, Shelli F Farhadian, Ryk Ward, et al.

Linkage disequilibrium in the human genome. *Nature*, 411(6834):199–204, 2001. [Cited on page 15.]

[17] David G Clayton. Generalized linear mixed models. *Markov chain Monte Carlo in practice*, 1:275–302, 1996. [Cited on page 15.]

[18] MARK S. FOX. An organizational view of distributed systems. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 140–150. Morgan Kaufmann, 1988. [Cited on page 16.]

[19] Eric A. Brewer. Towards robust distributed systems (abstract). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '00, page 7, New York, NY, USA, 2000. Association for Computing Machinery. [Cited on page 16.]

[20] Linus Torvalds. Linux–a free unix-386 kernel, 1991. [Cited on page 16.]

[21] Douglas R. Smith. The design of divide and conquer algorithms. *Science of Computer Programming*, 5:37–58, 1985. [Cited on page 16.]

[22] David A. Patterson, Garth Gibson, and Randy H. Katz. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*, SIGMOD '88, page 109116, New York, NY, USA, 1988. Association for Computing Machinery. [Cited on page 17.]

[23] Rashmi Vinayak. *Erasure Coding for Big-data Systems: Theory and Practice*. PhD thesis, EECS Department, University of California, Berkeley, Sep 2016. [Cited on page 17.]

[24] H. Lamehamedi, B. Szymanski, Z. Shentu, and E. Deelman. Data replication strategies in grid environments. In *Fifth International Conference on Algorithms and Architectures for Parallel Processing, 2002. Proceedings.*, pages 378–383, 2002. [Cited on page 17.]

[25] Sikha Bagui and Loi Tang Nguyen. Database sharding: to provide fault tolerance and scalability of big data on the cloud. *International Journal of Cloud Applications and Computing (IJCAC)*, 5(2):36–52, 2015. [Cited on page 17.]

[26] Xiufeng Liu, Christian Thomsen, and Torben Bach Pedersen. Mapreduce-based dimensional etl made easy. *Proceedings of the VLDB Endowment*, 5(12):1882–1885, 2012. [Cited on page 17.]

[27] Database basics: Acid transactions. https://towardsdatascience.com/database-basics-acid-transactions-bf4d38bd8e26. Accessed: 12-Aug-2022. [Cited on page 18.]

[28] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377387, jun 1970. [Cited on page 18.]

[29] Mysql server. https://www.mysql.com. Accessed: 2018-01-28. [Cited on pages 18 and 30.]

[30] Xiaojie Yang. Analysis of dbms: Mysql vs postgresql. *The Bachelors Thesis Information Technology program Kemi 2011*, 2011. [Cited on page 18.]

[31] Baron Schwartz, Peter Zaitsev, and Vadim Tkachenko. *High performance MySQL: optimization, backups, and replication*. " O'Reilly Media, Inc.", 2012. [Cited on page 19.]

[32] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):1–26, 2008. [Cited on page 19.]

[33] cloudduggu.com. Cassandra architecture, 2022. [Cited on page 19.]

[34] Lars George. *HBase: the definitive guide: random access to your planet-size data*. " O'Reilly Media, Inc.", 2011. [Cited on page 19.]

[35] Backblaze. Hard drive data and stats, Sep 2020. [Cited on pages 21 and 83.]

[36] Bjorn Andersen and Tom Fagerhaug. *Root cause analysis*. Quality Press, 2006. [Cited on page 23.]

[37] Christopher Kluse. Failure modes and effects analysis (fmea): Factors affecting execution and implementation of the fmea and an alternate method for process risk assessment. *Journal of Management & Engineering Integration*, 10(1):106–116, 2017. [Cited on page 23.]

[38] Sepideh Safari, Mohsen Ansari, Heba Khdr, Pourya Gohari-Nazari, Sina Yari-Karin, Amir Yeganeh-Khaksar, Shaahin Hessabi, Alireza Ejlali, and Jörg Henkel. A survey of fault-tolerance techniques for embedded systems from the perspective of power, energy, and thermal issues. *IEEE Access*, 10:12229–12251, 2022. [Cited on page 24.]

[39] Ana Gainaru and Franck Cappello. *Errors and Faults*, pages 89–144. Springer International Publishing, Cham, 2015. [Cited on page 24.]

[40] Roberto Natella, Domenico Cotroneo, and Henrique S. Madeira. Assessing dependability with software fault injection: A survey. *ACM Comput. Surv.*, 48(3), feb 2016. [Cited on page 24.]

[41] B Bockelman, T Cartwright, J Frey, E M Fajardo, B Lin, M Selmeci, T Tannenbaum, and M Zvada. Commissioning the htcondor-ce for the open science grid. *Journal of Physics: Conference Series*, 664(6):062003, 2015. [Cited on page 25.]

[42] Ruth Pordes, Don Petravick, Bill Kramer, Doug Olson, Miron Livny, Alain Roy, Paul Avery, Kent Blackburn, Torre Wenaus, Frank Wrthwein, Ian Foster, Rob Gardner, Mike Wilde, Alan Blatecky, John McGee, and Rob Quick. The open science grid. *Journal of Physics: Conference Series*, 78(1):012057, 2007. [Cited on pages 27, 75, and 105.]

[43] Charles E. Grant, Timothy L. Bailey, and William Stafford Noble. FIMO: scanning for occurrences of a given motif. *Bioinformatics*, 27(7):1017–1018, 02 2011. [Cited on page 29.]

[44] Geoff Macintyre, James Bailey, Izhak Haviv, and Adam Kowalczyk. is-rsnp: a novel technique for in silico regulatory snp detection. *Bioinformatics*, 26(18):i524–i530, 2010. [Cited on page 29.]

[45] Sunil Kumar, Giovanna Ambrosini, and Philipp Bucher. Snp2tfbs–a database of regulatory snps affecting predicted transcription factor binding site affinity. *Nucleic acids research*, 45(D1):D139–D144, 2017. [Cited on page 30.]

[46] Malin C Andersen, Pär G Engström, Stuart Lithwick, David Arenillas, Per Eriksson, Boris Lenhard, Wyeth W Wasserman, and Jacob Odeberg. In silico detection of sequence variations modifying transcriptional regulation. *PLoS computational biology*, 4(1):e5, 2008. [Cited on page 30.]

[47] Dilmi Perera, Diego Chacon, Julie AI Thoms, Rebecca C Poulos, Adam Shlien, Dominik Beck, Peter J Campbell, John E Pimanda, and Jason WH Wong. Oncocis: annotation of cis-regulatory mutations in cancer. *Genome biology*, 15:1–14, 2014. [Cited on page 30.]

[48] Laura Clarke, Xiangqun Zheng-Bradley, Richard Smith, Eugene Kulesha, Chunlin Xiao, Iliana Toneva, Brendan Vaughan, Don Preuss, Rasko Leinonen, Martin Shumway, et al. The 1000 genomes project: data management and community access. *Nature methods*, 9(5):459–462, 2012. [Cited on page 30.]

[49] Peter M Haverty, Ulla Hansen, and Zhiping Weng. Computational inference of transcriptional regulatory networks from expression profiling and transcription factor binding site identification. *Nucleic acids research*, 32(1):179–188, 2004. [Cited on page 30.]

[50] Sean D. Mooney, Vidhya G. Krishnan, and Uday S. Evani. Bioinformatic tools for identifying disease gene and snp candidates. *Methods Mol Biol*, 628:307–319, 2010. 20238089[pmid]. [Cited on page 30.]

[51] dbsnp short genetic variations. https://www.ncbi.nlm.nih.gov/SNP/. Accessed: 2018-01-03. [Cited on pages 30 and 38.]

[52] Matthew D. Mailman, Michael Feolo, Yumi Jin, Masato Kimura, Kimberly Tryka, Rinat Bagoutdinov, Luning Hao, Anne Kiang, Justin Paschall, Lon Phan, Natalia Popova, Stephanie Pretel, Lora Ziyabari, Yu Shao, Zhen Y. Wang, Karl Sirotkin, Minghong Ward, Michael Kholodov, Kerry Zbicz, Jeffrey Beck, Michael Kimelman, Sergey Shevelev, Don Preuss, Eugene Yaschenko, Alan Graeff, James Ostell, and Stephen T. Sherry. The ncbi dbgap database of genotypes and phenotypes. *Nat Genet*, 39(10):1181–1186, Oct 2007. 17898773[pmid]. [Cited on page 30.]

[53] Shuhui Song, Dongmei Tian, Cuiping Li, Bixia Tang, Lili Dong, Jingfa Xiao, Yiming Bao, Wenming Zhao, Hang He, and Zhang Zhang. Genome variation map: a data repository of genome variations in big data center. *Nucleic Acids Research*, 46(D1):D944–D949, 2018. [Cited on page 30.]

[54] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, SIGMOD '79, pages 23–34, New York, NY, USA, 1979. ACM. [Cited on page 30.]

[55] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, march 1976. [Cited on page 30.]

[56] Ameya Nayak, Anil Poriya, and Dikshay Poojary. Type of nosql databases and its comparison with relational databases. *International Journal of Applied Information Systems*, 5(4):16–19, 2013. [Cited on page 30.]

[57] Apache Software Foundation. Cassandra. http://cassandra.apache.org/. accessed: 12-Aug-2018, version: 2.1.14. [Cited on page 30.]

[58] 7 reasons why netflix uses cassandra databases. https://www.jcount.com/7-reasons-netflix-uses-cassandra-databases/. Accessed: 2017-09-20. [Cited on page 30.]

[59] Project Voldemort. Voldemort. http://www.project-voldemort.com/voldemort/. date : 2017-08-30, version : 1.10.25. [Cited on page 30.]

[60] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. In *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 06)*, Seattle, WA, 2006. USENIX Association. [Cited on page 30.]

[61] G. Vaish. *Getting Started with Nosql*. Packt Publishing, 2013. [Cited on page 31.]

[62] Rick Cattell. Scalable sql and nosql data stores. *SIGMOD Rec.*, 39(4):12–27, May 2011. [Cited on page 31.]

[63] John Klein, Ian Gorton, Neil Ernst, Patrick Donohoe, Kim Pham, and Chrisjan Matser. Performance evaluation of nosql databases: A case study. In *Proceedings of the 1st Workshop on Performance Analysis of Big Data Systems*, PABS '15, pages 5–10, New York, NY, USA, 2015. ACM. [Cited on page 31.]

[64] Y. Li and S. Manoharan. A performance comparison of sql and nosql databases. In *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 15–19, Aug 2013. [Cited on page 31.]

[65] W. Puangsaijai and S. Puntheeranurak. A comparative study of relational database and key-value database for big data applications. In *2017 International Electrical Engineering Congress (iEECON)*, pages 1–4, 2017. [Cited on page 31.]

[66] A comparison between cassandra and mysql. https://adataanalyst.com/data-analysis-resources/a-comparison-between-cassandra-and-mysql/, 2019. Accessed: 12-Aug-2022. [Cited on page 31.]

[67] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. [Cited on page 32.]

[68] Robert C Gentleman, Vincent J Carey, Douglas M Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, et al. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, 5(10):1–16, 2004. [Cited on page 32.]

[69] Kushal K Dey, Dongyue Xie, and Matthew Stephens. A new sequence logo plot to highlight enrichment and depletion. *BMC bioinformatics*, 19(1):1–9, 2018. [Cited on page 32.]

[70] Jianhong Ou, Scot A Wolfe, Michael H Brodsky, and Lihua Julie Zhu. motifstack for the analysis of transcription factor binding site evolution. *Nature methods*, 15(1):8–9, 2018. [Cited on page 32.]

[71] Spencer L Nystrom and Daniel J McKay. Memes: A motif analysis environment in r using tools from the meme suite. *PLoS Computational Biology*, 17(9):e1008991, 2021. [Cited on page 32.]

[72] MJ Bly. Ghostscript–postscript interpreter/previewer. *Starlink User Note*, 197, 1997. [Cited on page 32.]

[73] Henry Pratt and Zhiping Weng. Logojs: a javascript package for creating sequence logos and embedding them in web applications. *Bioinformatics*, 36(11):3573–3575, 2020. [Cited on page 32.]

[74] Hana Mallek, Faiza Ghozzi, Olivier Teste, and Faiez Gargouri. Bigdimetl with nosql database. *Procedia Computer Science*, 126:798–807, 2018. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 22nd International Conference, KES-2018, Belgrade, Serbia. [Cited on page 32.]

[75] Ardhian Agung Yulianto. Extract transform load (etl) process in distributed database academic data warehouse. *APTIKOM Journal on Computer Science and Information Technologies*, 4(2):61–68, 2019. [Cited on page 32.]

[76] Dhruba Borthakur. The hadoop distributed file system: Architecture and design. *Hadoop Project Website*, 11(2007):21, 2007. [Cited on page 32.]

[77] Alkis Simitsis, Chetan Gupta, Song Wang, and Umeshwar Dayal. Partitioning real-time etl workflows. In *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*, pages 159–162. IEEE, 2010. [Cited on page 32.]

[78] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. [Cited on pages 32 and 33.]

[79] Shruti Tekadpande and Mrs Leena Deshpande. A survey on hadoop technology to develop etl for efficient datawarehouse. *Journal Impact Factor*, 6(1):11–17, 2015. [Cited on page 33.]

[80] Amil Ahmad Ilham, Syahrul Usman, et al. Performance analysis of extract, transform, load (etl) in apache hadoop atop nas storage using iscsi. In *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*, pages 1–5. IEEE, 2017. [Cited on page 33.]

[81] enlyft. Companies using apache cassandra. https://enlyft.com/tech/products/apache-cassandra, 2022. Online; Accessed: 12-Aug-2022. [Cited on page 33.]

[82] Ink Sky Wheel. How many mysql instances are running in the world? here's a copy of the data. https://chowdera.com/2022/168/202206171023055567.html, 2022. Online; Accessed: 12-Aug-2022. [Cited on page 33.]

[83] Elastic Inc. Elastic reports strong second quarter fiscal 2022 financial results. https://www.elastic.co/about/press/elastic-reports-strong-second-quarter-fiscal-2022-financial-results, 2022. Online; Accessed: 12-Aug-2022. [Cited on page 33.]

[84] Myung-Hoon Cha, Dong-Oh Kim, Hong-Yeon Kim, and Young-Kyun Kim. Adaptive metadata rebalance in exascale file system. *The Journal of Supercomputing*, 73(4):1337–1359, Apr 2017. [Cited on page 33.]

[85] Sudhakar K. and Shivendra Pandey. *Advances in Intelligent Systems and Computing*, volume 710, chapter An Approach to Improve Load Balancing in Distributed

Storage Systems for NoSQL Databases: MongoDB, pages 529–538. Proceedings of ICCAN 2017, 04 2018. [Cited on page 34.]

[86] Xiangdong Huang, Jianmin Wang, Yu Zhong, and Philip S. Yu. Optimizing data partition for nosql cluster. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pages 962–969, 2015. [Cited on page 34.]

[87] Ceph Inc. Diskprediction module. https://docs.ceph.com/en/quincy/mgr/diskprediction/, 2022. Online; Accessed: 12-Aug-2022. [Cited on page 34.]

[88] Spyridon Chouliaras and Stelios Sotiriadis. Real-time anomaly detection of nosql systems based on resource usage monitoring. *IEEE Transactions on Industrial Informatics*, 16(9):6042–6049, 2020. [Cited on page 34.]

[89] Flora Karniavoura and Kostas Magoutis. A measurement-based approach to performance prediction in nosql systems. In *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 255–262, 2017. [Cited on page 34.]

[90] Ding Yuan, Yu Luo, Xin Zhuang, Guilherme Renna Rodrigues, Xu Zhao, Yongle Zhang, Pranay U Jain, and Michael Stumm. Simple testing can prevent most critical failures: An analysis of production failures in distributed {Data-Intensive} systems. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 249–265, 2014. [Cited on page 34.]

[91] Huan Ke, Haryadi S. Gunawi, Dominic Manno, David Bonnie, and Bradley W. Settlemyer. Fractional-overlap declustered parity: Evaluating reliability for storage systems. In *2020 IEEE/ACM Fifth International Parallel Data Systems Workshop (PDSW)*, pages 1–6, 2020. [Cited on page 34.]

[92] Accessed: Jan 18 2024. [Cited on page 35.]

[93] Luis Velasco and Danish Rafique. Fault management based on machine learning. In *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3. IEEE, 2019. [Cited on pages xiii, 35, and 59.]

[94] NCBI dbSNP. dbsnp short genetic variations. https://www.ncbi.nlm.nih.gov/projects/SNP/snp_summary.cgi?view+summary=view+summary&build_id=144, 2015. [Cited on page 38.]

[95] Genome reference consortium human build 38. https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.26/. Accessed: 2016-05-05. [Cited on page 38.]

[96] Martin Kreitman. Nucleotide polymorphism at the alcohol dehydrogenase locus of drosophila melanogaster. *Nature*, 304:412 EP –, Aug 1983. [Cited on page 39.]

[97] David Altshuler, Victor J. Pollara, Chris R. Cowles, William J. Van Etten, Jennifer Baldwin, Lauren Linton, and Eric S. Lander. An snp map of the human genome generated by reduced representation shotgun sequencing. *Nature*, 407:513 EP –, Sep 2000. [Cited on page 39.]

[98] MySQL. Limits on table column count and row size. https://dev.mysql.com/doc/mysql-reslimits-excerpt/5.7/en/column-count-limit.html. accessed: 12-Aug-2022. [Cited on page 39.]

[99] Weiping Qu and Stefan Dessloch. Distributed snapshot maintenance in wide-column nosql databases using partitioned incremental etl pipelines. *Information Systems*, 70:48–58, 2017. Advances in databases and Information Systems. [Cited on page 42.]

[100] Apache cassandra nosql performance benchmarks. https://academy.datastax.com/planet-cassandra/nosql-performance-benchmarks. Accessed: 2017-01-20. [Cited on page 42.]

[101] Improve mysql insert performance. https://kvz.io/blog/improve-mysql-insert-performance.html. Accessed: 2024-07-08. [Cited on page 43.]

[102] Apache Software foundataion. Apache lucene. https://lucene.apache.org/. Accessed: 30-Nov-2023. [Cited on page 44.]

[103] Gary D. Stormo, Thomas D. Schneider, Larry Gold, and Andrzej Ehrenfeucht. Use of the 'perceptron' algorithm to distinguish translational initiation sites in e. coli. *Nucleic Acids Research*, 10(9):2997–3011, 1982. [Cited on page 46.]

[104] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0. [Cited on page 47.]

[105] Pouria Pirzadeh, Junichi Tatemura, Oliver Po, and Hakan Hacıgümüş. Performance evaluation of range queries in key value stores. *Journal of Grid Computing*, 10(1):109–132, Mar 2012. [Cited on page 48.]

[106] Understanding mysql architecture. https://www.rathishkumar.in/2016/04/understanding-mysql-architecture.html. Accessed: 2023-10-30. [Cited on pages xiii and 49.]

[107] Amar Kapadia, Sreedhar Varma, and Kris Rajana. *Implementing Cloud Storage with OpenStack Swift*. Packt Publishing Birmingham, UK, 2014. [Cited on page 50.]

[108] Elijah Meeks. *D3. js in Action: Data visualization with JavaScript*. Simon and Schuster, 2017. [Cited on page 50.]

[109] Ændrew H Rininsland, Michael Heydt, and Pablo Navarro Castillo. *D3. js: cutting-edge data visualization*. Packt Publishing Ltd, 2017. [Cited on page 50.]

[110] Christopher Harrison and Rebecca Hudson. Source code for atsnp composite logo plots. https://github.com/RebeccaHudson/ss_search_viewer. Accessed: 2023-12-17. [Cited on page 50.]

[111] Elastic Inc. Rest apis. https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html. accessed: 12-Aug-2022. [Cited on page 51.]

[112] Bharat Singhal and Alok Aggarwal. Etl, elt and reverse etl: A business case study. In *2022 Second International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE)*, pages 1–4, 2022. [Cited on page 51.]

[113] Jehan-François Pâris and Darrell D. E. Long. Using device diversity to protect data against batch-correlated disk failures. In *Proceedings of the Second ACM Workshop on Storage Security and Survivability*, StorageSS '06, page 4752, New York, NY, USA, 2006. Association for Computing Machinery. [Cited on pages 54 and 124.]

[114] Shujie Han, Patrick PC Lee, Fan Xu, Yi Liu, Cheng He, and Jiongzhou Liu. An {In-Depth} study of correlated failures in production {SSD-Based} data centers. In *19th*

USENIX Conference on File and Storage Technologies (FAST 21), pages 417–429, 2021. [Cited on page 54.]

[115] Abdullah Al Mamun, GuoXiao Guo, and Chao Bi. *Hard disk drive: mechatronics and control*. CRC press, 2017. [Cited on page 58.]

[116] Sara Antomarioni, Filippo Emanuele Ciarapica, and Maurizio Bevilacqua. Data-driven approach to predict the sequence of component failures: a framework and a case study on a process industry. *International Journal of Quality & Reliability Management*, ahead-of-print, 01 2022. [Cited on page 58.]

[117] Seagate Product Marketing. Get smart for reliability. Technical report, Technical report, Seagate Technology Paper, 1999. [Cited on pages 58, 59, and 60.]

[118] Bianca Schroeder and Garth A. Gibson. Understanding disk failure rates: What does an mttf of 1,000,000 hours mean to you? *ACM Trans. Storage*, 3(3):8es, oct 2007. [Cited on page 59.]

[119] J.G. Elerath and S. Shah. Server class disk drives: how reliable are they? In *Annual Symposium Reliability and Maintainability, 2004 - RAMS*, pages 151–156, Jan 2004. [Cited on page 59.]

[120] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz André Barroso. Failure trends in a large disk drive population. *Proceedings of the Conference on File and Storage Technologies (FAST)*, 2007. [Cited on page 59.]

[121] Brian Beach. Hard drive smart stats, Sep 2020. [Cited on page 59.]

[122] Andy Klein. How Long Do Disk Drives Last? — backblaze.com. https://www.backblaze.com/blog/how-long-do-disk-drives-last/. [Accessed 25-06-2024]. [Cited on pages xiii, 60, 61, and 118.]

[123] Wyatts derivative work: McSush. File:Bathtub curve.svg - Wikimedia Commons — commons.wikimedia.org. https://commons.wikimedia.org/wiki/File:Bathtub_curve.svg. [Accessed 25-06-2024]. [Cited on pages xiii, 60, and 118.]

[124] Nicolas Aussel, Samuel Jaulin, Guillaume Gandon, Yohan Petetin, Eriza Fazli, and Sophie Chabridon. Predictive models of hard drive failures based on operational data. In *ICMLA 2017 : 16th IEEE International Conference On Machine Learning And*

Applications, pages 619 – 625, Cancun, Mexico, 2017. IEEE Computer Society. [Cited on pages 63 and 72.]

[125] Usama M Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, et al. Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, volume 96, pages 82–88, 1996. [Cited on page 64.]

[126] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011. [Cited on pages 64, 67, and 70.]

[127] Errin W Fulp, Glenn A Fink, and Jereme N Haack. Predicting computer system failures using support vector machines. *WASL*, 8:5–5, 2008. [Cited on page 65.]

[128] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. [Cited on page 65.]

[129] Will Koehrsen. Random forest simple explanation, Aug 2020. [Cited on page 66.]

[130] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144152, New York, NY, USA, 1992. Association for Computing Machinery. [Cited on page 66.]

[131] Pier Paolo Ippolito. Support vector machines, Jun 2019. [Cited on page 66.]

[132] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144152, New York, NY, USA, 1992. Association for Computing Machinery. [Cited on page 67.]

[133] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1), December 2012. [Cited on page 67.]

[134] Australian Transport Assessment and Planning. 6. forecasting and evaluation, Oct 2019. [Cited on page 68.]

[135] Helmut Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005. [Cited on pages 67 and 68.]

[136] Shay Palachy. Stationarity in time series analysis, Sep 2019. [Cited on page 69.]

[137] James H Stock and Mark W Watson. Vector autoregressions. *Journal of Economic perspectives*, 15(4):101–115, 2001. [Cited on page 68.]

[138] Douglas Holtz-Eakin, Whitney Newey, and Harvey S Rosen. Estimating vector autoregressions with panel data. *Econometrica: Journal of the econometric society*, pages 1371–1395, 1988. [Cited on page 68.]

[139] KDnuggets. The 5 most useful techniques to handle imbalanced datasets. [Cited on page 70.]

[140] MC.AI. Confusion matrix no more confusing, Sep 2018. [Cited on page 71.]

[141] DeepAI. Evaluation metrics, May 2019. [Cited on page 71.]

[142] Alex Smolyak, Orr Levy, Irena Vodenska, Sergey Buldyrev, and Shlomo Havlin. Mitigation of cascading failures in complex networks. *Scientific reports*, 10(1):16124, 2020. [Cited on page 71.]

[143] Jianwei Wang. Mitigation of cascading failures on complex networks. *Nonlinear Dynamics*, 70:1959–1967, 2012. [Cited on pages 71 and 72.]

[144] Lucas D Valdez, Louis Shekhtman, Cristian E La Rocca, Xin Zhang, Sergey V Buldyrev, Paul A Trunfio, Lidia A Braunstein, and Shlomo Havlin. Cascading failures in complex networks. *Journal of Complex Networks*, 8(2):cnaa013, 2020. [Cited on page 71.]

[145] Jianwei Wang. Robustness of complex networks with the local protection strategy against cascading failures. *Safety Science*, 53:219–225, 2013. [Cited on pages 71 and 72.]

[146] Ziyang Jin, Dongli Duan, and Ning Wang. Cascading failure of complex networks based on load redistribution and epidemic process. *Physica A: Statistical Mechanics and its Applications*, 606:128041, 2022. [Cited on page 71.]

[147] Ji Wang, Weidong Bao, Lei Zheng, Xiaomin Zhu, and Philip S. Yu. An attention-augmented deep architecture for hard drive status monitoring in large-scale storage systems. *ACM Trans. Storage*, 15(3), August 2019. [Cited on page 72.]

[148] Jing Shen, Jian Wan, Se-Jung Lim, and Lifeng Yu. Random-forest-based failure prediction for hard disk drives. *International Journal of Distributed Sensor Networks*, 14(11):1550147718806480, 2018. [Cited on page 73.]

[149] Jing Li, Rebecca J Stones, Gang Wang, Xiaoguang Liu, Zhongwei Li, and Ming Xu. Hard drive failure prediction using decision trees. *Reliability Engineering & System Safety*, 164:55–65, 2017. [Cited on page 73.]

[150] Ying Zhao, Xiang Liu, Siqing Gan, and Weimin Zheng. Predicting disk failures with hmm and hsmm-based approaches. In *Predicting disk failures with HMM and HSMM-based approaches*, volume 6171, pages 390–404, 07 2010. [Cited on page 73.]

[151] Argonne National Laboratory. Argonne national laboratory - theta hpc cluster. https://www.alcf.anl.gov/alcf-resources/theta, 2023. [Cited on page 74.]

[152] Dale Carder. Rrdtool scalability. http://net.doit.wisc.edu/~dwcarder/rrdcache/, 2007. Online; Accessed 9-Sept-2018. [Cited on page 75.]

[153] Inc. The Cacti Group. Cacti Monitoring Tool. https://www.cacti.net/, 2018. Online; Accessed: 24-Sept-2018. [Cited on page 75.]

[154] Matt Massie, Bernard Li, Brad Nicholes, Vladimir Vuksan, Robert Alexander, Jeff Buchbinder, Frederiko Costa, Alex Dean, Dave Josephsen, Peter Phaal, and Daniel Pocock. *Monitoring with Ganglia*. OŔeilly Media, Inc., 1 edition, 2012. [Cited on page 75.]

[155] Wolfgang Barth. *Nagios: System and Network Monitoring*. No Starch Press, San Francisco, CA, USA, 2006. [Cited on page 75.]

[156] Zabbix LLC. Zabbix monitoring system. https://www.zabbix.com/, 2018. Online; Accessed 24-Sept-2018. [Cited on page 75.]

[157] J. Case, M. Fedor, M. Schoffstall, and J. Davin. Simple network management protocol (snmp). RFC 1157, Internet Engineering Task Force, May 1990. [Cited on page 75.]

[158] Sajjad Haider and Babar Nazir. Fault tolerance in computational grids: perspectives, challenges, and issues. *SpringerPlus*, 5, 12 2016. [Cited on page 75.]

[159] J. M. Brandt, A. C. Gentile, D. J. Hale, and P. P. Pebay. Ovis: a tool for intelligent, real-time monitoring of computational clusters. In *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, pages 8 pp.–, April 2006. [Cited on page 75.]

[160] T. Evans, W. L. Barth, J. C. Browne, R. L. DeLeon, T. R. Furlani, S. M. Gallo, M. D. Jones, and A. K. Patra. Comprehensive resource use monitoring for hpc systems with tacc stats. In *2014 First International Workshop on HPC User Support Tools*, pages 13–21, Nov 2014. [Cited on page 75.]

[161] Qiang Guan, Ziming Zhang, and Song Fu. Ensemble of bayesian predictors and decision trees for proactive failure management in cloud computing systems. *JCM*, 7:52–61, 2012. [Cited on page 76.]

[162] Jiechao Gao, Haoyu Wang, and Haiying Shen. Task failure prediction in cloud data centers using deep learning. *IEEE Transactions on Services Computing*, 15(3):1411–1422, 2022. [Cited on page 76.]

[163] Carla Sauvanaud, Mohamed Kaniche, Karama Kanoun, Kahina Lazri, and Guthemberg Da Silva Silvestre. Anomaly detection and diagnosis for cloud services: Practical experiments and lessons learned. *Journal of Systems and Software*, 139:84–106, 2018. [Cited on page 76.]

[164] Teerat Pitakrat, Duan Okanovi, Andr van Hoorn, and Lars Grunske. Hora: Architecture-aware online failure prediction. *Journal of Systems and Software*, 137:669–685, 2018. [Cited on page 76.]

[165] Olumuyiwa Ibidunmoye, Francisco Hernández-Rodriguez, and Erik Elmroth. Performance anomaly detection and bottleneck identification. *ACM Computing Surveys (CSUR)*, 48(1):1–35, 2015. [Cited on page 76.]

[166] Yujun Chen, Xian Yang, Qingwei Lin, Hongyu Zhang, Feng Gao, Zhangwei Xu, Yingnong Dang, Dongmei Zhang, Hang Dong, Yong Xu, et al. Outage prediction and diagnosis for cloud service systems. In *The World Wide Web Conference*, pages 2659–2665, 2019. [Cited on page 76.]

[167] Eric A Brewer. Kubernetes and the path to cloud native. In *Proceedings of the sixth ACM symposium on cloud computing*, pages 167–167, 2015. [Cited on page 76.]

[168] C. Chang, S. Yang, E. Yeh, P. Lin, and J. Jeng. A kubernetes-based monitoring platform for dynamic cloud resource provisioning. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, Dec 2017. [Cited on page 77.]

[169] L. Wei, C. H. Foh, B. He, and J. Cai. Towards efficient resource allocation for heterogeneous workloads in iaas clouds. *IEEE Transactions on Cloud Computing*, 6(1):264–275, Jan 2018. [Cited on page 77.]

[170] Eduardo Pinheiro, Ricardo Bianchini, Enrique V. Carrera, and Taliver Heath. Load balancing and unbalancing for power and performance in cluster-based systems, 2001. [Cited on page 77.]

[171] Alex Homer, John Sharp, Larry Brader, Masashi Narumoto, and Trent Swanson. *Cloud Design Patterns: Prescriptive Architecture Guidance for Cloud Applications*. Microsoft patterns & practices, 2014. [Cited on page 77.]

[172] H S Guruprasad and bhavani B H. Resource provisioning techniques in cloud computing environment: A survey. *International Journal of Research in Computer and Communication Technology*, 3:395–401, 03 2014. [Cited on page 77.]

[173] San Diego Supercomputer Center, UC San Diego. Tscc purchase program. http://www.sdsc.edu/services/hpc/tscc-purchase.html, 2018. Online; Accessed 3-Dec-2018. [Cited on page 77.]

[174] Gabriel Mateescu, Wolfgang Gentzsch, and Calvin J. Ribbens. Hybrid computing-where hpc meets grid and cloud computing. *Future Generation Computer Systems*, 27(5):440 – 453, 2011. [Cited on page 77.]

[175] Abel Souza, Kristiaan Pelckmans, and Johan Tordsson. A hpc co-scheduler with reinforcement learning. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 126–148. Springer, 2021. [Cited on page 77.]

[176] Yuping Fan, Zhiling Lan, Taylor Childers, Paul Rich, William Allcock, and Michael E Papka. Deep reinforcement agent for scheduling in hpc. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 807–816. IEEE, 2021. [Cited on page 77.]

[177] Connor Imes, Steven Hofmeyr, and Henry Hoffmann. Energy-efficient application resource scheduling using machine learning classifiers. In *Proceedings of the 47th International Conference on Parallel Processing*, pages 1–11, 2018. [Cited on page 77.]

[178] Gonzalo P. Rodrigo, Erik Elmroth, Per-Olov Östberg, and Lavanya Ramakrishnan. Enabling workflow-aware scheduling on hpc systems. In *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '17, pages 3–14, New York, NY, USA, 2017. ACM. [Cited on page 77.]

[179] G. Juve, B. Tovar, R. F. d. Silva, D. Krl, D. Thain, E. Deelman, W. Allcock, and M. Livny. Practical resource monitoring for robust high throughput computing. In *2015 IEEE International Conference on Cluster Computing*, pages 650–657, Sept 2015. [Cited on page 78.]

[180] Serhiy Yanchuk and Giovanni Giacomelli. Spatio-temporal phenomena in complex systems with time delays. *Journal of Physics A: Mathematical and Theoretical*, 50(10):103001, 2017. [Cited on page 78.]

[181] David Jauk, Dai Yang, and Martin Schulz. Predicting faults in high performance computing systems: An in-depth survey of the state-of-the-practice. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '19, New York, NY, USA, 2019. Association for Computing Machinery. [Cited on page 78.]

[182] Anwesha Das, Frank Mueller, Paul Hargrove, Eric Roman, and Scott Baden. Doomsday: predicting which node will fail when on supercomputers. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 108–121. IEEE, 2018. [Cited on page 78.]

[183] Elizabeth Bautista, Nitin Sukhija, and Siqi Deng. Shasta log aggregation, monitoring and alerting in hpc environments with grafana loki and servicenow. In *2022 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 602–610. IEEE, 2022. [Cited on page 78.]

[184] Burak Aksar, Efe Sencan, Benjamin Schwaller, Omar Aaziz, Vitus J Leung, Jim Brandt, Brian Kulis, Manuel Egele, and Ayse K Coskun. Prodigy: Towards unsupervised anomaly detection in production hpc systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14, 2023. [Cited on page 78.]

[185] Alja Ferencek and Mirjana Kljaji Bortnar. Data quality assessment in product failure prediction models. *Journal of Decision Systems*, 29(sup1):79–86, 2020. [Cited on page 78.]

[186] 104th US. Congress. Health insurance portability and accountability act. In *United States of America,Federal Register, HIPAA Statute*. United States of America, Federal Statute, http://legislink.org/us/stat-110-1936, 1996. [Cited on page 105.]

[187] Inc. MongoDB. MongoDB. https://www.mongodb.com/, 2018. Online; Accessed: 14-Dec-2018. [Cited on page 107.]

[188] Moshe Dubiner. Bucketing coding and information theory for the statistical high-dimensional nearest-neighbor problem. *IEEE Transactions on Information Theory*, 56(8):4166–4179, 2010. [Cited on page 110.]

[189] Gilles Louppe. *Understanding Random Forests: From Theory to Practice*. PhD thesis, University of Liege, Belgium, 10 2014. arXiv:1407.7502. [Cited on page 111.]

[190] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. [Cited on page 111.]

[191] Ryan Adamson, Tim Osborne, Corwin Lester, and Rachel Palumbo. Stream: A scalable federated hpc telemetry platform. *osti.gov*, 5 2023. [Cited on page 122.]

[192] The black hole node syndrome: How to prevent a productivity killer. https://www.hpcwire.com/2012/09/24/the_black_hole_node_syndrome_how_to_prevent_a_productivity_killer/. Accessed: 2023-09-17. [Cited on page 123.]

[193] Po-Ru Loh, Gleb Kichaev, Steven Gazal, Armin P Schoech, and Alkes L Price. Mixed-model association for biobank-scale datasets. *Nature genetics*, 50(7):906–908, 2018. [Cited on page 123.]