FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Tecnologias e modelos de suporte a analytics sobre séries temporais

Paulo Manuel da Silva Faria

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Doutor João Moreira

Junho de 2017

Tecnologias e modelos de suporte a analytics sobre séries temporais

Paulo Manuel da Silva Faria

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Doutor Rui Camacho

Arguente: Doutor José Moreira

Orientador: Doutor João Pedro Mendes Moreira

_____

23 de Junho de 2017

Resumo

Numa sociedade em que a informação é transmitida e consumida em grandes quantidades e com enorme rapidez é necessário tratá-la da melhor maneira e maximizando os proveitos decorrentes da sua exploração. Com este trabalho pretende-se indagar acerca da utilização de tecnologias capazes de recolher, modelar e tratar esses dados. É neste contexto que surge a necessidade de análise de séries temporais, identificando "como", "quando" e "onde" os clientes do operador interagem com os serviços, para conseguir controlar os vários momentos da utilização dos mesmos, resultando em grandes quantidades de dados que requerem tratamento.

Os objetivos são os de encontrar tecnologias e modelizações adequadas, de forma a utilizar técnicas analíticas sobre séries temporais, e atendendo a fatores como a escalabilidade, o desempenho, o custo das ferramentas, da infraestrutura física de suporte às ferramentas e o custo de operação do sistema. As interrogações realizadas a agregações de informação são, devido às quantidades e tipos de dados, dispendiosas em termos de recursos computacionais e de tempo gasto, o que causa transtornos, na medida em que existe a necessidade de serem utilizadas na obtenção de indicadores de desempenho para *Business Intelligence*[1] e estes devem estar disponíveis para a tomada de decisão. No caso do proponente, os operadores de telecomunicações distinguem-se cada vez mais pela qualidade de serviço que fornecem aos seus clientes, é importante definir metodologias e técnicas que permitam comparar as várias hipóteses de solução com o sistema existente para que o impacto possa ser a melhoria da qualidade do serviço e da experiência para os utilizadores deste tipo de serviços, ganhos ao nível da eficiência e para o operador na gestão dos dados.

Depois de ponderadas várias possibilidades selecionaram-se três alternativas, de acordo com os requisitos do utilizador -Postgres, Citus e Timescaledb- nas quais se modelaram e forneceram dados reais do proponente para serem submetidas a testes, quer de resposta às interrogações quer de monitorização das máquinas onde foram configuradas para avaliar o seu desempenho utilizando ferramentas já conhecidas na empresa (jmeter e zabbix). Após avaliação das alternativas, chegou-se à conclusão que a melhor seria o Citus, perante os resultados face às necessidades atuais e de escalabilidade da solução.

[1] Inteligência empresarial

Abstract

In a society where the data is transmitted and consumed in a large scale at great speed, it is important to find the best way to process it and maximize the outcome. The aim of this work is to find tecnologies that are able to gather, model and process the data. It is in this context that comes the necessity of time series analysis in order to identity "where", "when" and "how" the clients of the telecomunication company interact with the services, to control the moments of contact and resulting in alot of data that has to be processed.

The aim is to find tecnologies better suitable, using time series techniques, to factors like scalability, performance, cost of tools, physical support infrastructure and system operational costs. The queries performed to agregattion of information are, due to high amount and type of data, expensive in both computational resources and time. That is a problem because this data needs to be used for KPI's for Business Intelligence and those need to be available as fast as possible for the decision process. In the proponent case, the telecommunication companies, they diferentiate from one another through the quality of the service they offer to the customers, to do so it is important to define methodologies and techniques that can compare several alternatives versus what is currently implemmented and that way improve quality of services and get efficiency/economic benefits to the telecommunication company in their data management.

After searching for alternatives, 3 were selected according to the user requirements for this project -Postgres, Citus and Timescaledb- on which data was model and insered to match project's proponent real use case, then put under tests to see the query performance and monitoring the machines, using tools used by the company like jmeter and zabbix to get the results. After analysing the results, the conclusion is that the best choice is Citus, both to the current use case evaluated and thinking on the scalability of the solution.

Agradecimentos

Agradeço aos meus pais pela oportunidade e apoio para completar o mestrado, às minhas irmãs, colegas e amigos com quem convivi durante estes 5 anos. Obrigado ao professor João Moreira pela orientação e a todos os que me ajudaram na integração na Altice.

Paulo Faria

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

OLAP	Online analytical processing
SQL	Structured Query Language
CEM	Customer Experience Management
QUERY	Database Interrogation
JOIN	SQL Clause
SSH	Secure Shell
PSQL	PostgreSQL interactive terminal
XML	Extensible Markup Language

Capítulo 1

Introdução

Este documento resulta do trabalho realizado ao longo do 5ºano do Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto.

1.1 Contexto/Enquadramento

No caso dos operadores de telecomunicações, tratando-se de um negócio com forte concorrência, existe a necessidade da aposta na diferenciação pela qualidade de serviços oferecida aos seus clientes. Para alcançar estes padrões de qualidade é necessário monitorizar a relação com os clientes seguindo princípios de *CEM*. Estes princípios implicam o conhecimento profundo acerca dos consumidores de forma a poder adaptar a oferta de forma individualizada fortalecendo a marca e a lealdade dos clientes à mesma.

No âmbito da dissertação e bolsa surgiu a oportunidade de trabalhar em Aveiro durante 6 meses na Altice, em parceria com a Inova Ria, com o tema Tecnologias e modelos de suporte a *analytics* sobre séries temporais.

A Inova Ria é uma associação que funciona como um agente dinamizador para a área das tecnologias de Informação através do conhecimento e formação de pessoas, trabalhando em rede com empresas como Altice Labs.

A Altice labs é uma empresa que procura apoiar os seus clientes na construção de inovação tecnológica e criação de valor, de forma a melhorar a vida das pessoas.

Com este trabalho pretende-se indagar acerca da utilização de tecnologias capazes de recolher, modelar e tratar esses dados.

1.2 Motivação e Objectivos

A motivação do trabalho resulta da necessidade de identificar “como”, “quando” e “onde” os clientes do operador interagem com os serviços, para conseguir controlar os vários momentos da utilização dos mesmos.

Desta monitorização resultam grandes quantidades de dados que requerem tecnologias e modelizações adequadas, de forma a utilizar técnicas analícas sobre séries temporais, e atendendo a fatores como a escalabilidade, o desempenho, o custo das ferramentas e da infraestrutura física de suporte às mesmas e o custo de operação do sistema.

Neste primeiro capítulo faz-se a introdução ao problema em causa.

No capítulo seguinte faz-se a revisão bibliográfica acerca de tecnologias de persistência aplicadas a séries temporais, subdividindo a revisão em tipo de informação, tecnologias e modelos, séries temporais e benefícios, bem com as metodologias seguidas para testar as tecnologias.

No terceiro capítulo abordam-se os requisitos do utilizador e as tecnologias selecionadas para cumprir esses requisitos.

No quarto capítulo descreve-se a amostra de dados, ferramentas utilizadas na avaliação de tecnologias, plano de testes e resultados obtidos.

No último capítulo é realizado um balanço final do trabalho e solução encontrada para o problema.

Capítulo 2

Tecnologias de persistência aplicadas a séries temporais

Neste capítulo é apresentado o estado da arte sobre tecnologias de persistência aplicadas a séries temporais sendo apresentada a metodologia seguida para validação e a oportunidade de desenvolvimento.

Com este trabalho pretende-se investigar acerca da utilização de tecnologias capazes de recolher, modelar e tratar os dados recolhidos pela organização. É neste contexto que surge a necessidade de análise de séries temporais, para se conseguir maximizar os benefícios e reduzir os encargos.

2.1 Informação

O processamento de séries temporais e respetiva análise são importantes apesar da diversidade de dados provenientes de sensores. Essa diversidade requer a procura da tecnologia com maior capacidade para que esses recursos possam ser rentabilizados com consequente melhoria do respetivo negócio (Whipple 2016).

O Altaia, plataforma utilizada pela Altice, armazena dados relativos ao funcionamento da rede guardando a informação numa base de dados relacional, usando, para isso, formatos em estrela para representar um cubo (OLAP multidimensional) em que cada uma das faces corresponde a uma dimensão de informação do negócio. Os elementos de uma dimensão seguem uma hierarquia que facilita a análise do utilizador e permite as operações tipicas destes cenários: *slice and dice*, *drill down*, *roll up* e *pivot*.

- O *roll up* (Figura 1: Exemplo de roll up (roll_up)) efetua a agregação no cubo quer pela elevação de um conceito hierárquico da dimensão quer pela redução da dimensão ("Data warehousing OLAP").

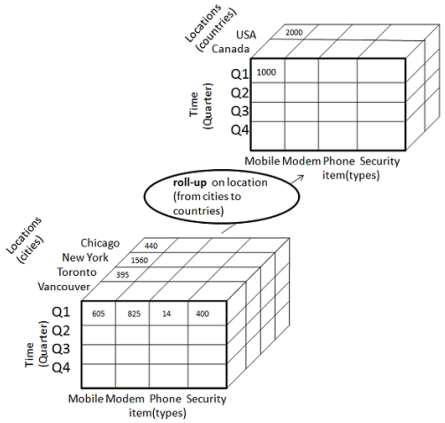


Figura 1: Exemplo de roll up

- O *drill down* efetua a operação oposta ao *roll up.*
- O *slice (*Figura 2: Exemplo de slice(slice)) faz um corte de uma certa dimensão do cubo para gerar um novo cubo ("Data warehousing OLAP").

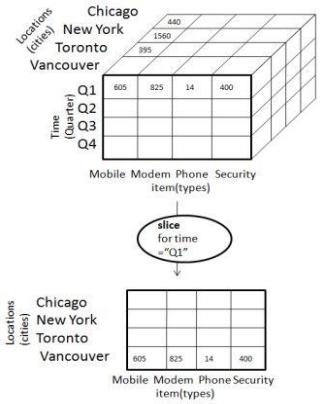


Figura 2: Exemplo de slice(slice)

- O *dice* seleciona 2 ou mais dimensões do cubo para gerar um novo cubo ("Data warehousing OLAP").

- O *pivot (*Figura 3: Exemplo de Pivot (pivot)) roda o eixo de forma a obter uma perspectiva da informação ("Data warehousing OLAP").

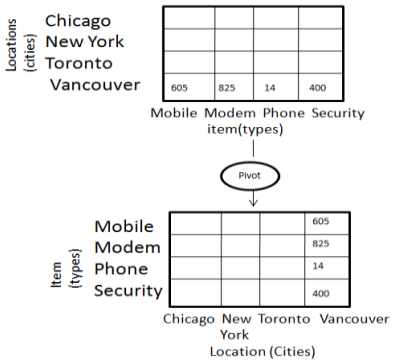


Figura 3: Exemplo de Pivot (pivot)

2.2 Tecnologias

O principal requisito funcional que as tecnologias de base de dados devem respeitar é a possibilidade de, durante o tempo da interrogação, permitir agregação temporal das séries em hierarquias para, deste modo, acelerar consultas e suportando as funções de agregação (Tabela 1: Funções de agregação).

Tabela 1: Funções de agregação

SUM	AVG	MAX	COUNT
COUNT_DISTINCT	MEDIAN	FIRST	LAST
STDDEV	VARIANCE	PERCENTILE_CONT	PERCENTILE_DISC

As agregações necessitam de ser armazenadas na base de dados em tabelas de factos relacionadas com os níveis superiores da hierarquia não necessitando assim da granularidade maior. Estas agregações recorrem ao *group by* que pode ter dois níveis- outer e inner query- podendo, devido à estrutura em cubo multidimensional, relacionar tempo com outros atributos da tabela de factos ou dimensões.

É importante neste projeto investigar e testar soluções tecnológicas de bases de dados adequadas a séries temporais e ao contexto da empresa. As seguintes tecnologias foram consideradas:

- **Cloudera Impala**

O *Cloudera Impala* é uma ferramenta *open source* que corre em Apache Hadoop permitindo a utilização de tecnologias de base de dados escaláveis e paralelas e que proporciona respostas mais rápidas às *queries* do que as tecnologias Hadoop (Cloudera 2017).

- ## Apache Spark

O *Apache Spark* é uma ferramenta que permite acelerar o processamento de grandes volumes de dados, combinando SQL, *streaming* e *analytics* funcionando em Hadoop, Mesos, entre outros ("Apache Spark").

- ## GreenPlum

O *GreenPlum* é um armazém de dados open source com capacidade e optimizado para análise de volumes de dados na ordem dos *petabytes* (GreenPlum).

- ## Blinkdb

O *Blinkdb* é uma ferramenta que agiliza as pesquisas SQL, em grandes quantidades de informação, permitindo aos utilizadores efetuarem uma cedência entre o tempo de resposta e a exatidão das interrogações SQL ("BlinkDB").

- ## Opentsdb

O *opentsdb* consiste numa tecnologia que utiliza TSDB(bases de dados temporais) em que cada base de dados recorre a uma base de dados *open source* chamada Hbase para guardar e ler dados de séries temporais ("opentsdb").

- ## Monetdb

O *Monetdb* é uma tecnologia de base de dados orientada a colunas e *open source,* permitindo a obtenção de elevados valores de desempenho para OLAP e extração de dados (monetdb).

- ## Eventql

O *Eventql* é uma tecnologia de base de dados orientada a colunas, para a recolha de eventos e análises em grande escala, correndo eficientemente queries SQL e *MapReduce* ("The database for large-scale event analytics" 2016).

- ## Postgres

O *PostgreSql* é uma tecnologia de base de dados relacional com experiência e boa reputação na área das tecnologias de bases de dados ("PostgreSql Documentation").

- ## Citus

O *Citus* é uma plataforma distribuída que extende o Postgres, permitindo o *sharding*(partição dos dados por várias bases de dados) e interrogações de análise em tempo real e escalável (citusdata).

- **Kudu**

O *Kudu* é uma tecnologia de base de dados analítica desenvolvida pela Apache. Organiza-se de forma orientada a colunas, o que lhe permite reduzir a quantidade de informação para operações de entrada/saída, tendo maior eficiência de codificação e compressão (APACHE).

- **Apache Hadoop com Apache Crunch**

O *Apache Hadoop* é uma plataforma *open source* para armazenamento e processamento distribuídos de grandes conjuntos de dados enquanto que o *Apache Crunch* é uma biblioteca Java que funciona em conjunto com o *Apache Hadoop* permitindo um melhor tratamento de tarefas, como a junção dos conjuntos de dados e encadeamento de procedimentos de manutenção da base de dados (Hortonworks; Beard 2014).

- **Gorilla Beringei**

O Gorilla é uma tecnologia de base de dados de séries temporais, em memória, que é atualmente utilizada pelo *Facebook* para guardar medidas de sistema associados a marcadores na análise da dados agregados e não individuais, com interrogações a executarem dentro dos 10 milisegundos (Pelkonen et al. 2015).

- **ElasticSearch**

O *Elasticsearch* é uma ferramenta de pesquisa baseada no *Apache Lucene*, conhecida pelas suas capacidades de pesquisa e indexação. Apresenta capacidade de armazenamento, pesquisa, análise de dados, estruturados ou não, métricas de séries temporais e manipulação do cálculo das métricas caso seja necessário *(*Elliot 2015*;* Barnsteiner*)*.

- **InfluxDB/InfluxData**

O *InfluxDB*, também conhecido por *InfluxData,* é uma tecnologia de base de dados SQL para séries temporais que permite a recolha, armazenamento, visualização e monitorização deste tipo de dados. As suas características - simplicidade, escalabilidade, *open source*, integrada - permitem análises em tempo real, gestão de tempo e espaço e tendo sido criada para séries temporais ("InfluxDB system Properties"; InfluxData 2017; "InfluxDb - Time series data storage"; Persen 2016).

- **Apache Cassandra**

O *Apache Cassandra* é uma tecnologia de base de dados *NoSQL* bastante utilizada para armazenamento de dados como medidas de desempenho e registos de atividade, ou seja, semelhantes a séries temporais utilizada por exemplo pela Netflix, eBay, Cern. Permite obter uma

solução linear escalável, no caso do Netflix cerca de 1 milhão operações de escrita por segundo, sem perda de performance, com tolerância à falha (replicação) e solução sem pontos únicos de falha (descentralizada) (Hobbs 2012; "What is Cassandra?"; Sheahan 2011).

2.3 Séries Temporais

Uma série temporal consiste numa sequência de números reais em que a cada valor está associado um marcador temporal, ou seja, no formato {marcador_temporal_x, valor}. Os dados associados a séries temporais são uma ferramenta que deve ser utilizada pela organização permitindo revelar padrões, alterações e ciclos com respostas em tempo real, optimizando a gestão das operações relativamente a previsões calculadas. Os problemas com os dados de séries temporais resultam da organização dos armazéns de dados em células (factos), que, por sua vez, têm associadas dimensões.

Normalmente às séries temporais estão associadas maioritariamente operações de escrita sequenciais, à medida que os dados chegam, e as leituras são feitas por séries sequenciais {valor, marcador_temporal_x} (Schwartz) .

2.3.1 Beneficios

As interrogações realizadas a agregados de informação são, devido às quantidades e tipos de dados, dispendiosas em termos de recursos computacionais e de tempo gasto, o que causa transtornos. Isso advém da necessidade de serem utilizadas para obtenção de indicadores de desempenho para Business Intelligence para o proponente e estes devem estar disponíveis para a tomada de decisão de forma mais rápida possível e ao menor custo, permitindo assim ao gestor a tomada de decisões corretas para o bem da organização.

2.4 Metodologia

Os testes serão feitos comparando com o Postgres, para efeitos de *benchmarking*, e na mesma máquina, assegurando assim as mesmas condições entre as tecnologias de bases de dados. Os parâmetros de comparação para as tecnologias avaliadas serão os seguintes: escalabilidade, desempenho, custos de operação do sistema, das ferramentas e da infraestrutura de suporte. Para isso serão testadas várias interrogações em diferentes cenários e será tida em consideração a documentação disponibilizada de cada tecnologia de base de dados. Os cenários para teste serão um conjunto de interrogações do altaia (Tabela 2: Cardinalidade de algumas tabelas do altaia modificada por motivos de confidencialidade), plataforma utilizada pela Altice, e o desempenho destas em amostras de dados recolhida pela organização (Tabela 3: Exemplo interrogação altaia modificada por motivos de confidencialidade).

Tabela 2: Cardinalidade de algumas tabelas do altaia

Nome da Tabela	Número de linhas
LS_OKX_CELL_ANTENA	27382
LS_OKX_CELL_ZONECODE	508
J_OKX_CELL	161907454
J_ENTITY_OKX_MSG_SMS	24511087

Tabela 3: Exemplo interrogação altaia

```
-- R3G_Cell evolution | geo
SELECT T3."year",
      T3."month",
      T3."day_of_month",
      T3."hour",
      T5."rncvendor",
      T21."celllac",
      T21."cellrac",
      T9."sitecode",
      T9."sitename",
      T13."nodeblatitude",
      T13."nodeblongitude",
      T2."antennaazimuth",
      T2."antennabeamwidth",
      T16."nodebnutsi",
      T16."nodebnutsii",
      T16."nodebnutsiii",
      T11."distrito",
      T11."concelho",
      T11."freguesia",
      T4."cellabbreviation",
      Max(T3."report_date_time"),
      Max(T4."entity_code"),
      SUM(T1."m31"),
      SUM(T1."m48"),
      SUM(T1."m21706253"),
      SUM(T1."m188416"),
      SUM(T1."m188409"),
      SUM(T1."m125912"),
      SUM(T1."m125893"),
      SUM(T1."m125912"),
      SUM(T1."m125913"),
      SUM(T1."m125896")
FROM   dbn1.f_r3g_cell T1,
      dbn1.da_r3g_cell_antenna T2,
      dbn1.d_time T3,
      dbn1.d_entity_r3g_cell T4,
      dbn1.da_r3g_rnc_vendor T5,
      dbn1.da_r3g_nodeb_site T9,
      dbn1.da_r3g_nodeb_geograph T11,
      dbn1.da_r3g_nodeb_coordin T13,
      dbn1.da_r3g_nodeb_add_info T16,
      dbn1.da_r3g_cell_areacode T21
WHERE  ( T2."aa_r3g_cell_antenna_key" = T1."aa_r3g_cell_antenna_key" )
      AND ( T3."time_key" = T1."time_key" )
      AND ( T4."entity_key" = T1."entity_key" )
      AND ( T5."aa_r3g_rnc_vendor_key" = T1."aa_r3g_rnc_vendor_key" )
```

```
    AND ( T9."aa_r3g_nodeb_site_key" = T1."aa_r3g_nodeb_site_key" )
    AND ( T11."aa_r3g_nodeb_geograph_key" = T1."aa_r3g_nodeb_geograph_key" )
    AND ( T13."aa_r3g_nodeb_coordin_key" = T1."aa_r3g_nodeb_coordin_key" )
    AND ( T16."aa_r3g_nodeb_add_info_key" = T1."aa_r3g_nodeb_add_info_key" )
    AND ( T21."aa_r3g_cell_areacode_key" = T1."aa_r3g_cell_areacode_key" )
    AND ( ( Substr(T1."time_key", 15, 5) ) = '20001' )
    AND ( T1."time_key" BETWEEN 20170208000000000000 AND 20170208235999999999 )
    AND ( T4."cellabbreviation" IN( 'a', 'b', 'c', 'd',
                                    'e', 'f', 'g', 'h',
                                    'i', 'j', 'k', 'l',
                                    'm', 'n', 'o' ) )
    AND ( T9."sitename" IN( 'local_a', "local_b, "local_c',
                            "local_d",
                            'local_e', 'local_f', 'local_g')
        )
    AND ( T11."freguesia" IN( 'ALMEIDA', 'ALVARENGA', 'BUÇOS', 'CABREIROS',
                              'CACIA', 'EIXO', 'ESGUEIRA', 'FREIXO' ) )
    AND ( T11."concelho" IN( 'ALMEIDA', 'AROUCA', 'AVEIRO', 'CABECEIRAS
DE BASTO' ) )
    AND ( T11."distrito" IN( 'AVEIRO', 'BRAGA', 'GUARDA' ) )
GROUP  BY T3."year",
          T3."month",
          T3."day_of_month",
          T3."hour",
          T5."rncvendor",
          T21."celllac",
          T21."cellrac",
          T9."sitecode",
          T9."sitename",
          T13."nodeblatitude",
          T13."nodeblongitude",
          T2."antennaazimuth",
          T2."antennabeamwidth",
          T16."nodebnutsi",
          T16."nodebnutsii",
          T16."nodebnutsiii",
          T11."distrito",
          T11."concelho",
          T11."freguesia",
          T4."cellabbreviation"
ORDER  BY T3."year",
          T3."month",
          T3."day_of_month",
          T3."hour",
          T5."rncvendor",
          T21."celllac",
          T21."cellrac",
          T9."sitecode",
          T9."sitename",
          T13."nodeblatitude",
          T13."nodeblongitude",
          T2."antennaazimuth",
          T2."antennabeamwidth",
          T16."nodebnutsi",
          T16."nodebnutsii",
          T16."nodebnutsiii",
          T11."distrito",
          T11."concelho",
          T11."freguesia",
          T4."cellabbreviation";
```

Os testes envolvem inserções de diferentes grandezas e interrogações cruzadas (ex: interrogação_1 com interrogação_2). Estes testes poderão ou não confirmar a primazia dada às tecnologias que melhor se enquadram nos requisitos do utilizador, criando oportunidades de desenvolvimento de soluções que melhor se adaptem à organização. Isto será explicado com mais detalhe no capítulo de testes (Testes).

Capítulo 3

Tecnologias sobre séries temporais

3.1 Requisitos do utilizador

3.1.1 Requisitos funcionais

- Suportar séries temporais;
- Permitir inserir e actualizar registos de forma *adhoc;*
- Permitir apagar registos anteriores a uma determinada data, ou os últimos;
- Permitir definir intervalos de datas nas condições de filtragem:
    - data >= "xxx" *and* data < "yyy"
    - T1."REPORT_DATE_TIME" BETWEEN TO_DATE('10-01-2017 23:00:00', 'DD-MM-YYYY HH24:MI:SS') AND TO_DATE('24-01-2017 23:59:59', 'DD-MM-YYYY HH24:MI:SS')
- Funções:
    - nvl – conversão de valores *null;*
    - trunc – para truncar datas a uma granularidade definida (aos 15 minutos, à hora, ao dia, …);
    - like – match parcial com strings e wildcards;
    - in – match com uma lista de valores;
    - upper – converter para *upper case;*

  - coalesce – selecção do 1º valor não *null* numa lista de valores;

- Funções analíticas (com suporte para over(order by xx, yyy partition by zzz)):
  - RANK
  - DENSE_RANK
- Não precisa de suportar transações;
- Tem de permitir agregar e/ou filtrar os dados por um conjunto de dimensões;
- Tem de permitir ordenar o resultado de uma interrogação à base de dados;
- Tem de permitir múltiplas agregações em que umas pegam no resultado das anteriores, *inner queries,* por exemplo agregar por dia e depois agregar por distrito
- Deve suportar as seguintes funções de agregação:

SUM	AVG	MAX	COUNT
COUNT_DISTINCT	MEDIAN	FIRST	LAST
STDDEV	VARIANCE	PERCENTILE_CONT	PERCENTILE_DISC

- Se a mesma tabela permitir guardar várias séries temporais que partilhem o mesmo contexto não é necessário cruzar informação com outras tabelas, caso isso não se verifique é preciso uma operação equivalente ao join;
- Preferencialmente suportar uma *query language* semelhante ao sql;
- Preferencialmente permitir inserções, alterações e remoções numa estrutura semelhante ao sql.

3.1.2 Requisitos não funcionais

- Tem de ser escalável;
- Tem de suportar "tabelas" com mais de 10^{12} registos;
- Tem de suportar bases de dados com mais de um *Petabyte*(10^{15});
- Tem de ter tempos de resposta a queries baixo;
- Deve ser open source;
- Suportar replicação de dados entre diferentes centros de dados;

- Simplicidade de instalação/configuração/operação. Por exemplo:
  - *Backups online/offline;*
  - Monitoria que permita obter informação útil para *tunning* e despiste de problemas;
- Comunidade de suporte activa;
- Documentação clara;
- Tem de suportar inserir/atualizar mais de 10^9 operações por hora.

3.2 Postgres

O *Postgres(v9.6)* foi utilizado como referência de comparação perante as restantes tecnologias de base de dados a avaliar. Este fornece uma solução robusta e estável, comprovada ao longo dos vários anos de funcionamento ao serviço dos seus utilizadores, é altamente compatível com as mais diversas plataformas e também é *open-source* sem custos de licenciamentos adicionais ao *software*.

Através da seguinte tabela (Tabela 4: Postgres limitações ("PostgreSql Documentation")) é possível verificar que o *postgres* cumpre os requisitos não funcionais estabelecidos:

Tabela 4: Postgres limitações ("PostgreSql Documentation")

Limitação	**Valor**
Tamanho da base de dados	ilimitado
Tamanho de tabelas	32TB
Tamanho da linha	1.6TB
Tamanho do campo	1GB
Linhas por tabela	ilimitado
Colunas por tabela	250 -1600, dependendo do tipo da coluna
Indexes por tabela	ilimitado

Uma solução possível para a aplicação de séries temporais seria a seguinte (grisha 2015):

1. Criação de duas tabelas uma(rrd) para a posição e data e outra tabela(ts) para guardar os dados dessa data

```
Create TABLE rrd (
 id SERIAL NOT NULL PRIMARY KEY,
 last_date DATE,
 last_pos INT);


CREATE TABLE ts (
 rrd_id INT NOT NULL,
 n INT NOT NULL,
 dp DOUBLE PRECISION[] NOT NULL DEFAULT '{}');
```

2. Inserção de dados:

```
INSERT INTO rrd (id, last_date,last_pos) VALUES (1, '2008-04-01' , 24);


INSERT INTO ts VALUES (1,1, '{64,67,70,71,72,69,67}');
INSERT INTO ts VALUES (1,2, '{65,60,58,59,62,68,70}');
INSERT INTO ts VALUES (1,3, '{71,72,77,70,71,73,75}'):
```

3. A atualização da data para dia dois de Abril seria:

```
UPDATE ts SET dp[4] = 92 WHERE rrd_id = 1 AND n =4;
UPDATE rrd SET last_date = '2008-04-02', last_pos = 25 WHERE id = 1;
```

Permite a utilização da função *date_trunc* ,por exemplo ("Postgres timeseries-tips"):

```
SELECT date_trunc('minute',measured_at) as mins, sum(activity_count)
   FROM activity_tseries
   GROUP BY date_trunc('minute', measured_at)
   ORDER BY date_trunc('minute', measured_at) asc;
```

O postgres permite a modificação dos registos bem como a possibilidade de filtragem por datas.

Para converter "*null"* pode-se utilizar o seguinte (SELECT COALESCE(field,0))

Possui as funções analíticas *rank()* e *dense_rank()* exemplo:

```
SELECT depname,
      empno,
      salary,
      rank()
        OVER (
          PARTITION BY depname
          ORDER BY salary DESC)
FROM   empsalary;
```

Permite a utilização de *subqueries*(ou também chamada de *inner query* ou *inner select)* em que uma agregação utiliza o resultado de outra anterior. Possui todas as funções de agregação indicadas nos requisitos do utilizador, por exemplo, a função *count_distinct()* pode ser feita da seguinte forma:

```
SELECT Count(*)
FROM   (SELECT DISTINCT column_name
       FROM   table_name) AS temp;
```

Funções como "*median*", "*first", "last"* não são diretamente suportadas pelo postgres, mas existem bibliotecas disponíveis com as respectivas implementações.

A replicação dos dados entre diferentes centros de processamentos de dados é possível através de um processo denominado "*Log-Shipping",* através do armazenamento continuo da informação, permitindo assim uma grande disponibilidade e garantia que se um servidor falhar existe outro pronto.

As cópias de segurança podem ser *online* ou offline. No caso de ser *online* o postgres possui diários (*log)* no quais se que descreve todas as alterações efetuadas aos dados assim, em caso de falha, basta utilizar estes diários que para restaurar a base de dados. Já no modo offline, é possível copiar diretamente os ficheiros ("Postgres File system level backup") tendo em atenção que o servidor deve ser desligado antes de efetuar a cópia de segurança e não deve ser restaurada apenas parte da base de dados (certas tabelas) pois o restauro parcial gera problemas de compatibilidade.

Para a monitorização do desempenho para melhoria e deteção rápida de problemas existem ferramentas como:

- pgcluu("pgcluu- PostgreSQL Cluster utilization");
- pganalyze("pganalyze- PostgreSQL Performance Monitoring");
- opm("opm- open postgresql monitoring").

Para além de tudo isto possui uma comunidade de suporte bastante ativa.

3.3 CitusData

O Citus (versão 6.1) é uma tecnologia de base de dados *open source* que extende as funcionalidades do PostgreSQL recorrendo a *sharding, replication* e à paralelização das interrogações (usando o seu *query distributed engine)*, podendo escalar horizontalmente por várias máquinas, com tolerância a falhas e tirando proveito do uso simultâneo dos múltiplos *cores* disponíveis no cluster. Adota, como o Postgres, uma estrutura *row-oriented* e com o objetivo de ser utilizada para grandes volumes de dados relacionados com séries temporais.

Para alcançar estes objetivos, o Citus segue uma estrutura *Master/Workers* (configurada com 4 *workers)* em que o *Master* guarda apenas informação (metadata) sobre os seus *Workers* e localização de onde eles guardam os fragmentos (*shards).* O Citus particiona, pelos *Workers*, as interrogações em vários fragmentos (Citusdata) com a seguinte arquitetura (Figura 4: arquitetura do Citus), assim sendo as configurações do número de fragmentos devem ser de acordo com o número de CPU *cores* disponíveis no *cluster* para tirar um maior proveito do paralelismo. No fim, o *Master* junta cada um dos resultados parciais, recolhidos pelos *Workers,* e devolve o resultado final da interrogação.

O Citus usa a mesma sintaxe SQL que o Postgres para a criação de tabelas, interrogações e inserções tirando partido da confiança deste, bem como das compatibilidades já existentes com extensões, ferramentas e *drivers* ao mesmo tempo que adiciona novas funcionalidades (*create_distributed_table (table,distributed_column)* e *create_reference_table (table)* por exemplo). Ao transformar as tabelas em *distributed_table* são criados fragmentos (*shards)* nos *workers* utilizando os valores definidos nos parâmetros de configuração *citus.shard_count (*com valor 32) e *citus.shard_replication_factor (*com valor 1) e selecionando para *distributed column* as colunas mais frequentes nas interrogações utilizadas no caso real do proponente. Com as *reference_tables* é possível distribuir tabelas para um único fragmento (*shard),* em vez de distribuir por múltiplos fragmentos na horizontal, e depois replicar esse fragmento por todos os *workers* em causa, permitindo assim a existência de *joins* locais com outras tabelas com custos de inserção, mas ganhos nas leituras para as interrogações.

Para cópias de segurança podem ser utilizadas as mesmas ferramentas do Postgres, em caso de falha nos *workers* o Citus completa a interrogação recorrendo a outros workers que disponham da mesma cópia do fragmento (*shard)* em falha. O Citus não se adequa a queries não-agregadas ou que envolvam transações Sql, caso se pretenda atualizar os dados presentes e libertar espaço no disco é necessário correr no *master* a seguinte interrogação de forma a apagar os fragmentos(*shards)* pretendidos:

- SELECT * FROM master_apply_delete_command (“DELETE FROM distributed_table_x WHERE time_key <= “data_dados_a_guardar”);

Para o setup de avaliação foi utilizada a versão *community* gratuita, mas existem soluções pagas nas vertentes *cloud* enterprise tendo disponível um *live chat* de apoio aos utilizadores e

canal na aplicação Slack. O servidor Citus está sobre licença GNU affero General Public License v3.0 e os *drivers* de cliente sobre a licença Postgres. Para se consultar os metadados guardados no *Master* é possível recorrer às seguinter interrogações, na porta do *Master:*

- SELECT * FROM pg_dist_partition;
- SELECT * FROM pg_dist_colocation.

A primeira interrogação permite ter acesso aos metados que contêm informação acerca das tabelas da base de dados que são distribuídas e, para cada uma dessas, informação sobre a sua distribuição. O pg_dist_colocation disponibiliza informação, para o caso de ser necessário, acerca da forma recomendada de distribuir, pelas tabelas, os fragmentos. O Citus cumpre, portanto, os requisitos funcionais e não funcionais para ser avaliada num ambiente de caso de uso do proponente.

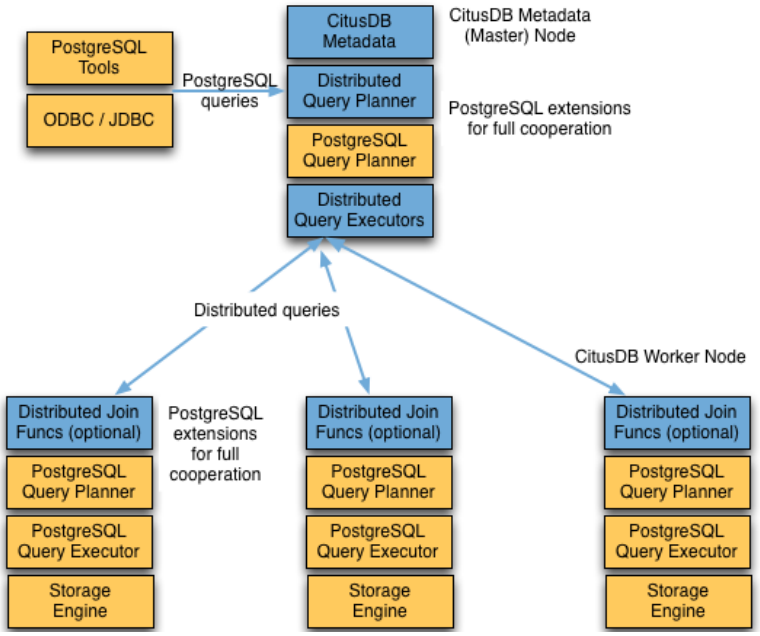


Figura 4: arquitetura do Citus (Processamento das interrogações no Citus)

3.4 TimescaleDB

A Timescaledb é uma tecnologia de base de dados SQL, baseada no postgres, *open-source* sendo distribuída de acordo com a licença Apache 2.0, orientada para séries temporais, de forma a responder às crescentes exigências de escalabilidade e de complexidade das interrogações atualmente nos vários sectores de actividade. Tradicionalmente, a escolha implicava a opção pela escalabilidade (NOSQL) ou então uma compatibilidade completa com SQL (modelos relacionais). Com o objetivo de fazer a ligação entre estas duas escolhas que desenvolveu-se a Timescaledb. Para isto, o foco é a otimização de novas inserções de dados e não as atualizações dos já existentes, recorrendo a um mecanismo de *hypertables* que permite criar tabelas, às quais existe já um valor temporal associado, que se tornam ocultamente (ao utilizador) subdivididas em vários *chunks* (Figura 5: hypertable (hypertables)), ou seja, em partições com dimensões tempo/espaço de acordo com o tamanho da tabela origem. Essas partições são geradas para garantir a escalabilidade horizontal por vários servidores e cuja manipulação (por ex. inserções/interrogações) beneficia da paralelização, pelos vários chunks e servidores, otimizada para o ambiente disponível (*single node* ou *cluster),* sendo apenas utilizados o mínimo indispensável de *chunks* para as interrogações ou agregações em causa, recorrendo para isto à análise da SQL *parse tree* para estas escolhas e atribuições aos *chunks*.

A Timescaledb disponibiliza uma interface completamente sql facilitando, assim, a interação dos utilizadores, sem que tenham de aprender uma nova linguagem e sem que estes se tenham de encarregar da partição dos dados ou das políticas de retenção de dados, pois a tecnologia encarrega-se disso de forma automática de acordo com os tamanhos das tabelas (timescaledb: sql made scalable for time-series).

Existem limitações na implementação atual, já identificadas para resolução, tais como a falta de níveis de acesso aos dados das *hypertables* (qualquer utilizador consegue aceder) e a impossibilidade de criação de *hypertables* a partir de tabelas não-vazias. Apesar disto, passa para uma avaliação de configuração e desempenho no contexto das necessidades do proponente.

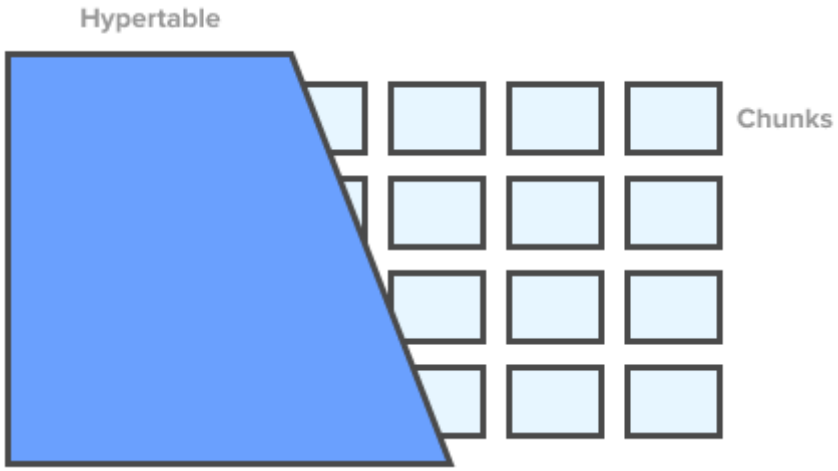


Figura 5: hypertable (hypertables)

Capítulo 4

Testes

4.1 Dados

A DBN0 possui os dados tal como são lidos pelos sensores enquanto que a DBN1 possui os dados agregados segundo critérios definidos para o Altaia. Os dados da DBN0, com informações reais, foram extraídos do Impala criando um projecto Maven no Intellij. Para isso, é criado um novo impalaClient com um método *executeQuery* que recebe uma *string* com a *query* a executar no Impala e que, depois de validada a autenticação de segurança Hadoop, estabelece a ligação e retorna os resultados da respetiva *query.* Esses dados são depois exportados para o formato .csv para posterior inserção da totalidade dos dados na mesma tabela (Figura 6: DBN0 flat simplificada) e comparação com os mesmos dados da mesma tabela, mas numa estrutura modelizada em *star* (Figura 7: DBN0 Star simplificada*)* e apenas no âmbito da tecnologia PostgreSQL.

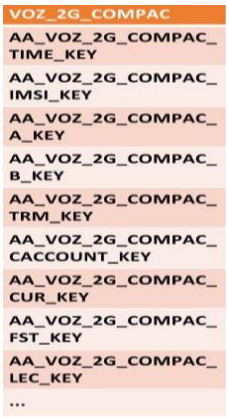


Figura 6: DBN0 flat simplificada

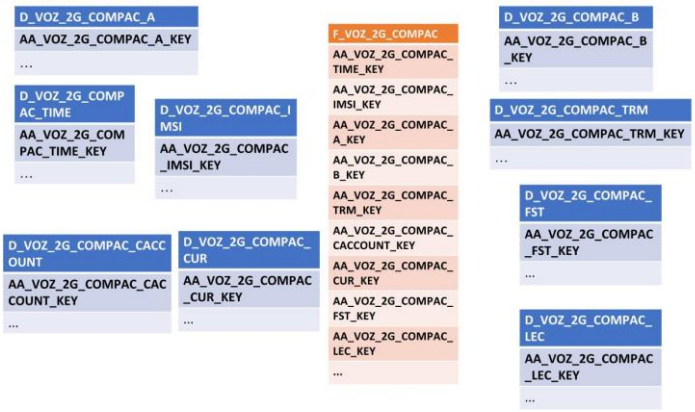


Figura 7: DBN0 Star simplificada

Relativamente à DBN1, os dados estavam armazenados numa base de dados MongoDB, que depois de estabelecer ligação ssh, foram extraídos da seguinte forma:

```
mongoexport --port "port" -d "database" -c "DA_MSG_SMS_LARGE_ACCOUNT" --fields="AA_MSG_SMS_LARGE_ACCOUNT_KEY,LARGE_ACCOUNT_NAME" --out DA_MSG_SMS_LARGE_ACCOUNT.csv --type=csv
```

Obtendo 3 estrelas F_RXG (Figura 8: DBN1 F_RXG simplificada), F_R3G, F_PLT_MSG_SMS e respetivas dimensões associadas no formato csv para, posteriormente, serem inseridos nas tabelas respetivas.

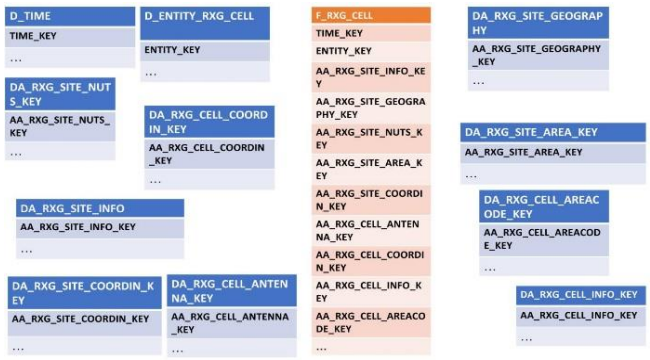


Figura 8: DBN1 F_RXG simplificada

No caso da estrela referida (Figura 8: DBN1 F_RXG simplificada), o *create table* surge neste formato:

```
CREATE TABLE F_RXG_CELL
(  TIME_KEY DOUBLE PRECISION,
   ENTITY_KEY BIGINT,
   AA_RXG_SITE_INFO_KEY DOUBLE PRECISION DEFAULT 0,
```

```
    AA_RXG_SITE_GEOGRAPHY_KEY DOUBLE PRECISION DEFAULT 0,
    AA_RXG_SITE_NUTS_KEY DOUBLE PRECISION DEFAULT 0,
    AA_RXG_SITE_AREA_KEY DOUBLE PRECISION DEFAULT 0,
    AA_RXG_SITE_COORDIN_KEY DOUBLE PRECISION DEFAULT 0,
    AA_RXG_CELL_ANTENNA_KEY DOUBLE PRECISION DEFAULT 0,
    AA_RXG_CELL_COORDIN_KEY DOUBLE PRECISION DEFAULT 0,
    AA_RXG_CELL_INFO_KEY DOUBLE PRECISION DEFAULT 0,
    AA_RXG_CELL_AREACODE_KEY DOUBLE PRECISION DEFAULT 0,
    Mx DOUBLE PRECISION,
    Ma DOUBLE PRECISION,
    Mb DOUBLE PRECISION,
    Mc DOUBLE PRECISION,
    Md DOUBLE PRECISION,
    Me DOUBLE PRECISION,
    Mf DOUBLE PRECISION,
    Mg DOUBLE PRECISION,
    Mh DOUBLE PRECISION,
    Mi DOUBLE PRECISION,
    Mj DOUBLE PRECISION,
    Mk DOUBLE PRECISION,
    Ml DOUBLE PRECISION,
    Mm DOUBLE PRECISION,
    Mn DOUBLE PRECISION,
    Mo DOUBLE PRECISION,
    Mp DOUBLE PRECISION,
    Mq DOUBLE PRECISION,
    Mr DOUBLE PRECISION
  );
```

Depois foram criadas partições para cada uma das 3 estrelas da DBN1, partições essas feitas de acordo com os valores da coluna “time_key”, exemplo:

```
    CREATE OR replace FUNCTION create_partition_dbn1_rxgpartition_4( t_name VARCHAR(30)) returns
void AS $func$ DECLARE table_number INT;
    BEGIN FOR table_number IN 20161101..20161114 LOOP EXECUTE format(
    ' CREATE TABLE IF NOT EXISTS %I( )INHERITS(F_RXG_CELL)',
    t_name || '_p_' || table_number
    );
    END LOOP;
    END $func$ LANGUAGE plpgsql;
```

Os indexes para todas as dimensões e estrelas foram sendo elaborados de acordo com o tipo de *query* utilizada e melhorados/acrescentados há medida que se detetavam pontos de estrangulamento no *query planner* e nos testes elaborados com essas mesmas *queries* (Tabela 5: Query_4 DBN1 modificada). Possuindo essas *queries* diferentes graus de complexidade na medida que, no caso da query 4 referida, o tipo de atributos das tabelas consultadas, colunas com o tipo *double precision,* têm um impacto cerca de 30% superior a inteiros de 4bytes. Tal como o impacto com os *scans* feitos a partições com maior horizonte temporal (pesquisa mais alargada pelo parâmetro *time_key)*. Para facilitar as consultas foram criados indexes, por exemplo, um dos que envolvem a estrela F_RXG :

```
CREATE INDEX PK_F_RXG_CELL ON F_RXG_CELL (
TIME_KEY, ENTITY_KEY, AA_RXG_SITE_INFO_KEY,
AA_RXG_SITE_GEOGRAPHY_KEY, AA_RXG_SITE_NUTS_KEY,
AA_RXG_SITE_AREA_KEY, AA_RXG_SITE_COORDIN_KEY,
AA_RXG_CELL_ANTENNA_KEY, AA_RXG_CELL_COORDIN_KEY,
AA_RXG_CELL_INFO_KEY, AA_RXG_CELL_AREACODE_KEY
);
```

Tabela 5: Query_4 DBN1 modificada

```
SELECT
T10.GENCELLRATVENDOR,
T10.GENCELLRATTYPE,
T6.GENCONTROLLER,
 MAX(T1.Mx),
 SUM(T1.Mb),
 SUM(T1.Mc),
 SUM(T1.Md)
 FROM
F_RXG_CELL T1,
DA_RXG_SITE_INFO T6,
DA_RXG_CELL_INFO T10
 WHERE
(T6.AA_RXG_SITE_INFO_KEY = T1.AA_RXG_SITE_INFO_KEY) AND
(T10.AA_RXG_CELL_INFO_KEY = T1.AA_RXG_CELL_INFO_KEY) AND
```

```
(((T1.TIME_KEY BETWEEN 2016100300000000000 AND

2016100323599999999) ) OR (T1.TIME_KEY BETWEEN 2016100500000000000 AND

20161005032359999999)) AND

(T6.GENCONTROLLER IN('k5', 'l2', 'j2', 'hI6',

't3', 'y5', 'u3')) AND

(T10.GENCELLRATTYPE IN('2G', '3G'))

 GROUP BY

T10.GENCELLRATVENDOR,

T10.GENCELLRATTYPE,

T6.GENCONTROLLER

 ORDER BY

T10.GENCELLRATVENDOR,

T10.GENCELLRATTYPE,

T6.GENCONTROLLER

LIMIT 20;
```

As inserções são feitas com recurso a *scripts,* como este exemplo (modificado por confidencialidade):

```
	STARTTIME=$(date +%s)
	for x in $(ls F_RXG_CELLPARTITION_P_*.csv);
	do
	psql -c "\copy F_RXG_CELL(Ma,Mb,Mc,Md,AA_RXG_CELL_INFO_KEY,Me,Mf,Mg,Mh,AA_RXG_CELL_ANTENNA_KEY,AA_RXG_SITE_NUTS_KEY,Mi,AA_RXG_SITE_AREA_KEY,Mj,AA_RXG_CELL_AREACODE_KEY,Mk,Ml,AA_RXG_SITE_GEOGRAPHY_KEY,Mm,Mn,TIME_KEY,Mo,Mp,Mq,AA_RXG_SITE_INFO_KEY,Mr,AA_RXG_SITE_COORDIN_KEY,ENTITY_KEY,Ms,AA_RXG_CELL_COORDIN_KEY) FROM '/home/paulo/Desktop/DBN1_csvs/$x' DELIMITER ',' NULL '' CSV HEADER;" -h host_info -U user_info -d database_info; ENDTIME=$(date +%s)
	done
	echo "It took $(($ENDTIME - $STARTTIME)) seconds to complete all the inserts to F_RXG_CELL..." >> "$(date +"%Y_%m_%d_%I_%M_%p").txt"
```

4.2 Ferramenta de testes

4.2.1 Jmeter

Para verificar o desempenho das tecnologias de base de dados foi utilizada a ferramenta de testes Jmeter (versão 3.1), sendo colocada num ambiente diferente e separando-a das tecnologias a analisar.

O Jmeter é uma ferramenta Java para testes de carga e performance de livre acesso para utilização (tutorialspoint). Destaca-se as seguintes características:

- Interface Gráfica;
- Funciona com servidores web como:
  - HTTP, HTTPS, SOAP, DATABASE via JDBC, LDAP, JMS, MAIL -POP3 entre outros;
- Independente da plataforma (vantagem de ser ferramenta Java);
- Planos de testes guardados e configuráveis em xml;
- *Framework multi-threading* para java (necessita, portanto, da disponibilidade do JDK 1.6+ no ambiente);
- Possibilidade de testes automatizados e funcionais (Figura 9: Funcionamento Jmeter (tutorialspoint));
- Possui um modo servidor para testes distribuídos.

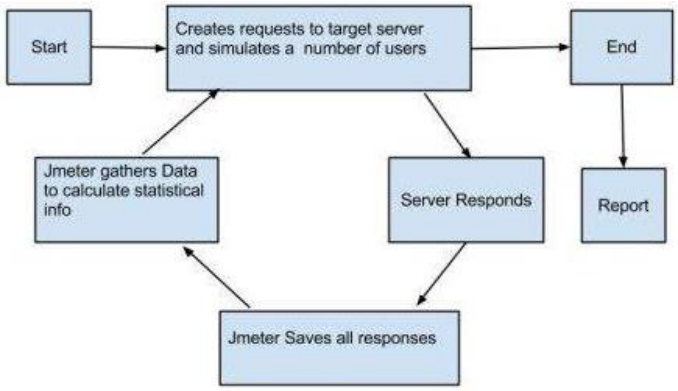


Figura 9: Funcionamento Jmeter (tutorialspoint)

Ao executar o script jmeter.sh obtemos uma janela (Figura 10: Execução jmeter) sem elementos adicionais, com um nó(*node)* para o plano de testes e um ambiente de testes. A estes é possível adicionar/remover os seguintes elementos:

- **Thread Groups**

  Todos os planos de teste precisam de pelo menos um *thread group,* podendo depois ter outros elementos para além deste. Cada thread representa um utilizador a efetuar um pedido à aplicação, sendo possível:

  - Definir o número de threads (utilizadores neste caso) com um máximo de 300 (por questões de limitações de hardware);
  - Modificar o *Ramp-up* que indica o tempo que demora para que o Jmeter consiga ter todas as *threads* definidas a funcionar;
  - Definir o número de iterações de teste.
- **Samplers**

  Permite especificar o tipo de pedidos que o jmeter envia a um servidor, tais como:
  - HTTP Request;
  - FTP Request;
  - JDBC Request (que será utilizado neste caso) pois permite estabelecer a ligação com a base de dados a analisar em cada teste;
  - Java Request;
  - SOAP/XML Request;
  - RPC Requests.
- **Logic controllers**

  Permitem gerir a ordem do processamento dos Samplers com recurso aos seguintes controladores:
  - Simple;
  - Loop;
  - Once only;
  - Random;
  - Throughput;
  - Runtime;
  - If;
  - While;
  - Switch;
  - ForEach;
  - Module;
  - Include;
  - Transaction;
  - Recording.
- **Listeners**

  Permitem a visualização dos resultados dos *samplers* e podem ser adicionados em qualquer local dos testes para atuarem sobre elementos abaixo ou ao mesmo nível. Exemplos de Listeners são:
  - Sample Result Save Configuration;
  - Graph Full Results;
  - Graph Results;
  - Spline Visualizer;

- Assertion Results;
- View Results Tree;
- Aggregate Report (usado nos testes):
  - Cria uma linha para cada um dos JDBC Request presentes no teste, neste caso, para cada uma das interrogações à base de dados em questão. Em cada linha é indicada a:
    - Quantidade de pedidos (*#Samples)* que varia consoante o número de *threads* e *loops* (repetições de cada teste);
    - Min, max e average(média), todos com resultados em milissegundos(ms) e a *error rate* em % (pedidos falhados do jmeter por indisponibilidade do servidor em relação ao número total de *samples* (pedidos totais));
    - Percentil 90%. 95% e 99%, significam que 90,95 ou 99 em cada 100 do total de pedidos ficaram dentro destes valores obtidos;
    - *Throughput (requests per second/minute/hour*, ao gravar os resultados para csv é utilizada a medida *requests per second)* e *kilobytes per second throughput* do ponto de vista do sampler (JDBC *target* = a base de dados).
- Summary Report (usado nos testes):
  - Cria uma linha para cada um dos JDBC Request presentes no teste, neste caso, para cada uma linhas das interrogações a analisar. Em cada linha são indicados os seguintes elementos (para além dos já vistos no listener Aggregate Report):
    - Standard Deviation, variação em milissegundos da amostra recolhida em relação à média;
    - Average Bytes, tamanho (em média) das respostas aos pedidos (*samples),* ou seja, valor (em bytes) médios da informação descarregada do servidor.
- View Results in Table;
- Simple Data Writer;
- Monitor Results;
- Distribution Graph;
- Aggregate Graph;
- Mailer Visualizer;
- BeanShell Listener.

- **Timers**

  Por predefinição, as *threads* no Jmeter enviam os pedidos sem pausas entre cada *sampler.* Isso pode ser modificado da seguinte forma:

  - Constant Timer;
  - Gaussian Random Timer;
  - Uniform Random Timer;
  - Constant Throughput Timer;
  - Synchronizing Timer (usado nos testes):
    - Permite bloquear as *threads* até que se atinja um número definido sendo depois desbloqueadas todas em simultâneo e com um *timeout* máximo de tempo de espera. É utilizado para testes de carga.
  - JSR223 Time;
  - BeanShell Time;
  - BSF Time;
  - Poisson Random Time.
- **Assertions**

  Permitem a inclusão de testes de validação às respostas aos pedidos do Sampler, servindo, assim, para atestar que os dados retornados são os corretos:
  - Beanshell Assertion;
  - BSF Assertion;
  - Compare Assertion;
  - JSR223 Assertion;
  - Response Assertion;
  - Duration Assertion;
  - Size Assertion;
  - XML Assertion;
  - BeanShell Assertion;
  - MD5Hex Assertion;
  - HTML Assertion;
  - XPath Assertion;
  - XML Schema Assertion.
- **Configuration Elements**

  Permitem a criação de variáveis para adicionar ou modificar pedidos do Sampler, estando disponíveis os seguintes:
  - Counter;
  - CSV Data Set Config;
  - FTP Request Defaults;
  - HTTP Authorization Manager;
  - HTTP Cache Manager;
  - HTTP Cookie Manager;
  - HTTP Proxy Server;
  - HTTP Request Defaults;

- Http Header Manager;
- Java Request Defaults;
- Keystore Configuration;
- JDBC Connection Configuration (Usada nos testes):
  - Serve para estabelecer a ligação com as tecnologias de bases de dados em análise, especificando os parâmetros: Database URL, JDBC Drive class, Username e password.
- Login Config Element;
- LDAP Request Defaults;
- LDAP Extended Request Defaults;
- LDAP Extended Request Defaults;
- TCP Sampler Config;
- User Defined Variables;
- Simple Config Element;
- Random Variable.

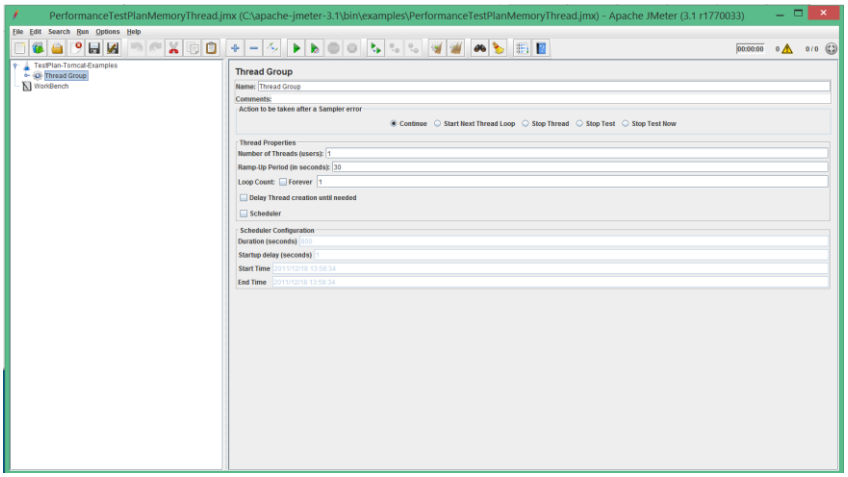


Figura 10: Execução jmeter

4.2.2 Plano de testes

Testes:

- Desempenho: para validar se o desempenho exibido nos testes corresponde ao que a configuração pressupõe;
- Carga: avaliam como o sistema se comporta com a carga máxima indicada;
- Stress: avaliam a resistência máxima do sistema a erros.

Objectivos gerais:

- Comparar as 3 tecnologias de bases de dados Postgres, Citus e TimescaleDB;

- Verificar se cada uma cumpre os requisitos (Requisitos do utilizador):
    - Funcionais
        - Linguagem de interrogação semelhante ao SQL;
        - Suporte a séries temporais;
        - Filtragem por intervalos;
        - Agregar ou filtrar os dados por um conjunto de dimensões;
        - Ordenar o resultado da interrogação.
    - Não Funcionais
        - Escalável;
        - Tempo de resposta a interrogações baixo;
        - Suportar o tamanho das tabelas e volume de dados típicos do utilizador, neste caso cerca 60GB para DBN1.
- Verificar o comportamento em condições normais e de stress durante os testes;
- Verificar se os resultados estão dentro dos valores considerados aceitáveis para a organização.

A testar:

- Interrogações que sejam usadas na própria organização do proponente testadas de forma individual (interrogação 1, interrogação 2, interrogação 3 …) ou então de forma cruzada (interrogação 1 e 2 ou ordem inversa)
- Variação do número de utilizadores (aqui equiparados a *threads*) que fazem os pedidos, quer sejam 1, 5, 10 ou 50 utilizadores
- Mudar parâmetros das interrogações fornecidas com recurso ao *config element* do tipo *random variable*
- Repetição de cada teste N vezes (aumentando as amostras(*samples)* disponíveis) alterando o *loop count* no *thread Group* do jmeter

Resultado esperado:

- Tempos de resposta a interrogações com dados relativos a *delay* médio, mínimo, máximo, percentil 90, 95 e 99, ou seja, em 10 amostras, ordenadas de forma ascendente pelo tempo de resposta, 9 dessas amostras (no caso do percentil 90) ficam abaixo do tempo de resposta indicado
- Número de pedidos (*workload)* por unidade de tempo (*throughput)*

- Para monitorização das máquinas (cpu, rede, I/O) onde a tecnologia de base de dados foi colocada recorreu-se à ferramenta zabbix (versão 2.2):
    - Para esse efeito é colocado um Zabbix agente *(*Figura 11: agente zabbix (Zabbix linux example)*)* nesse ambiente para monitorizar. Depois o servidor Zabbix requer,periodicamente, valores a este agente que podem ser filtrados mais tarde pela hora, dia, mês que se pretende monitorizar(neste caso pela altura em que se correram os testes jmeter) ("Zabbix documentation");
    - Posteriormente é possível criar *screens* bastando, para isso, definir qual o agente e escolher o tipo de template desejado (consoante o tipo de informação que se pretende colectar), sendo assim possível acompanhar a evolução dos testes efetuados.
- Para otimizações das interrogações é possível recorrer ao comando sql "EXPLAIN ANALYZE", para obter o plano de execução e com isso analizar os recursos ao disco gasto para cada tabela/índex/join em causa

```
# ps u -C zabbix_agentd
USER       PID %CPU %MEM    VSZ   RSS   STAT TIME COMMAND
zabbix   15778  0.0  0.0  48212   460   SN   0:00 /usr/sbin/zabbix_agentd
zabbix   15780  0.0  0.0  48212   748   SN   9:27 /usr/sbin/zabbix_agentd
zabbix   15781  0.0  0.0  48212   424   SN   0:00 /usr/sbin/zabbix_agentd
zabbix   15782  0.0  0.0  48212   424   SN   0:00 /usr/sbin/zabbix_agentd
zabbix   15783  0.0  0.0  48212   424   SN   0:00 /usr/sbin/zabbix_agentd
zabbix   15784  0.0  0.0  48220   612   SN   0:17 /usr/sbin/zabbix_agentd
```

Figura 11: agente zabbix (Zabbix linux example)

4.2.3 Configurações do jmeter

Depois de ter o ambiente de jmeter devidamente instalado é necessário verificar se existe:

- Uma base de dados;
- As tabelas;
- As inserções, com dados verídicos da organização, para as respectivas tabelas;
- Adicionar o driver JDBC na pasta "apache-jmeter.../libr".

Criar o plano de testes do Jmeter:

- Abrir o Jmeter (/bin/jmeter.sh);
- Adicionar os utilizadores:
    - Criando um *Thread Group:*

        - Selecionar o plano de testes, depois adicionar *threads(*utilizadores) e thread group, no qual é definido o número de utilizadores desejados, repetições (loop counter) de acordo com os objetivos para o teste em causa.

- Adicionar os pedidos JDBC (que definem quais serão as tarefas a realizar pelos utilizadores em causa):
    - Selecionar o *Thread Group* (criado anteriormente) e adicionar um *Config Element* do tipo JDBC Connection Configuration, com as seguintes configurações:
        - Nome único definido "N";
        - URL da base de dados em análise, por exemplo:
            - Jdbc:postgresql://xx.xxx.xx.xx/altaia
        - JDBC Driver class, por exemplo: org.postgresql.Driver;
        - Username e password para acesso à base de dados.
    - Selecionar o mesmo *Thread Group* e adicionar um *Sampler* do tipo JDBC Request, com as seguintes configurações:
        - Nome único idêntico ao definido no JDBC Connection Configuration "N"
        - Tipo de interrogação – SELECT STATEMENT e juntar a respetiva interrogação em baixo
- Adicionar o elemento Listener, que será responsável por interpretar todos os resultados dos pedidos do JDBC (apartir do .jtl), fornecendo assim uma representação visual dos resultados dos testes. Para isso:
    - Selecionar o *Thread Group* e adicionar os *listeners*: *summary reports* e *aggregate report,* colocando-os, no entanto, como não ativos para agilizar os testes e deixando a parte gráfica apenas para a análise final dos testes e não durante a sua execução (modo *ECO).*
- Para verificar o resultado final de cada teste é necessário abrir o respetivo plano de testes (ficheiros .jmx), selecionar qual o *listener* que se pretende ter como activo e abrir aí o ficheiro .jtl gerado pela execução do teste.

Jmeter Modo ECO

Para reduzir os recursos consumidos pelo Jmeter durante os testes foram adotados os seguintes procedimentos:

- Inativação dos listeners, especialmente aqueles que utilizam gráficos, durante a execução dos testes;
- Utilização do formato csv em vez de xml guardando apenas os campos de resposta necessários para cada caso. Posteriormente, com o ficheiro de resultados(.jtl), podemos carregá-lo no Jmeter(modo Gui) e selecionar o listener desejado;
- Usar o modo non-Gui, correndo na linha de comandos "./jmeter.sh -n -t nomePlanoTestes.jmx -l test.jtl":
    - Sendo nomePlanoTestes.jmx o ficheiro de testes e o test.jtl o ficheiro com os resultados desse mesmo teste.

Recursos disponibilizados pelo ambiente

Ambiente comum a todas a tecnologias de base de dados testadas: máquina com 120 GB de RAM e 2*8 *cores*, cuja utilização máxima da cache atribuída às tecnologas de base de dados foi de 30%, tendo sido utilizados 2 discos com 800gb com velocidade de 120mb/s para uma rede interna de 1gb/s. Todas as tecnologias foram instaladas a partir dos *source (*para as versões já indicadas na descrição de cada tecnologia), obtendo-se assim as versões mais recentes e configuradas de acordo com a documentação atualizada e recomendada pelos criadores das mesmas, separando-se do ambiente de onde foram executados os testes às tecnologias de base de dados (Jmeter).

4.2.4 Desenho dos testes

Estes testes serão comuns a todas as tecnogias de bases de dados em análise (Postgres, Citus e TimescaleDB) e incidem sobre a DBN1 (conjunto de dados agregados) do altaia(Informação). No caso do Postgres, será feito um teste extra para comparação dos dados da DBN0 (dados dos sensores) do altaia, modelizando em *flat* (uma única tabela) e *star (*tabelas dimensionais ligadas a uma tabela de factos).

1. **Plano de testes condições normais (*baseline)***

São testes com condicionantes importantes para a organização, com situações expectáveis de sucederem na utilização da tecnologia de base de dados, seguindo o seguinte padrão para o nome dos planos de teste: "Nomedb_dbnX_PlanoY_QueryZ.jmx". Utilizou-se um *loop counter* 200 para a DBN0 e 10 para DBN1

- **Plano 1 –** Para servir de comparação para os restantes planos

- Interrogações 1 a 5 em separado;
- Interrogações 1 e 2 em conjunto e também pela ordem inversa.

- **Plano 2 –** Para verificar o desempenho perante a variação de utilizadores
  - Interrogação 4 com 5,10 e 50 utilizadores (*threads).*
- **Plano 3**- Para verificar o desempenho perante a variação dos parâmetros das interrogações sendo utilizada uma *seed* com um valor numério 2, garantindo assim os mesmos valores aleatórios em todas as repetições dos testes
  - Interrogações 1 a 3 em separado.

2. **Plano de testes condições carga**

- **Plano 4-** Para averiguar o comportamento perante condições improváveis, mas cuja resistência da BD em causa é importante aferir. Para isso, são criadas barreiras na execução do teste com um recurso a temporizador para sincronizar os *threads(utilizadores)* e coordenar a libertação dos pedidos em simultâneo
  - Interrogações 1 e 2 sendo para cada uma delas elaborados os seguintes procedimentos:
    - Bloqueando 2 em 6 *threads*
    - Bloqueando 3 em 6 *threads*
    - Bloqueando 2 em 12 *threads*
    - Bloqueando 3 em 12 *threads*

Fase de criação dos scripts para os testes

Criar um ficheiro .jmx no Jmeter, usando a interface gráfica, com um plano para cada teste, de acordo com as especificações necessárias para os planos 1 a 4 **(**Figura 12: Teste exemplo Citus**)**

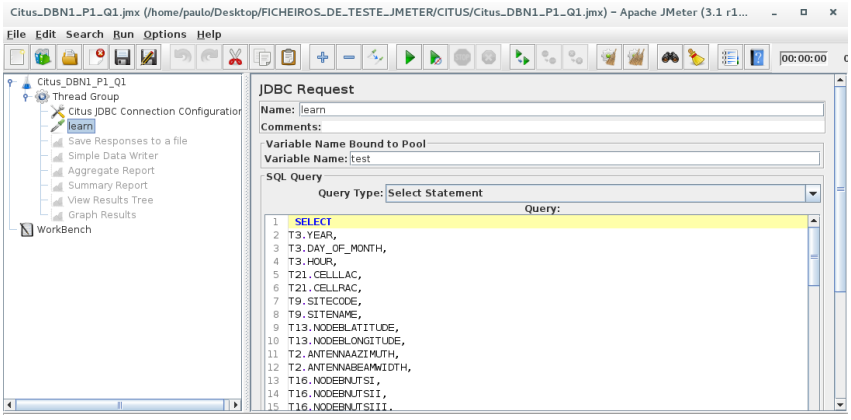


Figura 12: Teste exemplo Citus

Fase de execução dos testes

Para correr os testes, é necessário criar um script (.sh) de acordo com o tipo de teste, e executá-lo da seguinte forma:

- Time sh ./nomeFicheiro.sh (cujo exemplo está em baixo)

```
for x in $(ls Postgres_DBN1_*.jmx);
do
filename=$(basename "$x");
filename=  "${filename%.*}_";
~/Desktop/apache-jmeter-3.1/bin/jmeter.sh -n -t $x -l $filename$(date + "%Y_%m_%d_%I_%M_%p").jtl
Done
```

- Caso se pretenda agendar a execução dos testes para uma data pré-definida:
    - Echo "time sh ~/Desktop/nomeFicheiro.sh" | at 13:33 April 10

Depois de executar os testes são usados os *listeners "agregate report"* e "*summary report"* que por sua vez usam os ficheiros(.jtl) que contêm os resultados dos testes.

Glossário

Performance	Combinação de dados de *Delays, Throughput* e utilização de recursos de uma aplicação
Delay	O *Delay* é uma medida do tempo de resposta de uma aplicação a um pedido de um cliente
Throughput	*Throughput* é o número de unidades de trabalho que podem ser tratadas por unidade de tempo; por exemplo, pedidos por segundo, chamadas por dia etc
Workload	É a carga aplicada a um sistema em termos de unidades de trabalho por unidade de tempo. Tipicamente em pedidos por segundo.
Threads	Simulam o número de utilizadores que, simultâneo, estão a efetuar pedidos sobre um sistema.

4.3 Avaliação dos resultados

- **DBN0 Postgres *flat* e *star***

Nos testes do Jmeter, a modelização *flat (*anexo Postgres DBN0 Flat*)* obteve melhores resultados no plano 1(baseline) em 6 dos 7 testes, no parâmetro valores médios de resposta, face à modelização *star (*anexo Postgres DBN0 Star*)*. No plano 2(aumento do número de utilizadores), a modelização *star* obteve melhores resultados em todos os 3 testes deste plano. O plano 3 (variação dos parâmetros) serviu para validar os resultados e atestar que estes não ficaram em *cache* de umas iterações para as outras. No plano 4 (testes de carga), o modelo *flat* obteve tempos médio de resposta mais baixos mas tempos máximos mais elevados.

Nos testes do Zabbix, a modelização *flat (*anexo Postgres DBN0 Flat) registou valores na monitorização da máquina de 9.02%, na utilização do cpu e 0.0005% no tempo de utilização para operações de escrita e leitura (*IO)* contra os 20.18% e 0.27% da modelização *star (*anexo Postgres DBN0 Star), verificando-se assim uma maior capacidade do processamento em memória no caso da modelização *flat.*

- **Comparação DBN1 Postgres com Citus**

Os resultados com a ferramenta Jmeter comparam as tecnologias Postgres (anexo Postgres DBN1) e Citus (anexo Citus DBN1). Nos resultados obtidos do Jmeter, o Citus alcançou tempos de resposta mais baixos, parâmetro percentil 90, em todos os testes às interrogações (Figura 13: Comparação DBN1 Postgres com Citus usando jmeter).

No caso dos resultados com a ferramenta Zabbix, de monitorização dos ambientes onde estavam as bases de dados, o Citus (anexo Citus DBN1) obteve uma utilização superior do CPU e com menor tempo de espera em operações de leitura e escrita (I/O) cerca de 0.001% valores de utilização máxima no período temporal de realização dos testes (Figura 14: Comparação DBN1 Postgres com Citus usando o zabbix), conseguindo processar em memória os pedidos e com valores mais elevados de transmissão e recepção na rede (Figura 15: Comparação DBN1 Postgres com Citus usando o zabbix (rede)) em comparação com o Postgres (anexo Postgres DBN1).

A explicação para estes resultados superiores do Citus decorre do facto de a arquitetura contemplar a distribuição da informação por cada um dos *workers*, 4 na configuração utilizada, e cada um destes processar em paralelo uma parte da interrogação (fragmento) o que leva a que os joins sejam executados em paralelo com *nested loops.* A funcionalidade responsável por tratar da distribuição das interrogações é o *Task Tracker Executor,* que se liga a cada *worker* e distribui entre eles os fragmentos das interrogações, garantindo também o acesso eficiente caso estes envolvam o particionamento dos dados por diferentes *workers.* No caso dos testes, a configuração

utilizada foi com todos os *workers* na mesma máquina e assim em condições idênticas aos testes com o Postgres (Citusdata).

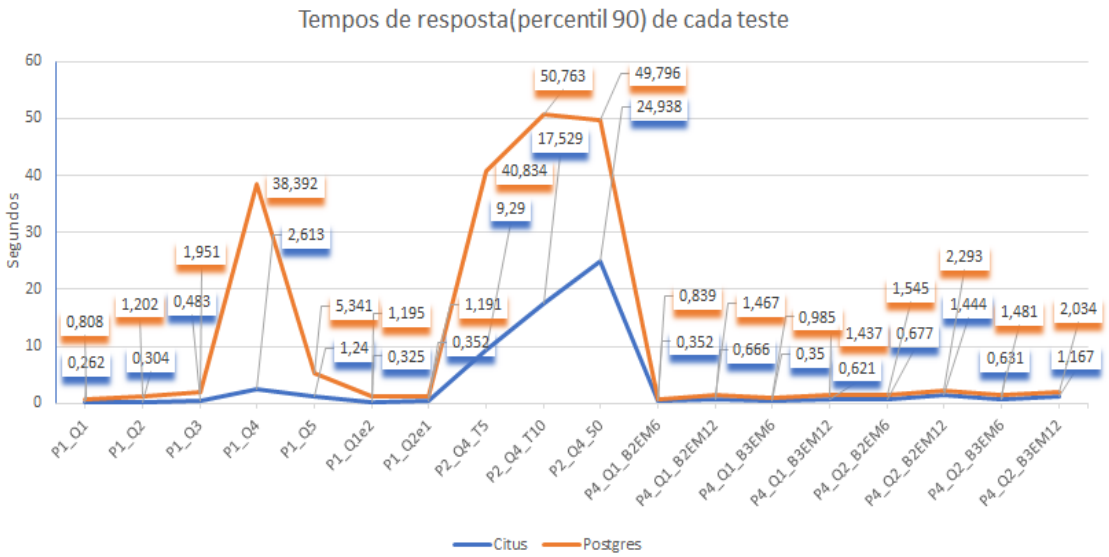


Figura 13: Comparação DBN1 Postgres com Citus usando jmeter

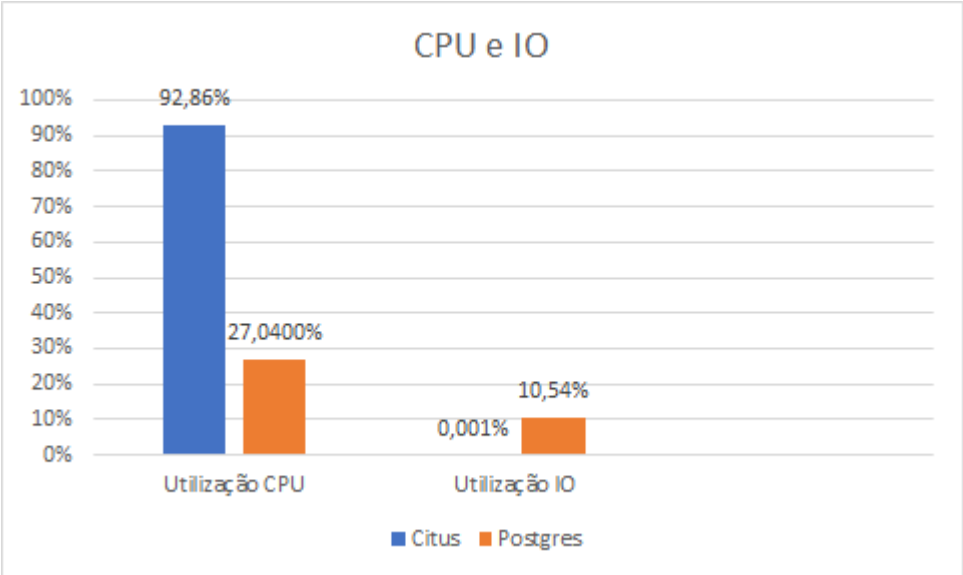


Figura 14: Comparação DBN1 Postgres com Citus usando o zabbix

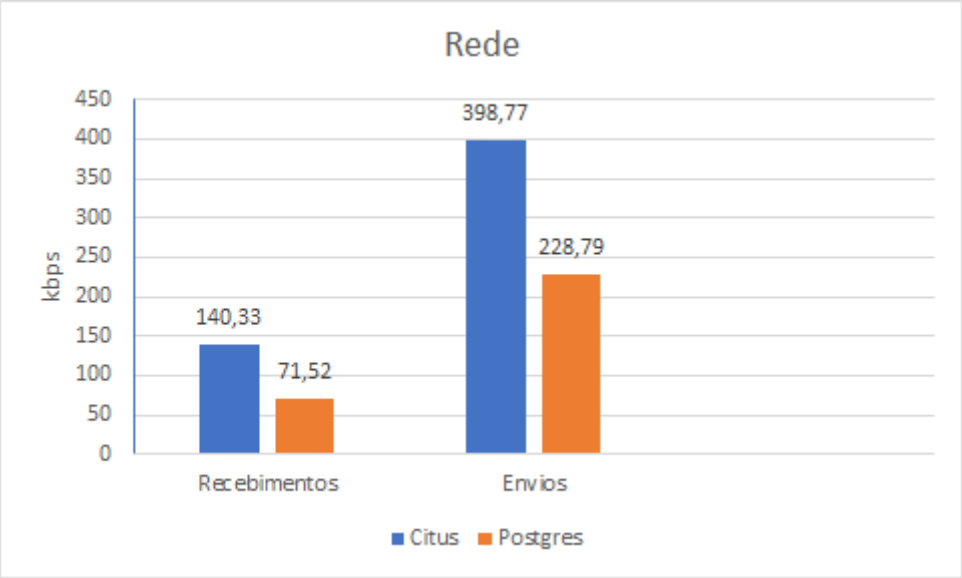


Figura 15: Comparação DBN1 Postgres com Citus usando o zabbix (rede)

- **Comparação DBN1 Postgres com Timescaledb**

Os resultados com a ferramenta Jmeter comparam as tecnologias Postgres e Timescaledb (anexo Timescaled DBN1). Nestes testes (Figura 16: Comparação DBN1 Postgres com Timescaledb usando o jmeter), a Timescaledb obteve melhores desempenhos em todos eles, sendo que isto se deve ao facto de esta tecnologia ser otimizada para interrogações complexas, como as testadas e utilizadas pelo altaia, interrogações essas que envolvem múltiplas medições com um dado marcador temporal. Nas operações de indexação e organização de informação (*group by/order by),* são selecionados apenas os *chunks* consoante a interrogação necessita, minimizando, assim, as pesquisas (*scanning)* completas uma vez que dispõe de informação sobre os intervalos de valores temporais com os quais os *chunks* foram criados (timescaledb: sql made scalable for time-series).

Comparativamente ao Citus, a Timescaledb apresenta uma performance inferior em todos os testes (Figura 17: Comparação DBN1 Citus com Timescaledb usando o jmeter).

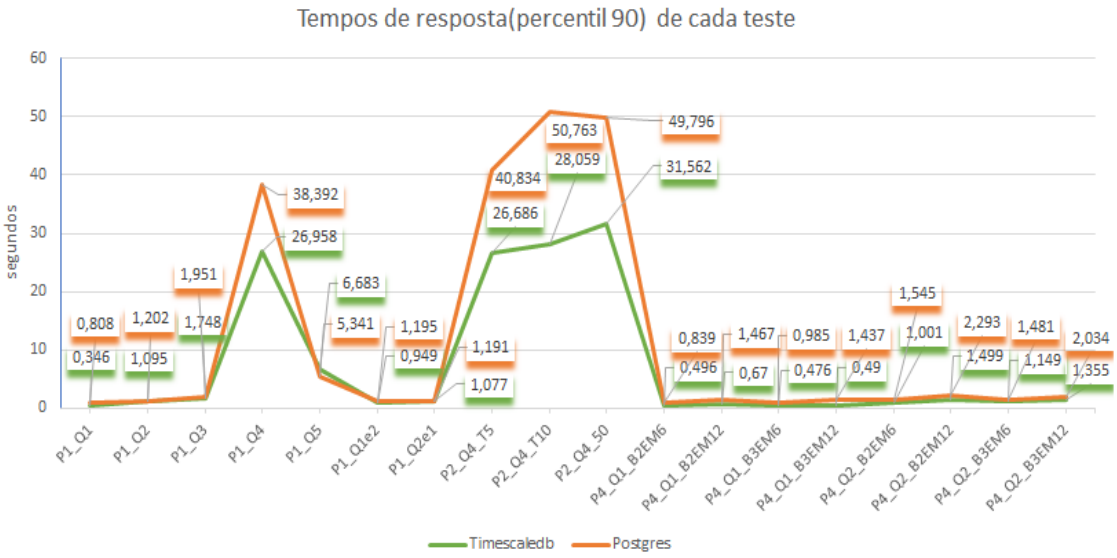


Figura 16: Comparação DBN1 Postgres com Timescaledb usando o jmeter

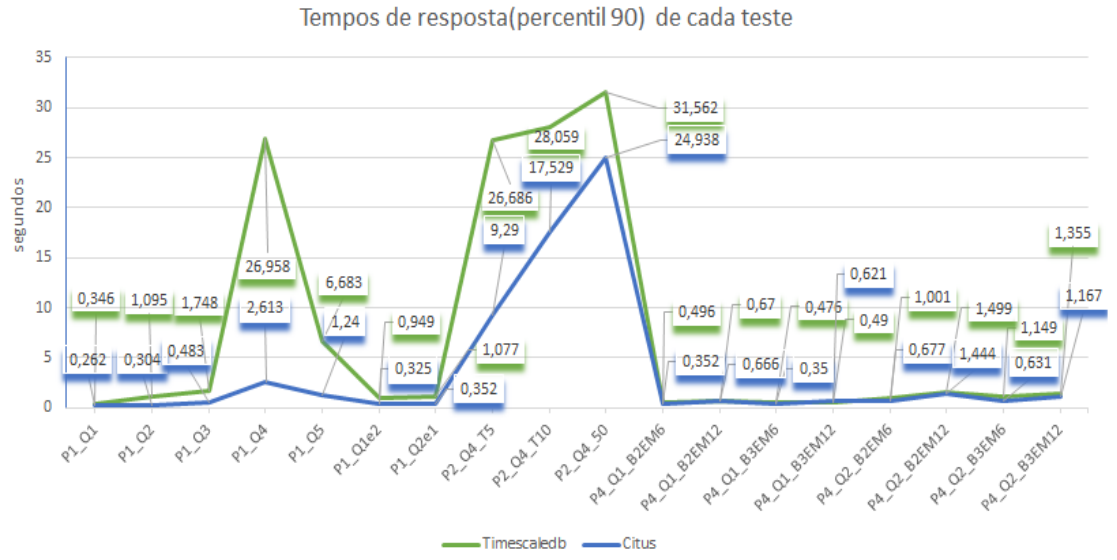


Figura 17: Comparação DBN1 Citus com Timescaledb usando o jmeter

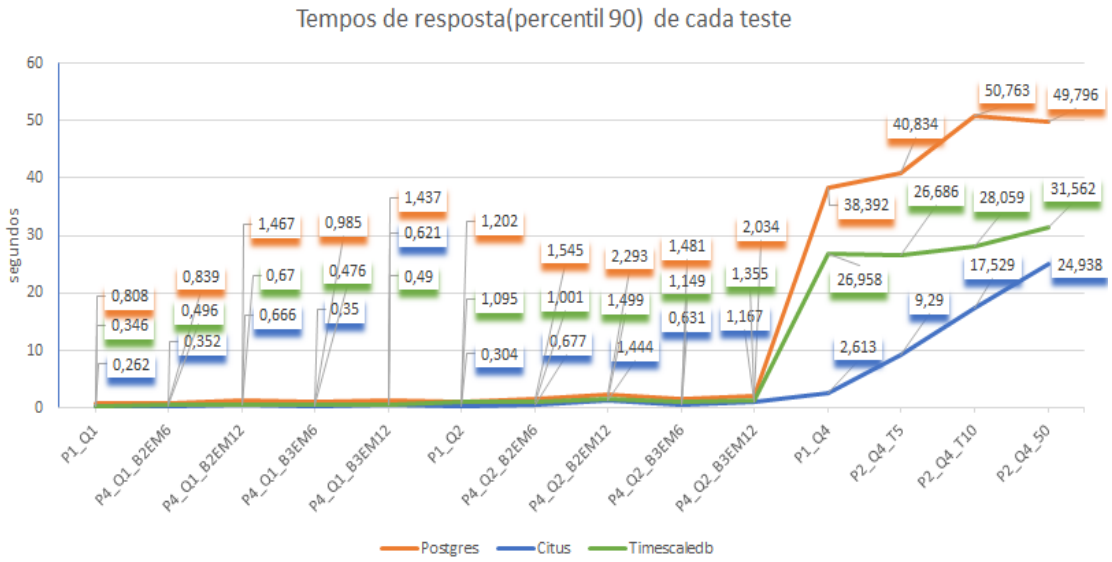


Figura 18: Comparação DBN1 Postgres, Citus e Timescaledb para cada interrogação

Capítulo 5

Conclusões

Relativamente às tecnologias que chegaram à fase de testes, estas cumpriam a grande maioria dos requisitos do utilizador, tais como, uma linguagem de interrogação semelhante ao SQL, o suporte a séries temporais, adequação ao tamanho das tabelas e volume de dados típicos do utilizador (60GB no caso dos dados reais usados), filtragem por intervalos de tempo, agregação e ordenação dos resultados da interrogação, escalabilidade, tempos de resposta a interrogações baixos em condições normais (baseline) e de carga (com barreiras). Perante cada resultado obtido nos testes, foram encontradas soluções (modificando ou acrexentado indices e particionamento) para melhorar o comportamento obtido em termos de tempos de resposta e diminuição das falhas. Como fatores importantes temos também a simplicidade de configuração, fiabilidade da tecnologia para implementação futura, documentação clara, comunidade de suporte ativa e solução *open source,* pelo que foram avaliadas de acordo com o caso de uso do proponente, verificando se o grau de cumprimento e de performance estava dentro dos valores aceitáveis para a organização em cada um dos pontos enunciados.

Para além dos requisitos exigidos, explorou-se também a DBN0 no Postgres, modelizada nas vertentes flat e star. No caso da flat, com toda a informação condensada na mesma tabela sem necessidade de joins para obter as respostas às interrogações e na vertente star com os dados espalhados por um conjunto de dimensões associadas, cada uma, à estrela correspondente. No caso particular da Timescaledb ainda existe falta de maturidade da tecnologia que permita, nesta fase, uma utilização fiável como se constatou durante a sua configuração. Também não possuí uma comunidade de suporte ativa, tornando a configuração dificil e com problemas, como a existência de dependências de extensões dblink na variável "shared_preload_libraries" ainda presentes no *source code*, que como os próprios responsáveis afirmam e como se verificou na configuração do ambiente, esta extensão está negativamente sinalizada por alguns fornecedores

de *cloud-hosting* o que poderá trazer problemas de disponibilidade e recuperação de falhas mais tarde pelo que não cumpre os requisitos do utilizador.

Perante o volume considerado de dados (cerca de 60GB) e a complexidade das interrogações da DBN1, ambos idealizados a partir do caso de uso do proponente, o Citus obteve os melhores resultados nos testes (Figura 18: Comparação DBN1 Postgres, Citus e Timescaledb para cada interrogação), revelando-se assim a melhor solução atual e permitindo também a escalabilidade que o Postgres não dispõe, e, assim, um crescimento futuro da solução. Relativamente à escalabilidade, para além da vertente da quantidade de ligações suportadas (número de utilizadores), faltou verificar o impacto da alteração dos recursos de hardware disponibilizados nos ambientes para, assim, conseguir aferir se a esse incremento corresponde ou não uma redução linear nos tempos de resposta nos testes efetuados. O Citus permite também uma maior facilidade de aprendizagem (SQL) face a outras tecnologias de séries temporais (InfluxDB) ou NOSQL (por exemplo o Cassandra), mencionadas aquando da pesquisa inicial de uma solução, visto que estas exigiriam um maior investimento de tempo (e por inerência económico) e alterações ao modelo do proponente(DNB1), impondo restrições à utilização dos JOINS entre tabelas. Como próximos passos seria interessante uma abordagem comparativa com a tecnologia InfluxDB.

Anexos

Anexo A: Resultados Jmeter

1.1 Postgres DBN0 Flat

<table>
<tr><td>Plano 1</td><td colspan="8">O objetivo deste plano de testes é estabelecer um ponto de partida de comparação(baseline) com os planos posteriores, testando individualmente as queries bem como o seu comportamento quando cruzado (query 1 com a 2 e a 2 com 1) para comparação também com o mesmo plano efetuado na modelização em star da DBN0.</td></tr>
<tr><td colspan="9">Análise dos Dados
Resultados dos listeners do jmeter aggregate report e summary report para cada teste</td></tr>
<tr><td colspan="9">Postgres_DBN0FLAT_P1_Q1</td></tr>
<tr><td>Interrogação</td><td>Amostras</td><td>Média (ms)</td><td>Mediana (ms)</td><td>Percentil 90% (ms)</td><td>Percentil 95% (ms)</td><td>Percentil 99% (ms)</td><td>Minimo (ms)</td><td>Máximo (ms)</td></tr>
<tr><td>1</td><td>200</td><td>16</td><td>14</td><td>15</td><td>16</td><td>22</td><td>10</td><td>467</td></tr>
<tr><td>Falhas %</td><td colspan="2">Pedidos por segundo(throughput)</td><td colspan="2">Recebidos (kb/sec)</td><td>Enviados(kb/sec)</td><td>Std. Dev. (ms)</td><td colspan="2">Tamanho médio das respostas (bytes)</td></tr>
<tr><td>0</td><td colspan="2">60.09615</td><td colspan="2">3.17</td><td>0.0</td><td>32.01</td><td colspan="2">54.0</td></tr>
<tr><td colspan="9">Postgres_DBN0FLAT_P1_Q2</td></tr>
</table>

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	200	20	18	19	27	29	13	477
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	47.16981		34.18		0	32.46	755.0	
Postgres_DBN0FLAT_P1_Q3								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
3	200	20	17	16	28	31	12	477
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	47.90419		56.23		0	32.67	1202.0	
Postgres_DBN0FLAT_P1_Q4								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	200	16	14	20	21	21	10	496
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	59.97001		2.64		0	34.11	45.0	
Postgres_DBN0FLAT_P1_Q5								

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
5	200	19	16	17	28	30	14	476
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	50.67140		33.80		0	32.50	683.0	
Postgres_DBN0FLAT_P1_Q1e2								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
1 e 2	400	14	13	16	21	23	8	476
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	69.50478		27.46		0	23.34	404.5	
Postgres_DBN0FLAT_P1_Q2e1								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2 e 1	400	15	13	16	22	28	9	469
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	63.30116		25.01		0	22.98	404.5	

Interpretação
Resultados Listeners
	As *queries* adaptadas para DBN0(modelização *flat* e *star)* são uma versão simplificada das utilizadas para a DBN1 daí os tempos na ordem dos milésimos de segundo. Dito isto, há a salientar:
	• Discrepância entre os tempos máximos e minimos (centenas vs poucos milésimos de segundo) verificado em todos os testes do plano, ainda que depois os percentis indiquem que a vasta maioria dos resultados permanece na ordem dos poucos milésimos de segundo, o que indica que os máximos foram residuais no contexto das amostras totais realizadas.
	• comparação das *queries* 1 e 2 em separado e simultâneo
	  ○ Em separado média dá 16ms e 20ms
	  ○ Em simultâneo a média dá 14ms (q1eq2) e dá 15ms se invertermos a ordem(2e1)
	  ○ Ora isto indica que a ordem por que são cruzadas não tem impacto relevante

Monitorização zabixx	Plano1 (anexo Postgres DBN0 Flat) das 14:03:11 até às 14:03:52

Plano 2	O objetivo deste plano de testes é aferir o impacto da variação do número de utilizadores(threads) e comparar também com o mesmo plano efetuado na modelização em *star* da DBN0.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

Postgres_DBN0FLAT_P2_Q4_T5

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	1000	15	14	17	20	22	9	554

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	271.59153	11.94	0	21.14	45.0

Postgres_DBN0FLAT_P2_Q4_T10

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil	Percentil 95% (ms)	Percentil	Minimo (ms)	Máximo (ms)

				90% (ms)		99% (ms)		
4	2000	17	16	21	21	31	10	560
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	469.15318		20.62		0	19.32	45.0	
Postgres_DBN0FLAT_P2_Q4_T50								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
4	10000	69	68	117	129	194	10	798
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	675.49311		29.68		0	43.60	45.0	
Interpretação Resultados Listeners	As *queries* adaptadas para DBN0(modelização *flat* e *star)* são uma versão simplificada das utilizadas para a DBN1 daí os tempos na ordem dos milésimos de segundo. Dito isto, há a salientar: • Esperado aumento dos tempos médios de resposta face ao aumento do número de utilizadores(*threads)* testados 1->5->10->50 culminando neste último caso com tempos sensivelmente 5 vezes superiores(16ms/69ms) como se pode verificar nos dados recolhidos.							
Monitorização zabixx	Plano2 (anexo Postgres DBN0 Flat) das 14:03:53 até às 14:04:19							

Plano3	O objetivo deste plano de testes é aferir o impacto da variação de parâmetros das *queries* e comparar também com o mesmo plano efetuado na modelização em *star* da DBN0.
Análise dos Dados Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste	

Postgres_DBN0FLAT_P3_Q1								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
3	200	16	13	21	22	22	10	495
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	59.40006		3.07		0	34.07	53.0	
Postgres_DBN0FLAT_P3_Q2								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	200	18	16	24	25	27	10	492
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	51.34788		37.20		0	33.77	741.9	
Postgres_DBN0FLAT_P3_Q3								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
3	200	20	16	25	26	30	11	470
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	48.30918		56.19		0	32.20	1191.0	

Interpretação Resultados Listeners	As *queries* adaptadas para DBN0(modelização *flat* e *star)* são uma versão simplificada das utilizadas para a DBN1 daí os tempos na ordem dos milésimos de segundo. Dito isto, há a salientar: • Impacto da variação dos parâmetros na query: o Q1. O impacto da variação dos parâmetros da query foram imperceptíveis, quando comparados com o plano1(*baseline*), servindo apenas para aumentar o valor máximo da resposta (467ms -> 495) e o Std dev (32ms -> 34) o Q2. Neste, como na query anterior, a variação dos parâmetros não teve correspondência expressiva nos tempos. o Q3. Maior impacto que nas queries anteriores com aumento dos tempos médios de resposta dos 17ms -> 20ms
Monitorização zabixx	Plano3 (anexo Postgres DBN0 Flat) das 14:04:21 até às 14:04:36

Plano 4	O objetivo deste plano de testes é aferir o impacto em situações de maior carga com a colocação de barreiras. Para isso são sincronizados os pedidos de vários *threads* para que esses pedidos sejam acionados em simultâneo. Pretende-se também a comparação com o mesmo plano efetuado na modelização em *star* da DBN0.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

Postgres_DBN0FLAT_P4_Q1_B2EM6

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
1	1200	17	15	19	21	24	11	529

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	41.19606	2.17	0	22.41	54.0

Postgres_DBN0FLAT_P4_Q1_B2EM12

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
1	2400	19	17	24	26	30	11	661
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	249.11771		13.14		0.0	25.63	54.0	
Postgres_DBN0FLAT_P4_Q1_B3EM6								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
1	1200	20	17	24	27	29	11	566
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	123.82623		6.53		0	28.64	54.0	
Postgres_DBN0FLAT_P4_Q1_B3EM12								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
1	2400	20	18	24	26	33	11	697
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	160.44926		8.46		0.0	30.13	54.0	
Postgres_DBN0FLAT_P4_Q2_B2EM6								

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	1200	19	17	27	29	32	13	595
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	48.80231		35.98		0.0	26.27	755.0	
Postgres_DBN0FLAT_P4_Q2_B2EM12								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	2400	22	20	27	30	39	13	702
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	59.39271		43.79		0	27.99	755.0	
Postgres_DBN0FLAT_P4_Q2_B3EM6								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	1200	21	18	25	28	35	13	579
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	119.82027		88.34		0	28.47	755.0	
Postgres_DBN0FLAT_P4_Q2_B3EM12								

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	2400	22	19	27	29	39	14	728

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	78.80738	58.11	0	31.6	755

Interpretação Resultados Listeners	As *queries* adaptadas para DBN0(modelização *flat* e *star)* são uma versão simplificada das utilizadas para a DBN1 daí os tempos na ordem dos milésimos de segundo. Dito isto, há a salientar:
	• Q1. vs *baseline*
	  ○ De destacar sobretudo o facto de não terem ocorrido erros (0% falhas) nos 4 testes á *query1* pela carga imposta e os os aumentos nos tempos máximos associados ao aumento do número de threads em causa.
	• Q1. vs *bloqueio de 2 ou de 3*
	  ○ Aumento nos tempos de resposta 17 ->20 associado á sincronização passar a ser de 3 e não 2 threads em simultâneo
	• Q2. vs *baseline*
	  ○ De destacar sobretudo o facto de não terem ocorrido erros (0% falhas) nos 4 testes á *query2 pela* carga imposta e os os aumentos nos tempos máximos associados ao aumento do número de threads em causa.
	• Q2. vs *bloqueio de 2 ou de 3*
	  ○ Aumento nos tempos de resposta 19 ->21 associado á sincronização passar a ser de 3 e não 2 threads em simultâneo

Monitorização zabixx	Plano4 (anexo Postgres DBN0 Flat) das 14:04:37 até às 14:07:39

1.2 Postgres DBN0 Star

Plano 1	O objetivo deste plano de testes é estabelecer um ponto de partida de comparação(*baseline)* com os planos posteriores, testando individualmente as *queries* bem como o seu comportamento quando cruzado (query 1 com a 2 e a 2 com 1) para comparação também com o mesmo plano efetuado na modelização flat da DBN0.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

Postgres_DBN0STAR_P1_Q1

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	200	18	16	17	24	25	12	414

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	53.87931	2.84	0	53.87931	54.0

Postgres_DBN0STAR_P1_Q2

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	200	21	19	33	34	36	15	405

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	44.66280	33.63	0	27.67	771.0

Postgres_DBN0STAR_P1_Q3

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil	Percentil 95% (ms)	Percentil	Minimo (ms)	Máximo (ms)

				90% (ms)		99% (ms)		
3	200	24	20	36	37	40	18	429
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	40.96682		47.09		0	29.23	1177.0	
Postgres_DBN0STAR_P1_Q4								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
4	200	9	7	8	8	8	7	395
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	105.09721		1.33		0	27.35	13.0	
Postgres_DBN0STAR_P1_Q5								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
5	200	26	24	25	26	47	21	530
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	37.14710		14.26		0	37.1471	393.0	
Postgres_DBN0STAR_P1_Q1e2								

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
1 e 2	400	17	15	20	21	35	11	427

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	56.44137	22.74	0	21.04	412.5

Postgres_DBN0STAR_P1_Q2e1

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2 e 1	400	16	16	19	25	35	11	399

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	58.43682	23.54	0	19.64	412.5

Interpretação Resultados Listeners	As *queries* adaptadas para DBN0(modelização *flat* e *star)* são uma versão simplificada das utilizadas para a DBN1 daí os tempos na ordem dos milésimos de segundo. Dito isto, há a salientar:
	• Picos máximos bastante acima da média (tal como foi visto também no modelo flat)
	• Nos 7 testes, obtivemos em:
	  ○ 6 deles valores médios superiores à modelização *flat*
	  ○ 7 deles valores máximos inferiores à modelização *flat*
	• A query 4 obteve valores significativamente mais baixos de tempos médios de resposta no modelo *star(9ms média vs 16ms)*

Monitorização zabixx	Plano1 (anexo Postgres DBN0 Star) das 16:41:39 até às 16:42:23

Plano 2	O objetivo deste plano de testes é aferir o impacto da variação do número de utilizadores(threads) e comparar também com o mesmo plano efetuado na modelização flat da DBN0.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

Postgres_DBN0STAR_P2_Q4_T5

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	1000	6	6	9	10	11	3	389

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	537.63441	6.83	0	13.69	13.0

Postgres_DBN0STAR_P2_Q4_T10

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	2000	6	6	9	11	12	3	397

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	1031.45952	13.09	0	11.78	13.0

Postgres_DBN0STAR_P2_Q4_T50

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	10000	22	22	31	44	79	3	537

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	1882.1758	23.89	0	21.74	13.0

Interpretação Resultados Listeners	As *queries* adaptadas para DBN0(modelização *flat* e *star)* são uma versão simplificada das utilizadas para a DBN1 daí os tempos na ordem dos milésimos de segundo. Dito isto, há a salientar: • Bom desempenho até aos 10 utilizadores alcançando inclusive valores inferiores médios, porém com 50 utilizadores a degradação da performance é visível demorando em média o dobro do tempo (9ms ->22ms). • Nos 3 testes: o tempos médios inferiores ao modelo *flat* nos 3 o dos 1->50 utilizadores, o modelo *star* duplica enquanto o modelo *flat* quintuplica
Monitorização zabixx	Plano2 (anexo Postgres DBN0 Star) das 16:42:24 até às 16:42:37

Plano3	O objetivo deste plano de testes é aferir o impacto da variação de parâmetros das *queries* e comparar também com o mesmo plano efetuado na modelização flat da DBN0.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

Postgres_DBN0STAR_P3_Q1

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
1	200	17	16	17	18	26	10	418

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	57.5374	3.08	0	28.59	54.8

Postgres_DBN0STAR_P3_Q2

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	200	23	20	23	37	44	16	426
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	42.24757		29.75		0	29.14	721.1	
Postgres_DBN0STAR_P3_Q3								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
3	200	26	28	38	39	42	17	452
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	36.98225		42.50		0	30.73	1176.9	

Interpretação Resultados Listeners	As *queries* adaptadas para DBN0(modelização *flat* e *star)* são uma versão simplificada das utilizadas para a DBN1 daí os tempos na ordem dos milésimos de segundo. Dito isto, há a salientar:
	• Impacto da variação dos parâmetros na query:
	  ○ Q1. O impacto da variação dos parâmetros da query foram imperceptíveis, quando comparados com o plano1(*baseline*).
	  ○ Q2 e Q3. Aumento de 2ms nos tempos médios de cada query.
	• Variações similares ao modelo flat

Monitorização zabixx	Plano3 (anexo Postgres DBN0 Star) das 16:42:38 até às 16:42:55

Plano 4	O objetivo deste plano de testes é aferir o impacto em situações de maior carga com a colocação de barreiras. Para isso são sincronizados os pedidos de vários threads para que esses

	pedidos sejam acionados em simultâneo. Pretende-se também a comparação com o mesmo plano efetuado na modelização *flat* da DBN0.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

Postgres_DBN0STAR_P4_Q1_B2EM6

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	1200	24	22	32	36	45	16	431

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	214.74588	246.83	0	17.79	1177.0

Postgres_DBN0STAR_P4_Q1_B2EM12

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	2400	27	26	36	39	52	14	440

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	57.77703	66.41	0	15.98	1177.0

Postgres_DBN0STAR_P4_Q1_B3EM6

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	1200	28	26	37	41	48	16	447

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	72.22389	83.02	0	22.12	1177.0

Postgres_DBN0STAR_P4_Q1_B3EM12

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	2400	26	23	34	39	49	16	475

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	147.65596	169.72	0	18.65	1177.0

Postgres_DBN0STAR_P4_Q2_B2EM6

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	1200	22	20	28	34	41	15	476

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	79.50179	59.86	0	19.97	771.0

Postgres_DBN0STAR_P4_Q2_B2EM12

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	2400	23	21	30	35	46	15	451

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	230.30419	173.4	0	15.95	771.0

Postgres_DBN0STAR_P4_Q2_B3EM6

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	1200	24	21	32	35	43	15	455

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	76.26311	57.42	0	22.40	771.0

Postgres_DBN0STAR_P4_Q2_B3EM12

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	2400	22	20	27	35	46	14	475

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	117.68744	88.61	0	18.59	771.0

Interpretação Resultados Listeners	As *queries* adaptadas para DBN0(modelização *flat* e *star)* são uma versão simplificada das utilizadas para a DBN1 daí os tempos na ordem dos milésimos de segundo. Dito isto, há a salientar: • Q1. vs *baseline* ○ De destacar sobretudo o facto de não terem ocorrido erros (0% falhas) nos 4 testes á *query1* pela carga imposta e os aumentos dos tempos médios (18ms ->24ms) • Q1. vs *bloqueio de 2 ou de 3*

	○ Aumento nos tempos de resposta 24ms ->28ms associado á sincronização passar a ser de 3 e não 2 threads em simultâneo • Q2. vs *baseline* ○ De destacar sobretudo o facto de não terem ocorrido erros (0% falhas) nos 4 testes à *query2* pela *carga imposta* e com aumentos nos tempos inferiores ao aumento na query1 • Q2. vs *bloqueio de 2 ou de 3* ○ Aumento nos tempos de resposta 22ms->24ms associado à sincronização passar a ser de 3 e não 2 threads em simultâneo • vs modelo flat: ○ q1: ▪ ambos sem erros (0% falhas) ▪ tempos médios superiores ▪ valores máximos inferiores ○ q2: ▪ ambos sem erros (0% falhas) ▪ tempos médios superiores ▪ valores máximos inferiores
Monitorização zabixx	Plano4 (anexo Postgres DBN0 Star) das 16:42:56 até às 16:45:31

1.3 Postgres DBN1

Plano 1	O objetivo deste plano de testes é estabelecer um ponto de partida de comparação(*baseline)* com os planos posteriores, testando individualmente as *queries* bem como o seu comportamento quando cruzado (*query* 1 com a 2 e a 2 com 1) para comparação também com o mesmo plano efetuado para outras tecnologias de bases de dados temporais.
Análise dos Dados Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste	
Postgres_DBN1_P1_Q1	

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
1	10	866	805	808	808	1416	803	1416
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	1.15393		3.34		0	183.34	2964.0	
Postgres_DBN1_P1_Q2								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	10	1262	1198	1202	1202	1847	1194	1847
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.79158		1.97		0	194.75	2547.0	
Postgres_DBN1_P1_Q3								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
3	10	1838	1663	1951	1951	3132	1661	3132
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.54348		1.54		0	439.69	2911.0	

Postgres_DBN1_P1_Q4								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	10	38111	37656	38392	38392	41391	37225	41391
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.02624		0.02		0	1137.71	658.0	
Postgres_DBN1_P1_Q5								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
5	10	5552	5057	5341	5341	9730	5054	9730
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.18006		0.63		0	1395.15	3583.0	
Postgres_DBN1_P1_Q1e2								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1 e 2	20	1609	1192	1195	1558	12310	801	12310
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	

0	0.61994		1.67		0	2463.85	2755.5	
Postgres_DBN1_P1_Q2e1								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2 e 1	20	1030	827	1191	1198	1806	805	1806
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.96913		2.61		0	257.43	2755.5	
Interpretação Resultados Listeners	Os tempos de resposta estão na ordem dos segundos, a destacar a interrogação nº4 que apresenta valores bastante acima das restantes (<5s *vs* 40s) na medida em que é suposto testar queries mais complexas e a nº4 contempla a pesquisa a dias diferentes (e por por isso a partições diferentes) com 2 dias de dados comparando o desempenho da mesma noutras tecnologias de bases de dados. A execução das interrogações cruzadas 1e2 e 2e1 revelou que a execução na ordem 2e1 tem vantagens nos tempos de resposta 1030ms vs 1609ms nos tempos médios de resposta do conjunto das 2 interrogações.							
Monitorização zabixx	Plano 1 (anexo Postgres DBN1) 16:29:05 -16:38:04							

Plano 2	O objetivo deste plano de testes é aferir o impacto da variação do número de utilizadores(threads) e comparar também com o mesmo plano efetuado noutras tecnologias de bases de dados de séries temporais.
Análise dos Dados Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste	

Postgres_DBN1_P2_Q4_T5								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)

4	50	37772	36837	40834	43337	50548	36325	50548
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas bytes)	
0	0.12814		0.08		0	2756.07	658.0	
Postgres_DBN1_P2_Q4_T10								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	100	45034	42432	50763	71688	79158	37483	79946
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.21492		0.14		0	8948.41	658.0	
Postgres_DBN1_P2_Q4_T50								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	500	14734	10000	41521	49796	57744	10000	60593
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
87.6%	1.43007		0.2		0	12789.45	140.3	
Interpretação Resultados Listeners	Bom desempenho até aos 5 utilizadores com tempos de resposta do parâmetro percentil 90% apenas ligeiramente superiores ao *baseline (*plano 1*)* 38392ms vs 40834ms.							

	Com 10 utilizadores o impacto torna-se visível com tempos médios de resposta do pâmetro percentil 90% a passarem dos 38392ms *baseline* para os 50763ms. Com 50 utilizadores existem bastantes erros das respostas aos pedidos cerca de 87.6% de falhas em 500 amostras com percentil 90% 41521ms e o percentil 99% de 57744ms.
Monitorização zabixx	Plano 2 (anexo Postgres DBN1) das 16:38:04 -16:58:15

Plano3	O objetivo deste plano de testes é aferir o impacto da variação de parâmetros das *queries,* que os resultados não ficam em *cache* (ou seja que os resultados obtidos a cada repetição dos testes no *baseline* são válidos) e comparar também com o mesmo plano efetuado em outras tecnologias de bases de dados de séries temporais.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

Postgres_DBN1_P3_Q1

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	10	846	780	781	781	1449	778	1449

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	1.1805	3.41	0	200.87	2959.0

Postgres_DBN1_P3_Q2

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	10	1261	1194	1194	1194	1879	1191	1879

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	0.79164	1.97	0	205.74	2547.0

Postgres_DBN1_P3_Q3								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
3	10	1667	1604	1615	1615	2228	1602	2228

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	0.59945	1.69	0	186.93	2880.0

Interpretação Resultados Listeners	• Impacto da variação dos parâmetros na query: ○ Sem impacto quando comparados com o plano1(*baseline)* pelo que valida o facto que os resultados nao ficam em *cache*
Monitorização zabixx	Plano 3 (anexo Postgres DBN1) das 16:58:15 - 16:58:56

Plano 4	O objetivo deste plano de testes é aferir o impacto em situações de maior carga com a colocação de barreiras. Para isso são sincronizados os pedidos de vários *threads* para que esses pedidos sejam acionados em simultâneo. Pretende-se também a comparação com o mesmo plano efetuado em outras tecnologias de bases de dados de séries temporais.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

Postgres_DBN1_P4_Q1_B2EM6								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)

1	60	827	798	839	1082	1400	764	1400
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	6.95249		20.12		0	120.48	2965.0	
Postgres_DBN1_P4_Q1_B2EM12								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	120	963	830	1467	1525	1536	779	1536
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0.00	11.37117		32.91		0	246.84	2964.0	
Postgres_DBN1_P4_Q1_B3EM6								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	60	897	868	985	1064	1378	745	1504
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	6.17602		17.88		0	139.92	2964.0	
Postgres_DBN1_P4_Q1_B3EM12								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)

1	120	930	831	1437	1469	1570	765	1601
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0.00	10.36807		30.01		0	223.83	2964.0	
Postgres_DBN1_P4_Q2_B2EM6								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	60	1369	1306	1545	1662	1993	1258	1994
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	4.16898		10.37		0	150.02	2547.0	
Postgres_DBN1_P4_Q2_B2EM12								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	120	1574	1401	2293	2659	2663	1236	2778
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	5.10117		12.69		0	383.80	2547.0	
Postgres_DBN1_P4_Q2_B3EM6								

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	60	1367	1317	1481	1525	1976	1253	1977
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	4.22416		10.51		0	153-43	2547.0	
Postgres_DBN1_P4_Q2_B3EM12								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	120	1540	1407	2034	2350	2609	1301	2639
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	6.84424		17.02		0	336.75	2547.0	

Interpretação
Resultados Listeners
	• Q1. vs *baseline*
	  ○ De destacar sobretudo o facto de não terem ocorrido erros (0% falhas), e do aumento dos tempos de resposta máximos (1416ms -> 1601 teste b3em12)
	• Q1. vs *bloqueio de 2 ou de 3*
	  ○ Sem diferenças significativas nos tempos de resposta
	• Q2. vs *baseline*
	  ○ De destacar sobretudo o facto de não terem ocorrido erros (0% falhas), e do aumento dos tempos de resposta, sobretudo os tempos máximos (1847ms -> 2778ms no teste b2em12 da query2)
	• Q2. vs *bloqueio de 2 ou de 3*
	  ○ Sem diferenças significativas nos tempos de resposta

Monitorização zabixx	Plano 4 (anexo Postgres DBN1) 16:58:56 - 17:01:00

1.4 Citus DBN1

Plano 1	O objetivo deste plano de testes é estabelecer um ponto de partida de comparação(*baseline)* com os planos posteriores, testando individualmente as *queries* bem como o seu comportamento quando cruzado (*query* 1 com a 2 e a 2 com 1) para comparação também com o mesmo plano efetuado para outras tecnologias de bases de dados temporais.							
Análise dos Dados Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste								
Citus_DBN1_P1_Q1								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	10	275	249	262	262	634	170	634
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	3.61664		10.47		0	112.8	2964.0	
Citus_DBN1_P1_Q2								

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	10	308	244	304	304	747	218	747
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	3.23311		8.04		0	149.17	2547.0	
Citus_DBN1_P1_Q3								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
3	10	464	437	483	483	838	350	838
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	2.15100		6.11		0	130.73	2911.0	
Citus_DBN1_P1_Q4								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
4	10	2359	2268	2613	2613	2909	2075	2909
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.42355		0.27		0	241.41	652.0	

Citus_DBN1_P1_Q5								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
5	10	1217	1174	1240	1240	1624	1120	1624
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.82068		2.87		0	139.50	3583.0	
Citus_DBN1_P1_Q1e2								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1 e 2	20	306	289	325	337	833	198	833
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	3.25256		8.75		0	126.81	2755.5	
Citus_DBN1_P1_Q2e1								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2 e 1	20	292	261	352	386	644	214	644
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	

0	3.39443	9.13	0	91.84	2755.5

Interpretação Resultados Listeners	Os tempos de resposta médios estão na ordem das décimas de segundo, a destacar a interrogação 4(2359ms) que apresenta valores acima das restantes, pois esta efetua uma pesquisa com um intervalo de tempo superior (2 dias). A execução das queries cruzadas 1e2 e 2e1 revelou tempos máximos de respectivamente 833ms e 644ms a ponderar na ordenação da sua execução.
Monitorização zabixx	Plano 1 (anexo Citus DBN1) 17:28:16 -17:29:24

Plano 2	O objetivo deste plano de testes é aferir o impacto da variação do número de utilizadores(*threads*) e comparar também com o mesmo plano efetuado noutras tecnologias de bases de dados de séries temporais.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

Citus_DBN1_P2_Q4_T5

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	50	9149	9125	9290	9368	9667	8815	9667

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	0.54502	0.34	0	156.87	646.9

Citus_DBN1_P2_Q4_T10

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	100	17398	17385	17529	17623	17794	17065	17886

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	0.57307	0.36	0	127.79	645.9

Citus_DBN1_P2_Q4_T50								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	500	12592	10000	24938	25019	27331	10000	27410

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
82.0%	3.17015	0.53	0	5643.28	170.9

Interpretação Resultados Listeners	De 1 utilizador para 5 obteve-se um tempo bastante superior ao *baseline* (plano 1), passando de um tempo médio de 2359ms para 9149ms **.** Com 10 utilizadores os tempos médios de resposta face ao teste com 5 utilizadores atingem os 17398ms (cerca de 7x o baseline). Com 50 utilizadores existem bastantes erros das respostas aos pedidos cerca de 82.0% de falhas em 500 amostras com percentil 99% de 27331ms (cerca de 11x o *baseline).*
Monitorização zabixx	Plano 2 (anexo Citus DBN1) 17:29:25 -17:36:42

Plano3	O objetivo deste plano de testes é aferir o impacto da variação de parâmetros das *queries,* que os resultados não ficam em *cache* (ou seja que os resultados obtidos a cada repetição dos testes no *baseline* são válidos) e comparar também com o mesmo plano efetuado em outras tecnologias de bases de dados de séries temporais.
Análise dos Dados Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste	
Citus_DBN1_P3_Q1	

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	10	298	240	255	255	830	226	830
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	3.35008		9.68		0	177.45	2959.0	
Citus_DBN1_P3_Q2								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	10	342	303	336	710	710	263	710
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	2.91206		7.24		0	124.27	2547.0	
Citus_DBN1_P3_Q3								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
3	10	478	416	508	508	898	372	868
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	2.08725		5.87		0	145.49	2880.0	
Interpretação	• Impacto da variação dos parâmetros na query:							

Resultados Listeners	○ Aumento dos tempos médios em todas as 3 queries mas com valores similares aos já obtidos na *baseline* o que valida que os resultados de repetições anteriores não ficam em cache.
Monitorização zabixx	Plano 3 (anexo Citus DBN1) 17:36:43 - 17:36:58

Plano 4	O objetivo deste plano de testes é aferir o impacto em situações de maior carga com a colocação de barreiras. Para isso são sincronizados os pedidos de vários threads para que esses pedidos sejam acionados em simultâneo. Pretende-se também a comparação com o mesmo plano efetuado em outras tecnologias de bases de dados de séries temporais.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

Citus_DBN1_P4_Q1_B2EM6

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	60	278	266	352	373	683	169	697

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	17.21170	49.82	0	92.59	2964.0

Citus_DBN1_P4_Q1_B2EM12

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	120	507	502	666	721	892	150	939

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)

0	10.94018		57.72		0	130.73	2964.0	
Citus_DBN1_P4_Q1_B3EM6								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	60	290	261	350	417	724	158	726
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	16.26016		47.07		0	110.85	2964.0	
Citus_DBN1_P4_Q1_B3EM12								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	120	484	479	621	696	820	161	849
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	19.72711		57.10		0	119.65	2964.0	
Citus_DBN1_P4_Q2_B2EM6								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	60	519	532	677	733	931	223	932
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das	

					respostas (bytes)			
0	9.80232		24.38		0	151.52	2547.0	
Citus_DBN1_P4_Q2_B2EM12								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	120	956	866	1444	1633	1867	225	1874
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	10.97394		27.30		0	301.06	2547.0	
Citus_DBN1_P4_Q2_B3EM6								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	60	568	574	631	644	1117	280	1120
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	9.32401		23.19		0	149.40	2547.0	
Citus_DBN1_P4_Q2_B3EM12								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	120	885	865	1167	1280	1672	274	1701

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	11.44492	28.47	0	227.71	2547.0

Interpretação Resultados Listeners	• Q1. vs *baseline*
	  ○ Sem erros (tal como na *baseline)* com a resposta aos pedidos efetuados.
	• Q1. vs *bloqueio de 2 ou de 3*
	  ○ Sem diferenças significativas nos tempos de resposta
	• Q2. vs *baseline*
	  ○ De destacar sobretudo o facto de não terem ocorrido erros (0% falhas) e do aumento dos tempos de resposta, sobretudo os tempos máximos (747ms -> 1874ms no teste b2em12 da query2)
	• Q2. vs *bloqueio de 2 ou de 3*
	  ○ Aumento de 519ms ->568ms nos tempos médios dos testes b2em6 para b3em6 e diminuição de 956ms para 885ms dos testes b2em12 para b3em12

Monitorização zabixx	Plano 4 (anexo Citus DBN1) 17:36:59 - 17:38:04

1.5 Timescaled DBN1

Plano 1	O objetivo deste plano de testes é estabelecer um ponto de partida de comparação(*baseline)* com os planos posteriores, testando individualmente as *queries* bem como o seu comportamento quando cruzado (*query* 1 com a 2 e a 2 com 1) para comparação também com o mesmo plano efetuado para outras tecnologias de bases de dados temporais.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

TimescaleDB_DBN1_P1_Q1

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)

1	10	400	343	346	346	919	337	919
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	2.49439		7.22		0	172.92	2964.0	
TimescaleDB_DBN1_P1_Q2								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	10	1131	1068	1095	1095	1718	1009	1718
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.88230		2.19		0	196.58	2547.0	
TimescaleDB_DBN1_P1_Q3								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
3	10	1683	1687	1748	1748	2338	1337	2338
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.59358		1.69		0	265.21	2911.0	
TimescaleDB_DBN1_P1_Q4								

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
4	10	27178	26770	26958	26958	30671	26755	30671
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.036792		0.02		0	27178	658.0	
TimescaleDB_DBN1_P1_Q5								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
5	10	6066	5945	6693	6693	6721	5626	6721
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.16480		0.58		0	339.67	3583.0	
TimescaleDB_DBN1_P1_Q1e2								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
1 e 2	20	681	943	949	1091	1149	318	1149
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	1.46466		3.94		0	331.18	2755.5	

TimescaleDB_DBN1_P1_Q2e1								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2 e 1	20	740	456	1077	1078	1713	336	1713
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	1.34816		3.63		0	412.51	2755.5	
Interpretação Resultados Listeners	Os tempos de resposta estão na ordem dos segundos (<10s), a destacar a interrogação 4(36224ms) que apresenta valores acima das restantes (10* em alguns casos), pois esta efetua uma pesquisa com um intervalo de tempo superior (2 dias). A execução das queries cruzadas 1e2 e 2e1 revelou tempos máximos de respectivamente 1149ms e 1713ms a ponderar na ordenação da sua execução.							

Plano 2	O objetivo deste plano de testes é aferir o impacto da variação do número de utilizadores(threads) e comparar também com o mesmo plano efetuado noutras tecnologias de bases de dados de séries temporais.
Análise dos Dados Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste	

TimescaleDB_DBN1_P2_Q4_T5								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	50	27464	27249	29686	30042	30930	25807	30930
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	

0	0.18117		0.12		0	1380.30	656.5	
TimescaleDB_DBN1_P2_Q4_T10								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	100	28528	28059	30998	32643	33094	26852	33195
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	0.33825		0.22		0	1492.74	658.3	
TimescaleDB_DBN1_P2_Q4_T50								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
4	500	13125	10000	31562	35679	38133	10000	38894
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
87.20%	2.49851		0.35		0	8192.44	142.5	
Interpretação Resultados Listeners	De 1 utilizador para 5 obteve-se um tempo superior ao *baseline* (plano 1), passando de um tempo médio de 27178ms para 27464ms. Com 10 utilizadores os tempos médios de resposta atingem os 28528ms (aumento face ao teste com 5 utilizadores). Com 50 utilizadores existem bastantes erros das respostas aos pedidos cerca de 87.20% de falhas em 500 amostras com tempos no parâmetro percentil 99% de 38133ms .							

Plano3	O objetivo deste plano de testes é aferir o impacto da variação de parâmetros das *queries,* que os resultados não ficam em

	cache (ou seja que os resultados obtidos a cada repetição dos testes no *baseline* são válidos) e comparar também com o mesmo plano efetuado em outras tecnologias de bases de dados de séries temporais.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

TimescaleDB_DBN1_P3_Q1

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	10	353	289	341	341	872	287	872

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	2.81849	8.14	0	173.43	2959.0

TimescaleDB_DBN1_P3_Q2

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
2	10	1008	941	947	947	1614	938	1614

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	0.99108	2.47	0	201.82	2547.0

TimescaleDB_DBN1_P3_Q3

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)

3	10	1693	1615	1709	1709	2275	1509	2275

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	0.58990	1.66	0	200.48	2880.0

Interpretação Resultados Listeners	• Impacto da variação dos parâmetros na query: o Aumento dos tempos médios em todas as 3 *queries* mas com valores similares aos já obtidos na *baseline* o que valida que os resultados de queries anteriores não ficam em cache.

Plano 4	O objetivo deste plano de testes é aferir o impacto em situações de maior carga com a colocação de barreiras. Para isso são sincronizados os pedidos de vários threads para que esses pedidos sejam acionados em simultâneo. Pretende-se também a comparação com o mesmo plano efetuado em outras tecnologias de bases de dados de séries temporais.

Análise dos Dados
Resultados dos listeners do jmeter *aggregate report* e *summary report* para cada teste

TimescaleDB_DBN1_P4_Q1_B2EM6

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	60	462	445	496	717	1047	297	1049

Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	11.97844	34.67	0	126.97	2964.0

TimescaleDB_DBN1_P4_Q1_B2EM12

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)

1	120	429	399	537	670	973	253	998
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	22.90513		66.30		0	119.62	2964.0	
TimescaleDB_DBN1_P4_Q1_B3EM6								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	60	470	447	476	561	1038	306	1038
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	11.86474		34.34		0	138.20	2964.0	
TimescaleDB_DBN1_P4_Q1_B3EM12								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Minimo (ms)	Máximo (ms)
1	120	390	370	490	641	970	244	972
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	24.67613		71.43		0	125.92	2964.0	
TimescaleDB_DBN1_P4_Q2_B2EM6								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil	Percentil 95% (ms)	Percentil	Minimo (ms)	Máximo (ms)

				90% (ms)		99% (ms)		
2	60	979	939	1001	1344	1675	847	1678
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	5.87027		14.60		0	162.49	2547.0	
TimescaleDB_DBN1_P4_Q2_B2EM12								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	120	1092	986	1499	1581	1865	848	1906
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	9.43916		23.48		0	258.10	2547.0	
TimescaleDB_DBN1_P4_Q2_B3EM6								
Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	60	979	920	1149	1178	1645	764	1645
Falhas %	Pedidos por segundo*(throughput)*		Recebidos (kb/sec)		Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)	
0	5.34141		13.31		0	180.64	2547.0	
TimescaleDB_DBN1_P4_Q2_B3EM12								

Interrogação	Amostras	Média (ms)	Mediana (ms)	Percentil 90% (ms)	Percentil 95% (ms)	Percentil 99% (ms)	Mínimo (ms)	Máximo (ms)
2	120	1079	980	1355	1567	2313	786	2315

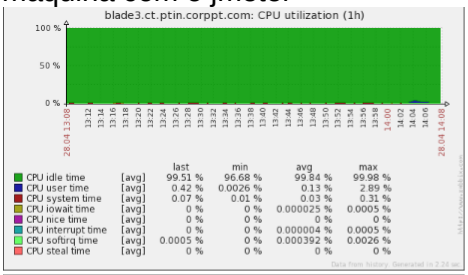
Falhas %	Pedidos por segundo*(throughput)*	Recebidos (kb/sec)	Enviados(kb/sec)	Std. Dev. (ms)	Tamanho médio das respostas (bytes)
0	9.39555	23.37	0	276.53	2547.0

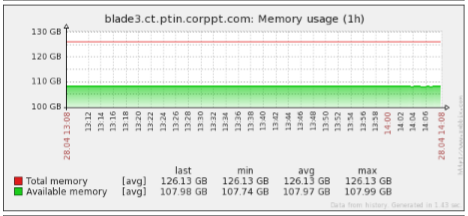
Interpretação
Resultados Listeners
	• Q1. vs *baseline*
	  ○ Sem erros (tal como na *baseline)* com a resposta aos pedidos efetuados.
	• Q1. vs *bloqueio de 2 ou de 3*
	  ○ Sem diferenças significativas nos tempos de resposta
	• Q2. vs *baseline*
	  ○ De destacar sobretudo o facto de não terem ocorrido erros (0% falhas) e do aumento dos tempos de resposta, sobretudo os tempos máximos (1718ms -> 2315ms no teste b3em12 da *query2*)
	• Q2. vs *bloqueio de 2 ou de 3*
	  ○ Sem diferenças significativas nos tempos de resposta

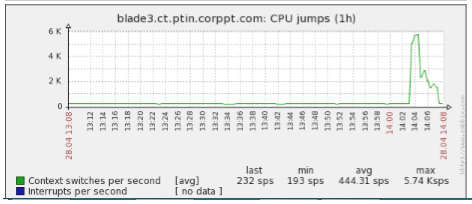
Anexo B: Resultados Zabbix

2.1 Postgres DBN0 Flat

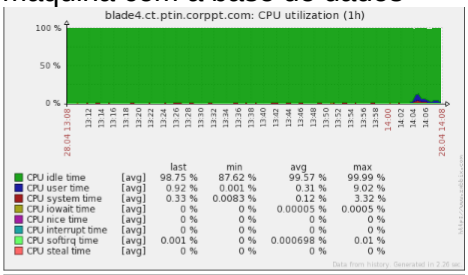
máquina com o jmeter

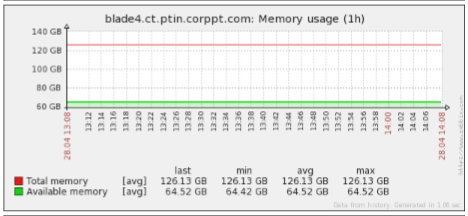


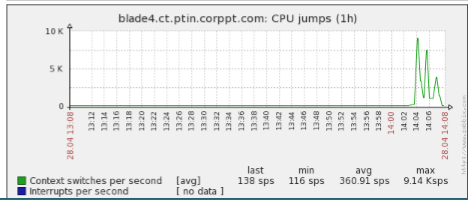


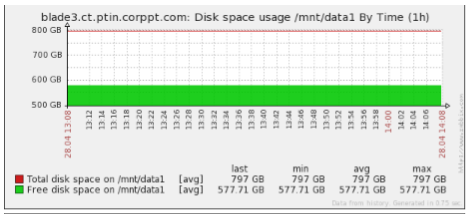


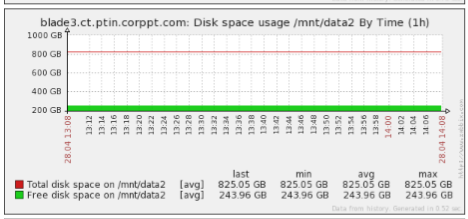
máquina com a base de dados

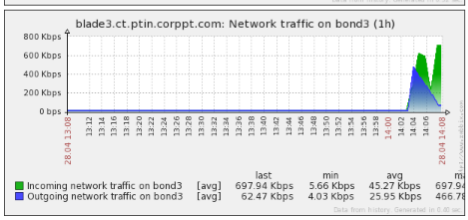


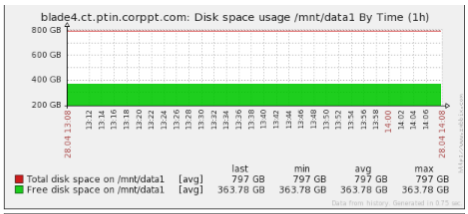


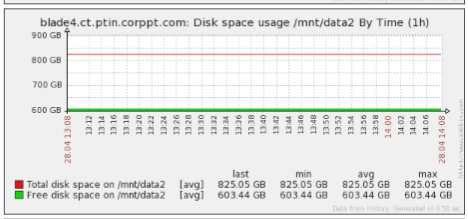


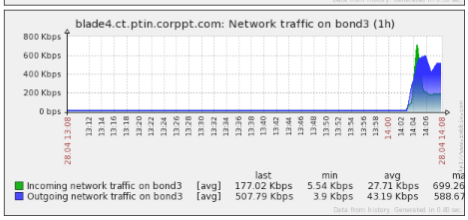








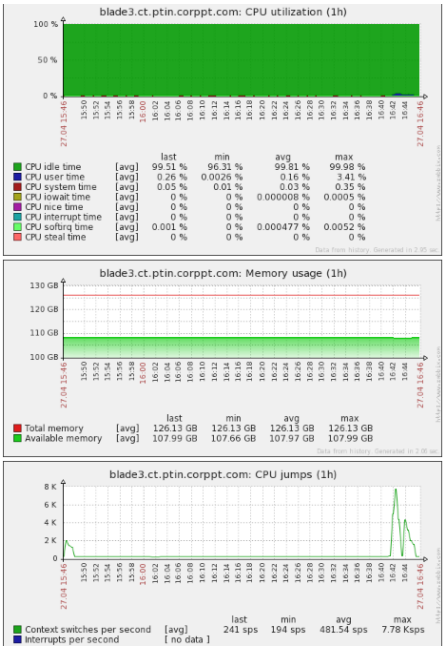




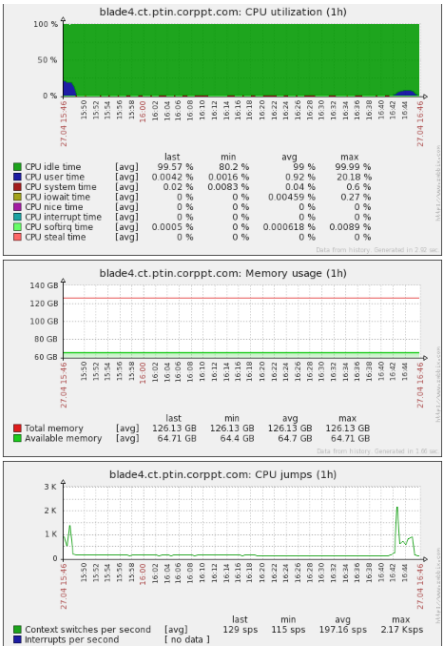
Análise à monitorização Zabbix (apartir das 14:03h)	**CPU e I/O**	**Network**
máquina testes	Utilização máxima do Cpu pelo utilizador = 2.89% e tempo de espera máximo para completar operações de *IO* (leitura e escrita) 0.0005%. Picos de 5740 *switches per second* durante os testes	máximos de recebimento de 697kbps e envio de 466kbps
máquina da base de dados	Utilização máxima do Cpu pelo utilizador = 9.02% e tempo de espera máximo para completar operações de IO(leitura e escrita) 0.0005%. Picos de 9140 *switches per second* durante os testes	máximos de recebimento de 699kbps e envio de 588kbps

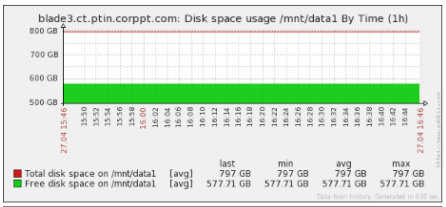
2.2 Postgres DBN0 Star

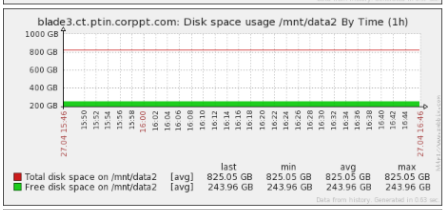
máquina com o jmeter

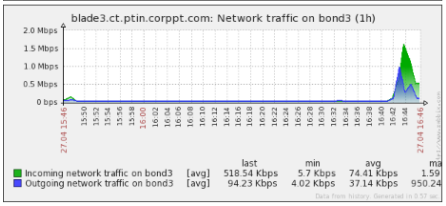


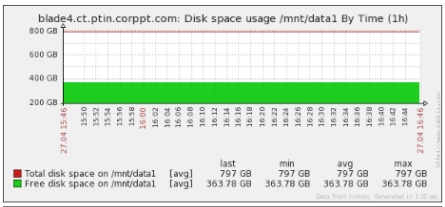
máquina com a base de dados

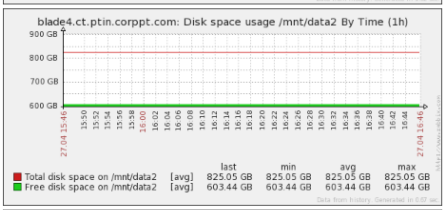


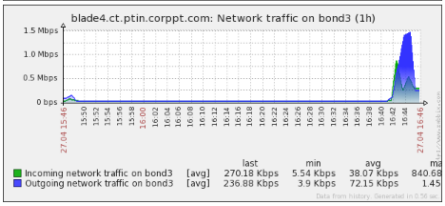








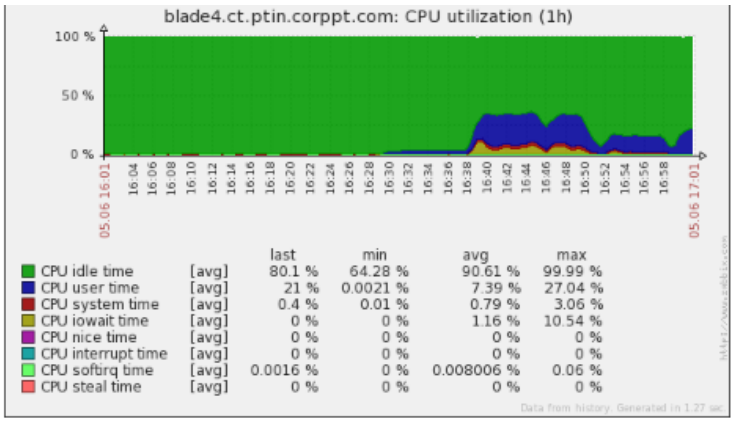


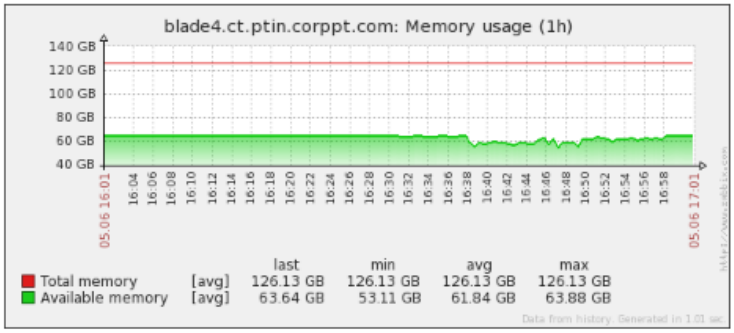


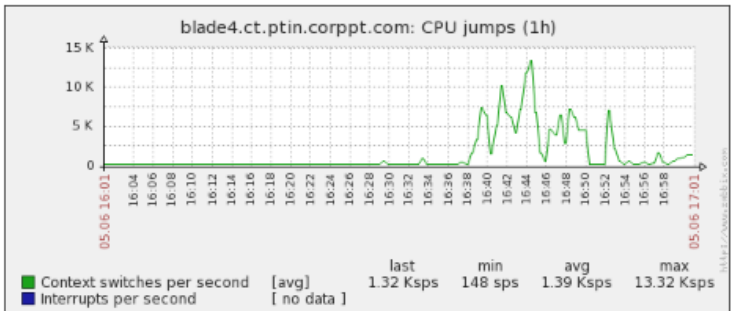
Análise à monitorização Zabbix(apartir das 16:41:39)	**CPU e I/O**	**Network**
máquina testes	Utilização máxima do Cpu pelo utilizador = 3.41% e tempo de espera máximo para completar operações de IO (leitura e escrita) 0.0005%. Picos de 7780 *switches per second* durante os testes (superiores aos 5740 do *flat)*	máximos de recepção de 1590kbps (vs 697kbps *flat)* e envio de 950kbps(vs 466kbps)
máquina da base de dados	Utilização máxima do Cpu pelo utilizador = 20.18% e tempo de espera máximo para completar operações de IO (leitura e escrita) 0.27%. Picos de 2170 *switches per second* durante os testes (inferiores aos 9140 do flat)	máximos de recebimento de 840kbps(vs 699 *flat)* e envio de 1450kbps (588 *flat)*

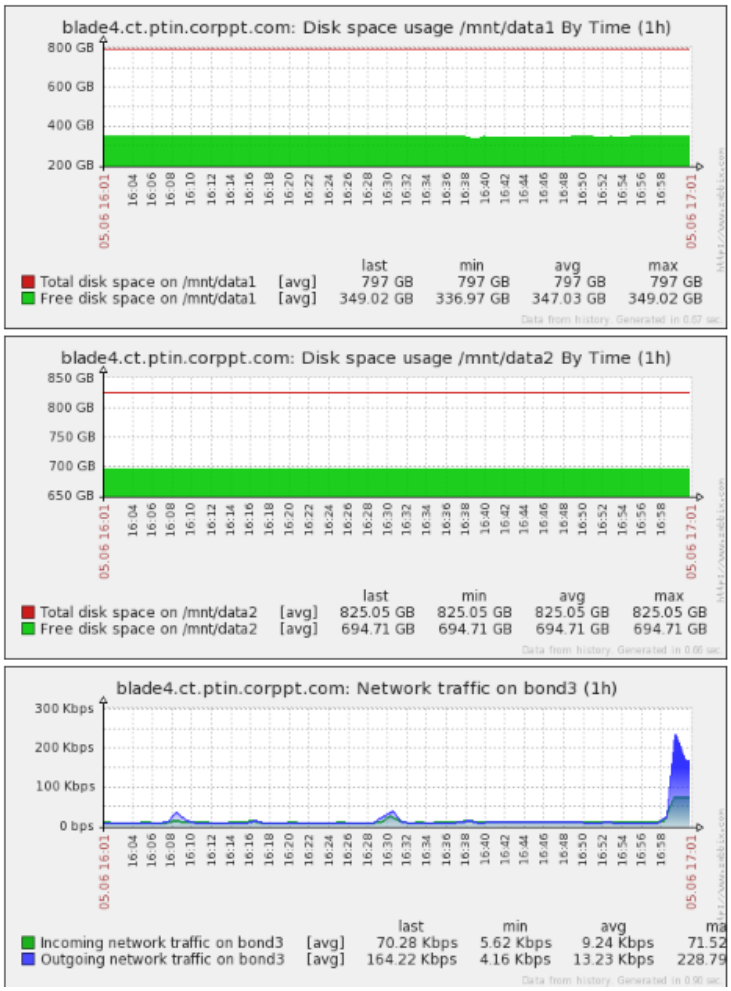
2.3 Postgres DBN1

máquina com a base de dados



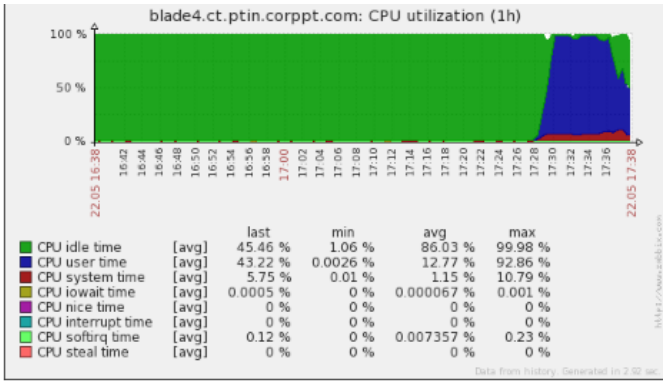


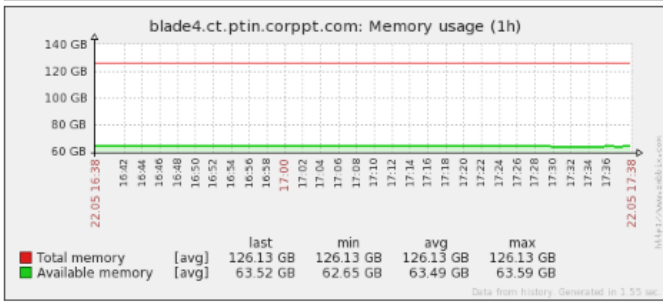


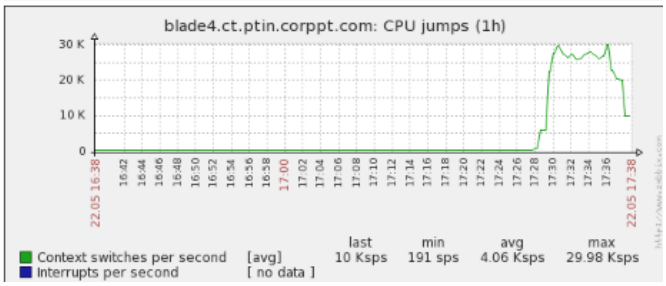


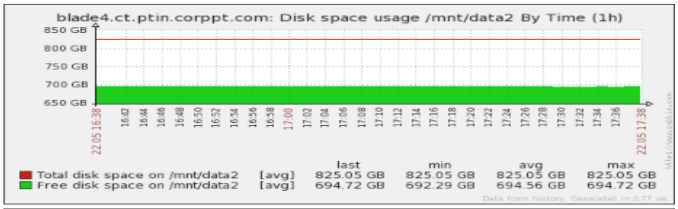
Análise à monitorização Zabbix(duração 31m58.877s)	**CPU e I/O**	**Network**
máquina da base de dados	Utilização máxima do Cpu pelo utilizador = 27.04% e tempo de espera máximo para completar operações de IO (leitura e escrita) 10.54% picos de 13320 *switches per second* durante os testes	máximos de recebimento de 71.52kbps e envio de 228.79kbps

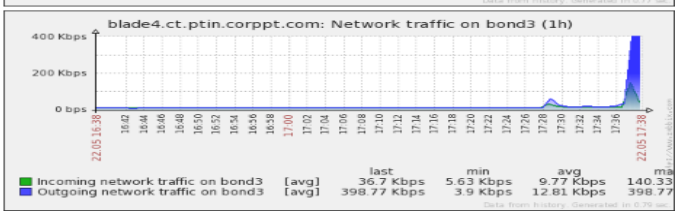
2.4 Citus DBN1











Análise à monitorização Zabbix(início 17:28:16 até 17:38:04)	**CPU e I/O**	**Network**
máquina com base de dados	Utilização máxima do Cpu pelo utilizador = 92.86% e tempo de espera máximo para completar operações de IO (leitura e escrita) 0,001% picos de 29980 *switches per second* durante os testes	máximos de recebimento de 140.33 e envio de 398.77kbps

Referências

APACHE. "Apache Kudu". Acedido a 3 de Fevereiro de 2017. https://kudu.apache.org/overview.html.

"Apache Spark". Acedido a 3 de Fevereiro de 2017. http://spark.apache.org/.

Barnsteiner, Felix. "Elasticsearch as a Time series Data Store", 4 de Novembro de 2015. Acedido a 26 de Janeiro de 2017. https://www.elastic.co/blog/elasticsearch-as-a-time-series-data-store.

Beard, Jeremy. 2014. "How to: Process Time-Series Data Using Apache Crunch". Acedido a 26 de Janeiro de 2017. https://blog.cloudera.com/blog/2014/05/how-to-process-time-series-data-using-apache-crunch/.

"BlinkDB". Acedido a 3 de Fevereiro de 2017. http://blinkdb.org/.

Citusdata. "citus query processing". Acedido a 17/5/2017. https://docs.citusdata.com/en/latest/performance/query_processing.html.

———. "Postgres that allows you to scale out horizontally". Acedido a 3 de Fevereiro de 2017. https://www.citusdata.com/product.

Cloudera. 2017. "Cloudera Impala guide". Acedido a 3 de Febreiro de 2017. http://www.cloudera.com/documentation/enterprise/5-4-x/topics/impala.html.

"Data warehousing OLAP". Acedido a 3 de Fevereiro de 2017. https://www.tutorialspoint.com/dwh/dwh_olap.htm.

"The database for large-scale event analytics". 2016. Acedido a 3 de Fevereiro de 2017. https://eventql.io/.

Elliot, Steve. 2015. "Elasticsearch as a Time Series Database - Does it work". Acedido a 26 de Janeiro de 2017. http://engineering.laterooms.com/elasticsearch-as-a-time-series-database-does-it-work-part-1/.

GreenPlum. "GreenPlum database". Acedido a 3 de Fevereiro de 2017. http://greenplum.org/.

grisha. 2015. "Armazenamento". Acedido a 9 de Fevereiro de 2017. https://grisha.org/blog/2015/09/23/storing-time-series-in-postgresql-efficiently/.

Hobbs, Tyler. 2012. "Advanced Time Series with Cassandra", 28 de Março de 2012. http://www.datastax.com/dev/blog/advanced-time-series-with-cassandra.

Hortonworks. "apache_hadoop". Acedido a 26 de Janeiro de 2017. http://hortonworks.com/apache/hadoop/.

hypertables. http://www.timescale.com/how-it-works.html

InfluxData. 2017. http://www.prnewswire.com/news-releases/influxenterprise-12-delivers-greater-than-50-percent-increase-in-write-performance-along-with-advanced-backup-and-restore-capabilities-300395352.html.

"InfluxDb - Time series data storage". Acedido a 26 de Janeiro de 2017. https://www.influxdata.com/time-series-platform/influxdb/.

"InfluxDB system Properties". Acedido a 26 de Janeiro de 2017. http://db-engines.com/en/system/InfluxDB.

monetdb. "The column-store pioneer". Acedido a 3 de Fevereiro de 2017. https://www.monetdb.org/Home.

"opentsdb". Acedido a 3 de Fevereiro de 2017. http://opentsdb.net/.

"opm- open postgresql monitoring". Acedido a 9 de Fevereiro de 2017. http://opm.io/.

Pelkonen, T., S. Franklin, J. Teller, P. Cavallaro, Q. Huang, J. Meza e K. Veeraraghavan. 2015. "Gorilla: A fast, scalable, in-memory time series database". *Proceedings of the VLDB Endowment* no. 8 (12):1816-1827. https://www.scopus.com/inward/record.uri?eid=2-s2.0-84953869901&partnerID=40&md5=42029c62641048aea4bae0228e58a0da.

Persen, Todd. 2016. "InfluxDB Markedly Outperforms Elasticsearch in Time-Series Data & Metrics Benchmark". Acedido a 26 de Janeiro de 2017. https://www.influxdata.com/influxdb-markedly-elasticsearch-in-time-series-data-metrics-benchmark/.

"pganalyze- PostgreSQL Performance Monitoring". Acedido a 9 de Fevereiro de 2017. https://pganalyze.com/.

"pgcluu- PostgreSQL Cluster utilization". Acedido a 9 de Fevereiro de 2017. http://pgcluu.darold.net/.

pivot. https://www.tutorialspoint.com/dwh/images/pivot.jpg.
"Postgres File system level backup". Acedido a 9 de Fevereiro de 2017. https://www.postgresql.org/docs/9.1/static/backup-file.html.
"Postgres timeseries-tips". Acedido a 9 de Fevereiro de 2017. http://no0p.github.io/postgresql/2014/05/08/timeseries-tips-pg.html.
"PostgreSql Documentation". https://www.postgresql.org/about/.
Processamento das interrogações no Citus. https://docs.citusdata.com/en/latest/performance/query_processing.html.
roll_up. https://www.tutorialspoint.com/dwh/images/rollup.jpg.
Schwartz, Baron. "Time-Series Database Requirements", 8 de Junho de 2014. https://www.xaprb.com/blog/2014/06/08/time-series-database-requirements/.
Sheahan, Adrian Cockcroft and Denis. 2011. "Benchmarking Cassandra Scalability on AWES - Over a million Writes per second". Acedido a 26 de Janeiro de 2017. http://techblog.netflix.com/2011/11/benchmarking-cassandra-scalability-on.html.
slice. https://www.tutorialspoint.com/dwh/images/slice.jpg.
timescaledb: sql made scalable for time-series. http://www.timescale.com/papers/timescaledb.pdf.
tutorialspoint. Jmeter How it works.
———. "jmeter Tutorial". Acedido a 17 de Março de 2017. https://www.tutorialspoint.com/jmeter/.
"What is Cassandra?". Acedido a 26 de Janeiro de 2017. http://cassandra.apache.org/.
Whipple, Karen. 2016. "Time Series Data Is the New Big Data". Acedido a 25 de Janeiro de 2017. https://www.mapr.com/blog/time-series-data-new-big-data.
"Zabbix documentation". Acedido a 11/4/2017. https://www.zabbix.com/documentation/2.2/manual/concepts/agent#supported_platforms.
Zabbix linux example. http://www.zabbix.com/zabbix_agent.