ANÁLISE DE DESEMPENHO DE CONSULTAS DE BANCO DE DADOS RELACIONAL EM AMBIENTE LOCAL E VIRTUALIZADO: Um estudo de caso utilizando o MySQL 8.0

Guilherme Alves Fernandes*

Hudson Silva de Souza**

RESUMO

A utilização da computação e do poder de processamento para atender as necessidades diárias dos usuários de serviços de programas tornou-se algo extremamente comum. Aplicativos que oferecem acesso a sistema de transmissão de grande quantidade de dados, como vídeos e imagens, são alguns dos mais utilizados pela maioria das pessoas. A demanda por desempenho computacional e pela crescente necessidade de se armazenar dados em bancos de dados tornou-se exponencialmente maior e uma forma muito interessante de atender esta demanda é a utilização de computadores na “nuvem”. Mas a utilização de computação na nuvem culmina na “virtualização” do hardware para se processar e distribuir de forma mais eficiente os recursos do equipamento, o que tende a causar uma perda de performance. Este trabalho busca analisar a perda de performance na utilização desse tipo de sistema quando se trabalha com bancos de dados do tipo MySQL. O foco deste trabalho é a utilização do *software* BenchmarkSQL para se produzir tabelas e gráficos comparativos que permitem averiguar como a performance é afetada ao se utilizar diferentes tipos de hardware que possuem a mesma arquitetura base.

Palavra-chave: Banco de dados, desempenho e servidores na nuvem.

ABSTRACT

Using computing and processing power to meet the daily needs of users of service programs has become extremely common. Applications that provide access to a large amount of data, such as videos and images, are some of the most used by most people. The demand for computational performance, therefore, has become exponentially greater and a very interesting way to meet this demand is the use of computers in the “cloud”. But the use of cloud computing entails the “virtualization” of the hardware to process and distribute equipment resources more efficiently, which causes a performance loss. This work seeks to analyze the loss of performance in the use of this type of system when working with MySQL databases. The focus of this work is the use of BenchmarkSQL *software* to produce comparative tables and graphs that allow us to verify how performance is affected when different types of hardware that have the same base architecture are used.

* Rede de Ensino Doctum – Unidade Caratinga – guilhermealvesfernande@gmail.com – graduando em Ciência da Computação

** Rede de Ensino Doctum – Unidade Caratinga – hudson@doctum.edu.br – Professor – Orientador(a) do trabalho

Keywords: Database, performance, Cloud server.

1. Introdução

A prática de utilizar a computação de nuvem se tornou tão simples e acessível que para a maioria das corporações contratar um serviço na nuvem é uma decisão direta e não questionada. Especificamente nos últimos anos, novas tecnologias permitiram a criação de cluster (conjunto de computadores) de processadores capazes de serem escalados à medida que for necessário e ampliar o acesso à funcionalidade proposta pela empresa.

Em relação a escalabilidade os servidores em nuvem, são inegavelmente superiores devido ao fato de serem construídos justamente para atender uma demanda de forma dinâmica, com uma grande disponibilidade de computadores. Embora a fácil ampliação do alcance dos serviços seja importante, um fator raramente levado em consideração é o efeito da virtualização no desempenho dos *softwares* (programas de computador) sendo executados em ambientes virtuais.

A utilização de *softwares* em nuvem coloca uma nova separação entre o *software* e o hardware (equipamento) diretamente. Quanto mais separações existem maior o tempo de processamento necessário para se resolver requisições e processos de hardware. Ao mesmo tempo, a capacidade de se escalar servidores na nuvem é sua maior vantagem, permitindo que se tenha acesso à quantidade certa de processamento para uma aplicação em determinado momento.

Várias empresas possuem interesse em reduzir seus custos e garantir a eficiência de suas aplicações. Portanto, a hipótese que este trabalho propõe é a de que embora possa existir uma diferença na performance entre o servidor local e o na nuvem, esta não será suficiente para sobrescrever a vantagem de possuir servidores escaláveis.

Acredita-se que, mesmo que um hardware local possua processadores mais potentes e maiores recursos físicos, os gastos podem ser inferiores quando se comparado a um ambiente na nuvem equivalente.

Este trabalho teve como objetivo analisar o sistema MySQL utilizando um benchmark (testes com objetivo de avaliar desempenho) que possibilitou averiguar a diferença de desempenho entre um computador pessoal e um servidor na nuvem. O trabalho não foi focado na questão de escalabilidade, mas sim na utilização de um hardware de alto nível em comparação a um equivalente na nuvem. Como objetivo secundário também foram efetuados testes em diferentes modelos de hardware para analisar se existe maior diferença que pode ser observada.

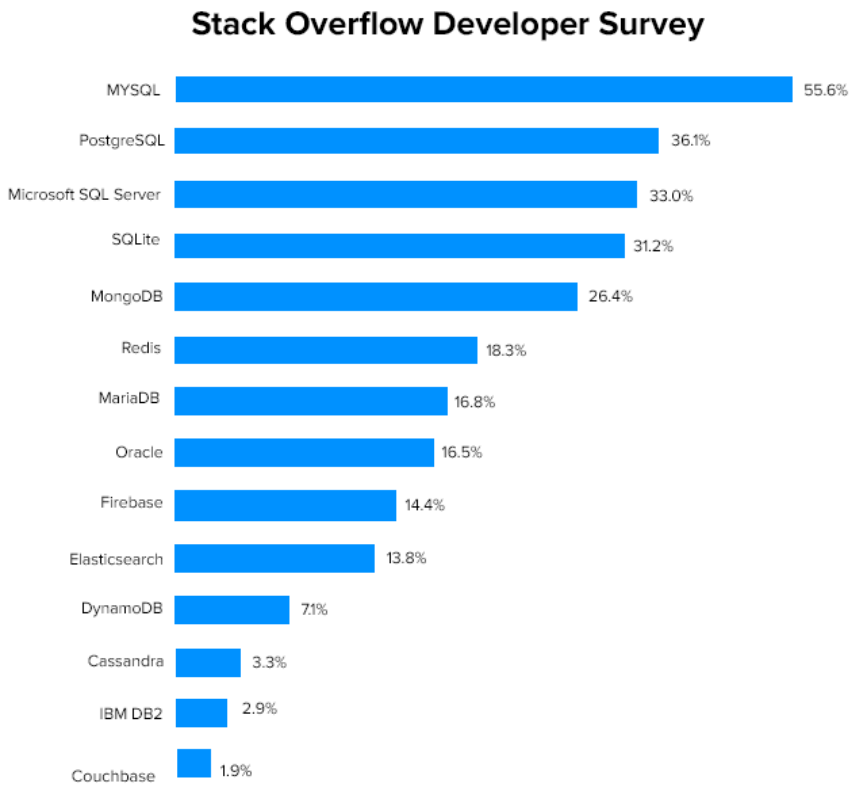
2. Referencial Teórico

Este capítulo aborda as principais definições, a seleção de autores para validar o trabalho e os principais temas utilizados como alicerce para o desenvolvimento.

2.1. Banco de dados

Criado inicialmente em 1985 por Edgar F. Codd, os bancos de dados relacionais, embora não sejam os únicos, são os mais amplamente utilizados por empresas e por entidades públicas, segundo pesquisa publicada no site appinventiv em Julho de 2021. Dentre eles, o que mais se destaca é o MySQL, sendo utilizado por mais de 55% dos desenvolvedores pesquisados.

Figura 1 - Comparativo sobre popularidade de bancos de dados (2021)



Fonte: Artigo publicado no site appinvent: Top 15 Databases to Use in 2021 and Beyond

Segundo o site de distribuição do MySQL sua popularidade em sua utilização decorre de vários fatores, dentre os quais: boa velocidade de processamento, facilidade de utilização, compatível com o uso de querys (consulta ao banco de dados) SQL (*Structured Query Language*) que são o padrão da maioria dos bancos atuais, possui acesso *multi-thread* possibilitando o acesso de diversos clientes ao mesmo tempo, arquivo compacto, facilidade de utilização e instalação em diversos sistemas operacionais, e documentação aberta que facilita resolução de problemas.

Uma técnica comumente utilizada dentro da área de computação é a abstração, em que se reconstrói o problema isolando os aspectos relevantes de um todo e permite repetir o mesmo de forma individual. Machado (2020) apresenta algumas dessas formas em conjunto com o conceito de cardinalidade que permite a conexão de dados dentro de uma estrutura lógica, possibilitando que bancos de

dados sejam relacionados e assim agregando grande quantidade de informações de forma concisa e fácil acesso.

2.2. Computação em Nuvem

A definição direta de computação da nuvem vem do *National Institute of Standards and Technology* (NIST), que define a mesma como:

> A computação em nuvem é um modelo para permitir acesso onipresente, conveniente e sobre demanda à rede a um pool compartilhado de recursos de computação configuráveis que podem ser rapidamente provisionados e liberados com mínimo esforço de gerenciamento ou interação com o provedor de serviços. Este modelo de nuvem é composto por cinco características essenciais, três modelos de serviço e quatro modelos de implantação.

Desde a concepção das primeiras redes de computadores em 1997 pela *Advanced Research Projects Agency Network* (ARPANET), a ideia de dividir e delegar tarefas já era implementada para agilizar os diferentes processos que os computadores de baixo desempenho eram capazes de efetuar. Esse conceito foi popularizado com redes mais complexas e o uso comercial da internet.

Segundo Luo (2009), os primeiros computadores de nuvem foram criados para empresas de grande porte com o objetivo de replicar os sistemas das mesmas e unificar suas bases de informação e garantir acesso ao redor do globo. Com sua popularização, empresas responsáveis por oferecer o hardware sem se responsabilizar pelo *software* surgiram.

Luo (2009) ainda afirma que com sistemas online de fácil acesso e preços mais palpáveis, empresas gigantes como a Amazon expandiram os serviços de armazenamento e de computação desde o início dos anos 2000. Desde então, é possível alugar um servidor virtualizado na nuvem e dinamicamente expandir suas capacidades à medida que o sistema requisitar, sem que com isso fosse necessário a expansão e investimento em equipamentos físicos.

2.3. Teste de Software

O escopo do teste de *software* é amplo, devido ao enorme número de variáveis que podem ser analisadas, mas uma boa definição para o objetivo de se testar e analisar um *software* é proposta por Souza (2018). Segundo seu estudo, "O teste de *software* é um processo relacionado ao desenvolvimento de *software* que tem como principal objetivo revelar falhas por meio da análise dinâmica de programas". O autor também aponta para a norma ISO/IEC 25010:2011, que proporciona uma série de requisitos e métricas que são padrões da indústria de *software* para a obtenção de resultados em análises de desempenho de sistemas, como demonstrado na Figura 2.

Figura 2 - Tipos de testes de desempenho conforme a norma ISO/IEC/IEEE

Tipo de teste	Definição
Teste de carga	Avaliar o comportamento de um item de teste em condições **esperadas de carga** variável (uso baixo, típico e de pico).
Teste de estresse	Avaliar o comportamento de um item de teste em condições de **carga acima dos requisitos** de capacidade antecipados ou especificados ou da disponibilidade de recursos abaixo dos requisitos mínimos exigidos.
Teste de capacidade	Avaliar o nível no qual o **aumento de carga** (de usuários, transações, armazenamento de dados, etc.) **compromete a capacidade** de um item de teste para sustentar o desempenho requerido.
Teste de resistência	Avaliar se um item de teste pode sustentar uma **carga necessária continuamente** por um período de tempo determinado.
Teste de volume	Avaliar a capacidade do item de teste processar **volumes de dados** especificados (usualmente no ou **próximos ao limite máximo de capacidade** especificada) em termos de *throughput*, capacidade de armazenamento ou ambos.

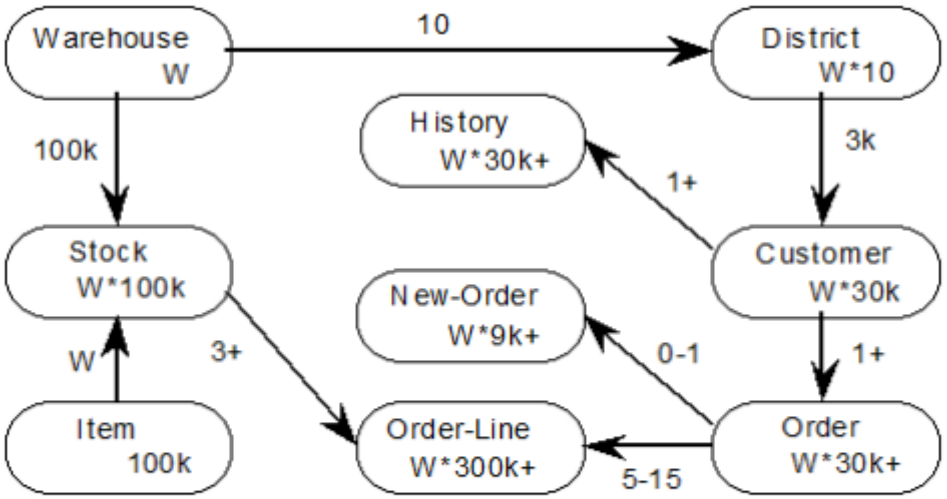
Fonte: Souza, Thiago (2018 p.147)

Outro autor que complementa o assunto sobre testes de *softwares* é Mesquita (2011). Referido autor estabelece que um sistema computacional que inicia algum tipo de requisição para realizar ação ou serviços, pode ter seu desempenho analisado em relação à velocidade de navegação, capacidade de armazenamento, número de acessos simultâneos, tamanho de requisições, número de relações efetuadas, inserções e remoções de dados. Ele também leva em conta quais tipos de dados devem ser analisados: "A avaliação de desempenho dos sistemas computacionais deverá considerar o tipo de requisições recebidas, o tipo de clientes e o tipo do serviço a ser realizado".

Um dos padrões mais utilizados para medir o desempenho de bancos de dados é o TPC-C. Aprovado em 1992, o mesmo é considerado referência para análise de desempenho devido a sua maior complexidade e escalabilidade. O mesmo foi desenvolvido com a capacidade de produzir diversos tipos de transações simultaneamente, em um banco de dados mais complexo e menos especializado, sendo assim facilmente utilizado em diferentes casos, desde comércio, indústria e serviços.

Segundo Maiello (2016), “As transações projetadas nesse Benchmark simulam um típico sistema de aquisição de produtos, com as principais funcionalidades encontradas nesse tipo de aplicação”. O sistema é composto por um total de 9 tabelas que possuem conexões diversas entre si. A Figura 3 demonstra uma representação gráfica do mesmo.

Figura 3 - Relações de banco de dados no protocolo TPC-C



Fonte: site TPC-C benchmark

Um dos *softwares* que são capazes de produzir os resultados esperados pelo protocolo TPC-C é o BenchmarkSQL. Segundo sua documentação o mesmo simula 5 tipos de transações com o objetivo de simular o uso real de um sistema sendo

elas: novas ordens (*New-Order*), operações de pagamento (*Payment*), ordem de fila de status (*Order-Status*), bens enviados (*Delivery*), fila de inventário (*Stock-Level*).

Uma leitura da documentação produzida pelo site TPC permite que seja retirado as informações necessárias para se avaliar o teste e quais métricas são observadas pelo mesmo, sendo elas:

New-Order representa uma nova compra de um usuário, onde uma quantidade semi aleatória de itens são selecionados para um cliente e produzem um novo processo de venda. 1% das vendas são retornadas como uma forma hipotética de simular falha do usuário. A mesma possui um peso médio, com grande frequência de escrita e leitura e precisa de uma resposta rápida do sistema.

Payment produz um pagamento de um usuário aleatório para uma *warehouse* e o processo de pagamento é logado no sistema. O método é considerado leve, com frequentes leituras e escritas e uma resposta rápida.

O *Order-Status* seleciona um cliente aleatoriamente, lê sua última compra e disponibiliza o status de cada item. Peso médio, baixa frequência de leitura com resposta rápida.

Delivery seleciona aleatoriamente um pacote em trânsito, faz *update* da balança do mesmo e deleta a ordem das novas ordens. Baixa leitura e escrita com tempo de resposta imprevisível.

Por fim, *Stock-Level* que seleciona aleatoriamente os últimos 20 pedidos checa o inventário por aqueles que possuem uma quantidade menor que um valor aleatório e exibe os resultados. Pesado, com baixa leitura e alto tempo de resposta.

Os resultados são então produzidos em um tempo especificado pelo usuário e a métrica de *Throughput* (tpmC) é produzida, indicando quantos novos pedidos podem ser produzidos por minuto enquanto o sistema mantém todas as outras funcionalidades necessárias para a utilização do mesmo como um ambiente válido de trabalho. O resultado leva em consideração várias métricas como quantidade de transações efetuadas, tempo de resposta das mesmas, e definições de aceitabilidade definidas no padrão TPC-C.

2.4. Softwares

2.4.1. PHP

Sendo uma das linguagens de desenvolvimento web mais utilizadas no mundo, o PHP (PHP: *Hypertext Preprocessor*) está envolvido em grande parte das aplicações produzidas e desenvolvidas para serem utilizadas através da internet. Oliveira (2018) mostra como a linguagem pode ser utilizada para diversas tarefas e sua aplicação.

Segundo seu manual online gratuito, PHP foi desenvolvido em 1994 por Rasmus Lerdof, inicialmente com o objetivo de ser “um simples conjunto de binários *Common Gateway Interface* (CGI) escrito em linguagem de programação C”. A linguagem foi liberada em 1995 como PHP *Tools* como um *software* aberto e com capacidade de conexão de banco de dados.

PHP, como é conhecido atualmente, foi inicialmente liberado em 1997 com a versão 2.0 e possui suporte nativo a banco de dados DBM, mSQL, e Postgres95, *cookies*, funções de apoio definidas pelo usuário, documentação mais robusta, novos padrões de sistema, acesso a controle de navegação e funções mais complexas de controle de ambiente.

Segundo Crawford (2017), PHP é uma das tecnologias com maior abrangência na produção de sites. Em seu estudo de 2016, é demonstrado que aproximadamente 11,7% dos sites da internet utilizam a linguagem. Em termos práticos, isso indica aproximadamente 42 milhões de sites utilizando o mesmo. Crawford (2017) compara em seu estudo as outras linguagens populares e demonstra a popularidade de PHP, sendo mais utilizada que as outras linguagens por um fator de mais de 10 vezes.

Uma das maiores vantagens que o PHP possui é seu suporte nativo a diversos formatos de banco de dados e sintaxes de dados. Sendo o mais importante entre eles o SQL. Segundo DB-Engines, um site especializado em manter um registro atualizado da popularidade de sistemas de banco de dados, o padrão que mais se destaca é o MySQL.

2.4.2. MySQL

Segundo Cardoso (2009), ao pensar em banco de dados ou um sistema do mesmo, deve-se levar em consideração como os dados são armazenados, as estruturas que mantêm a ligação entre eles, a segurança que estes dados devem possuir e a capacidade de armazenamento.

Cardoso (2009) introduz os parâmetros de funcionamento da estrutura SQL, explicando e exemplificando como cada instrução deve ser executada e como cada comando pode ser utilizado. O autor também demonstra como ampliar a complexidade de instruções, adicionando mais tabelas relacionadas, aplicando sub-pesquisas, maior tempo de resposta de inserções e de atualizações de várias tabelas.

2.4.3. Java

Outras tecnologias utilizadas para se comunicar com bancos de dados é a linguagem Java. Como indicado em seu próprio site, a mesma é a maior linguagem de programação do mundo, com mais de 3 bilhões de dispositivos a utilizando.

Como mencionado em sua documentação, Java funciona ao se utilizar uma máquina virtual que serve de ambiente seguro para a execução do código, o que em princípio permite que o programa seja capaz de executar exatamente da mesma forma em qualquer máquina que o mesmo seja colocado, basta que seja instalado o sistema de virtualização que é distribuído pela Oracle, sua empresa pai. Isso torna Java uma excelente ferramenta para se criar programas e algoritmos que devem ser executados em qualquer ambiente de sistema operacional.

Um exemplo de ambiente em que Java é capaz de compilar é o sistema operacional Ubuntu, que é um sistema operacional de código aberto, construído a partir do núcleo Linux, desenvolvido e mantido com o objetivo de ser utilizado por usuários e servidores através do globo.

2.4.4. Linux

Em sua documentação o Ubuntu explica sobre as funcionalidades de um sistema operacional, onde o mesmo é capaz de administrar e gerenciar os recursos de um sistema, desde componentes de hardware e sistemas de arquivos a programas de terceiros, estabelecendo a interface entre o computador e o usuário. Devido ao fato de sua instalação ser gratuita e ser uma das maiores distribuições Linux, como informado em seu site, o Ubuntu permite que vários servidores e desenvolvedores produzam as mais diversas tecnologias para solucionar problemas através das diversas linguagens de programação do mundo.

3. Metodologia

O principal objetivo deste trabalho é analisar a diferença de desempenho entre as duas formas comuns de se trabalhar com serviços online: através de um servidor físico local ou acessando um servidor de nuvem. Inicialmente, deve-se garantir que os ambientes de teste são comparáveis com todos os *softwares* possuindo a exata mesma versão, os hardwares escolhidos devem ser devidamente documentados e garantir que apenas as diferenças escolhidas sejam fatores importantes no resultado. Por fim, os *softwares* de *benchmarking* são aplicados e executados produzindo um resultado.

3.1. Metodologia de testes

Como base para os testes, foi utilizado uma importante norma definida pela ISO/IEC/IEEE 29119-2:2013 que informa quais passos devem ser tomados para conseguir desenvolver uma boa metodologia de testes. Tais passos são: configuração do ambiente de testes, planejamento de testes, design e implementação de testes, execução de testes, comunicação de incidentes e conclusão do projeto de testes.

3.1.1. Configuração do ambiente de testes

O sistema operacional selecionado para os testes foi o Ubuntu em sua versão 20.04.03 LTS. Sua escolha foi devido a sua grande utilização em ambientes de desenvolvimento, como informado na seção 2.4.4, e pela familiaridade com o próprio. A versão do kernel utilizada é a 5.11.0-40-generic em todos os ambientes de teste.

Para a execução dos testes, foram utilizados os compiladores PHP, Java e Apache Ant. A versão utilizada do PHP foi a 7.4.3 devido a sua estabilidade e facilidade de encontrar informações sobre utilização. No Java, foi utilizado a versão "11.0.11" do openjdk, devido a sua estabilidade e compatibilidade com o *software* de testes. O responsável por compilar o código final do ambiente de testes foi o Apache Ant(TM) versão 1.10.7.

Como ambiente de banco de dados foi selecionado o MySQL 8.0.26, devido a sua compatibilidade com o *software* de teste e sua estabilidade, como apontado nas seções de Testes de *Softwares* e *Softwares* no Referencial Teórico.

Os *softwares* de teste escolhidos foram o dbt2 e o BenchmarkSQL. A versão mais atual de ambos foi selecionada de seu ambiente GitHub, sendo estas a 0.37.50.16 para o primeiro e a 5.0 para o segundo.

Por fim, os *hardwares* foram escolhidos para garantir que os mesmos fossem baseados em arquiteturas similares. O motivo da escolha dos hardwares foi devido a seus processadores serem do tipo *chiplet*, com grupos de 8 núcleos que podem ser combinados para formar um processador com mais núcleos que se comunicam através de uma banda de comunicação de alta velocidade chamada *infinity fabric*. Os mesmos possuem uma arquitetura padrão que é utilizada tanto em servidores quanto em computadores de trabalho, permitindo que o desempenho núcleo a núcleo entre processadores da mesma geração seja extremamente similar mesmo entre diferentes níveis de hardware.

Sendo assim, os seguintes computadores foram selecionados para os testes, sendo referidos com seu respectivo cenário.

Tabela 1: Sistemas de teste

	CPU	Memória	SSD
Cenário 1A	Ryzen 7 5800x 3.8 - 4.7 Ghz com 8 Threads	32 GB, 3200 MT/s	NVME 512 GB 1800 MBps
Cenário 1B	Ryzen 7 5800x 3.8 - 4.7 Ghz com 16 Threads	32 GB, 3200 MT/s	NVME 512 GB 1800 MBps
Cenário 2	Ryzen 5 2600x 3.6 - 4.2 Ghz com 12 Threads	16 GB, 3000 MT/s	SATA 480 GB 520 MBps
Cenário 3	Ryzen 9 3900xt 3.8 - 4.7 Ghz com 24 Threads	64 GB, 3200 MT/s	NVME 512 GB 1800 MBps
Cenário 4	AMD Epic Milan 2.4 - 3.45 Ghz com 8 Núcleos (virtualizado)	64 GB, 3200 MT/s	NVME 128 GB 1800 MBps

Fonte: Elaborada pelo autor com base nos computadores disponíveis.

Os scripts utilizados para o teste se encontram em uma *fork* do GitHub[1] bem quanto as configurações necessárias para preparar o ambiente de teste.

3.1.2. Planejamento de testes

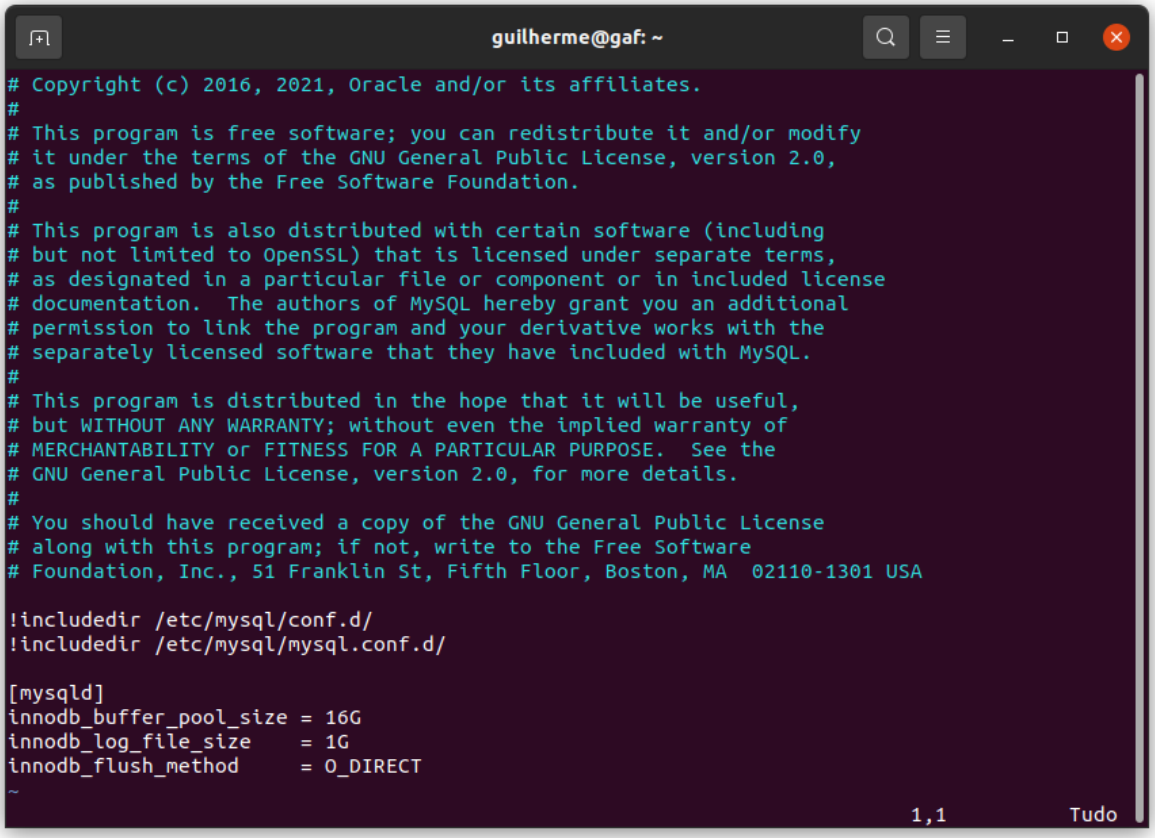
Para a execução dos testes foi necessário utilizar uma heurística analítica para encontrar um padrão de execução e de configuração dos mesmos. Foram testados diversos padrões, mas alguns se mostraram mais viáveis e estáveis.

Para o MySQL, as seguintes configurações foram definidas como demonstrado na figura 4

:

[1] Disponível em: <https://github.com/GuilhermeAlvesFernandes/benchmarksql-1/tree/5.0-mysql-support-opt-2.1/run>. Acesso em: 16 de novembro de 2021.

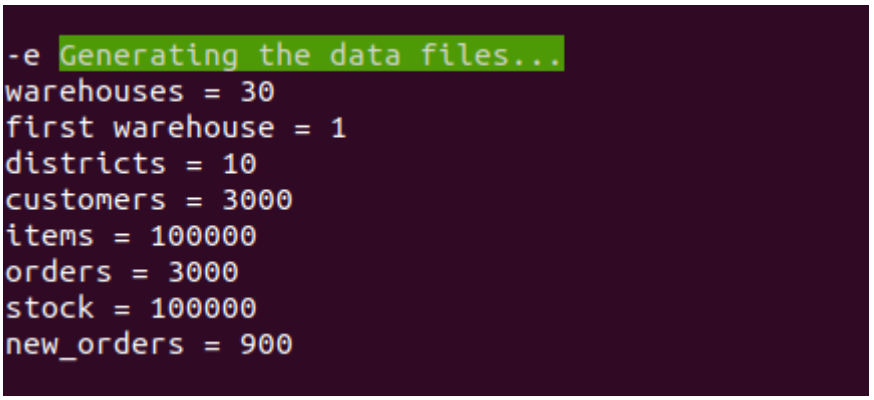
Figura 4 - Arquivo de configuração do MySQL



Fonte: Compilação do autor

Para a execução do dbt2 foram escolhidas as seguinte configurações, demonstrado na Figura 5:

Figura 5 - Configurações de execução dbt2



Fonte: Compilação do autor

Para a execução do Benchmarksql foram escolhidas as seguintes configurações, demonstrado na Figura 6:

Figura 6 - Configurações de execução do BenchmarSQL

```
db=mysql
driver=com.mysql.cj.jdbc.Driver
conn=jdbc:mysql://localhost:3306/tpcc?-
useSSL=false&useServerPrepStmts=true&useConfigs=maxPerformance&rewriteBatchedStatements=true
user=guilherme
password=123

warehouses=100
loadWorkers=10

terminals=16
//To run specified transactions per terminal- runMins must equal zero
runTxnsPerTerminal=0
//To run for specified minutes- runTxnsPerTerminal must equal zero
runMins=10
//Number of total transactions per minute
limitTxnsPerMin=0

//Set to true to run in 4.x compatible mode. Set to false to use the
//entire configured database evenly.
terminalWarehouseFixed=true

//The following five values must add up to 100
//The default percentages of 45, 43, 4, 4 & 4 match the TPC-C spec
newOrderWeight=45
paymentWeight=43
orderStatusWeight=4
deliveryWeight=4
stockLevelWeight=4

// Directory name to create for collecting detailed result data.
// Comment this out to suppress.
//resultDirectory=my_result_%tY-%tm-%td_%tH%tM%tS
//osCollectorScript=./misc/os_collector_linux.py
//osCollectorInterval=1
//osCollectorSSHAddr=user@dbhost
//osCollectorDevices=net_eth0 blk_sda
```

Fonte: Compilação do autor

As configurações foram replicadas de forma independente e idêntica em todos os hardwares utilizados.

3.1.3. Design e implementação de testes

Com o objetivo de tornar os testes o mais replicável possível, foram criados *scripts* de automatização dos mesmos. Os *softwares* foram analisados e os comandos necessários para sua execução foram implementados em um único arquivo que, quando chamado, executa todos os outros. Para garantir que os testes

ocorram de forma limpa, o banco de dados foi excluído e os arquivos de log deletados para evitar influência dos mesmos após cada interação.

O arquivo de automatização permite que os testes sejam executados um número controlado de vezes e também salva os resultados em um arquivo de log de texto, como demonstrado na Figura 7 e Figura 8.

Figura 7 - Arquivo de automatização e parametrização do dbt2

```
for filename in `find /tmp/dbt2/data -type f -name *.data`; do
    echo $filename
    mv $filename $filename.bak
    iconv -f ISO-8859-1 -t UTF-8 $filename.bak -o $filename
    rm $filename.bak
done

  log 'Loading the data into the database...'
  scripts/mysql/mysql_load_db.sh \
    --path /tmp/dbt2/data \
    --local \
    --mysql-path "$mysql_path" \
    --host "$host" \
    --user "$user" \
    --password "$pass"

  log 'Loading the stored procedures...'
  scripts/mysql/mysql_load_sp.sh \
    --client-path "$mysql_dir_path" \
    --sp-path storedproc/mysql \
    --host "$host" \
    --user "$user" \
    --password "$pass"

for i in 1 2 3 4 5 6 7 8 9 10
do
  log "Executando o teste &i"
  log 'Running the benchmark...'
  scripts/run_mysql.sh \
    --connections 20 \
    --time 300   \
    --warehouses 30 \
    --zero-delay \
    --host "$host" \
    --user "$user" \
    --password "$pass"
  echo "PURGE BINARY LOGS BEFORE NOW();" | mysql -u $user
done
  log 'Cleaning database...'
  echo "DROP DATABASE dbt2;" | mysql -u $user

# rm -rf /tmp/dbt2
```

Fonte: Compilação do autor

Figura 8 - Arquivo de automatização do BenchmarkSQL

```
#!/bin/sh

for i in 1 2 3 4 5 6
do
    service mysql restart
    echo "PURGE BINARY LOGS BEFORE NOW();" | mysql -u super

    ./runDatabaseDestroy.sh props.mysql
    ./runDatabaseBuild.sh   props.mysql
    ./runBenchmark.sh       props.mysql > "Log_teste_$i.txt"

    service mysql restart
    echo "PURGE BINARY LOGS BEFORE NOW();" | mysql -u super

done
```

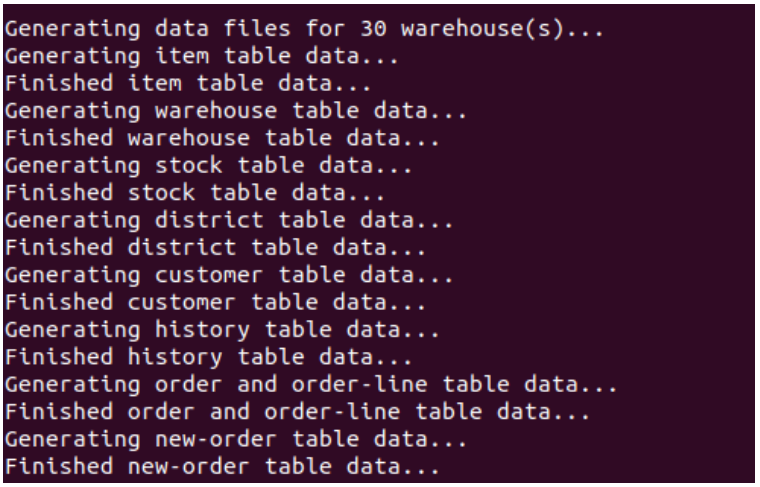
Fonte: Compilação do autor

3.1.4. Execução de testes

Para evitar perda de comunicação com o servidor durante a execução, foi utilizado o *software* tmux, que permite a criação de terminais diversos e em que a execução se encontra de forma interna o que evita que alguma variação na conexão afete os resultados.

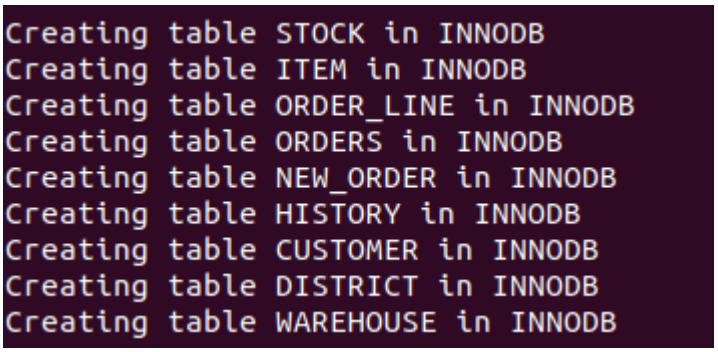
Os testes, após compilados e automatizados, necessitam apenas da chamada de seu arquivo de automatização para sua execução. O processo que segue, prepara todo o ambiente de testes do banco de dados e repete os testes 6 vezes cada para analisar variações de resultado, como demonstrado na Figura 9, Figura 10, Figura 11 e Figura 12.

Figura 9 - Geração de warehouses dbt2



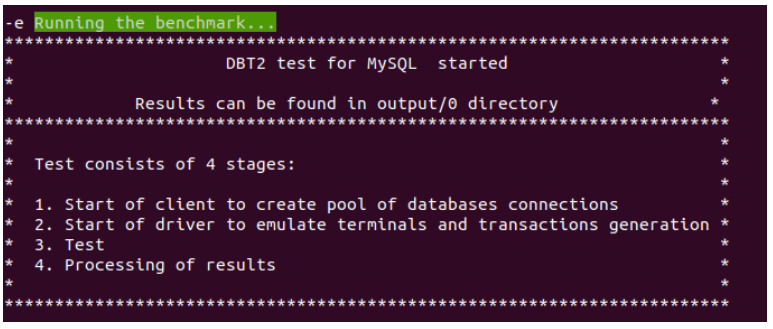
Fonte: Compilação do autor

Figura 10 - Tabelas utilizadas pelo dbt2



Fonte: Compilação do autor

Figura 11 - arquivo de execução dbt2



Fonte: Compilação do autor

Figura 12 - Leitura de parâmetros dbt2

```
DATABASE NAME:                  dbt2
DATABASE USER:                  super
CLIENT PORT:                    30000
DATABASE HOST:                  127.0.0.1
DATABASE PORT:                  3306
DATABASE CONNECTIONS:           20
TERMINAL THREADS:               300
SCALE FACTOR(WAREHOUSES):       30
FIRST WAREHOUSE:                1
LAST WAREHOUSE:                 30
SPREAD                          1
TERMINALS PER WAREHOUSE:        10
DURATION OF TEST(in sec):       300
SLEEPY in (msec)                100
ZERO DELAYS MODE:               1
```

Fonte: Compilação do autor

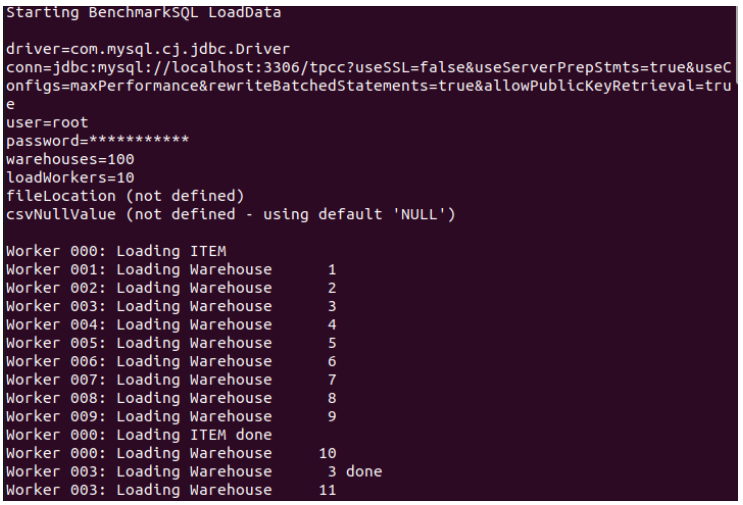
A execução do teste BenchmarkSQL é demonstrado na Figura 13, Figura 14, Figura 15 e Figura 16.

Figura 13 - Distribuição de criação das tabelas de teste BenchmarkSQL

```
# ------------------------------------------------------------
# Loading SQL file ./sql.mysql/tableDrops.sql
# ------------------------------------------------------------
drop table bmsql_config;
drop table bmsql_new_order;
drop table bmsql_order_line;
drop table bmsql_oorder;
drop table bmsql_history;
drop table bmsql_customer;
drop table bmsql_stock;
drop table bmsql_item;
drop table bmsql_district;
drop table bmsql_warehouse;
drop table bmsql_config;
Unknown table 'tpcc.bmsql_config'
# ------------------------------------------------------------
# Loading SQL file ./sql.mysql/tableCreates.sql
# ------------------------------------------------------------
```

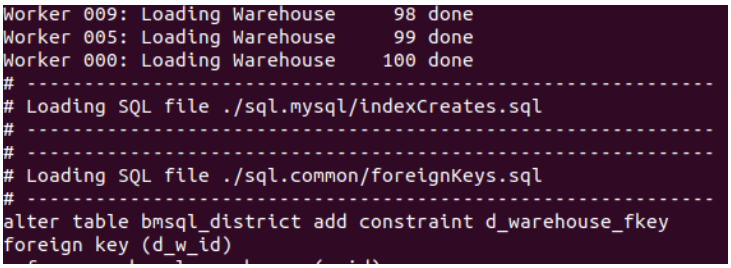
Fonte: Compilação do autor]

Figura 14 - Conexão e criação de warehouses BenchmarkSQL



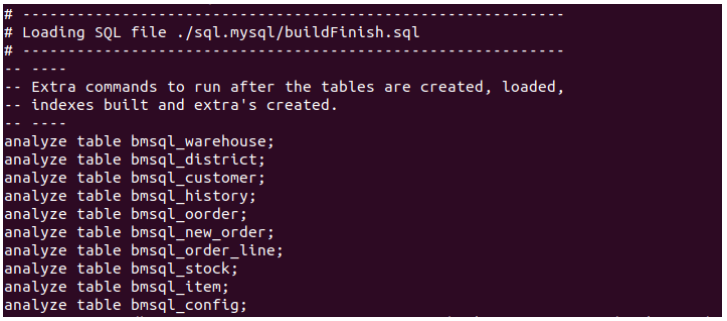
Fonte: Compilação do autor

Figura 15 - Criação de index e conexão de chaves estrangeiras



Fonte: Compilação do autor

Figura 16 - Execução do BenchmarkSQL



Fonte: Compilação do autor

3.1.5. Comunicação de incidentes

Durante a execução dos testes do projeto dbt2, foi observado que existe uma grande inconsistência nos resultados, sendo que os mesmos variam por mais de quatro vezes o valor e até mesmo retornam números negativos como resultado. Ao se analisar mais a fundo o projeto foi constatado que o mesmo possui dificuldades para ser executado em computadores atuais, sendo incapaz de escalar com hardware moderno.

Após deliberação foi utilizado o BenchmarkSQL produzido pelo mesmo desenvolvedor, mas que possui ferramentas mais modernas em seu código, possibilitando escalar mais facilmente. Sendo assim os resultados obtidos pelo dbt2 foram descartados.

3.1.6. Conclusão do projeto de testes

A bateria de testes foi corretamente concluída e uma tabela foi produzida com os resultados. Os mesmos são discutidos no item 4. RESULTADOS.

4. Resultados

Os resultados dos experimentos baseados nos cenários apresentados na Tabela 1 e discutidos na seção 3 são apresentados na subseção 4.1 bem como uma discussão sobre os valores encontrados é feita na subseção 4.2.

4.1. Resultados obtidos

Os parâmetros usados na execução dos testes foram escolhidos de forma empírica visando maximizar a utilização do conjunto CPU e memória possuída pelo mesmo. Com o objetivo de se conseguir uma comparação mais válida foi efetuada a aplicação de testes estatísticos. Segundo Kanji (2006), testes estatísticos são úteis para se descobrir o que testar para evitar armadilhas e má interpretação da conclusão das análises. Ao se observar as colocações de Rodriguez (2015), é necessário se encontrar dois resultados para os testes. O primeiro possui o objetivo

de averiguar a distinção dos resultados e o segundo a real diferença entre os mesmos.

Existem diversos testes estatísticos que podem ser utilizados e escolher o mais adequado para certo problema não é uma tarefa trivial. Sendo assim, a ferramenta STAC[2] apresentada por Rodríguez (2015) foi utilizada para indicar e executar os testes feitos neste trabalho.

O teste de Friedman foi utilizado para garantir com 95% de significância estatística que os cenários avaliados são realmente distintos entre si. O mesmo é utilizado para se criar um Ranking entre os resultados.

Após ser averiguado que existe diferença significativa entre os resultados foi aplicado o teste de Nemenyi para identificar quais cenários são diferentes dos demais. A significância estatística utilizada é de 95%.

De acordo com o estudo de Rodríguez (2015) este algoritmo é aplicável em situações onde se deseja descobrir a diferença entre duas situações com valores aproximados e referenciar caso os mesmos não sejam capazes de fornecer resultados estatisticamente significativos.

A Tabela 2 apresenta os resultados obtidos. A primeira coluna apresenta qual métrica o resultado se refere e as demais colunas apresentam as médias e o desvio padrão (entre parênteses) obtido nos resultados dos experimentos de cada um dos cenários. São destacados individualmente os melhores resultados obtidos por métrica através da utilização da ferramenta de negrito. As informações sobre as métricas dispostas nas linhas estão detalhadas na seção 2.3, com exceção da métrica de Ranking que foi obtida através do teste de Friedman cujo valor é inversamente proporcional ao resultado obtido.

Tabela 2 - Comparação de desempenho entre cenários analisados.

	Cenário 1A	Cenário 1B	Cenário 2	Cenário 3	Cenário 4
Measured tpmC (NewOrders)	74169.14 (597.10)	87641.36 (1242.56)	13883.97 (1434.36)	**89913.76 (1729.51)**	33841.29 (506.91)
Measured tpmTOTAL	164831.55	195193.68	30852.90	**199828.61**	75224.74

[2]STAC <http://tec.citius.usc.es/stac/>

	(1300.52)	(2708.71)	(3191.92)	**(3714.01)**	(1078.57)
Transaction Count	1648362.17 (13007.42)	1950174.00 (26677.95)	308754.83 (31820.35)	**1998370.67 (37142.72)**	752288.00 (10786.22)
executeTime[Payment]	3308788.17 (17340.51)	2948268.17 (27294.30)	**3906854.67 (149709.76)**	2734013.17 (13047.77)	3199462.33 (43101.52)
executeTime[Order-Status]	**46016.83 (442.04)**	43103.50 (659.46)	24805.17 (1055.67)	42607.83 (655.03)	38531.50 (968.29)
executeTime[Delivery]	626909.67 (5168.60)	671233.83 (8211.02)	592083.50 (38094.56)	**703639.17 (10411.60)**	647984.83 (14020.60)
executeTime[Stock-Level]	817210.33 (19993.77)	1489075.00 (38751.16)	263105.50 (34888.50)	**1778001.33 (56253.15)**	1135030.83 (42857.91)
executeTime[New-Order]	4799814.83 (31768.85)	4461686.17 (35615.88)	**4814195.33 (151947.79)**	4339678.67 (51957.12)	4577858.00 (66970.16)
Rank	2.75	2.4375	4	2.27083	3.54167

Fonte: Produzida pelo autor com base nos resultados obtidos.

A Tabela 3 apresenta os resultados do teste de Nemenyi apresentando em quais situações, em geral, os resultados são significativamente diferentes. A primeira coluna (Cenários comparados) mostra os pares de cenários comparados, onde a segunda (Resultado) demonstra sua equiparação.

Tabela 3 - Comparação geral entre cenários baseado na comparação do teste de Nemenyi.

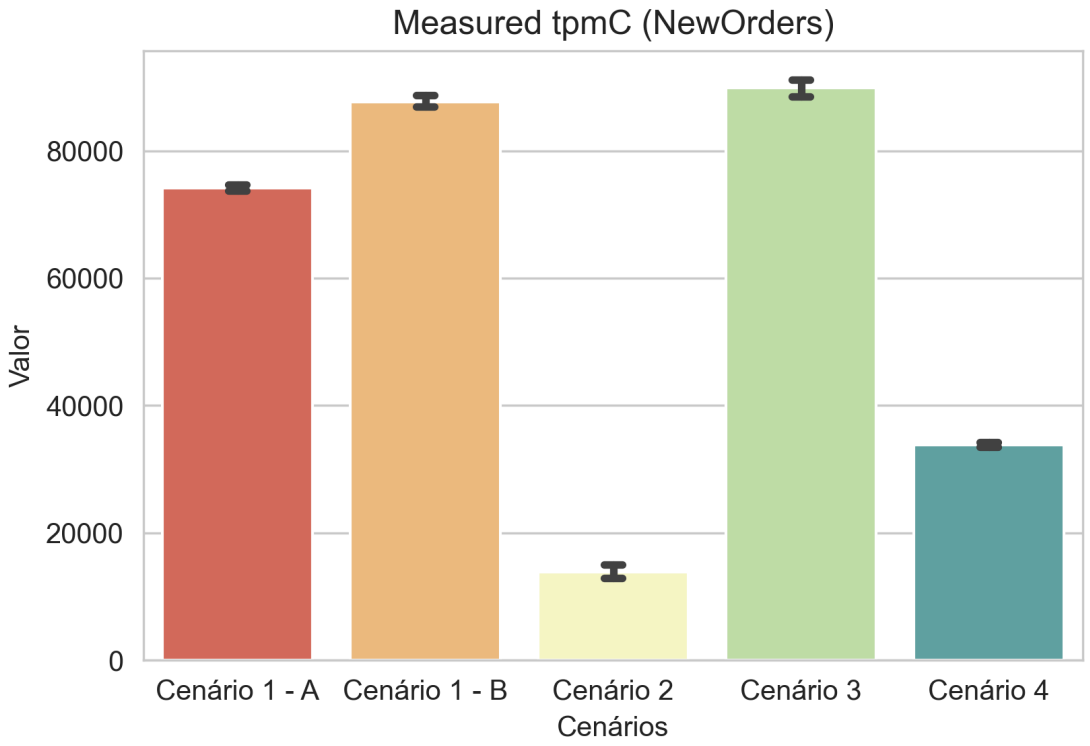
Cenários comparados	Resultado
Cenário 2 vs Cenário 3	Diferentes
Cenário 2 vs Cenário 1B	Diferentes
Cenário 3 vs Cenário 4	Diferentes
Cenário 2 vs Cenário 1A	Diferentes

Cenário 1B vs Cenário 4	Diferentes
Cenário 1A vs Cenário 4	Iguais
Cenário 3 vs Cenário 1A	Iguais
Cenário 2 vs Cenário 4	Iguais
Cenário 1B vs Cenário 1A	Iguais
Cenário 3 vs Cenário 1B	Iguais

Fonte: Produzida pelo autor com base nos resultados obtidos.

A principal métrica apresentada pelo BenchmarkSQL é o tpmC (New Orders), pois o mesmo indica quantos pedidos foram efetivamente concluídos. Seus resultados estão dispostos na Figura 16, que se trata de um gráfico Valor/Cenário.

Figura 16 - Tabela comparativa entre os resultados obtidos.



Fonte: Compilação do autor

A partir dos resultados obtidos é possível se chegar em diferentes conclusões, que são apresentadas e discutidas na seção a seguir.

4.2. Discussões

Através da Tabela - 2 é possível ver que o Cenário 3 apresenta melhores resultados médios em cinco métricas (Measured tpmC (NewOrders), Measured tpmTOTAL, Transaction Count, executeTime[Delivery] e executeTime[Stock-Level]), enquanto o Cenário 2 apresenta melhores resultados em duas métricas (executeTime[Payment] e executeTime[New-Order]) e o Cenário 1 apresenta melhores resultados considerando uma métrica (executeTime[Order-Status]).

Considerando a Tabela - 3, nota-se que no geral os cenários 1A, 1B e 3 apresentam resultados bem similares, indicando que a quantidade de threads ou a quantidade de memória podem não influenciar nos resultados, já que eles possuem diferentes valores para esses parâmetros. Enquanto que os cenários 2 e 4 apresentam resultados bem distintos dos demais, indicando que o clock, ou quantidade de RAM ou velocidade do SSD ou virtualização podem influenciar nos desempenhos gerais.

Analisando a Figura 16, é possível observar uma grande variação entre os cenários analisados considerando a métrica mais importante dos testes (Measured tpmC (NewOrders)). Ao analisar com mais atenção, observa-se que o Cenário 1A foi capaz de produzir 119% mais ordens concluídas do que o ambiente na nuvem (Cenário 4). Embora esse resultado seja indicativo de uma grande limitação de performance por um ambiente na nuvem, é necessário observar que ao se levar em consideração todos os resultados, os dois cenários se aproximam muito na média.

A análise primordial desse resultado tendenciosamente leva a validar o objetivo do trabalho, mas algumas considerações devem ser feitas sobre os resultados obtidos.

Dentre os parâmetros controlados dos cenários analisados (Tabela 1), a diferença de desempenho entre os cenários 1A e 4 (Figura 16) pode existir devido a

diferença nos valores de clock (definição de tempo de computador) entre os processadores e/ou devido a virtualização dos núcleos de cpu. Em ambos os casos a melhor performance do ambiente 1A aponta que o mesmo seja a melhor opção para se retirar o melhor desempenho possível de um servidor.

Sendo assim, a presente pesquisa aponta resultados indicativos de que a utilização de um servidor na nuvem pode trazer perda significativa de desempenho com relação a um servidor local. Seja devido a processadores locais serem capazes de atingir maiores valores de clock ou devido a virtualização causar lentidão e atraso na comunicação do sistema com o banco de dados.

Entretanto, ao se observar a média geral de todos os resultados é possível se observar que os dois se aproximam muito. Sendo que é possível se expandir o processador na nuvem de forma muito mais simplificada, devido a expansibilidade de um sistema virtual. Sendo então necessário que seja observado qual o tipo de atividade o servidor necessita realizar onde, em alguns casos, a diferença de performance é negligenciável.

Vale ressaltar que nem sempre a expansão no número de núcleos proporciona um aumento significativo no resultado final. Isso pode ser comprovado de duas formas. Uma é ao se relacionar o Cenário 2 com o Cenário 1B, onde o primeiro possui 50% maior número de núcleos que o segundo e a performance dos dois pode ser considerada dentro da margem de erro. Como também ao relacionar o cenário 1A e 1B, onde o segundo possui o dobro de núcleos, mas a performance geral não é significativamente maior. Isso indica que ao se atingir o patamar de performance necessário para se completar a tarefa existe uma dependência maior da capacidade individual de cada núcleo do que do número de núcleos disponíveis.

É necessário observar também que o Cenário 2 possui um desempenho consideravelmente inferior aos demais. Considerando as limitações do mesmo, o fator de performance mais significativo nesta situação é a utilização de um SSD sata em contrapartida a um SSD NVME. Portanto deve-se garantir que outras situações não estão sendo os fatores limitantes anteriormente a se ampliar a performance de CPU.

5. Conclusão

A utilização de servidores em nuvem e ambientes de tratamento de dados traz grandes benefícios para empresas, organizações e governos. Para garantir a contínua operação de seus sistemas é necessário uma infraestrutura que possibilite que não haja lentidão ou desperdício de performance para com o mesmo.

O presente trabalho permitiu a observação de um fragmento de uma área de bancos de dados onde existe pouca análise quando se busca investir em tecnologias. Foi utilizada a ferramenta BenchmarkSQL para executar experimentos e analisar o comportamento de diferentes sistemas com relação a configuração de hardware e encontrar em quais situações é possível se observar diferenças significativas de performance dentro de um ambiente MySQL.

Sendo assim é observável que existe uma correlação entre as parametrizações do sistema e o desempenho do sistema. No caso do MySQL os parâmetros que mais afetaram o resultado foram a virtualização dos núcleos para a métrica de maior importância, enquanto para o resultado geral o fator mais significativo é a velocidade do sistema de armazenamento.

Embora a premissa do trabalho se mostrou válida, também foi possível se observar que outros parâmetros podem ser bem mais influentes no resultado. Sendo necessário uma análise detalhada do ambiente que se deseja utilizar. Deve-se levar em consideração um número maior de informações para se possuir uma resposta concreta sobre qual sistema melhor soluciona o problema apresentado para a empresa.

Sendo assim, a melhor solução para um problema de desempenho em banco de dados deve levar em consideração não somente o desempenho bruto proporcionado por um sistema, mas também a necessidade do mesmo. Para maximizar o resultado de uma forma onde o custo benefício seja de extrema importância, torna-se relevante a análise dos fatores apresentados neste trabalho e a necessidade do sistema.

Por fim, sugere-se que em trabalhos futuros sejam apresentados novas métricas para serem analisadas, como por exemplo efetuar testes em diferentes

tipos de banco de dados, como PostgreSQL e MariaDB. É possível também realizar testes em uma quantidade maior de hardware criando-se mais cenários, como maior variação na quantidade de memória, maior controle sobre o desempenho do SSD e maior controle sobre os valores de clock da CPU.

Referências

Cardoso, Virgínia. Linguagem SQL, fundamentos e práticas – 1ª edição. Editora Saraiva, 2009

Crawford, Tyler e Hussain, Tauqeer. A Comparison of Server Side Scripting Technologies. Kutztown University of Pennsylvania, Kutztown, PA – U.S.A (2017).

DB-Engines, Knowledge Base of Relational and NoSQL Database Management Systems (2021) Disponível em: https://db-engines.com/en/ranking. Acesso em 22 de Junho de 2021

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO/IEC/IEEE 29119-2:2013 – Software and systems engineering – Software testing. (2013)

Java, O que é java?. Disponível em https://www.java.com/en/download/help/whatis_java.html Acesso em 23 de novembro de 2021

Kanji, Gopal K. 100 Statistical Tests - 3ª edição. Editora SAGE 2006

Machado, Felipe Nery Rodrigues. Banco de Dados – Projeto e Implementação – 4ª edição. Editora Saraiva/Erica, 2020

Maiello, Pollyana Espinosa (2016). Comparação dos sistemas gerenciadores de bancos de dados oracle e postgresql com o uso da ferramenta BenchmarkSQL Universidade de Araraquara (UNIARA)

Medeiros, Fernanda Oliveira de. *Estudo sobre benchmarks para avaliação de desempenhos orientados à nuvem.* IFMG - 2020

Mesquita, Johnson, Thienne De Melo E Silva, e COUTINHO, Mauro Margalho *Avaliação de Desempenho de Sistemas Computacionais*. Grupo GEN, 2011.

Metha, Apeksha. Top 15 Databases to Use in 2021 and Beyond. Appinventiv, 2021. Disponível em https://appinventiv.com/blog/top-web-app-database-list/ Acesso em 22 de Agosto de 2021

Oliveira, Saraiva, Maurício De, e Barreto, Jeanine dos Santos Desenvolvimento de sistemas com PHP. Grupo A, 2018.

PHP, Documentação – histórico. Disponível em https://www.php.net/manual/pt_BR/history.php.php Acesso em 08 de outubro de 2021

Rodríguez-Fdez, STAC: a web platform for the comparison of algorithms using statistical tests (2015) Disponível em: http://tec.citius.usc.es/stac/index.html. Acesso em 19 de novembro de 2021

Thiago Silva de Souza (2018) – Testes de Desempenho de Software: Teoria e Prática – V Escola Regional de Sistemas de Informação do Rio de Janeiro, SBC, 1ª Ed.

TPC, Documentation & About. Disponível em http://tpc.org/ Acesso em 10 de Novembro de 2021

Ubuntu, *What is Ubuntu?* Disponível em https://help.ubuntu.com/lts/installation-guide/s390x/ch01s01.html Acesso em 23, novembro de 2021

Zhiguo Luo, (2009) Cloud Computing: An Overview. Primeira conferência internacional de computação na nuvem, CloudCom, Beijing, China

Apêndices

Resultados Absolutos

Ex	Métrica	Cenário 1A	Cenário 1B	Cenário 2	Cenário 3	Cenário 4
1	**Measured tpmC (NewOrders)**	**73717.22**	**89937.30**	**16211.92**	91930.12	33016.46
1	Measured tpmTOTAL	163921.22	199780.20	36028.49	204147.81	73402.51
1	Transaction Count	1639244.00	1997841.00	360299.00	2041552.00	734056.00
1	executeTime[Payment]	3324288.00	2910834.00	4209296.00	2711533.00	3281873.00
1	executeTime[Order-Status]	45750.00	44395.00	23836.00	43062.00	36695.00
1	executeTime[Delivery]	623708.00	687355.00	532723.00	714635.00	622457.00
1	executeTime[Stock-Level]	823658.00	1426097.00	326080.00	1743663.00	1202336.00
1	executeTime[New-Order]	4781348.00	4529948.00	4506366.00	4385129.00	4455432.00
2	**Measured tpmC (NewOrders)**	74397.79	87454.78	14647.34	88936.95	33908.72

2	Measured tpmTOTAL	165116.12	194549.64	32563.43	197926.84	75386.81
2	Transaction Count	1651207.00	1945596.00	325857.00	1979373.00	753891.00
2	executeTime[Payment]	3289544.00	2940721.00	3845967.00	2736027.00	3184678.00
2	executeTime[Order-Status]	46104.00	42929.00	26237.00	42387.00	38685.00
2	executeTime[Delivery]	633978.00	671432.00	586126.00	700696.00	646949.00
2	executeTime[Stock-Level]	796003.00	1500202.00	274737.00	1814137.00	1148646.00
2	executeTime[New-Order]	4832934.00	4443577.00	4867412.00	4304808.00	4579899.00
3	**Measured tpmC (NewOrders)**	74305.74	87663.36	14257.45	90001.84	33846.19
3	Measured tpmTOTAL	165110.77	195053.07	31693.54	200116.56	75319.30
3	Transaction Count	1651159.00	1950624.00	317160.00	2001268.00	753241.00
3	executeTime[Payment]	3298619.00	2924461.00	3821387.00	2738447.00	3164352.00
3	executeTime[Order-Status]	46524.00	42929.00	25112.00	42683.00	39507.00
3	executeTime[Delivery]	626866.00	669782.00	630440.00	699988.00	658060.00
3	executeTime[Stock-Level]	799212.00	1523594.00	264309.00	1765743.00	1116832.00
3	executeTime[New-Order]	4827577.00	4438032.00	4859375.00	4351226.00	4619939.00
4	**Measured tpmC (NewOrders)**	73255.51	87292.79	13024.09	89818.90	34191.10
4	Measured tpmTOTAL	162886.42	194047.58	28893.76	199607.85	75939.08
4	Transaction Count	1628923.00	1940504.00	289345.00	1996164.00	759443.00
4	executeTime[Payment]	3335076.00	2962446.00	3839959.00	2748312.00	3168086.00
4	executeTime[Order-Status]	45280.00	42474.00	24061.00	42514.00	39100.00
4	executeTime[Delivery]	621846.00	666892.00	588570.00	707092.00	662162.00
4	executeTime[Stock-Level]	848149.00	1485008.00	242421.00	1742615.00	1104502.00
4	executeTime[New-Order]	4748467.00	4441843.00	4910211.00	4357020.00	4625070.00
5	**Measured tpmC (NewOrders)**	74962.92	86157.32	12662.62	91563.73	34487.86
5	Measured tpmTOTAL	166705.30	191588.21	28129.37	203248.87	76541.51
5	Transaction Count	1667116.00	1915945.00	281470.00	2032569.00	765446.00
5	executeTime[Payment]	3299295.00	2977531.00	3838492.00	2727258.00	3193381.00
5	executeTime[Order-Status]	46306.00	42935.00	23809.00	43463.00	38661.00

5	executeTime[Delivery]	632375.00	665331.00	636892.00	713060.00	652674.00
5	executeTime[Stock-Level]	807522.00	1468301.00	239802.00	1727156.00	1082279.00
5	executeTime[New-Order]	4813258.00	4444585.00	4862022.00	4387038.00	4631807.00
6	**Measured tpmC (NewOrders)**	74375.63	87342.59	12500.42	87231.03	33597.41
6	Measured tpmTOTAL	165249.47	196143.40	27808.80	193923.75	74759.21
6	Transaction Count	1652524.00	1950534.00	278398.00	1939298.00	747651.00
6	executeTime[Payment]	3305907.00	2973616.00	3886027.00	2742502.00	3204404.00
6	executeTime[Order-Status]	46137.00	42959.00	25776.00	41538.00	38541.00
6	executeTime[Delivery]	622685.00	666611.00	577750.00	686364.00	645607.00
6	executeTime[Stock-Level]	828718.00	1531248.00	231284.00	1874694.00	1155590.00
6	executeTime[New-Order]	4795305.00	4472132.00	4879786.00	4252851.00	4555001.00