Uma Avaliação de Desempenho de Contêineres Docker Executando Diferentes SGBDs Relacionais

Brena Santos[∓], Patricia Takako Endo[⋄] e Francisco Airton Silva[∓]

[1] ∓Universidade Federal do Piauí (UFPI), Campus Picos
⋄Universidade de Pernambuco (UPE)
E-mail: brenamaia12@gmail.com, patricia.endo@upe.br, faps@ufpi.edu.br

Abstract. *The concept of computational virtualization, though old, has become a commonplace in enabling applications that share computing resources in the cloud infrastructure. Virtual machines (VMs) have brought benefits such as the ability to perform live migration to provide fault tolerance and load balancing. However, despite the advantages, VMs add extra layers of abstraction, resulting in loss of efficiency. As an alternative to VMs, today's containers present themselves as a lighter, better-performing solution. This work performs a performance evaluation based on sensitivity analysis with the objective of identifying the factors that most impact on the efficiency of a Database Management System (DBMS) using Docker as a container.*

Resumo. *O conceito de virtualização computacional, apesar de antigo, se tornou lugar-comum na viabilização de aplicações que compartilham recursos computacionais na infraestrutura da nuvem. As máquinas virtuais (do inglês virtual machines, VMs) trouxeram benefícios como a possibilidade de realizar live migration para prover tolerância a falha e balanceamento de carga. Porém, apesar das vantagens, as VMs adicionam camadas extras de abstração, resultando em perda de eficiência. Como alternativa as VMs, atualmente os contêineres se apresentam como uma solução mais leve e com bom desempenho. Este trabalho realiza uma avaliação de desempenho baseada em análise de sensibilidade com o objetivo de identificar os fatores que mais impactam na eficiência de um Sistema de Gerenciamento de Banco de Dados (SGBD) utilizando Docker como contêiner.*

1. Introdução

A virtualização é uma tecnologia estabelecida e em expansão. Muitos aplicativos, sistemas de gerenciamento, modelos e simuladores foram criados para promover e melhorar seu desempenho. Algumas das principais vantagens da virtualização incluem flexibilidade, capacidade, poder de processamento, crescimento da demanda e eficiência energética. No entanto, recentemente esse paradigma vem mudando. Um novo modelo de virtualização a nível do sistema operacional que permite instâncias com vários espaços de usuário adquiriu apoio de grandes empresas. Esse modelo de virtualização é chamado de contêiner. Esta tecnologia é semelhante a Máquinas Virtuais (VMs), porém os contêineres não exigem uma camada de emulação para serem executados, tornando seu funcionamento computacionalmente mais leve que as VMs [Dua et al. 2014]. O Docker é uma plataforma para contêineres das mais populares, que fornece uma maneira automatizada de hospedar aplicações.

O Docker permite a criação de contêineres em menos de um segundo, eliminando a sobrecarga bem conhecida da inicialização de *hypervisors* [Fink 2014]. Esta plataforma vem sendo bastante utilizada para atender um novo paradigma arquitetural que é a Internet das Coisas (Internet of Things - IoT) [Amento et al. 2018, Hsieh et al. 2018, Morabito 2017, Niu et al. 2017]. Sabemos que alguns trabalhos na área de IoT utilizam bancos de dados para armazenar informações, como [Jiang et al. 2014] que propõe uma estrutura de armazenamento de dados para IoT e [Gupta et al. 2016] que coletam dados de sensores e os enviam para um servidor de banco de dados para que sejam analisados e mantidos de forma estática. Sendo assim, é importante que sejam realizadas avaliações de desempenho dos contêineres junto a bancos de dados, como um modo de auxiliar na melhoria da qualidade de serviço de aplicações. Alguns trabalhos analisaram contêineres Docker com banco de dados relacionais, tais como [Velásquez et al. 2018, Xavier et al. 2016], que se concentraram na avaliação do desempenho da Rede e Disk I/O, respectivamente. No entanto, esses trabalhos não realizam uma análise de sensibilidade aprofundada, tornando necessária uma investigação que utilize diversos fatores e níveis de forma conjunta, pois dessa forma é possível extrair informações mais precisas.

Este trabalho realiza uma análise sobre o uso de contêineres Docker executando banco de dados relacionais. Para tanto, foi realizada uma avaliação de desempenho baseada em análise de sensibilidade sobre o contêiner Docker em comparação com diferentes bancos de dados. Foram considerados os seguintes fatores: SGBD utilizado, número de tabelas e número de tuplas. E como variáveis dependentes, foram adotadas as seguintes: Tempo de Execução para processamento dos dados e Nível de Falha de requisições. Assim, este trabalho tem como principal contribuição o provimento de indicativos de qual banco de dados relacional deve ser adotado junto ao Docker visando um melhor desempenho.

O restante deste trabalho está dividido da seguinte maneira: a Seção 2 discute os principais trabalhos relacionados a esta pesquisa; na Seção 3 apresentamos a metodologia de *Design of Experiments*, utilizada neste trabalho; a Seção 4 descreve os experimentos realizados e discute os resultados obtidos; por fim, na Seção 5 encontra-se a conclusão e os planos para trabalhos futuros.

2. Trabalhos Relacionados

Esta seção apresenta alguns trabalhos relacionados ao presente estudo, ou seja, avaliaram o desempenho do contêiner Docker em conjunto com algum SGBD.

Os autores em [Morabito 2016] avaliaram o uso de contêineres Docker em ambientes restritos. Os autores avaliaram por meio de diferentes ferramentas de benchmark, o desempenho do Docker quando executado em um computador de placa única, no caso o Raspberry Pi (RPi2). Os resultados mostram um impacto quase insignificante da camada de virtualização de contêiner em termos de desempenho, se comparado à execução nativa.

Os autores em [Barik et al. 2016] avaliaram o desempenho de VMs e contêineres simultaneamente em computadores tradicionais. Os autores mostram que contêineres têm algumas deficiências nas áreas de segurança e isolamento, pois proporciona um isolamento de segurança mais baixo do que a virtualização.

Os autores em [Velásquez et al. 2018] apresentam uma análise dos diferentes sis-

temas de banco de dados para armazenar as informações coletadas por uma rede de sensores sem fio. Diferentes testes são apresentados para avaliar SGBDs relacionais (MySQL, PostgreSQL) e No-SQL (MongoDB, Couch, Neo4J). Os SGBDs foram implantados em contêineres Docker. O objetivo do trabalho foi fornecer informações necessárias para a correta escolha da base de dados para esse tipo de sistema. Os resultados indicaram que o melhor SGBD analisado para a ingestão de dados por segundo é o PostgreSQL, mas, para a quantidade de armazenamento total, o MongoDB obteve melhor desempenho.

Os autores em [Shirinbab et al. 2017] apresentam uma extensa comparação de desempenho entre o contêiner VMware e o Docker, enquanto executa o Apache Cassandra. O Apache Cassandra lidera o ranking de SGBD distribuído NoSQL quando se trata de plataformas de Big Data. Como linha de base para comparações, os autores avaliaram o Cassandra focando na infraestrutura física. O estudo mostrou que o Docker tinha uma sobrecarga menor em comparação com o VMware ao executar o Cassandra. O desempenho do Cassandra na infraestrutura com Docker era tão bom quanto no ambiente sem virtualização alguma.

Todos os trabalhos acima destacados utilizaram a ferramenta Docker como hospedeiro para executar alguma instância de SGBD. Pôde-se verificar várias conclusões comparativas entre bancos de dados relacionais e até não relacionais. No entanto, nenhum dos trabalhos observados realizaram uma análise de sensibilidade sobre fatores que interferem no desempenho do Docker junto ao SGBD. A Tabela 1 resume as comparações.

Tabela 1. Trabalhos Relacionados

Trabalho Relacionado	Métricas	SGBD	Hardware	Análise de Sensibilidade
[Morabito 2016]	CPU, Memória, Tipo de Transação	MySQL	Raspberry	Não
[Barik et al. 2016]	Segurança	MySQL	Computador Desktop	Não
[Shirinbab et al. 2017]	CPU, Tipo de Transação	Apache cassandra	Computador Desktop	Não
[Velásquez et al. 2018]	Segurança	MySQL, PostgreSQL, MongoDB, Couch, Neo4J	Computador Desktop	Não
O presente trabalho	Tempo de Execução, Falhas	MySQL, PostgreSQL, SQLServer	Computador Desktop	Sim

3. Metodologia *Design of Experiments*

O Desing of Experiments (DoE -Projeto de Experimentos) é uma metodologia estatística que foi aplicada no presente artigo. O DoE foi amplamente utilizado com sucesso em vários campos, tais como otimização [Gunst 1996], química [Lazic 2006] e engenharia de software [Kuhn and Reilly 2002]. Existe uma quantidade considerável de livros sobre tal metodologia [Siebertz et al. 2017, Seltman 2012, Antony 2006]; Portanto, a metodologia geral aqui será descrita brevemente. Em cada parte da aplicação, será exemplificado cada caso metodológico para o estudo de caso.

O conceito geral de DoE foi criado por uma série de experimentos reais simulados em um sistema ou modelo de sistema sob observação. Em cada experimento, um ou

vários parâmetros de projeto são alternados e o impacto no comportamento do sistema é avaliado. Quais parâmetros são alterados e como eles são alterados é definido usando um plano de experimento. O objetivo é obter o máximo de informações possíveis usando a menor quantidade de experimentos. O comportamento do sistema com base nas mudanças de parâmetros é observado usando conjuntos de saídas. No contexto do DoE, as saídas podem ser referenciadas como "indicadores de performance", os parâmetros de projeto como "fatores" e os valores das configurações de "níveis".

Para analisar o impacto de cada fator no sistema e a interação com os outros fatores, várias combinações de fatores precisam ser testadas e, portanto, exige um experimento. Devido à grande quantidade de combinações possíveis, este é um esforço muitas vezes inviável em termos de tempo e custos. O DoE oferece uma coleção de métodos - referidos como planos experimentais ou tabelas de design - para reduzir a quantidade de experimentos necessários para encontrar informações precisas com o menor número de experimentos possível. O tipo de plano usado depende do objetivo do experimento. Encontrar a tabela de projeto ideal, por exemplo, a menor quantidade de experimentos necessários para descrever o comportamento dos sistemas corretamente, tem sido objeto de intensa pesquisa. Usando um conjunto de métodos estatísticos de avaliação nos dados resultantes do experimento, o impacto, efeitos e interações dos fatores em relação aos indicadores de performance escolhidos é avaliado. Os resultados do experimento podem ser usados para formular um modelo substituto matemático, também chamado de metamodelo em disciplinas de engenharia [Miller et al. 2014, Thomas et al. 2014]. Neste trabalho foi aplicada a metodologia DoE para avaliação da arquitetura proposta. Três tipos de gráficos são adotados usualmente em estudos com DoE: Gráfico de Pareto, Gráfico de Efeitos Principais e Gráfico de Interação [Ruiz Espejo 2006].

3.1. Gráfico de Pareto

O gráfico de Pareto permite detectar qual o efeito da interação de fatores é mais importante para o processo ou estudo de otimização de projeto com o qual se deve lidar. Exibe os valores absolutos dos efeitos e desenha uma linha de referência no gráfico. Qualquer efeito que ultrapasse essa linha de referência é potencialmente importante. Um gráfico de Pareto é construído como na Figura 1, por exemplo. O gráfico mostra que os fatores B (tool geometry) e C (cutting angle) e a interação AC possui maior impacto.

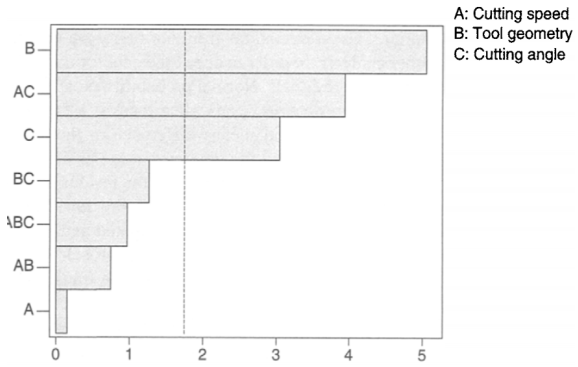

Figura 1. Gráfico de Pareto com Efeitos Padronizados

3.2. Gráfico de Efeitos Principais

O "Gráfico de Efeitos Principais" é o gráfico dos valores médios de resposta em cada nível de um parâmetro de projeto ou variável de processo. Pode-se usar esse gráfico para comparar a força relativa dos efeitos de vários fatores. O sinal e a magnitude de um efeito principal nos diriam o seguinte:

- O sinal de um efeito principal nos diz a direção do efeito, ou seja, se o valor médio da resposta aumenta ou diminui.
- A magnitude nos diz a força do efeito.

Se o efeito do parâmetro de projeto ou processo é positivo, isso implica que a resposta média é maior em nível alto do que em nível baixo da configuração do parâmetro. Em contraste, se o efeito for negativo, isso significa que a resposta média em nível baixo de configuração do parâmetro é maior que em nível alto. A Figura 2 ilustra o efeito principal da temperatura na resistência à tração de uma amostra de aço. Como pode ser visto na Figura, a resistência aumenta quando as configurações da temperatura variam para o nível baixo (ou seja, 1 para 1). O efeito de um parâmetro de projeto ou processo (ou fator) pode ser calculado matematicamente usando a simples Equação 1:

$$E_f = \overline{F}_{(+1)} - \overline{F}_{(-1)} \tag{1}$$

Quando $\overline{F}_{(+1)}$ significa a resposta média na configuração de alto nível de um fator, e $\overline{F}_{(-1)}$ significa a resposta média na configuração de baixo nível de um fator.

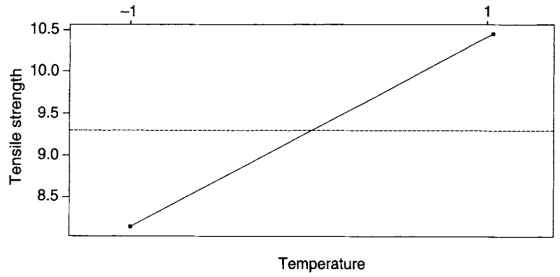

Figura 2. Gráfico de Efeitos Principais da Temperatura na Resistência à Tração

3.3. Gráfico de Interação

O processo de "Interações", identifica efeitos importantes e determina sua magnitude, logo, as interações entre os efeitos são cruciais. As interações ocorrem quando o efeito de um fator depende do nível de outro fator. Uma medida de design sempre aborda vários fatores. Entender como esses fatores interagem em que magnitude permite para escolher a melhor combinação de medidas, revelando combinações de fatores com efeito cumulativo ou degradante. A interação entre os fatores A e B pode ser calculada usando a Equação 2.

$$I_{A,B} = \frac{1}{2}(E_{A,B(+1)} - E_{A,B(-1)}) \tag{2}$$

O $E_{A,B(+1)}$ é o efeito do fator 'A' no nível alto do fator 'B' e $E_{A,B(-1)}$ é o efeito do fator 'A' no nível baixo do fator 'B'.

Para determinar se dois parâmetros de processo estão interagindo ou não, pode-se usar uma ferramenta gráfica simples, porém poderosa, chamada de gráficos de interação. Se as linhas no gráfico de interação forem paralelas, não haverá interação entre os parâmetros do processo. Isso implica que a mudança na resposta média do fator 'A' não depende dos níveis do fator 'B'. Por outro lado, se as linhas não são paralelas, existe uma interação entre os fatores. Quanto maior o grau de afastamento de ser paralelo, mais forte o efeito de interação. Para interação sinérgica, as linhas no gráfico não se cruzam. Por exemplo, a Figura 3(a) é um exemplo de interação sinérgica.

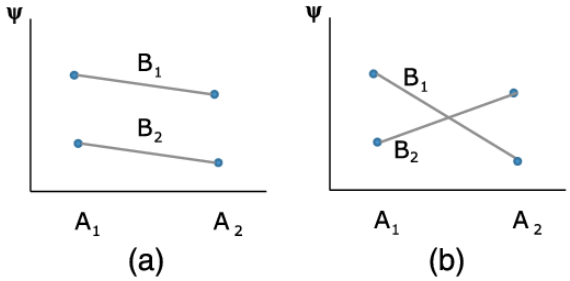

Figura 3. Gráfico de Interações Sinérgica e Antagônica

Na interação antagônica, as linhas no gráfico se cruzam. Isso pode ser ilustrado na Figura 3(b). Neste caso, a mudança na resposta média para o fator A no nível 'A_1' é alta em comparação ao nível 'A_2'. As mudanças nos níveis do fator 'A' para a resposta média, indica uma dependência do fator 'A' em relação aos níveis do fator 'B'.

4. Análise de Sensibilidade do Contêiner Docker e SGBDs Relacionais

Esta seção descreve os experimentos realizados para analisar os fatores com maior influência no desempenho de contêineres com SGBDs relacionais, baseado em duas métricas: tempo de execução e nível de falha na requisição.

4.1. Desenho Experimental

O ambiente de experimentos foi composto por dois computadores conectados à mesma rede: o Computador A foi utilizado para hospedar o contêiner Docker, enquanto o Computador B foi utilizado para enviar as requisições de acesso aos dados presentes no contêiner. A Tabela 2 apresenta as configurações das respectivas máquinas.

Tabela 2. Configurações dos Computadores A e B

	Processador	Memória	Disco Rígido	Sistema Operacional
Computador A	Intel i5	8 GB	1 TB	Linux Ubuntu 18.04 LTS
Computador B	Intel i3	4 GB	500 GB	Linux Ubuntu 16.04 LTS

Para realizar a avaliação de desempenho, duas métricas foram definidas: tempo de execução e nível de falha na requisição. Para a avaliação da métrica de tempo de execução, foi realizado o *Design of Experiments* (DoE), com três fatores estabelecidos: SGBD, número de tabelas e número de tuplas da tabela. O SGBD possui três níveis, que foram escolhidos por serem alguns dos mais populares: MySQL (v. 8.0.12), PostgreSQL (v. 11.0) e SQL Server (v. 2017). Vale ressaltar que as configurações padrão dos SGBDs

foram mantidas, e a consulta realizada buscava os dados de todas as colunas das tabelas (ID e NOME). O fator número de tabelas se refere à quantidade de tabelas que serão consultadas durante o experimento e possui dois níveis: 1 e 2. Já o fator número de tuplas é referente à quantidade de tuplas que cada tabela consultada possui e foram determinados três níveis: 1000, 3000 e 5000. A escolha do número de tuplas e tabelas foi baseada em sucessivos testes com níveis de fatores mínimos para garantir um controle sobre o test-bed. Nos experimentos em que são utilizadas duas tabelas, foi realizada a operação de junção, e o número de tuplas das mesmas devem ser iguais. A Tabela 3 apresenta os fatores e níveis escolhidos para a realização do DoE utilizando a métrica de tempo de execução.

Tabela 3. Fatores e níveis do DoE para a métrica tempo de execução

	Nível 1	Nível 2	Nível 3
SGBD	MySQL	PostgreSQL	SQL Server
Número de Tabelas	1	2	-
Número de Tuplas	1000	3000	5000

Com todos os fatores e níveis definidos, deve-se combinar os fatores e níveis para definir como os experimentos deverão ser realizados. A Tabela 4 apresenta as combinações possíveis entre os fatores e níveis.

Tabela 4. Combinações de fatores e níveis para a métrica tempo de execução

Nº	SGBD	Número de Tabelas	Número de Tuplas
1	MySQL	1	1000
2	MySQL	1	3000
3	MySQL	1	5000
4	MySQL	2	1000
5	MySQL	2	3000
6	MySQL	2	5000
7	PostgreSQL	1	1000
8	PostgreSQL	1	3000
9	PostgreSQL	1	5000
10	PostgreSQL	2	1000
11	PostgreSQL	2	3000
12	PostgreSQL	2	5000
13	SQL Server	1	1000
14	SQL Server	1	3000
15	SQL Server	1	5000
16	SQL Server	2	1000
17	SQL Server	2	3000
18	SQL Server	2	5000

Uma outra avaliação de desempenho foi realizada para a análise do nível de falha na requisição, com o objetivo de identificar qual é o banco de dados mais eficaz, ou seja, qual deles possui o menor nível de falha. Para esse experimento foram utilizados os mesmos SGBDs do DoE para a métrica tempo de execução: MySQL, PostgreSQL

e SQL Server. Porém, o número de tuplas e número de tabelas foram fixados em 5000 tuplas e uma única tabela, pois de acordo com sucessivos testes com níveis de fatores mínimos realizados para essa avaliação, foi constatado que a alteração desses fatores não influenciam na métrica em questão.

Na métrica tempo de execução é analisada a média do tempo total (em milissegundos) de duração do experimento. Para a análise do nível de falha na requisição é observada a média da porcentagem de erro encontrada durante o experimento, ou seja, a porcentagem de solicitações que não foram atendidas. Os valores para ambas as métricas foram coletados de acordo com os dados disponibilizado pela ferramenta de teste de carga utilizada, o JMeter [1]. Escolhemos essa ferramenta, pois a mesma foi utilizada em alguns trabalhos relacionados a este, como [Gillan et al. 2018] e [Gaur et al. 2017].

Para o cálculo do tempo de execução, configurou-se o Computador B com o JMeter, para que a mesma enviasse requisições de leitura para o SGBD presente no Docker, instalado no Computador A. Para tanto, o JMeter foi configurado para gerar requisições simultâneas de leitura emulando um grupo de 100 usuários. Já na avaliação da métrica de nível de falha na requisição, o grupo de usuários sofreu variações e os teste foram realizado com 500, 1000, 1500, 2000 e 2500 threads simultâneas. Em ambas as avaliações, foram realizadas 100 réplicas de experimento para cada combinação.

4.2. Resultados Obtidos

Esta seção apresenta os resultados obtidos de acordo com os dados coletados no ambiente de testes. A Figura 4 apresenta o gráfico de Pareto para os fatores relacionados a métrica tempo de execução. Quando um fator possui alto impacto nos testes, significa que ao alterar o nível dele são obtidos valores bem distintos. De acordo com os valores do p-value encontrados, os efeitos do fator SGBD possui maior relevância dentre os fatores deste estudo, portanto a escolha do SGBD a ser utilizado é determinante na eficiência do contêiner, com relação ao tempo de execução.

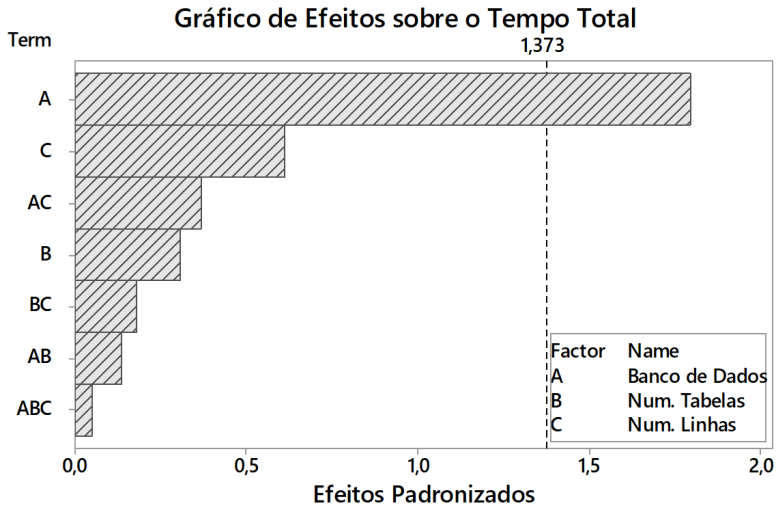

Figura 4. Influência dos fatores na métrica tempo de execução

[1] https://jmeter.apache.org/

A Figura 5 apresenta o gráfico de efeitos principais para a métrica tempo de execução. O gráfico representa o tempo médio para a realização dos testes de cada nível para cada um dos fatores definidos. Nesse gráfico, quanto mais horizontal for a linha, menos influência tem aquele fator, pois significa que os diferentes níveis do fator influenciam no resultado final de forma semelhante.

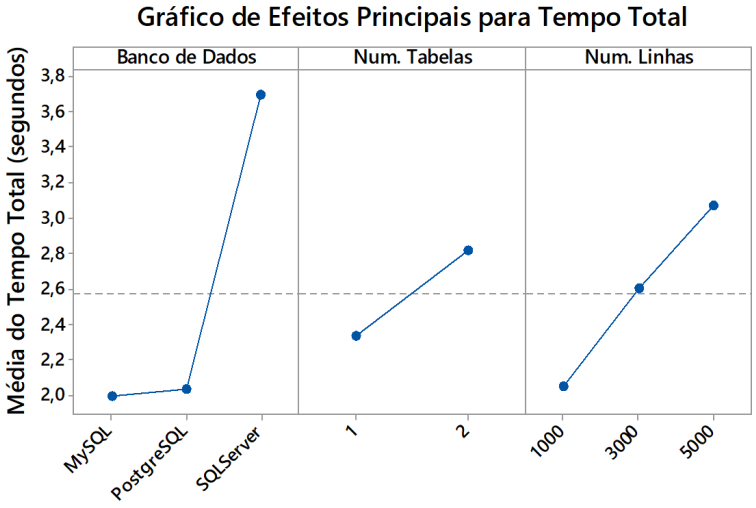

Figura 5. Efeitos principais para Tempo de Execução

Nesse estudo, através do gráfico de efeitos principais, pode-se concluir que todos os níveis dos fatores definidos interferem na métrica tempo de execução. O SGBD e o número de tuplas são os fatores que possuem maior efeito. Sobre o fator SGBD, o MySQL atingiu a melhor média de tempo de execução (1,993ms), com o PostgreSQL bem próximo (2,031ms), enquanto o SQL Server alcançou uma média de tempo muito maior que ambos (3,692ms). Sendo assim, pode-se dizer que o SGBD mais eficiente foi o MySQL, visto que alcançou a menor média de tempo de execução de leitura.

Com relação ao fator número de tabelas, percebe-se que o fator influencia no tempo médio total de acordo com a sua quantidade. Ao utilizar duas tabelas, o número de dados manipulados é dobrado, o que aumenta o tempo utilizado durante os testes. Por fim, o fator número de tuplas influencia de forma que à medida que a quantidade de tuplas aumenta, o tempo médio de leitura também aumenta. Quanto maior o número de tuplas, mais dados são manipulados, tornando necessário um tempo maior para a realização dos experimentos.

A Figura 6 exibe o gráfico de interações para cada combinação possível entre os fatores. Com esse gráfico, é possível analisar quais níveis possuem maior interferência no resultado final dos experimentos. Na primeira interação nota-se que na relação SGBD e tempo de execução, o número de tabelas influenciou no resultado. Percebe-se que ao manipular duas tabelas é obtido um tempo médio maior que o tempo de manipulação para apenas uma tabela, independente do SGBD. Da mesma forma, utilizando apenas uma tabela, apesar do desempenho ser semelhante ao caso anterior, o tempo médio total é inferior. Na segunda interação, percebe-se que o Número de Tuplas interfere no resultado da relação entre o SGBD e o Tempo de Execução. É possível constatar que o Tempo de Execução aumenta de acordo com o Número de Tuplas, independente do SGBD utilizado.

No entanto, a influência do Número de Tuplas que o SQL Server sofreu foi inferior aos demais. Na última interação é analisada a relação entre o Número de Tabelas e o Número de Tuplas. Conclui-se que a influência do Número de Tabelas no Tempo de Execução cresce de acordo com o aumento do Número de Tuplas. Tal resultado acontece pois quanto maior o Número de Tuplas nas tabelas, mais dados serão manipulados, o que exige maior processamento e consequentemente maior quantidade de tempo.

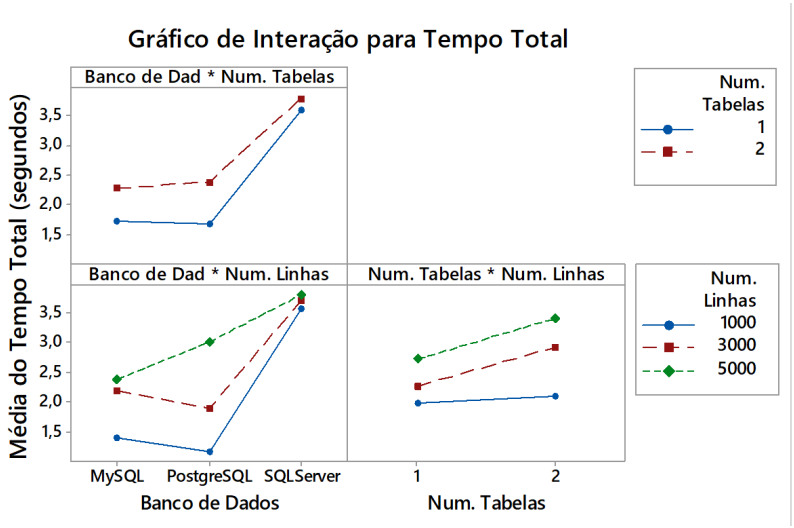

Figura 6. Interações dos fatores para Tempo de Execução

A Figura 7 apresenta o gráfico de otimização para a métrica tempo de execução. Esse gráfico mostra o efeito de cada fator na resposta do experimento. Os números destacados em vermelho que são exibidos no topo do gráfico e as linhas vermelhas verticais representam as configurações atuais do fator, ou seja, os níveis que estão sendo analisados. Os números e linhas azuis na horizontal representam o resultado para o nível do fator atual. Analisando o gráfico é perceptível que para obter o menor tempo possível (1,150ms), a melhor configuração para os fatores é o Banco de Dados do SGBD PostgreSQL, utilizando uma única tabela de 1000 tuplas.

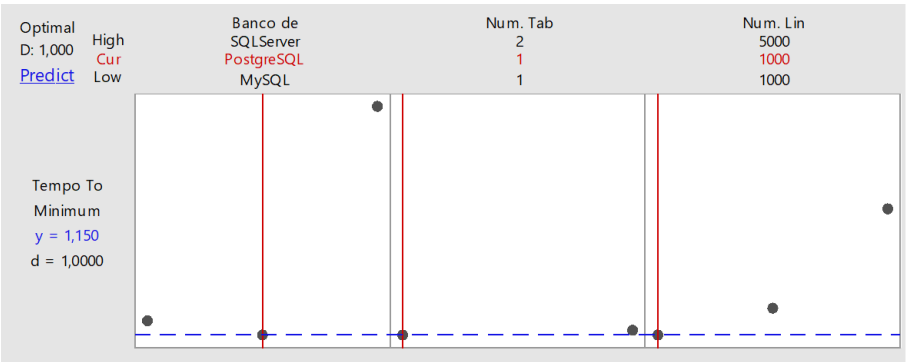

Figura 7. Gráfico de otimização para tempo de execução

A Figura 8 apresenta o gráfico de barras que exibe a porcentagem do nível de falha na requisição de cada SGBD, de acordo com a quantidade de threads simultâneas.

Analisando o gráfico, percebe-se que o SQL Server obteve a menor porcentagem de falha. Já o MySQL e o PostgreSQL obtiveram porcentagem de falha semelhantes entre eles, porém muito superior ao SQL Server. Acreditamos que a justificativa para esse resultado seja o fato de que por configuração padrão, o número máximo de conexões do MySQL e PostgreSQL é de 100 usuários simultâneos, enquanto o SQL Server esse número é dinâmico, podendo chegar a 32.767.

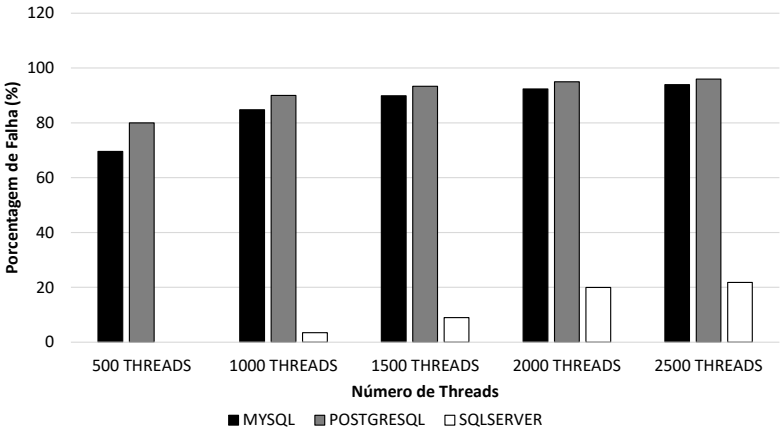

Figura 8. Porcentagem do nível de falha na requisição

A Figura 9 exibe o gráfico de barras que representa o tempo total dos testes para nível de falha na requisição, os valores para essa métrica foram coletados considerando o tempo médio das requisições bem sucedidas. Os SGBDs MySQL e PostgreSQL alcançaram os menores tempos de testes, e assim como na porcentagem do nível de falha na requisição, os resultados foram próximos um do outro. Nota-se também que o SQL Server utilizou uma quantidade de tempo muito superior aos demais, mas após observar as Figuras 8 e 9, percebe-se que apesar disso foi o SGBD que proporcionou menor número de falhas. O MySQL e o PostgreSQL mesmo com o bom desempenho relacionado ao tempo total dos testes, apresentaram grande porcentagem de falhas.

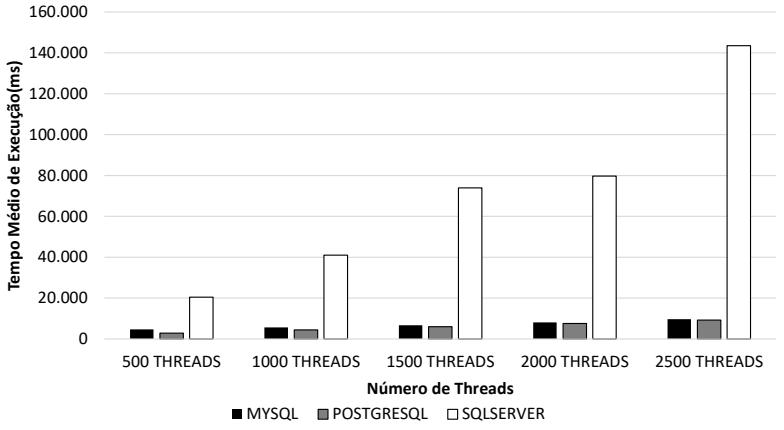

Figura 9. Tempo dos testes para nível de falha na requisição

A Figura 10 mostra as combinações que obtiveram melhor desempenho para as

métricas analisadas: tempo de execução e nível de falha na requisição. As combinações foram feitas com os melhores níveis de cada fator, de acordo com as métricas utilizadas.

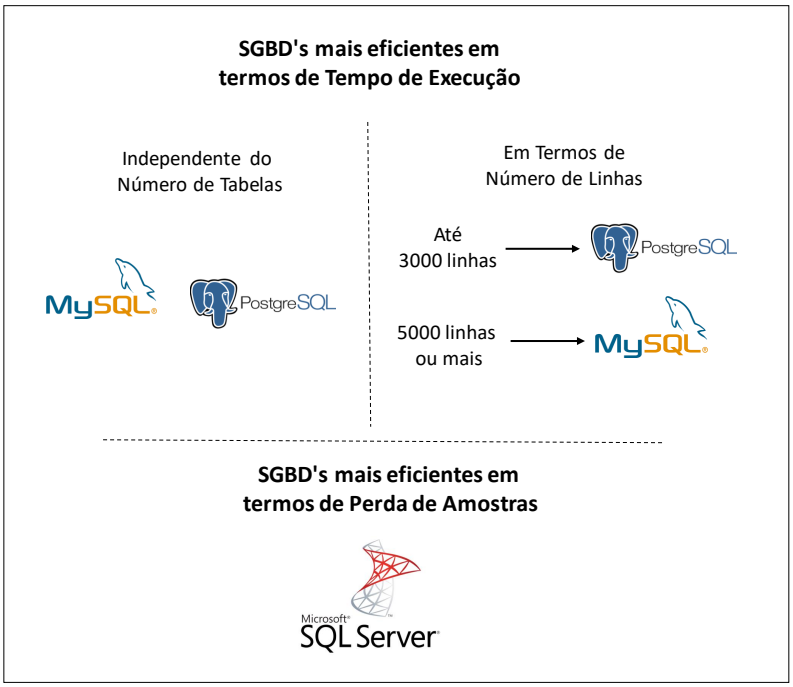

Figura 10. Melhores combinações obtidas

Sabendo quais são as melhores combinações, foram definidos dois cenários com a finalidade de identificar qual combinação obtém melhor desempenho, de acordo com as necessidades de cada um:

- **Cenário 1:** uma aplicação de pequeno porte precisa acessar o Banco de Dados para selecionar usuários cadastrados. Nesse caso, para obter o melhor desempenho é importante que o tempo utilizado seja o menor possível. Sendo assim, é recomendado o PostgreSQL, pois é mais eficiente com pequeno Número de Tuplas na tabela.
- **Cenário 2:** uma aplicação de porte maior e com um número de usuários superior que o cenário 1 precisa acessar o Banco de Dados para selecionar usuários cadastrados. Nesse caso, o mais importante é garantir a consistência dos dados, evitando falha na requisição dos mesmos durante a consulta. Assim, é recomendada a utilização do SQL Server, pois independente de levar mais tempo para realizar a tarefa, é o SGBD que possui menor nível de falha na requisição.

5. Conclusão

Neste trabalho foi realizada uma avaliação de desempenho sobre o contêiner docker. Esta pesquisa teve como objetivo principal identificar a melhor configuração para a utilização de um banco de dados instalado no contêiner docker. Para isso, foram realizadas avaliações de desempenho para as métricas: tempo de execução e nível de falha na requisição. Nas avaliações executadas foram observados alguns fatores, são eles: SGBD, número de tabelas e número de tuplas.

De acordo com o que foi analisado neste trabalho, nota-se que as configurações que proporcionam melhor desempenho é relativo à métrica escolhida. Quando se trata de menor tempo de execução, a melhor combinação encontrada dentre os fatores e níveis analisados foi o uso do PostgreSQL utilizando uma única tabela com 1000 tuplas, obtendo o tempo total de 1,150ms, e a pior combinação foi o uso do SQL Server com 2 tabelas e 5000 tuplas, com o tempo de 3,962ms. Já para menor nível de falha na requisição, a configuração que possibilitou melhor desempenho foi o SQL Server com 500 threads simultâneas, pois proporcionou falha de 0% nas requisições, e a pior combinação foi o PostgreSQL com 2500 threads simultâneas, pois resultou em 96% de falha.

Como trabalhos futuros, pretende-se analisar outros SGBDs, bem como utilizar outros fatores e níveis para analisar mais profundamente o impacto dos mesmos na disponibilidade de um Banco de Dados. Além disso, também planeja-se testar orquestração de contêineres com SGBDs.

Referências

Amento, B., Hall, R. J., Joshi, K., and Purdy, K. H. (2018). Systems and methods for allocating and managing resources in an internet of things environment using location based focus of attention. US Patent App. 15/432,042.

Antony, J. (2006). Taguchi or classical design of experiments: a perspective from a practitioner. *Sensor Review*, 26(3):227–230.

Barik, R. K., Lenka, R. K., Rao, K. R., and Ghose, D. (2016). Performance analysis of virtual machines and containers in cloud computing. In *Computing, Communication and Automation (ICCCA), 2016 International Conference on*, pages 1204–1210. IEEE.

Dua, R., Raja, A. R., and Kakadia, D. (2014). Virtualization vs containerization to support paas. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 610–614. IEEE.

Fink, J. (2014). Docker: a software as a service, operating system-level virtualization framework. *Code4Lib Journal*, 25:29.

Gaur, N., Joshi, P., and Srivastava, R. (2017). Modelling database server sizing for concurrent users using coloured petri-nets. In *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, pages 90–94. IEEE.

Gillan, C. J., Novakovic, A., Marshall, A. H., Shyamsundar, M., and Nikolopoulos, D. S. (2018). Expediting assessments of database performance for streams of respiratory parameters. *Computers in biology and medicine*, 100:186–195.

Gunst, R. F. (1996). Response surface methodology: process and product optimization using designed experiments.

Gupta, P., Agrawal, D., Chhabra, J., and Dhir, P. K. (2016). Iot based smart healthcare kit. In *2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT)*, pages 237–242. IEEE.

Hsieh, Y.-C., Hong, H.-J., Tsai, P.-H., Wang, Y.-R., Zhu, Q., Uddin, M. Y. S., Venkatasubramanian, N., and Hsu, C.-H. (2018). Managed edge computing on internet-of-things devices for smart city applications. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–2. IEEE.

Jiang, L., Da Xu, L., Cai, H., Jiang, Z., Bu, F., and Xu, B. (2014). An iot-oriented data storage framework in cloud computing platform. *IEEE Transactions on Industrial Informatics*, 10(2):1443–1451.

Kuhn, D. R. and Reilly, M. J. (2002). An investigation of the applicability of design of experiments to software testing. In *Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE*, pages 91–95. IEEE.

Lazic, Z. R. (2006). *Design of experiments in chemical engineering: a practical guide*. John Wiley & Sons.

Miller, C., Thomas, D., Irigoyen, S. D., Hersberger, C., Nagy, Z., Rossi, D., and Schlueter, A. (2014). Bim-extracted energyplus model calibration for retrofit analysis of a historically listed building in switzerland. *Proceedings of SimBuild*, 2014.

Morabito, R. (2016). A performance evaluation of container technologies on internet of things devices. In *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*, pages 999–1000. IEEE.

Morabito, R. (2017). Virtualization on internet of things edge devices with container technologies: a performance evaluation. *IEEE Access*, 5:8835–8850.

Niu, M., Cheng, B., Zhai, Z., Feng, Y., and Chen, J. (2017). Poster: Docker-based self-organizing iot services architecture for smarthome. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 153–153. ACM.

Ruiz Espejo, M. (2006). Design of experiments for engineers and scientists.

Seltman, H. J. (2012). Experimental design and analysis. *Online at: http://www. stat. cmu. edu/, hseltman/309/Book/Book. pdf*.

Shirinbab, S., Lundberg, L., and Casalicchio, E. (2017). Performance evaluation of container and virtual machine running cassandra workload. In *Cloud Computing Technologies and Applications (CloudTech), 2017 3rd International Conference of*, pages 1–8. IEEE.

Siebertz, K., Van Bebber, D., and Hochkirchen, T. (2017). *Statistische Versuchsplanung: design of experiments (DoE)*. Springer-Verlag.

Thomas, D., Miller, C., Kämpf, J., and Schlueter, A. (2014). Multiscale co-simulation of energyplus and citysim models derived from a building information model. In *Bausim 2014: Fifth German-Austrian IBPSA Conference*.

Velásquez, W., Munoz-Arcentales, A., and Rodriguez, J. S. (2018). A case study: Ingestion analysis of wsn data in databases using docker. In *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, pages 1–6. IEEE.

Xavier, M. G., Israel C. De Oliveira, F. D. R., Robson D. Dos Passos, K. J. M., and Rose, C. A. F. D. (2016). Containers or hypervisors, which is better for database consolidation?