Avaliação de Desempenho de Nuvens usando Microsserviços

Leonel Feitosa, Carlos Victor, Juliana Carvalho, Francisco Airton Silva

[1] Laboratório de Pesquisas Aplicadas a Sistemas Distribuídos (PASID)
Universidade Federal do Piauí (UFPI)
Picos, PI, Brasil

{leonelfeitosa,carlosvictor,julianaoc,faps}@ufpi.edu.br

Abstract. *The process of using microservices in application development allows these microservices to be distributed in different cloud providers independently, but generating the impression that they are deployed in a single service. Thus, the use of microservices gains strength in the market, having a greater predominance in large corporations. This work aims to evaluate the performance of a system that uses the microservices architecture deployed in multiple cloud. For the experiments, three different providers were used, AWS (Amazon Web Services), Google Cloud and Azure, all conditions work independently. The experiments were carried out with an application in a centralized and distributed way. The results show the performance for the obtained scenarios and the particularities of each one of them.*

Resumo. *O processo de utilização de microsserviços no desenvolvimento de aplicações permite que estes microsserviços sejam distribuídos em diferentes provedores de nuvem de forma independente, porém gerando uma impressão que estão implantados em um único serviço. Dessa forma a utilização de microsserviços ganha força no mercado, possuindo uma maior predominância nas grandes corporações. O presente trabalho tem como objetivo avaliar o desempenho de um sistema que utiliza arquitetura de microsserviços implantada em múltiplas nuvens. Para os experimentos foram utilizados três provedores diferentes, AWS (Amazon Web Services), Google Cloud e Azure, todas as instâncias funcionam de forma independente. Os experimentos foram realizados com a aplicação de forma centralizada e distribuída. Os resultados mostram o desempenho para os cenários avaliados e as particularidades de cada um deles.*

1. Introdução

A cada ano, a computação em nuvem evolui à medida que as empresas buscam maneiras mais eficientes e eficazes de utilizar seu investimento em TI. A computação em nuvem fornece capacidade de implementação de serviços flexíveis e escaláveis sem ter os recursos de computação instalados diretamente no sistema do usuário [Josyula et al. 2011]. A adoção de serviços baseados em nuvem tornou-se cada vez mais difundida, visto que esse paradigma oferece um ajuste perfeito para uma ampla gama de aplicativos, por exemplo, aqueles que aproveitam as potencialidades do domínio IoT [Botta et al. 2016].

As características essenciais do modelo de computação em nuvem incluem autoatendimento sob demanda, elasticidade rápida, diversidade de recursos e amplo acesso à rede. Com isso, a computação em nuvem ganhou muita popularidade devido a eliminação

da sobrecarga de planejamento do usuário, fornecendo recursos que estão disponíveis sob demanda, autoatendimento e a capacidade de escalar de acordo com os requisitos [Armbrust et al. 2009]. Com a computação em nuvem, empresas começaram a aumentar seus recursos de computação apenas quando necessário. Os modelos de implantação usados pela computação em nuvem são nuvem privada que permite acesso apenas aos membros da organização, nuvem pública (acessível para uso público em geral mediante algum pagamento) e *Multiple Cloud* (ambiente de computação em nuvem em que uma organização fornece e gerencia alguns recursos internamente e os outros serviços são fornecidos externamente) [Zhu et al. 2011].

Embora compreendemos os modelos de implantação da computação em nuvem, não existe uma definição precisa sobre a própria computação em nuvem, devido a cada indústria impor sua própria definição. Além disso, com base na arquitetura de computação em nuvem que já existe, os usuários da nuvem podem aplicá-la para projetar e construir seu próprio ambiente e infraestrutura de computação em nuvem.

Empresas vêm desenvolvendo e implantando sobre plataforma de computação em nuvem, como por exemplo: Google, Amazon, Spotify que são plataformas para soluções de desenvolvimento, armazenamento ou entretenimento. Vantagens associadas a esse processo de separação dos serviços é permitir que diferentes times possam definir os métodos de implantação, gerar escalabilidade de partes especificas de uma aplicação [Muller et al. 2020]. Este trabalho apresenta um caso real de implementação e implantação da aplicação utilizando microsserviços.

A seguir, o artigo se encontra dividido da seguinte forma: A Seção 2 apresenta os trabalhos relacionados Na Seção 3 está descrita a aplicação utilizada nos experimentos e as tecnologias usadas para construí-la. A Seção 4 apresenta o processo de implantação da aplicação nos provedores de nuvem escolhidos. A Seção 5 mostra os detalhes a respeito da realização dos experimentos. Na Seção 6 foi feita a análise dos resultados obtidos nos experimentos. Por fim, a Seção 7 sintetiza as descobertas dos experimentos e conclui o artigo.

2. Trabalhos Relacionados

Nesta seção serão apresentados brevemente alguns trabalhos semelhantes a este. Diante do contexto de microsserviços o trabalho [de Carvalho et al. 2018] propõe um sistema de tomada de decisão baseado em economicidade para implantar uma aplicação distribuída em múltiplos provedores de nuvem. Considerando somente aplicações baseadas em microsserviços, a solução proposta deve selecionar provedores que melhor atendam aos requisitos dos microsserviços e de um arquiteto de *software*. No que se refere a avaliação de desempenho utilizando múltiplas nuvens e microsserviços, o trabalho de [Satheler 2021] propõe uma abordagem de alta disponibilidade baseada em redundância ativo-passivo. Os resultados do trabalho mostram que é possível balancear as aplicações FaaS entre os provedores de nuvem (AWS e Microsoft Azure) e que o tempo de disponibilidade pode ultrapassar 99,999%.

Já o trabalho de [Esperança 2016] traz uma ideia e abordagem bem diferentes das propostas aqui. A proposta do trabalho é apresentar uma abordagem que utiliza conceitos de DSDM (Desenvolvimento de Software Dirigido a Modelos), para auxiliar os desenvolvedores a distribuírem uma aplicação, inicialmente em uma nuvem, para múltiplas

nuvens. O trabalho [Lopes 2021] apresenta um método de migração intitulado *Microservice Full Cycle - MFC*, inspirado no ciclo de vida de desenvolvimento de software e em estratégias *DevOps*, levando em consideração um contexto onde o sistema existente não utiliza microsserviços, e uma real dificuldade de migração de projetos com arquitetura monolítica. O objetivo é auxiliar sistemas de *software* legados a gradualmente evoluírem orientados por um conjunto de etapas e atividades comuns à arquitetura de microsserviços.

Por fim, o trabalho de [Cândido et al. 2019] tem como objetivo realizar a migração do *Context Acquisition and Offloading System (CAOS)* para uma arquitetura de microsserviços, visando alcançar os benefícios que essa arquitetura fornece. Nos experimentos foram realizados teste de desempenho e escalabilidade, os resultados indicaram que a nova versão, chamada então de CAOS Microsserviços (CAOS MS), apresenta ganhos de escalabilidade em relação à versão monolítica, sem também comprometer seu desempenho geral.

3. Aplicação

A aplicação tem como objetivo organizar o processo de venda e coleta de materiais recicláveis. O sistema de vendas é composto basicamente por dois tipos de usuários, os coletores e os vendedores, O coletor realiza o cadastro de todos os itens a serem coletados. Após realização da coleta, o coletor marca o item como coletado. O registro de coleta fica armazenado e disponível para consulta, pois os dados podem ser utilizados para uma possível verificação e balanço de entrada de material. O vendedor por sua vez realiza o processo de venda dos materiais recicláveis, contudo, antes de uma venda ser efetuada, é realizado um cadastro prévio do que será vendido. Após o cadastramento do item, o vendedor aguarda a realização da venda para modificar o status para reciclado.

Para a construção da aplicação foi utilizado a plataforma *Nodejs, framework Expressjs e ORM Typorm*, como banco de dados foi utilizado o *MySQL*. A aplicação consiste em um sistema onde é realizado o gerenciamento de coleta para reciclagem. Os serviços foram divididos em cadastros, coletor e vendedor, cada microsserviços está alocado em uma instância diferente com funcionando e banco de dados independente. O cadastro diz respeito ao serviço de cadastro de itens por parte de um usuário coletor. Os serviços de usuário foram divididos em dois, um para o coletor de lixo e outro para o vendedor desse lixo coletado.

4. Processo de Implantação

Para a implantação foram utilizados três servidores, a AWS (Amazon web Services), Microsoft Azure e o Google Cloud. Em cada um dos provedores foi criando uma instância localizada em pontos geográficos distintos. Para desenvolver a aplicação foi utilizada a linguagem de programação *TypeScript*, com o ORM (Object Relational mapper) *TypeOrm*, com a plataforma *Nodejs* e a *Framework expressjs*. Em seguida, uma *API REST* foi desenvolvida, cuja qual foi separada em serviços que funcionam de forma independente. Para fins de teste, visualização e integração, foi desenvolvido um *front-end* utilizando a f*ramework Javascript, Angular*.

Os serviços foram hospedados em instancias na Amazon AWS[1], Microsoft Azure[2]

[1]https://aws.amazon.com/pt/?nc2=h_lg

[2]https://azure.microsoft.com/en-us/

e Google Cloud[3]. O processo de implantação seguiu os seguintes passos: 1) Foi criada uma instância com o sistema operacional Ubuntu 20.04[4]; 2) Foi criada uma instância de banco de dados[5]; 3a) Na AWS, foram criados 3 *containers*, e em cada um deles foi alocado um serviço da aplicação (cadastro de usuário, cadastro de reciclagem, cadastro de coleta); 3b) Na Google Cloud, foi feito o *deploy* do serviço de cadastro de reciclagem; 3c) Na Microsoft Azure, foi feito o *deploy* do serviço de cadastro de coleta. Os serviços foram iniciados em portas diferentes nas 3 instâncias.

5. Experimento

Essa seção apresenta o *setup* e arquitetura utilizados nesse trabalho. Nessa arquitetura, o sistema está dividido em microsserviços que se comunicam de forma sequencial. A Figura 1 representa uma arquitetura utilizada no sistema, que é composta por três partes:

1. **Serviço de usuário** se trata do cadastros dos usuários e fornecimento das requisições de login;
2. **Coletor** responsável pelo o recebimento e cadastro dos itens a ser vendidos;
3. **Vendedor** responsável pela realização da venda dos itens cadastro pelo o coletor;

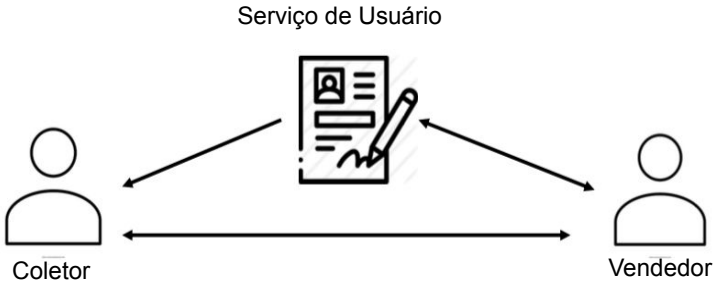

Figura 1. Arquitetura Base

O experimento consistiu em submeter o servidor a uma carga de trabalho através do envio de requisições realizadas pelo cliente e utilizando as operações de leitura nos bancos de dados. O vendedor acessa o *login* e solicita a leitura de um dado cadastrado por um coletor. Para efeito de variação de resultados foram utilizadas cargas de 1, 10, 50 e 100 requisições. Os provedores utilizados serão os que já foram descritos nas seções anteriores. Todos os provedores estão localizados nos Estados Unidos da América (EUA).

1. Cenário I: O sistema se encontra todo na AWS.
2. Cenário II: O sistema se encontra distribuído em três provedores diferentes, o microsserviço **serviço de usuário** se encontra na AWS, o microsserviço **vendedor** se encontra na Azure e o microsserviço **coletor** se encontra no Google Cloud.

6. Análise dos resultados

Nesta seção serão apresentados os resultados obtidos a partir da execução dos experimentos em diferentes cenários. Cada um dos dois cenários contêm um gráfico de *boxplot* e de Tukey, obtidos por meio de uma análise ANOVA na ferramenta Minitab[6] com os dados obtidos. O tamanho da amostra para cada quantidade de requisições (1, 10, 50, 100) é 15.

[3]https://cloud.google.com/
[4]Para o serviço AWS foi escolhida a EC2
[5]Na aws foi utilizado o recurso RDS
[6]https://www.minitab.com/en-us/

6.1. Resultados do cenário I

No primeiro cenário temos todos os serviços divididos, contudo todos eles estão numa instância da AWS localizada na região da Flórida nos EUA. A Figura 2 mostra um gráfico de *boxplot* com os resultados do experimento. Vale ressaltar que o tempo de reposta corresponde aqui ao *Round Trip Time* (RTT), ou seja, o tempo da requisição ir e voltar. No inicio, para 1 requisição por vez o tempo é bem curto, levando menos de 500ms mesmo no valor mais alto obtido. Entre 10 e 50 requisições houve um aumento de cerca de 100% na média de tempo, indo de cerca de 1s para aproximadamente 2s. Com 100 requisições simultâneas o valor médio chegou a ultrapassar 2,5s e em alguns casos chegou próximo a 3s, e pode-se observar um *outlier* onde o tempo chegou a próximo de 2s, valor médio de 50 requisições. Os valores são esperados uma vez que há um aumento na carga de trabalho. Contudo, quando se aumenta o número de requisições o serviço tende a ficar mais instável e ligeiramente variável, possivelmente por se tratar de uma máquina só para lidar com tudo.

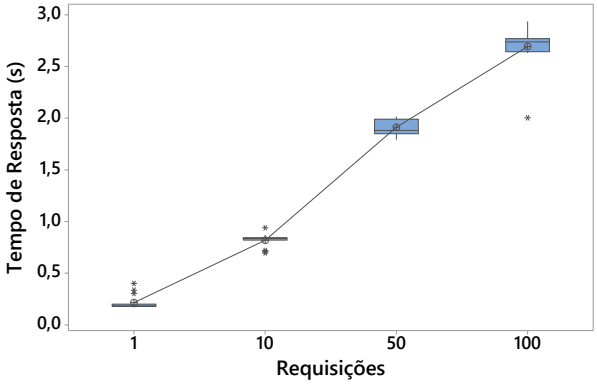

Figura 2. BoxPlot para microsserviços na AWS

Para mostrar melhor o impacto do aumento dessas requisições foi gerado um gráfico de Tukey. A Figura 3 mostra o gráfico de Tukey para a análise das amostras do experimento. O valor p obtido foi menor que 0,000, e isso implica que há uma diferença bem significativa entre os cenários, mostrando que o número de requisições é um fator bem importante para um sistema em nuvem. O gráfico mostra o nível de diferença entre a quantidade de requisições no tempo de resposta do sistema. A menor diferença entre quantidade de requisições foi com 1 e 10 requisições e também com 50 e 10, que são os que estão mais próximos de zero. A maior diferença foi entre os 1 e 100 requisições, já que há um aumento de mais de 2s. Também podemos ver que a diferença entre 10 e 100 requisições foi maior que a diferença entre 1 e 50. Por fim, como pudemos observar houve um aumento de tempo bem maior entre 10 e 50 requisições do que entre 50 e 100, isso porque há um aumento de 5 vezes entre 10 e 50, e entre 50 e 100 foi somente o dobro.

6.2. Resultados do cenário II

Neste cenário os 3 serviços apresentados foram colocados em provedores de nuvem diferentes, na AWS, na Azure e na Google Cloud. Da mesma forma que o cenário anterior, os serviços são acessados sequencialmente. A Figura 4 mostra um gráfico de *boxplot* com os resultados obtidos no experimento com múltiplas nuvens. Os valores obtidos com 1 requisição ficaram em torno de 300ms. Quando o número cresce para 10 requisições

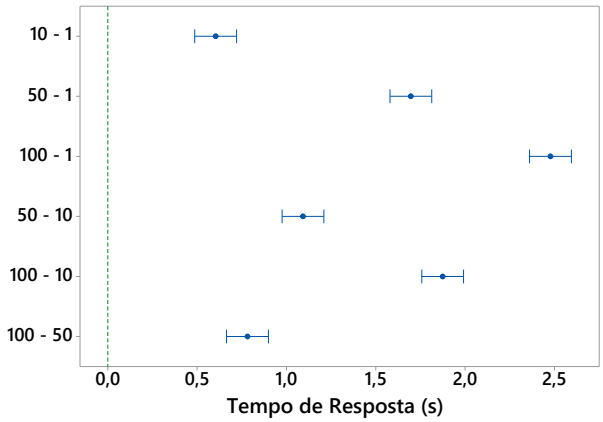

Figura 3. Tukey para microsserviços na AWS

temos um aumento de quase 10 vezes também, e o valor médio chegar a ser um pouco, mas que 2s. Entre 10 e 50 requisições tem-se um aumento de tempo de cerca de 4 vezes, chegando a em média 8,5s. Com 100 requisições o valor anterior dobra e alcança a marca de 16s, com alguns *outliers* beirando os 18s. Nesse cenário os tempos consideravelmente elevados se dão por causa que os 3 serviços estão em instancias diferentes, mas no mesmo estado, que é na Flórida nos EUA.

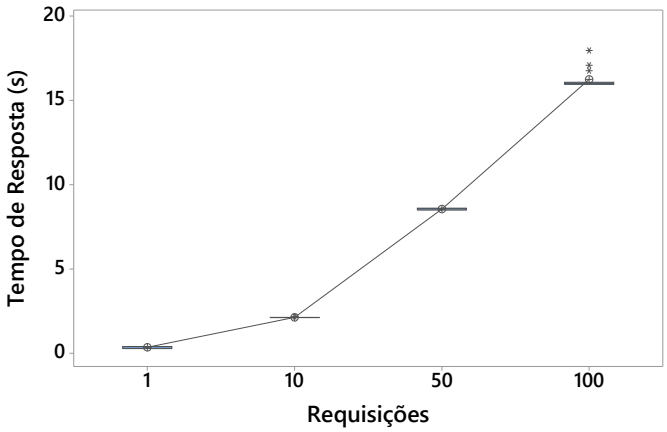

Figura 4. BoxPlot para microsserviços em múltiplas nuvens

A Figura 5 mostra melhor a diferença entre os resultados obtidos para diferentes quantidades de requisições. O valor p da análise ficou abaixo de 0,000, mostrando assim que há diferença entre um ou mais fatores. A menor diferença nos resultados foi entre 1 e 10 requisições, por mais que seja a menor ainda é uma diferença de 2s. A maior diferença foi entre 1 e 100 requisições, onde tem-se um aumento de 16s no tempo de resposta. A segunda maior diferença foi entre 10 e 100 requisições simultâneas, com uma diferença de 14 segundos. O restante das combinações ficou entre 6s e 8s. A diferença grande entre os diferentes números de requisições se dá devido a divisão dos serviços, como cada um está num provedor diferente pode ser que um ou mais deles tenha processos internos mais lento, ou muito rápido a ponto de criar um gargalo. A distância entre os servidores também pode ser um fator, eles estão todos num único estado, mas pode ser que haja uma distância interna que atrase a chegada do processo.

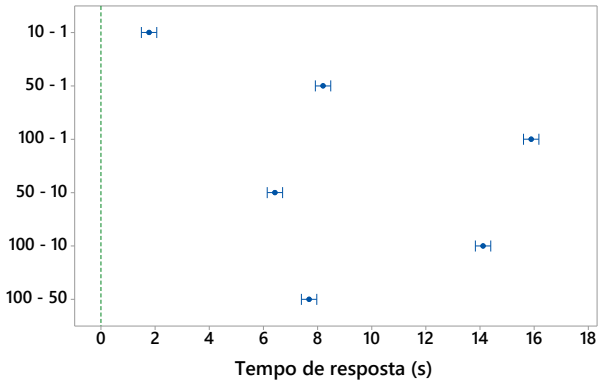

Figura 5. Tukey para microsserviços em múltiplas nuvens

6.3. Comparação entre cenários

Nessa sessão é feita uma comparação entre os resultados para checar se a distribuição de microsserviços impacta no tempo de resposta. A Tabela 1 mostra a média dos valores obtidos nos experimentos nos cenários I e II. Para 1 requisição os cenários I e II tiveram uma diferença de mais de 100ms, para algumas aplicações pode ser muito e para outras pouco. Com 10 requisições já pode-se ver um aumento considerável, com um aumento de cerca de 1,3s no cenário II. Quando se olha para os valores para 50 requisições o aumento é de mais que 6s no cenário II. Por fim, quando temos 100 requisições o valor aumenta em 6 vezes no cenário II indo de 2,7s para 16,24s.

Tabela 1. Média dos experimentos

Quantidade de Requisições	Cenário I	Cenário II
1	0,21s	0,36s
10	0,82s	2,13s
50	1,91s	8,56s
100	2,70s	16,24s

A diferença entre cenários é bem visível. O cenário I se mostrou bem mais rápido que o cenário II para processar as requisições. O motivo principal como já dito, é que no cenário II os microsserviços estão divididos em provedores diferentes. Essa divisão traz alguns fatores que podem aumentar o tempo, como a distância entre as máquinas, a largura de banda entre elas e até possíveis gargalos provenientes das diferenças de hardware de cada máquina.

7. Conclusão

O artigo propôs uma avaliação de performance de nuvens utilizando uma aplicação simples. A avaliação levou em conta o uso de nuvens centralizadas e distribuídas a fim de descobrir qual é mais rápida. Os experimentos foram feitos com os provedores de nuvem mais utilizados, AWS, Azure e Google Cloud. A quantidade de requisições se mostrou um fator bem importante e impactante nos diferentes cenários. O cenário centralizado se mostrou bem mais rápido para processar as requisições, mas apresentou uma variação maior que a do cenário distribuído. O cenário distribuído mostrou tempos de processamento maiores, porém a variabilidade de tempo por quantidade de requisições foi menor

que num cenário distribuído. Para concluir, o cenário centralizado se mostrou bem superior em questão de desempenho e por mais que houve uma variação maior, sua variação com 100 requisições foi cerca de 6 vezes menor que no cenário distribuído. O cenário distribuído pode ter levado mais tempo, mas entregou os dados com uma variação de tempo menor por quantidade de requisições. Como trabalho futuro, planeja-se fazer também uma análise de consumo de memória das nuvens para tentar descobrir se há vantagens para o cenário distribuído.

Referências

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., and Konwinski, A. (2009). Above the clouds: a berkeley view of cloud computing. *Technical report, EECS Department, University of California, Berkeley*.

Botta, A., Donato, W., Persico, V., and Pescapé, A. (2016). Integration of cloud computing and internet of things: A survey. *Future Generat. Comput. Syst.*, pages 684–700.

Cândido, A. L., Trinta, F. A., Rego, P. A., Rocha, L. S., Mendonça, N. C., and Garcia, V. C. (2019). Um relato sobre a migração de uma plataforma de offloading para microsserviços. In *Anais do VII Workshop on Software Visualization, Evolution and Maintenance (VEM)*, pages 29–36. SBC.

de Carvalho, J. O., Trinta, F., and Vieira, D. (2018). Pacificclouds: A flexible microservices based architecture for interoperability in multi-cloud environments. In *CLOSER*, pages 448–455.

Esperança, V. N. (2016). Uma abordagem dirigida por modelos para distribuição tardia de aplicações.

Josyula, V., Orr, M., and Page, G. (2011). *Cloud computing: Automating the virtualized data center*. Cisco Press.

Lopes, T. R. (2021). Método de migração de sistemas monolíticos legados para a arquitetura de microsserviços.

Muller, R. H., Meinhardt, C., and Mendizabal, O. M. (2020). Microsserviços aplicados no gerenciamento de dados de vistorias imobiliárias: um estudo de caso. In *Anais do XVIII Workshop em Clouds e Aplicações*, pages 41–54. SBC.

Satheler, G. B. (2021). Alta disponibilidade de funções como serviço em ambiente de múltiplas nuvens de computação.

Zhu, Y., Hu, H., Ahn, G., Han, Y., and Chen, S. (2011). Collaborative computing: Networking, applications and work sharing (collaboratecom). *7th International Conference on*, pages 191–200.