UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

PEDRO AUGUSTO ALCÂNTARA RIBEIRO MORAES

UMA ANÁLISE DE DESEMPENHO *MULTI-CLOUD* DE UMA APLICAÇÃO BASEADA EM MICROSSERVIÇOS

Orientador:
Dra. Juliana Oliveira de Carvalho

Picos – PI
Novembro de 2021

Pedro Augusto Alcântara Ribeiro Moraes
Orientadora: Juliana Oliveira de Carvalho

Uma análise de Desempenho Multi-Cloud de uma Aplicação Baseada em Microsserviços

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Universidade Federal do Piauí
Campus Senador Helvídio Nunes de Barros
Bacharelado em Sistemas de Informação

Picos - PI
Novembro de 2021

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Campus Senador Helvídio Nunes de Barros
Biblioteca Setorial José Albano de Macêdo
Serviço de Processamento Técnico

M828a Moraes, Pedro Augusto Alcântara Ribeiro
Uma análise de desempenho multi-cloud de uma aplicação baseada em microsserviços / Pedro Augusto Alcântara Ribeiro Moraes – 2021.

Texto digitado
Indexado no catálogo *online* da biblioteca José Albano de Macêdo-CSHNB
Aberto a pesquisadores, com as restrições da biblioteca

Trabalho de Conclusão de Curso (Graduação) – Universidade Federal do Piauí, Bacharelado em Sistemas de Informação, Picos-PI, 2021.

"Orientadora: Juliana Oliveira de Carvalho"

1. Microsserviços. 2. Multi-cloud. 3. Síncrono. 4. Assíncrono. I. Carvalho, Juliana Oliveira de. II. Título

CDD 004

Maria José Rodrigues de Castro CRB 3: CE-001510/O

Uma análise de Desempenho Multi-Cloud de uma Aplicação Baseada em Microsserviços

PEDRO AUGUSTO ALCÂNTARA RIBEIRO MORAES

Monografia apresentada como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 17 de NOVEMBRO de 2021

Profa. Juliana Oliveira de Carvalho

Prof. Francisco Airton Pereira da Silva

Prof. Glauber Dias Gonçalves

Agradecimentos

A Deus, por me dar forças e bençãos para vencer os obstáculos e chegar até aqui. À minha Orientadora, Juliana Oliveira de Carvalho, pela constante ajuda e orientação neste trabalho, e contribuição fundamental na minha formação. À minha querida Avó, Domingas Alcântara da Cruz (*in memoriam*) a qual sou grato por todos os ensinamentos em vida. À minha companheira e aos meus familiares por todo incentivo, apoio e carinho.

Por amor às causas perdidas (Engenheiros do Hawaii).

Resumo

A popularização da computação em nuvem abriu-se espaço para o surgimento de vários provedores de nuvens, oferecendo serviços diversos e especializados. Com isso e com a popularização de microsserviços, tem-se a oportunidade de alocar aplicações em diversas nuvens, aproveitando assim o melhor de cada provedor. Entretanto, apesar de ter-se arquiteturas de microsserviços independentes, é necessário que haja comunicação entre esses serviços, as quais podem ser síncronas e assíncronas. Ainda assim, pouco se sabe sobre o desempenho dessas tecnologias quando utilizados em ambientes *multi-cloud*. Desta forma, este trabalho propõe avaliar o desempenho da comunicação síncrona e assíncrona em ambientes *multi-cloud*, e para isto, construindo uma aplicação intitulada *e-bank*, distribuindo-a em ambientes *single-cloud* e *multi-cloud*, e além disso, realizando experimentos que levam em consideração o número de comunicações entre os microsserviços e a quantidade de acessos simultâneos (1, 100 e 1000) para a aplicação. Através dos resultados obtidos pode-se constatar que a comunicação síncrona tem desempenho melhor que comunicação assíncrona quando a aplicação está em ambiente *single-cloud* com até 100 acessos, porém tende a aumentar o tempo de resposta em relação a comunicação assíncrona a medida que o número de acessos simultâneos aumentam. Em *multi-cloud*, no geral, a comunicação assíncrona obteve melhores resultados em relação a comunicação síncrona, se mostrando mais escalável a medida que o número de acessos simultâneos aumentam. Logo, de acordo com esta análise pode se observar que os dois tipos de comunicação são viáveis para aplicações *multi-cloud*, entretanto a comunicação assíncrona tende a responder de forma mais rápida, ser mais escalável e oferecer melhor resiliência para os microsserviços.

Palavras-chaves: Microsserviços. Multi-cloud. síncrono. assíncrono.

Abstract

The popularization of cloud computing opportunities opened up for the emergence of several cloud providers, offering diverse and specialized services. In this context, and its dues the popularization of microservices, there is the opportunity to allocate applications in multiple clouds, thus taking advantage of the best of each provider. However, despite having independent microservices architectures, there must be communication between these services, and the market has several communication technologies such as Synchronous and asynchronous communication. Still, little is known about the performance of these technologies when used in multi-cloud environments. Thus, this work proposes to evaluate the performance of synchronous and asynchronous in multi-cloud environments, and for this, building an application called e-bank, distributing it in single-cloud and multi-cloud environments, and in addition, performing experiments which take into account the number of communications between microservices and the number of simultaneous accesses (1, 100 and 1000) for the application. The results obtained show that synchronous communication performs better than asynchronous communication when the application is in a single-cloud environment with up to 100 accesses. However, it tends to increase the response time about Queue as the number of accesses simultaneous increases. On the other hand, in multi-cloud, Queue generally obtained better results about REST, showing itself more scalable as the number of simultaneous accesses increases. Therefore, according to this analysis, it can be observed that both types of communication are viable for multi-cloud applications. However, asynchronous communication tends to respond faster, be more scalable, and offer better resilience for microservices.

Keywords: Microservices. Multi-cloud. asynchronous. Synchronous.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
APPS	*Azure Aplication Service*
AWS	*Amazon Web Services*
CPU	*Central Process Unit*
CSQL	*CloudSQL*
EBS	*Amazon Elastic Beanstalk*
EC2	*Amazon Elastic Compute Cloud*
GCP	*Google Cloud Platform*
JSON	*JavaScript Object Notation*
PaaS	*Platform as a service*
PF	Programação Funcional
SDB	*Azure SQL Database*
SQS	*Amazon Simple Queue Service*
RDS	*Amazon Relational Database Service*
REST	*Representational State Transfer*

Sumário

1 Introdução

A Internet está no centro de uma nova revolução, onde os recursos são globalmente conectados e podem ser facilmente compartilhados (LOMBARDI; PIETRO, 2011). Nesse cenário, a computação em nuvem constitui um dos principais componentes desta mudança de paradigma computacional e vem alterando toda a perspectiva da forma de como a computação é vista (OMOTUNDE et al., 2013).

Segundo Etro (2015), algumas das justificativas para o devido sucesso da computação em nuvem vão desde a facilidade na aquisição de recursos, a economia de escala obtida pela consolidação de servidores, e até benefícios relacionados à diminuição de custos com a infraestrutura e pessoal. A nuvem se apresenta como uma fonte "inesgotável" de recursos virtualizados, acessados como serviços pelos seus clientes, e permitindo o crescimento elástico de sistemas (BENLIAN et al., 2018).

Por apresentar esses benefícios, devem ser investido cerca de 623.3 bilhões de dólares no setor até o final de 2023, além de 94% das empresas já terem suas soluções tecnológicas em nuvens, e 30% de todos os orçamentos de TI já serem alocados para computação em nuvem (GALOV, 2021). Com isso, é observável que um futuro impulsionado pela nuvem está começando a se moldar de maneira mais intensa, à medida que mais e mais empresas migram para a infraestrutura remota.

Observando todo esse crescimento, a diversidade de provedores, suas diversas características e desempenho diversificado em determinados cenários, surge espaço para aplicações hospedadas em diversas nuvens, as chamadas *multi-cloud* que IBM (2021) define como o uso de serviços em nuvem de dois ou mais fornecedores, oferecendo às organizações mais flexibilidade para otimizar o desempenho, controlar custos e aproveitar as melhores tecnologias de nuvem disponíveis. Ainda no contexto de múltiplas nuvens, Carvalho, Vieira e Trinta (2019) afirmam que sua utilização é um método para mitigar bloqueio do fornecedor e permitem que os aplicativos sejam distribuídos em vários provedores de nuvem, a fim de adquirir melhores qualidades de serviço da perspectiva do arquiteto de *software*.

Embora a utilização de múltiplas nuvens tragam diversos benefícios, elas também trazem muitos desafios. Paraiso et al. (2012) destacam alguns deles, tais como: interoperabilidade, portabilidade e geo-diversidade. Carvalho, Vieira e Trinta (2019) cita que alguns desses problemas ocorrem pela especificidade impostas pelos diversos provedores.

Hootsuite (2018) afirma que em 2018 o número de usuários de Internet passavam de 4 bilhões (crescimento de 7% em comparação ao ano anterior), segundo a mesma pesquisa esse numero continuará crescendo. Com isso, a tendência é que as requisições de acesso às aplicações venham a crescer. Pensando em múltiplas nuvens, é necessário garantir que essas aplicações tenham um bom nível de escalabilidade que é definido por Marston et al.

(2011) como a capacidade que a infraestrutura básica tem de se expandir, adaptando-se ao número de utilizadores e aos dados suportados nas aplicações.

Uma das formas de garantir uma escalabilidade por parte do sistema é utilizando a arquitetura baseada em microsserviços que Clark (2016) define como uma forma de dividir a aplicação em componentes menores e completamente independentes, permitindo que ter maior agilidade, escalabilidade e disponibilidade.

Apesar de inúmeros benefícios, a utilização de arquitetura de microsserviço trás consigo alguns desafio como a migração de arquiteturas monolíticas para microsserviços. Em cenários onde temos uma aplicação monolítica densa, os engenheiros de *software* podem encontrar dificuldades em decompor essa aplicação em serviços especializados (IBM, 2019).

O surgimento de todo este cenário na computação trás consigo diversos problemas quando o assunto é a modelagem de *software* para a nuvem, tais como as tecnologias a serem utilizadas, técnicas de desenvolvimento das aplicações em questão e informações sobre o desempenho e disponibilidade de aplicativos baseados em microsserviços.

Um ponto bastante discutido em sistemas distribuídos tanto em trabalhos mais antigos como em Alexander (1991) e em trabalhos mais atuais como o de Abdelmawgoud, Jamshidi e Benavidez (2020) é o tempo de resposta de aplicações distribuídas, e que mesmo não fazendo parte desses trabalhos citados, microsserviços também compartilha desse problema, que por muitas vezes necessita de comunicação entre eles. Duas formas muito utilizadas para a comunicação de microsserviços são síncronos e assíncronos.

De acordo com os nosso conhecimentos, não há na literatura estudos relatando o desempenho de microsserviços distribuídos em múltiplas nuvens, principalmente quando levamos em consideração o tipo de comunicação entre esses serviços, desta forma, torna-se difícil escolher a forma de comunicação em cenários onde é necessário que o sistema seja resiliente e tenha tempo de resposta adequado, e um estudo levando em consideração múltiplas nuvens se torna ainda mais importante para concretizar a viabilidade de ambientes *multi-cloud*.

Dado o exposto, o objetivo principal deste trabalho é analisar o desempenho de uma aplicação baseada em microsserviços distribuídos em múltiplas nuvens, levando em consideração o tipo de comunicação que a mesma realizada (síncrono ou assincrono). Desta forma, contribuir para o desenvolvimento de *softwares* baseados em microsserviços para serem implantados em nuvem, dando um embasamento cientifico para os engenheiros que desenvolvem ou irão desenvolver esse tipo de aplicação.

1.1 Objetivos

Com o intuito de auxiliar pessoas responsáveis pelo o desenvolvimento de *software* a definirem quais os tipo de comunicação mais eficientes para ambientes distribuídos em

multi-cloud, este trabalho tem como objetivo geral avaliar o desempenho e funcionalidade de uma aplicação baseada em microsserviços distribuídos ou não em múltiplas nuvens, levando em consideração os tipos de comunicações Síncronos e Assíncronos.

Os objetivos específicos são:

1. Prover uma arquitetura de aplicação baseada em microsserviços para múltiplas nuvens, com o intuito de contribuir com os *engenheiros de softwares*

2. Mostrar que uma aplicação baseada em microsserviços distribuída em múltiplas nuvens pode usar tanto comunicação síncrona como assíncrona.

3. Mostrar que o paradigma puramente funcional pode ser utilizado no desenvolvimento de uma aplicação baseada em microsserviços distribuídos em múltiplas nuvens usando tanto comunicação síncrona como assíncrona, apresentando suas vantagens.

2 Referencial Teórico

Este capítulo apresenta os conceitos teóricos que são usados neste trabalho, e que foram a base de desenvolvimento desse projeto. A subseção 2.1 discute sobre Programação Funcional. Já na seção 2.2 explana sobre a arquitetura de microsserviços. A subseção 2.3 descreve sobre arquitetura de aplicações para nuvem. A seção 2.4 faz uma introdução sobre computação em nuvem. Por fim, a seção 2.5 descreve sobre comunicação síncrona, e a seção 2.6 sobre comunicação assíncrona.

2.1 Programação Funcional

Segundo Thomas (2009) programação funcional é um paradigma de programação totalmente diferente do modelo convencional, pois pode ser definida recursivamente como uma composição de funções em que cada função pode ser outra composição de funções ou um operador primitivo (como operadores aritméticos). O programador não precisa se preocupar com a especificação explícita de processos paralelos, pois as funções independentes são ativadas pelas funções predecessoras e pelas dependências de dados do programa. Isso também significa que o controle pode ser distribuído. Além disso, nenhum sistema de memória central é inerente ao modelo, pois os dados não são "gravados"por nenhuma instrução, mas são "transmitidos"de uma função para a seguinte. Dau (2017) cita as principais características, sendo elas:

- Programação sem estado: a qual significa que não existe um estado externo governando a execução do programa. Não há máquina de estado, o que destaca uma característica fundamental da programação funcional, funções puras.
- Funções puras: é uma função sem efeitos colaterais. Se a mesma função for executada com a mesma entrada, sempre fornecerá a mesma saída. Além disso, isso é um conceito crítico para a programação funcional porque permite que o compilador trabalhe nos bastidores e otimize o programa.
- A ausência de efeitos colaterais introduz o conceito de transparência referencial, o que significa que um argumento pode ser substituído por seu valor e ainda produzir o mesmo resultado. Por exemplo, se x = 3, chamar a função de fatorial x seria equivalente a fatorial 3.
- Dados Imutáveis: significam que uma vez que uma variável é instanciada, seu valor nunca pode ser alterado. Os programas funcionais “imitam” dados mutáveis mapeando uma função em uma estrutura de dados imutável e retornando uma nova

estrutura de dados como resultado. Uma vez que uma variável tenha sido definida, esse valor nunca mudará.

2.2 Microsserviços

Balalaie et al. (2018) afirmam que as arquiteturas de microsserviços estão se tornando o padrão para a construção de sistemas implantados continuamente. Ao mesmo tempo, há um crescimento substancial na demanda por migração de aplicações legadas no local para a nuvem. Nesse contexto, as organizações tendem a migrar suas arquiteturas tradicionais para arquiteturas nativas da nuvem usando microsserviços. Com base nisso, Newman (2015) cita algumas características que tornam os microsserviços diferentes dos sistemas monolíticos:

- são pequenos e focados em apenas uma parte do sistema, cada microsserviço é responsável por reunir partes das aplicações semelhantes.
- autônomo - O microsserviço é uma entidade separada. Pode ser implantado como um serviço isolado em um plataforma como serviço, ou pode ser seu próprio processo de sistema operacional.
- heterogeneidade das tecnologias - Com um sistema composto por vários serviços colaborativos, podemos decidir usar diferentes tecnologias diferentes dentro de cada uma. Isso permite escolher a ferramenta certa para cada trabalho, em vez de ter que selecionar uma abordagem mais padronizada e de tamanho único acaba sendo o menor denominador comum.
- resiliência - Em um sistema de microsserviços, se um componente de um sistema falha, essa falha não ocorrerá em cascata no restante do sistema, você pode isolar o problema e o resto do sistema pode continuar trabalhando.
- escalabilidade - Com um serviço grande e monolítico, precisa escalar todos os seus módulos juntos. Uma pequena parte de um sistema está restrito ao desempenho, mas se esse comportamento estiver bloqueado em uma aplicação monolítica gigante, tem-se que lidar com o dimensionamento de tudo como uma peça. Com serviços menores, pode-se apenas dimensionar os serviços que precisam de escala, permitindo executar outras partes do sistema em *hardware* menor e menos potente.

Um outro ponto destacado por Newman (2015) é a facilidade de modificação em questões de código. Uma simples modificação em uma aplicação monolítica, pode causa efeito cascata, produzindo erros por toda aplicação. Então, modificações nesse tipo de aplicação podem ser consideradas de grande impacto e alto risco, coisa que é reduzida em ambientes de microsserviços. Outro ponto citado é o alinhamento organizacional, ele

afirma que os microsserviços ajudam a dividir melhor as equipes para o desenvolvimento do *software*.

2.3 Arquitetura para Aplicação em microsserviços

A Figura 1 ilustra uma arquitetura simples para aplicações baseadas em microsserviços. Nela pode-se observar os componentes básicos de uma aplicação baseada em microsserviço, partindo desde a solicitação feita pelo cliente, passando pelo *gateway* que se trata de um centralizador de mensagens, até a chegada nas instancias do microsserviços, onde ocorre o processamento.

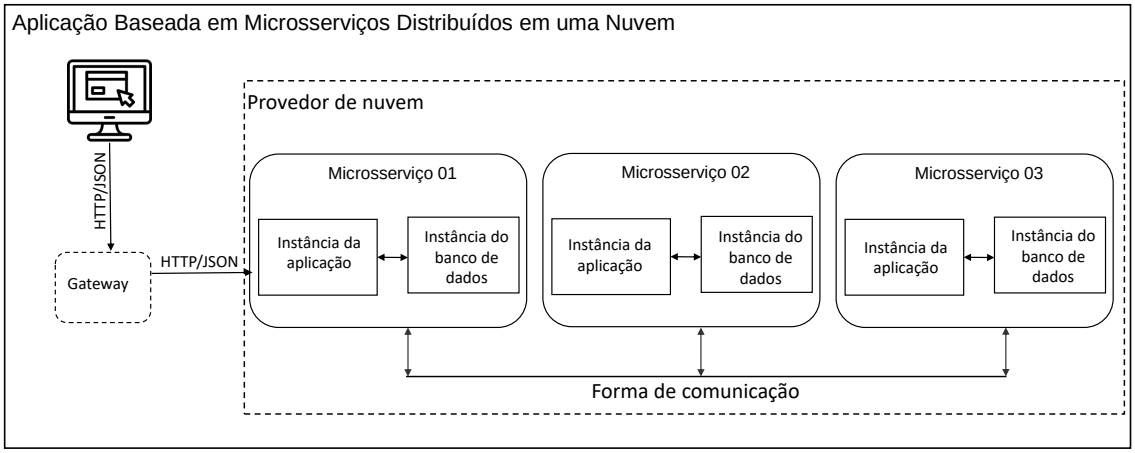

Figura 1 – Arquitetura de uma aplicação baseada em microsserviço

Para cada microsserviço tem-se uma ou mais instâncias de aplicação, a quantidade vai depender da escalabilidade e da necessidade de replicação desses microsserviços. Além destas instâncias, também existem as instâncias de banco de dados, onde Messina et al. (2016a) e Messina et al. (2016b) discutem sobre qual tipo de arquitetura de base de dados usar para a aplicação, sendo as principais arquiteturas: *shared database*, onde os microsserviços compartilham a mesma base de dados e *database per service*, onde cada microsserviço possui sua base de dados. A arquitetura utilizada neste trabalho foi *database per service*.

Um ponto importante em arquiteturas de microsserviços é o tipo de comunicação, onde existem formas síncrona e assíncronas. Dependendo do tipo de comunicação utilizada na aplicação, a mesma poderá ter algumas peculiaridades. Por exemplo, utilizando comunicação síncrona (Rest por exemplo), a aplicação precisará conhecer o endereço (ou pelo menos o *gateway*) de todos os microsserviços em que seja necessário estabelecer comunicação, fato que normalmente não acontece em comunicação síncrona (utilizando Fila, por exemplo), onde é necessário apenas conhecer o endereço do servidor e o nome da fila.

A comunicação síncrona pode ser utilizado por exemplo, em casos onde é necessário obter confirmação imediata de uma solicitação (seja de sucesso ou de falha), como no

caso de um sistema para alocação de passagens aéreas, pois se trata de um cenário onde o processamento sequencial (e pela ordem de solicitação), então, nesses casos a comunicação síncrona seria mais recomenda que a assíncrona

Ainda no exemplo de sistemas de passagem aéreas, pode-se exemplificar um cenário bastante utilizado para comunicação assíncrona. Após fazer a compra da passagem, o usuário precisa receber um e-mail (ou sms) para ser informado dos detalhes da viagem, nesse caso, o processamento pode acontecer de forma assíncrona, ou seja, mesmo se acontecer algum *delay* por conta de outros processamentos ocorrendo em outro microsserviço, um pequeno atraso no envio da mensagem não acarretará em grandes problemas.

Assim, nesta seção foi apresentada uma arquitetura básica para o funcionamento de uma arquitetura de microsserviços, mas, dependendo do tipo de aplicação, pode ser necessário outras tecnologias como por exemplo, gerenciador de logs, provedores de escalabilidade como kubertes ou até mesmo serviços de cluster para inteligencia artificial. Na arquitetura propostas neste trabalho não foram acrescentadas outras tecnologias, pois o objetivo do trabalho é verificar o desempenho de uma aplicação distribuída baseada em microsserviços implantada na nuvem.

2.4 Múltiplas Nuvens

Segundo Lombardi e Pietro (2011), a Internet está no centro de uma nova revolução, onde dados e *softwares* que antes ocupavam *desktops* e servidores corporativos estão sendo levado para a "nuvem". Esta mudança pode estar acontecendo por diversos fatores, dentre eles: um menor custo em comparação aos servidores locais, segurança ou até mesmo por ser mais fácil construir um *software* escalável na nuvem.

Mell, Grance et al. (2011) define computação em nuvem como um modelo para permitir acesso onipresente, conveniente e sob demanda da rede a um conjunto de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que pode ser rapidamente provisionado e liberado com o mínimo esforço de gerenciamento ou interação do provedor de serviços. Mell, Grance et al. (2011) citam as características essenciais da computação em nuvem:

- Acesso amplo à rede: os recursos mencionados anteriormente poderiam ser alcançado através de um sistema utilizando métodos heterogêneos, por exemplo, notebooks ou telefones celulares.

- Conjunto de recursos: conjuntos de recursos que estão disponíveis para os clientes. Isso é mencionado como multi-locação, onde, por exemplo, um servidor físico pode ter algumas máquinas virtuais tendo um lugar com clientes distintos.

- Rápida elasticidade: um cliente pode obter rapidamente mais recursos da nuvem ao expandir e pode voltar ao descarregar esses recursos quando não forem mais necessários.

- Serviço mensurado: a utilização dos recursos é medida por monitoramento do uso do armazenamento, horas da CPU, uso da largura de banda, etc.

Observando o último item, pode-se perceber que esse elemento é importante para que os provedores de computação em nuvem se diferencie. Serviços mensurados são aplicados a todas as nuvens, mas cada nuvem fornece aos usuários serviços em um nível diferente de abstração, que é uma alternativa a uma administração como relatado por (MELL; GRANCE et al., 2011).

Observando esse contexto, Carvalho, Vieira e Trinta (2019) relatam que atualmente várias aplicações complexas requerem múltiplos serviços e cada serviço também possui muitos requisitos. Estes aplicativos são atualizados considerando a complexidade, a quantidade de serviços e a plataforma de desenvolvimento. Com essa complexidade e diversidade impostas pelas atuais aplicações, Satzger et al. (2013) afirma que vários provedores de nuvem estão inundando o mercado com diversos serviços confusos, incluindo serviços de computação como Amazon Elastic Compute Cloud (EC2) e VMware vCloud.

Ainda segundo Satzger et al. (2013), alguns desses serviços são conceitualmente comparáveis entre si, enquanto outros são muito diferentes, mas todos são tecnicamente incompatíveis e não seguem nenhum padrão além dos seus. Este é um fator que pode "aprisionar"os usuários desses serviços (chamado de *vendor lock-in*) ao tentar mudar de uma provedor para outro. Carvalho, Vieira e Trinta (2018) afirmam que múltiplas nuvens podem ser usadas para reduzir o problema de *vendor lock-in* e aproveitar as vantagens da computação em nuvem, com isso, uma aplicação poderá ser implantado em vários provedores de nuvem que melhor atendam às necessidades de usuários e aplicativos. Os autores também citam os principais motivos para a utilização de *multi-cloud*, sendo eles:

- Escalabilidade, ampla disponibilidade de recursos e recuperação de desastres para redimensionar dinamicamente sua capacidade e possibilitar a cooperação entre várias nuvens para solucionar os problemas relacionados aos padrões de uso de serviço e implantação personalizados em um aplicativo em nuvem. O problema associado aos padrões de uso de serviço é que eles podem variar ao longo do tempo e na maior parte do tempo de maneira imprevisível, o que pode sobrecarregar um único provedor de nuvem, causando interrupções no serviço ou até mesmo tornando-os não confiáveis.

- Distribuição geográfica, baixa latência de acesso, questões legais, regulamentação, eficiência de custos, economia de energia, interoperabilidade e combate a fornecedores, permitindo desenvolver e implantar aplicações em nuvem altamente disponíveis

em várias nuvens, garantindo a qualidade e o desempenho do serviço. Por ser a única solução para atender às demandas dos consumidores de serviços geograficamente dispersos, que exigem baixo tempo de resposta e o uso simultâneo de várias nuvens.

2.5 Comunicação Síncrona

Segundo WILDE (2021), comunicação síncrona ocorre quando as mensagens só podem ser trocadas em tempo real. Requer que o transmissor e o receptor estejam presentes ao mesmo tempo e/ou espaço. Exemplos de comunicação síncrona são chamadas telefônicas ou videoconferências. Em contexto de desenvolvimentos de aplicações HAT (2020) ressalta as principais característica da comunicação síncrona, sendo elas:

- Uma arquitetura cliente-servidor composta de clientes, servidores e recursos, com solicitações gerenciadas normalmente por meio de HTTP.
- Dados armazenáveis em cache que otimizam as interações cliente-servidor.
- Um sistema em camadas que organiza cada tipo de servidor (os responsáveis pela segurança, balanceamento de carga, etc.) envolvia a recuperação das informações solicitadas em hierarquias, invisíveis para o cliente.
- *Code-on-demand*: a capacidade de enviar código executável do servidor para o cliente quando solicitado, estendendo a funcionalidade do cliente.

2.6 Comunicação Assíncrona

(MARTINS; JUSTINO; GABRIEL, 2010) afirma que a comunicação síncrona implica que os participantes se encontrem num mesmo espaço (físico ou online) e num momento específico de modo a poderem comunicar entre si ou trocar informações. Tem, por isso mesmo, como principal característica a interatividade gerada pela presença dos utilizadores e pode potenciar um clima de comunidade, onde o tempo de resposta dos participantes pode não acontecer momentaneamente. . No contexto de desenvolvimento de aplicações, Moraveis (2019) destaca os principais benefícios da utilização de comunicação assíncrona de aplicações:

- Redundância: Depois de armazenar a mensagem, a tecnoligia assíncrona garante que a mensagem só será removida quando o processo que a lê confirmar o sucesso na leitura e no processamento. Se algo der errado, a mensagem não será perdida e reprocessada posteriormente.

- Mensagens assíncronas: nos casos em que uma aplicação não exige uma resposta correta para um processo, o serviço que consome essas mensagens pode executar sua lógica de negócios em seu próprio ritmo.

- Resiliência: Digamos, por exemplo, que seu sistema seja composto por dois microsserviços, um para processamento de pedidos e outro para envio de emails. Ter uma processamento assíncrono para indicar que um e-mail precisa ser enviado significa que, mesmo se seu sistema de e-mail estiver inativo, o microsserviço de processamento de pedidos pode continuar recebendo e processando pedidos. Quando o serviço de e-mail estiver online novamente, ele pode começar a ler as mensagens e enviar os e-mails.

- Escalabilidade: o processamento assíncrono permitem desacoplar seu sistema em diferentes microsserviços e escaloná-los de acordo com a demanda.

3 Trabalhos Relacionados

Este capítulo são apresenta trabalhos relevantes relacionados ao nosso projeto e estão ordenados cronologicamente. Como fator de comparação foram utilizadas as seguintes métricas: se o trabalho utiliza microsserviços, se tem foco em ambientes *multi-cloud*, se utiliza a forma de comunicação síncrona ou assíncrona entre microsserviços e se há alguma análise de desempenho. Foram identificados 7 trabalhos, do ano de 2015 até 2018, os quais são destacados nos próximos parágrafos.

A fim de avaliar a implantação de aplicações que utilizam microsserviços, LIMA (2015) desenvolveu uma aplicação composta por um *backend* dividido em cinco serviços que foram implementados com a utilização de diferentes tipos de tecnologias, entre elas Java e Scala, sendo Scala uma linguagem multi-paradigma (podendo ser usado de forma orientado a objetos ou funcional). Ao fim do estudo, LIMA (2015) conclui que o maior desafio durante o desenvolvimento do aplicativo móvel foi a integração com o *backend*, que estava dividido em diversos serviços diferentes, que por sua vez respondiam em endereços e portas distintas.

Buscando diminuir o impacto da heterogeneidade presente no ambiente *multi-cloud*, Sousa, Rudametkin e Duchien (2016) proporam uma abordagem automatizada para a seleção e configuração de provedores de nuvem para aplicativos baseados em microsserviços distribuídos em várias nuvens. Dessa forma, os Sousa, Rudametkin e Duchien (2016) desenvolveram uma linguagem de modelagem específica para descrever os requisitos de diversas nuvens e fornecer um método sistemático para obter configurações adequadas para cumprir os requisitos do aplicativo e as restrições dos provedores de nuvem.

Villamizar et al. (2016) apresentam uma comparação de custos de um aplicativo Web desenvolvido e implantado usando os mesmos cenários escaláveis com três abordagens diferentes: 1) uma arquitetura monolítica, 2) uma arquitetura de microsserviço operada pelo cliente da nuvem e 3) uma arquitetura de microsserviço operada pela nuvem. Os resultados dos testes mostram que os microsserviços podem ajudar a reduzir os custos de infraestrutura em comparação com as arquiteturas monolíticas padrão. Além disso, o uso de serviços projetados especificamente para implantar e escalar microsserviços reduz os custos de infraestrutura em mais ou menos 70%.

Em seu trabalho, Singh e Peddoju (2017) analisaram diferentes desafios na implantação e integração contínua de microsserviços. Para superar esses desafios, eles propuseram, projetaram e implementaram um sistema automatizado que ajuda na implantação e integração contínua de microsserviços. Como resultado tem-se uma comparação de desempenho da abordagem monolítica e de microsserviço usando vários parâmetros, como tempo de resposta, taxa de transferência, tempo de implantação etc. Os resultados mostram que o aplicativo desenvolvido usando a abordagem de microsserviço e implantado usando o

design proposto reduz o tempo e o esforço para implantação e integração contínua do aplicativo.

Com o intuito de analisar a eficiência entre as comunicação síncronas REST e SOAP, Voigt e Junior (2017) realizam testes de desempenho nesses tipos de comunicação, onde como resultado é observado que o Web service na arquitetura REST possui um melhor desempenho em relação ao SOAP, nos testes de desempenho com 1, 10 e 100 usuários simultâneos realizando a mesma operação.

Carvalho, Trinta e Vieira (2018) propõem em seu estudo uma nova arquitetura intitulada *PacificClouds*, que se baseia em microsserviços para abordar interoperabilidade em ambientes *multi-clouds*, permitindo que o arquiteto de *software* escolha os requisitos e a arquitetura do aplicativo sem precisar se preocupar com a implantação do aplicativo na nuvem. O *PacificClouds* difere das outras arquiteturas pois fornece uma maior flexibilidade devido ao padrão arquitetural dos microsserviços. Por fim, os autores chegam a conclusão que além da flexibilidade (citada anteriormente), o *PacificClouds* oferece leveza, governança e facilidade para migração de aplicativos para múltiplas nuvens.

A fim de explorar os métodos de comunicação de aplicativos da Web de microsserviço, Hong, Yang e Kim (2018) usam RabbitMQ como comunicação assíncrona e REST API como comunicação síncrona, respectivamente, como um *middleware* orientado a mensagens para microsserviço. O objetivo é fornecer compreensão dentro dos dois métodos para aplicações web de microsserviço para que os provedores de serviço possam selecionar o método apropriado com base em sua necessidade. Os resultados experimentais obtidos mostram que quando um grande número de usuários envia solicitações ao aplicativo da web ao mesmo tempo, é mais estável usar o RabbitMQ como *middleware* orientado a mensagens do que o método de comunicação da API REST.

Com o a finalidade de realizar uma análise de desempenho entre aplicações monolítica e aplicações baseadas em microsserviços, Cunico (2018) desenvolve duas aplicações, que são destinados a funcionarem em diferentes ambientes, um para o ambiente monólito e outro para ambiente de microsserviços utilizando REST. Com isto, o autor demonstrou que a aplicação baseada em microsserviços teve um melhor desempenho quando comparado a aplicação monolítica.

Uma comparação entre os trabalhos apresentados anteriormente é mostrado na Tabela 1. Nós apresentamos na Tabela 1 cinco critérios de avaliação dos trabalhos, além da primeira coluna que contém a referência de cada trabalho. Na segunda coluna, nós verificamos se os trabalhos tratam de aplicações baseadas em microsserviços, assim temos uma métrica de trabalho onde levamos em consideração uma arquitetura de aplicação que pode ser distribuída. Na terceira coluna, nós verificamos se o trabalho trata de aplicações que estão distribuídas em múltiplas nuvens, com isso, temos uma base se os trabalhos, quando conveniente, tratam de distribuição em diversos provedores. Já na quarta e quinta coluna, nós verificamos se as aplicações usam comunicação síncrona e assíncrona respecti-

vamente, com isso, verificamos se os trabalhos realizam algum tipo de comunicação entre os componentes da aplicação e de que forma. Por fim, na sexta coluna, nós verificamos se o trabalho investiga o desempenho de uma aplicação, a fim de sabermos se existe algum trabalho que realiza alguma tipo análise de desempenho para comparação.

Embora alguns dos trabalhos forneçam soluções para microsserviços distribuídos em *single-cloud* ou *multi-cloud* e outros realizem a análise de desempenho entre tipos de comunicação (alguns levando em consideração apenas a comunicação síncrona e outros síncrono e assíncrono) entre microsserviços, este trabalho se diferencia dos demais pois realiza uma análise de desempenho de microsserviços distribuídos em *multi-cloud* levando em consideração os tipos de comunicação síncrono e assíncrono entre esses microsserviços.

Tabela 1 – Características de Trabalhos relacionados

Autor	É Microsserviço?	É Multi-Cloud?	Comunicação síncrona?	Comunicação assíncrona?	Analisa Desempenho?
LIMA (2015)	Sim	Não	Não	Não	Não
Sousa, Rudametkin e Duchien (2016)	Sim	Sim	Não	Não	Não
Villamizar et al. (2016)	Sim	Não	Não	Não	Sim
Singh e Peddoju (2017)	Sim	Não	Não	Não	Sim
Voigt e Junior (2017)	Sim	Não	Sim	Não	Sim
Carvalho, Trinta e Vieira (2018)	Sim	Sim	Não	Não	Não
Hong, Yang e Kim (2018)	Sim	Não	Sim	Sim	Sim
Este trabalho	Sim	Sim	Sim	Sim	Sim

4 e-bank: uma aplicação para nuvem

Com o intuito de analisar o desempenho de aplicações baseadas em microsserviços, foi desenvolvida uma aplicação para testes intitulada *e-bank*, onde a mesma tem a finalidade de aprovar transações bancárias com cartão de credito fictícia. *e-bank* foi desenvolvida baseada em microsserviços utilizando a linguagem de programação funcional Clojure. Essa aplicação contém 3 microsserviços que simulam uma transação bancária. Foram desenvolvidas duas arquiteturas para esta aplicação, onde uma utilizava como comunicação síncrona e outra utilizando comunicação assíncrona

Para realizar uma análise sobre desempenho em aplicações distribuídas em nuvens, também foram desenvolvidas arquiteturas que levavam em consideração cada microsserviço e os ambientes em que os mesmos estavam distribuídos. Nas próximas subseções, apresentam as tecnologias e as arquiteturas da aplicação e em seguida explana-se sobre a arquitetura *cloud* criada para a realização dos experimentos.

4.1 Funcionalidades

Com o intuito de realizar aprovações de transações bancárias, a aplicação *e-bank* possui 3 microsserviços, onde cada serviço possui seu banco de dados próprio. O microsserviço 1 era responsável pelas informações básicas do cliente, nele continha informações pessoais a qual identificava o cliente no sistema. Nesse microsserviços é possível realizar consulta, cadastro ou exclusão de clientes.

O microsserviço 2 realiza a validação de transações fictícias, podendo ser compras com cartão de crédito, débito ou boleto. Nele se tem informações de cartões, saldos e outras preferencias dos clientes. Ao chegar uma requisição neste microsserviço, ele deve analisar se o cliente está apto a realizar uma determinada compra, além de pode consultar ou adicionar informações de cartões.

O microsserviço 3 se tratava de simulador de validação de fraudes, nele é feita análises simples onde é analisado o histórico de compra do cliente e tem como o intuito de verificar se nova compra se trata de uma possível fraude, tendo sua importância para validações de compras.

Com a junção desses três microsserviços, temos um sistema onde um determinado cliente pode realizar comprar via cartão (crédito ou débito), onde dependendo dependendo do cenário onde essa compra é realizada, ela pode ser aprovada ou negada, seja por falta de saldo, limite ou suspeita de fraude.

4.2 Tecnologias da Aplicação

A Aplicação desenvolvida tem um contexto bancário, onde foram criados 3 microsserviços para a realização de transações de *gateway* para autorização de pagamentos. Cada microsserviço tem o seu banco de dados e regras de negócio próprios, além de ter a capacidade de recuperação de falha em comunicações entre os serviços utilizando técnicas de *retry*.

Para realizar o desenvolvimento da aplicação foi utilizado a Linguagem de Programação Clojure (Versão 1.10), os *frameworks* Compojure (Versão 1.6.1) e Ring (Versão 1.0), ambos para a construção e disponibilização de API's RESTs para comunicação síncrona. Para a aplicação com comunicação assíncrona foi utilizado fila de mensageria utilizando a biblioteca squeedo (versão 1.1.2).

A biblioteca Clojure/jdbc (Versão 0.7) também foi utilizada para integração com o banco de dados, além de utilizar Java 8 para a compilação do projeto. Para o banco de dados, foi utilizado o Sistema Gerenciador de banco de dados PostgreSQL (versão 12.7) para todos os serviços. Na parte de conteinerização dos microsserviços foi utilizado o Docker (versão 20.0).

Para disponibilizar as API's para os devidos experimentos foi necessário realizar a implantação das aplicações levando em consideração ambiente *multi-cloud* e *single-cloud* utilizando tanto a comunicação síncrona quanto a assíncrona. Para isso, foram criados duas aplicações, a primeira é a aplicação baseada em microsserviços em comunicação síncrona, a segunda possui as mesmas funcionalidades da primeira aplicação, porém, utiliza comunicação assíncrona para comunicação.

4.3 Arquitetura *Cloud* para aplicação *e-bank*

Com o intuito de realizar os testes da aplicação tanto em uma quanto em múltiplas nuvens, foram desenvolvidas quatro arquiteturas de *software* distintas, onde foi levado em consideração o tipo de comunicação e o tipo de distribuição da aplicação (*single-cloud* ou *multi-cloud*). Tendo em vista o vasto número de provedores de nuvem e seus inúmeros serviços de hospedagem, nas próximas subseções contêm informações sobre as tecnologias dos provedores utilizadas e em seguida a arquitetura desenvolvida.

Para a realização dos testes foram selecionados dos três provedores de nuvem mais utilizados atualmente, (Segundo Chand (2021)), *Amazon - AWS, Google Cloud* e *Microsoft Azure*, sempre utilizando os serviços PaaS oferecidos por cada provedor. A Tabela 2 destaca as tecnologias utilizadas em cada nuvem, levando em consideração a plataforma, o serviço de aplicação web (serviço de deploy da aplicação), o serviço de database e a *Queue Service*. Outra ferramenta *cloud* utilizada foi o *API Gateway* da Amazon, para centralizar as requisições da aplicação.

Tabela 2 – Lista de Serviços por Provedor

Plataforma	Applications Service	Database Service	Queue Service
AWS	Elastic Beanstalk	Relational Database	Simple Queue Service
GCP	App Engine	Cloud SQL	N/A
Azurre	App Service	SQL Database	N/A

Em relação as configurações de cada serviço ou instância foram utilizadas configurações idênticas para todas as máquinas (Seja Web Aplication, Database Service ou Queue). Para o Serviço de Web Aplication, cada instância teve a seguinte configuração: 2GB de memória RAM, 500 GB de HD em Imagens Docker. Para as bases de dados foram utilizadas instâncias com 1GB. Para a comunicação assíncrona, utilizou-se Fila de Mensageria com a ferramenta SQS, utilizando configuração padrão e utilizando o aspecto de melhor ordenação possível e sem criptografia.

Com base nas tecnologias nos parágrafos anteriores, foram desenvolvidas quatro arquiteturas: duas delas levando em consideração que a aplicação é implantada em apenas 1 nuvem, porém uma das arquiteturas leva em consideração que a aplicação se comunica de forma síncrona e a outra de forma assíncrona. As próximas subseções apresentam essas arquiteturas de forma mais detalhada.

4.3.1 Arquitetura da Aplicação utilizando comunicação síncrona

A Figura 2 mostra a arquitetura desenvolvida para o cenário em que a aplicação está em apenas uma nuvem e realizando comunicação síncrona, utilizando a tecnologia REST. Com isso, os microsserviços se comunicam diretamente, sem a necessidade de um *Gateway, Middleware*, ou outro tipo de ferramenta intermediária para a comunicação entre os serviços.

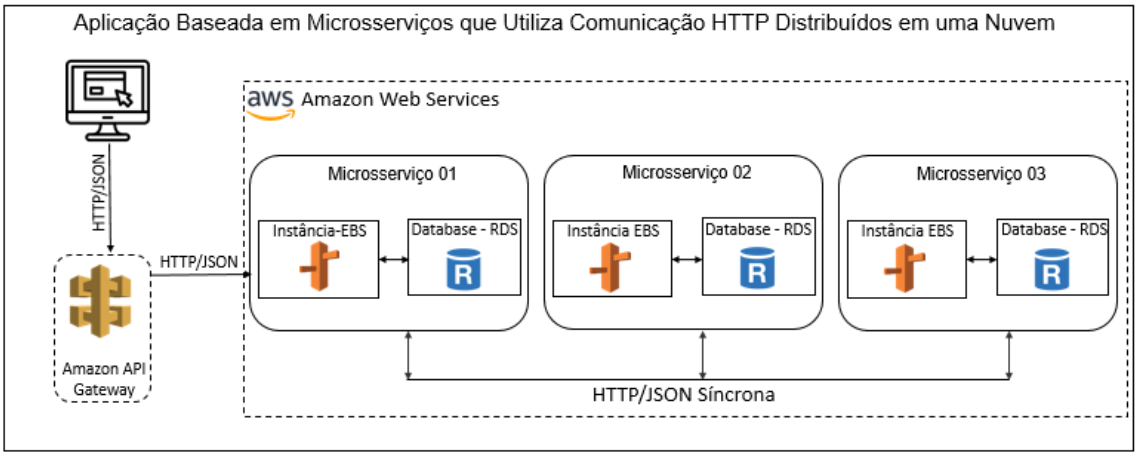

Figura 2 – Aplicação utilizando comunicação *single-cloud* síncrona

Nesta arquitetura temos um *Gateway* (foi utilizado a *API Gateway* da Amazon) que

centraliza as chamadas feitas pelo cliente, ou seja, é como se fosse um *Middleware* para tratar as requisições e orquestrar qual serviço deve ser chamado. Essa aplicação foi totalmente alocada em um provedor de nuvem, nesse caso, foi utilizado a *AWS*. Cada microsserviço executava em uma instancia do Amazon Elastic Beanstalk (EBS) e com seu próprio banco de dados, nesse caso, uma instância da Amazon Relational Database Service (RDS).

A comunicação síncrona entre os serviços foi realizada via REST, onde cada serviço conhecia o endereço que os outros serviços estavam alocados, quais parâmetros e métodos HTTP (POST, GET, PUT ou DELETE) eles esperavam para realizar uma comunicação com sucesso. No corpo da mensagem eram enviados JSON's para o envio dos parâmetros necessários para o processamento dos dados.

Seguindo o mesmo da aplicação síncrona distribuída em uma única nuvem, a Figura 3 representa a arquitetura desenvolvida para a aplicação distribuída em múltiplas nuvens que utiliza REST para a comunicação síncrona. Nesse caso, cada microsserviço foi implantado em uma nuvem, o microsserviço 01 foi implantado na AWS e utiliza 1 instância do EBS para executar a aplicação e uma instancia do RDS para a base de dados.

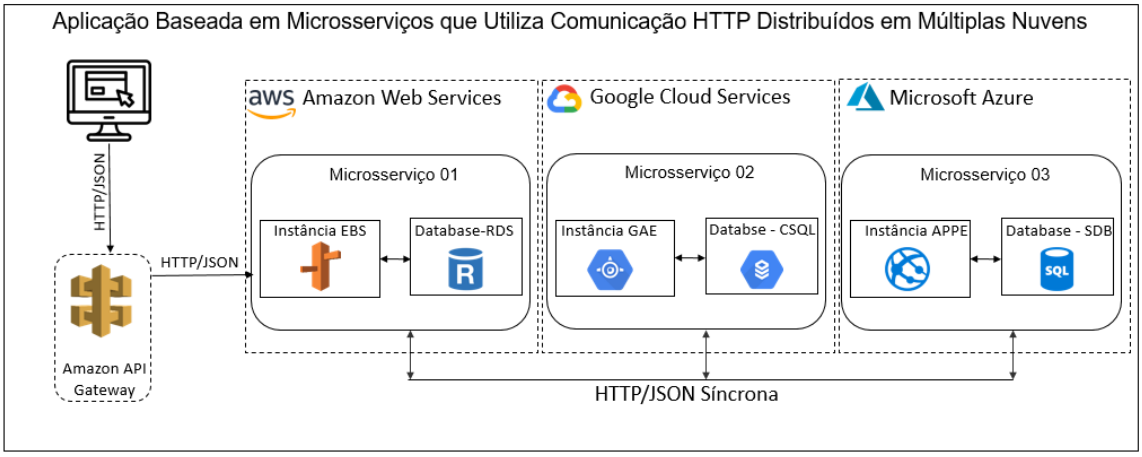

Figura 3 – Arquitetura da aplicação distribuída em múltiplas nuvens utilizando comunicação síncrona

O microsserviço 2 foi alocado no Google Cloud, onde foi criada uma instância da aplicação utilizando o Google App Engine (GAE) e uma instância para a base de dados utilizando o CloudSQL (CSQL). Já o terceiro microsserviço foi alocado na Azure, onde utiliza a ferramenta o APP Service (APPS) para executar a aplicação e o SQL Database para a instância da base de dados.

Como a comunicação síncrona é feita via REST, cada microsserviço conhece o endereço dos outros microsserviços a qual deseja realizar a comunicação, além de possuírem técnicas de *retry* para tratamento de falha de comunicação. É importante frisar que nesse caso já tem-se uma arquitetura onde a distancia geográfica entre os microsserviços são maiores se comparado aos cenários anteriores, mesmo que todas as instâncias estejam alocadas na região de São Paulo.

4.3.2 Arquitetura da Aplicação utilizando comunicação assíncrona

Assim como na arquitetura da Figura 2, essa arquitetura (Figura 4) também possui um *gateway* para a centralização das requisições vindas do cliente e também foi implantada em apenas uma nuvem (AWS), ela também utiliza instancias do EBS para o executar a aplicação e uma instancia RDS para a base de dados. e nesse caso, ao invés da aplicação conhecer o endereço dos outros microsserviços para obter a comunicação (nesse caso assíncrona), ele envia as mensagens apenas para uma instância da Amazon Simple Queue Service (SQS).

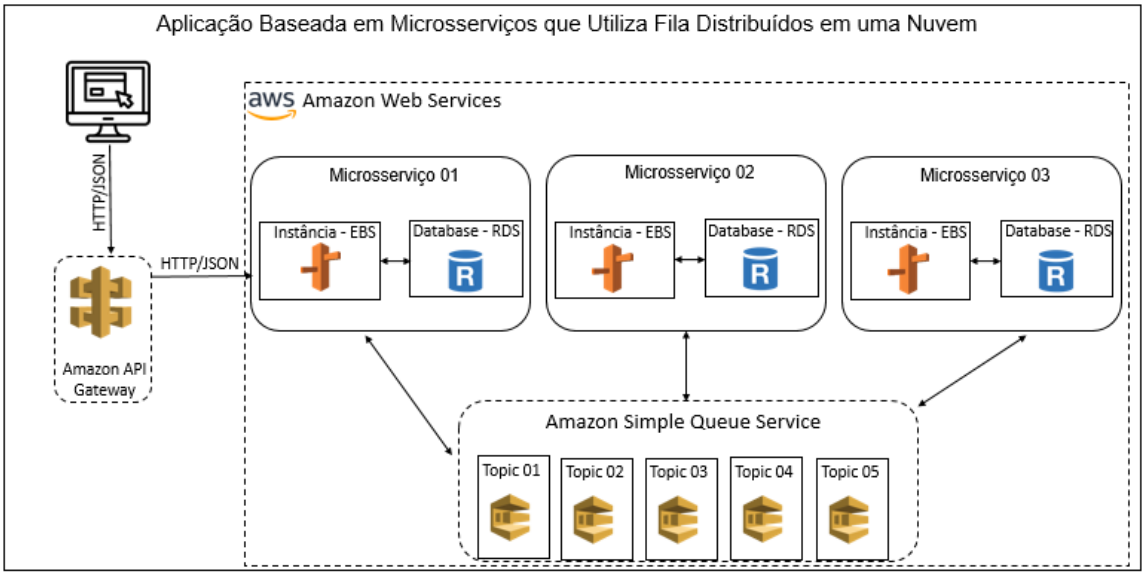

Figura 4 – Aplicação utilizando comunicação assíncrona

Nesse caso, quando a aplicação deseja acionar ou obter uma informação que está disponível em outro microsserviço, ela mandará uma mensagem para um tópico especifico no SQS. Assim, o microsserviço de destino será notificado e receberá a mensagem assim que possível, então ele fará o processamento necessário e mandará a resposta para um tópico especifico que será consumido pelo microsserviço que realizou a primeira requisição. Ao chegar a mensagem no microsserviço de origem, ele retornará a resposta ao cliente.

Aqui já pode-se observar o desacoplamento em relação a arquitetura anterior. O SQS garantirá que a mensagem seja consumida, caso ele tente entregar uma vez e a aplicação não esteja apta a receber a mensagem naquele momento, ele tentará novamente até receber a garantia que a aplicação de destino consumiu a mensagem.

Como mostrado na Figura 5, nesse cenário tem-se a aplicação distribuída em múltiplas nuvens e utilizando comunicação assíncrona. Sobre a distribuição dos microsserviços, ela leva a mesma configuração da arquitetura anterior (Figura 3), ou seja, o microsserviço 1 está alocado na AWS e utiliza EBS e o RDS para executar a aplicação e a base de dados respectivamente. O microsserviço 2 está alocado no Google Cloud, onde utiliza o GAE para executar a aplicação eo CSQL para a base de dados e o microsserviço 3 está

alocado na Azure, utilizando o APPS para executar a uma instancia da aplicação e o SQL Database (SDB) para a base de dados.

Para a comunicação foi utilizado o SQS para centralizar as mensagens, ou seja, tanto o serviço que está na AWS, quanto os que estão nos outros provedores irão consumir as mensagens nessa instância.

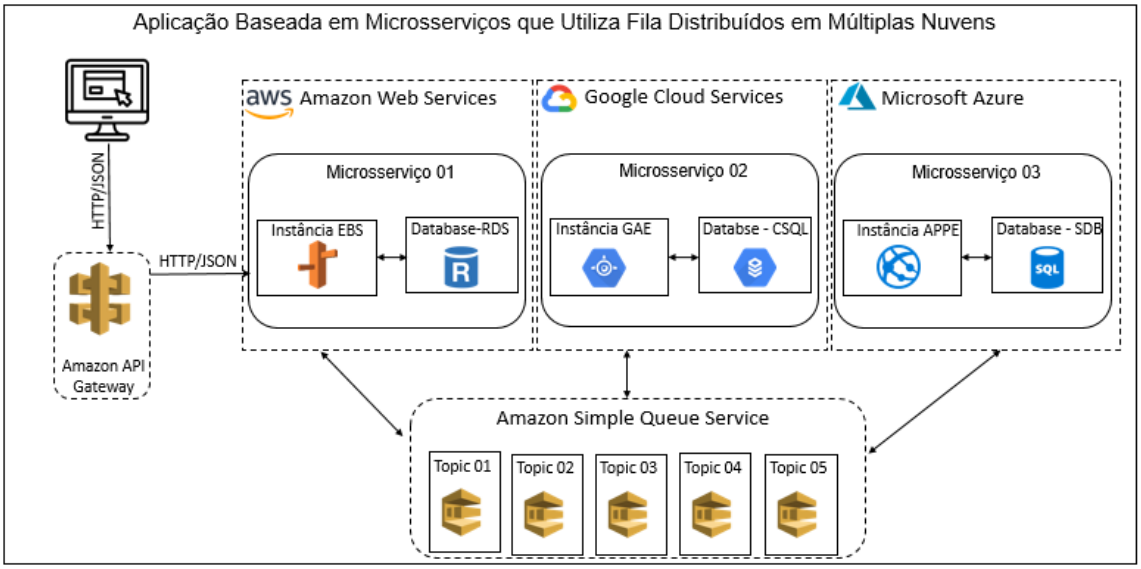

Figura 5 – Arquitetura da aplicação distribuída em múltiplas nuvens utilizando comunicação assíncrona

5 Experimentos e Análise dos Resultados

Com o intuito de descrever e analisar os experimentos e resultados obtidos nesse estudo, este capítulo apresenta na sessão na Seção 5.1 os cenários para avaliação do desempenho da aplicação e a Seção 5.2 apresenta os resultados obtidos com a realização dos experimentos. Além disso, a Seção 5.3 apresenta uma análise dos resultados levando em consideração cada tipo de comunicação e a Seção 5.4 dispõem de uma análise dos resultados comparando as formas de comunicação síncrona e assíncrona.

5.1 Descrição dos Cenários

Alguns cenários foram criados para avaliarmos o desempenho da aplicação *e-bank* tanto utilizando comunicação síncrona quanto usando assíncrona. Assim, três cenários foram criados para a comunicação síncrona: Cenário síncrono 01, Cenário síncrono 02 e Cenário síncrono 03. O Cenário síncrono 01 utiliza apenas 1 microsserviço para realizar a tarefa, o Cenário síncrono 02 utiliza 2 microsserviços, desta forma deve haver a comunicação entre eles e o Cenário síncrono 03 utiliza 3 microsserviços para realizar a tarefa tendo uma comunicação sequencial entre eles. Cenários com as mesmas características são criados para a comunicação assíncrona, cujo nome dos cenários são: Cenário assíncrono 01, Cenário assíncrono 02 e Cenário assíncrono 03.

Cada um dos experimentos foram realizados utilizando sub-cenários, nos quais varia-se o número de acessos simultâneos. No sub-cenário A, o número de acesso simultâneo é 0, para o sub-cenário B, o número de acessos simultâneos é 100 e o número de acessos simultâneos para o sub-cenário C é 1000. Como se pode observar na Tabela 3.

Tabela 3 – Cenários detalhados

Cenários	Single Cloud e Multi Cloud		
	Subcenário A	Subcenário B	Subcenário C
Síncrono 01	1 Microsserviço 0 acessos simultâneos	1 Microsserviço 100 acessos simultâneos	1 Microsserviço 1000 acessos simultâneos
Síncrono 02	2 Microsserviços 0 acessos simultâneos	2 Microsserviços 100 acessos simultâneos	2 Microsserviços 1000 acessos simultâneos
Síncrono 03	3 Microsserviços 0 acessos simultâneos	3 Microsserviços 100 acessos simultâneos	3 Microsserviços 1000 acessos simultâneos
Assíncrono 01	1 Microsserviço 0 acessos simultâneos	1 Microsserviço 100 acessos simultâneos	1 Microsserviço 1000 acessos simultâneos
Assíncrono 02	2 Microsserviços 0 acessos simultâneos	2 Microsserviços 100 acessos simultaneos	2 Microsserviços 1000 acessos simultâneos
Assíncrono 03	3 Microsserviços 0 acessos simultâneos	3 Microsserviços 100 acessos simultâneos	3 Microsserviços 1000 acessos simultâneos

5.2 Experimentos

Esta seção explana os experimentos. Os experimentos estão divididos em três seções gerais: a subseção 5.2.1 trata da aplicação baseada em microsserviços com comunicação síncrona, já a 5.2.2 trata da aplicação baseada em comunicação assíncrona.

5.2.1 Aplicação Baseada em Microsserviços com Comunicação síncrona

Esta subseção explanaremos os experimentos na aplicação que utiliza comunicação síncrona, levando em consideração os cenários descrito na seção anterior. A Tabela 4 apresenta os resultados para todos os subcenários e levanto em consideração as aplicações *single-cloud* e *multi-cloud*.

Tabela 4 – RTT (em ms) para os experimentos utilizando comunicação síncrona

Cenários	**Microsserviços**	**Single Cloud Subcenários**			**Multi-Cloud** Subcenários		
		A	**B**	**C**	**A**	**B**	**C**
		Acessos simultâneos			**Acessos simultâneos**		
		0	**100**	**1000**	**0**	**100**	**1000**
01	Cliente	153	391	757	156	394	758
02	Cliente e Produtos	161	410	787	193	431	799
03	Cliente, Produtos e Gateway	181	438	801	219	478	824

As Figuras 6 e 7 apresentam os gráficos dos resultados obtidos no experimento *single-cloud* e *multi-cloud* para todos os cenários. Pode-se notar uma leve diferença entre esses resultados levando em consideração o tempo médio em cada cenário. Uma análise de cada resultado é descrita nas próximas subseções.

5.2.1.1 Comunicação síncrona *single-cloud*

No Cenário 01 onde não existe comunicação interna entre microsserviços, o subcenário A precisou em média de 153 ms para retorna o resultado ao cliente. No subcenário B (onde existia 100 usuários simultâneos fazendo solicitações na aplicação), o tempo médio de resposta foi de 391 ms, um aumento de aproximadamente 155.6% se comparado com o subcenário A.

Esse primeiro experimento já trouxe resultados interessantes sobre o tempo de resposta da aplicação, onde se percebe claramente a necessidade de mais tempo e poder de processamento da aplicação a medida que os números de requisições aumentam. Isso fica mais claro quando se observa o subcenário C (1000 acessos simultâneos), onde o tempo médio de resposta foi de 757 ms. Um aumento de aproximadamente 394.7% em relação ao subcenário A e 93% em relação ao subcenário B.

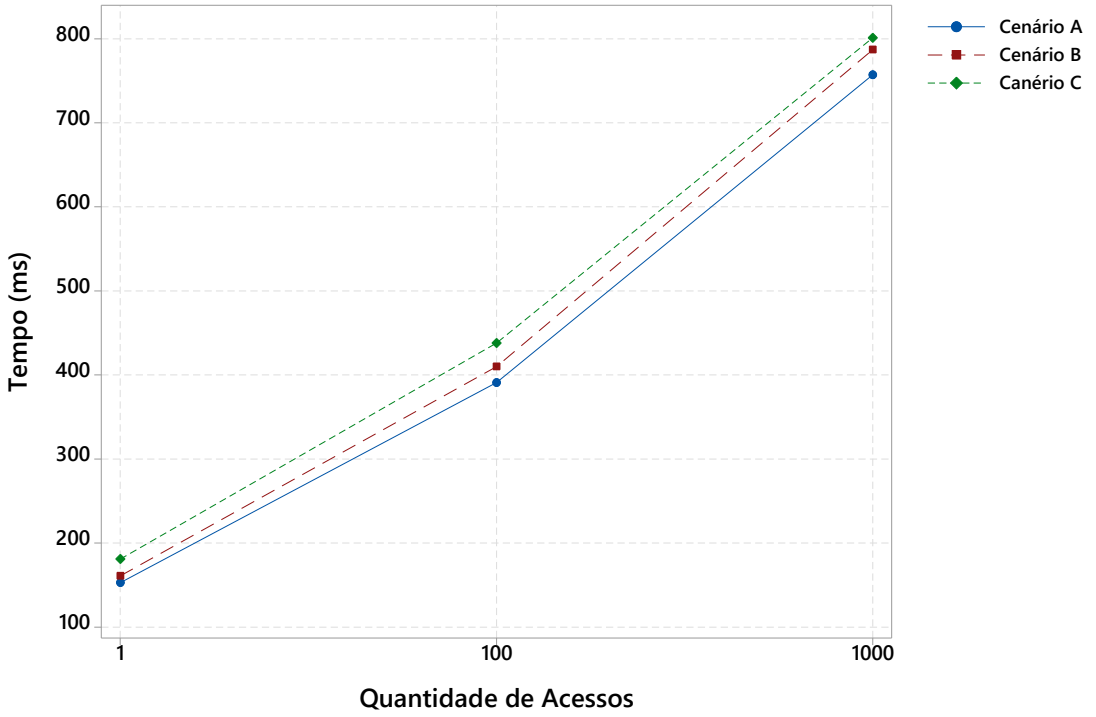

Figura 6 – Resultados para cenários usando comunicação síncrona em *Single-Cloud*

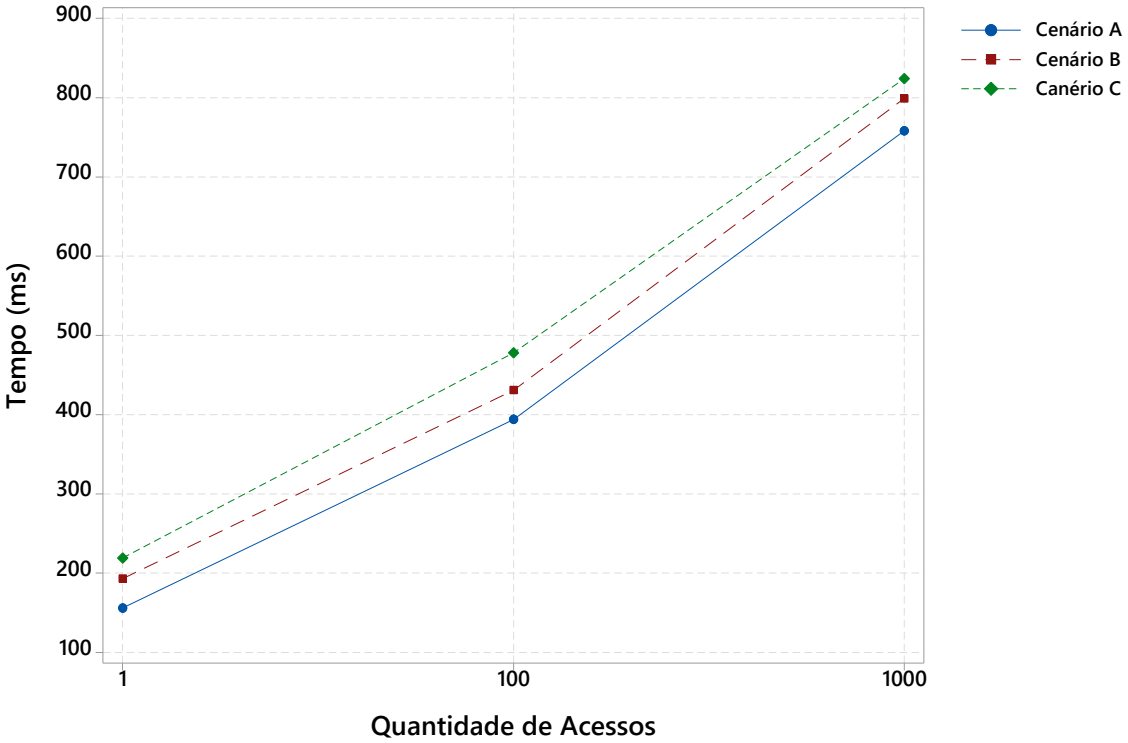

Figura 7 – Resultados dos 3 cenários para a aplicação síncrona em *Multi Cloud*

É notável que o tempo que a aplicação precisa para responder ao cliente não é proporcional ao número de acessos, ela age de forma diferente dependendo de quanto *stress*

(simulação carga de tráfego no sistema) ela está sofrendo, isso pode ocorrer por diversos fatores, dentre eles respostas já gradavas em cache, otimização por parte do compilador e até mesmo por otimização do servidor de aplicação (tomcat) para o gerenciamento das solicitação recebidas.

No cenário 02, tem-se a comunicação entre 02 microsserviços através do protocolo síncrono, no subcenário A, tem-se uma média de 161 ms para a resposta ao cliente. Quando se trata do subcenário B, a média para a resposta foi de 410 ms, uma diferença de aproximadamente 154.56% para resposta. E no subcenário C, a média de tempo obtido para o cliente foi de 757 ms, se comparado com o subcenário A, a diferença de tempo de resposta foi de aproximadamente 370.18% e com subcenário B a diferença foi menor, com 84.63% de diferença.

Ao contrário do cenário 01, nesse experimento além do tempo de rede, tempo de processamento de aplicação e processamento da base de dados, também tem-se o tempo de comunicação entre 2 microsserviços. E se comparamos com o cenário 01, podemos observar resultados bem interessantes quanto a diferença do tempo de resposta da aplicação. A Tabela 5 apresenta a comparação desses cenários.

Tabela 5 – Resultados comparativos entre Cenário 01 e 02

Comparação de Cenários 01 e 02	Tempo	Percentual
Subcenário A	8 ms	5.23%
Subcenário B	19 ms	4.86%
Subcenário C	30 ms	3.96%

Entre os cenário 1 e 2, a diferença no subcenário A foi de aproximadamente 5.23% ou 8 ms de diferença. O subcenário B, a diferença foi de 4.86% ou 19 ms. E no subcenário C, a diferença foi 3.96% de ou 30 ms. Estes resultados são bastantes interessantes, pois a diferença entre os cenários foi muito baixa em todos os subcenários. Obtendo o pior resultado (percentual) no subcenário C, e que mesmo assim foi bastante satisfatório.

No cenário 03 é necessário a comunicação entre os 03 microsserviços para retornar o resultado ao cliente. No subcenário A, o tempo médio de resposta foi de 181 ms. Já no subcenário B, a média de tempo foi de 438 ms, uma diferença média de aproximadamente 142% para o subcenário A. No Experimento com o subcenário C, a aplicação levou em média 801 ms para retornar a resposta ao cliente, aproximadamente 342.5% a mais de tempo que o subcenário A, e 82% que o subcenário B.

É importante destacar que o grande ponto nesse cenário é a comunicação entre todos os microsserviços, que mesmo o sistema de máquinas do Elastic Beanstalk da AWS estarem localizadas na mesma região e possivelmente na mesma rede, o tempo para comunicação entre os serviços tendem a permanecer linear. Para demonstrar isso, pode-se fazer uma comparação do cenário 03 com os experimentos dos outros cenários.

Ao comparar os subcenários A dos cenários 1 e 3 exposto na Tabela 6, tem-se uma

Tabela 6 – Comparação entre os cenários 01 e 03

Comparação de Cenários 01 e 03	Tempo	Percentual
Subcenário A	28 ms	18.3%
Subcenário B	47 ms	12%
Subcenário C	44 ms	5.81%

diferença de 18.3% ou 28 ms. Na comparação com do subcenário B, a diferença percentual foi ainda menor, com 12% com tempo de diferença média de 47 ms. E no subcenário C, tive-se uma diferença de 5.81% ou 44 ms de tempo médio, uma diferença uma diferença média de tempo um pouco menor, se comparamos com o subcenário B, esse fato está ligado mais a capacidade de processamento dividida entre os microsserviços que ao tempo de comunicação.

Na comparação entre os cenário 2 e 3 exposto na Tabela 7, o subcenário A teve uma diferença de 12% ou 20 ms entre os cenários. Na comparação entre os subcenários B, a diferença percentual foi 4.86%, uma diferença percentual bem menor se comparado com o subcenário A, porém, em tempo médio a diferença foi um pouco maior: 28 ms. No ultimo subcenário, a diferença percentual foi de aproximadamente 1.78% ou 14 ms. Nesse caso, obtemos valor percentual bem expressivo, onde fica destacado que em cenários onde há comunicação entre os serviços, a diferença de tempo pode ser bastante baixa.

Tabela 7 – Comparação entre os cenários 02 e 03

Comparação de Cenários 02 e 03	Tempo	Percentual
Subcenário A	20 ms	12%
Subcenário B	28 ms	4.86 %
Subcenário C	14 ms	1.78 %

5.2.1.2 Comunicação síncrona *multi-cloud*

O cenário 01 não requer comunicação entre os microsserviços, então neste caso, os experimentos são realizados em apenas 01 provedor de nuvem, algo muito parecido com os testes do cenário 01 da aplicação single-cloud. Mesmo assim, ele tem bastante importância quando comparado com os outros cenários.

Analisando o subcenário A, tem-se uma média de 156 ms para obter resposta do servidor. No subcenário B, esse tempo é de 394 segundos, ou 152.56% de tempo a mais que o subcenário anterior. Já no subcenário C, a aplicação levou em média 758 ms para responder, sendo aproximadamente 385.9% ms de tempo a mais que o subcenário A e 92.38% ou de tempo se comparado com o subcenário B.

No cenário 02, onde é necessário uma comunicação externa entre os microsserviços, ou seja, aqui se dá o primeiro cenário onde é necessário a comunicação entre serviços que estão alocado em provedores diferentes.

No subcenário A, teve-se uma média de resposta de 193 ms. Nos experimentos do subcenário B, foi necessário em média 431 ms para obter retorno do servidor, se comparado com o subcenário A, teve-se uma diferença de aproximadamente 123.32%. Já no subcenário C, a média de tempo foi de 799 ms, aproximadamente 313.99% a mais de tempo que no subcenário A, mas quando comparado com o subcenário B, a diferença cai para aproximadamente 85.38%.

Após verificar os resultados obtidos neste cenário, pode-se fazer uma breve análise comparativa com os resultados obtidos nos experimentos do cenário 01 e que estão destacados na Tabela 8. Ao verificar os resultados do subcenário A, a diferença entre ambos cenários foram de aproximadamente 23.72% ou 37 ms em tempo de resposta.

Tabela 8 – Comparação entre os cenários 01 e 02

Comparação de Cenários 01 e 02	Tempo	Percentual
Subcenário A	37 ms	23.72%
Subcenário B	37 ms	9.39%
Subcenário C	41 ms	5.91%

Ao fazer a mesma comparação no subcenário B pode-se perceber resultados bastante interessantes, onde diferença percentual foi menor, tendo uma diferença 9.39% de tempo entre os subcenários ou 37 ms. A diferença de tempo entre os resultados são exatamente o mesmo (37 ms) se comparado com o subcenário A. Com isso, podemos pressupor que a diferença de tempo se deu apenas no tempo de processamento da aplicação e que tivemos um tempo de rede igual entre os subcenários.

Na comparação realizada entre os resultados no cenários 1 e 2 no subcenário C, podemos observar que a diferença de tempo entre os cenários foi de aproximadamente 5.91% ou 41 ms. Essa diferença de tempo tem um valor muito próximo a comparação entre os resultados feitos nos parágrafos anteriores (37ms), onde mais uma vez pode-se destacar que o tempo da comunicação entre esses serviços continuam similares.

No cenário 03 a aplicação realiza comunicação entre 03 microsserviços, ou seja, ocorre a comunicação entre 03 provedores de nuvens diferentes antes de retornar o resultado final ao cliente.

No subcenário A teve-se uma média de tempo de 219 ms. No subcenário B a média é de 478, uma diferença percentual de aproximadamente 118.26% em comparação com o subcenário A. No subcenário C, o tempo médio para resposta foi de 824 ms, aproximadamente 276.25% a mais de tempo que no subcenário A, e 72.38% de tempo quando comparado com o subcenário B.

Na comparação realizada entre os resultados obtidos para os cenários 01 e 03 (Tabela 9), pode-se observar que no subcenário A teve uma diferença percentual de aproximadamente 40.38% ou 56 ms de tempo a mais para obter resposta do servidor. Comparando

com o subcenário B a diferença percentual foi de 21.31% ou 84 ms, no subcenário c, a diferença foi de apenas 8.70%, ou 66 ms.

Tabela 9 – Comparação entre os cenários 01 e 03

Comparação de Cenários 01 e 03	Tempo	Percentual
Subcenário A	63 ms	40.38%
Subcenário B	84 ms	21.31%
Subcenário C	66 ms	8.70%

A Tabela 10 apresenta uma comparação entre os resultados obtidos nos cenários 02 e 03, observando o subcenário A, pode-se notar que a diferença percentual entre os cenários foi de 13.47% ou 26 ms em média, ao fazermos a comparação no subcenário B, a diferença de tempo médio entre os cenários foi de 47 ms ms ou de 10.90% em comparação entre os cenários. Já no ultimo subcenário, tivemos uma leve diferença de 3.13% entre os tempos ou 25 ms no tempo médio de resposta.

Tabela 10 – Comparação entre os cenários 02 e 03

Comparação de Cenários 02 e 03	Tempo	Percentual
Subcenário A	26 ms	13.47%
Subcenário B	47 ms	10.90 %
Subcenário C	25 ms	3.13 %

5.2.2 Aplicação Baseada em Microsserviços com comunicação Assíncrona

Esta subseção elucida os resultados obtidos na aplicação que utilizava comunicação assíncrona. Assim, como na comunicação síncrona, foram realizados experimentos para os três cenários, tanto para aplicação em uma única nuvem, quanto para a aplicação baseada em *multi-cloud*. A Tabela 11 apresenta o resultado resumido para todos os cenários dos experimentos realizados. As Figuras 8 e 9 apresentam os gráficos dos resultados obtidos no experimentos.

Tabela 11 – RTT (em ms) para os experimentos utiizando comunicação assíncrona

Cenários	**Microsserviços**	**Single Cloud Subcenários**			**Multi-Cloud** Subcenários		
		A	**B**	**C**	**A**	**B**	**C**
		Acessos simultâneos			**Acessos simultâneos**		
		0	**100**	**1000**	**0**	**100**	**1000**
01	Cliente	154	392	738	158	391	757
02	Cliente e Produtos	160	408	751	174	422	776
03	Cliente, Produtos e Gateway	184	445	774	212	450	802

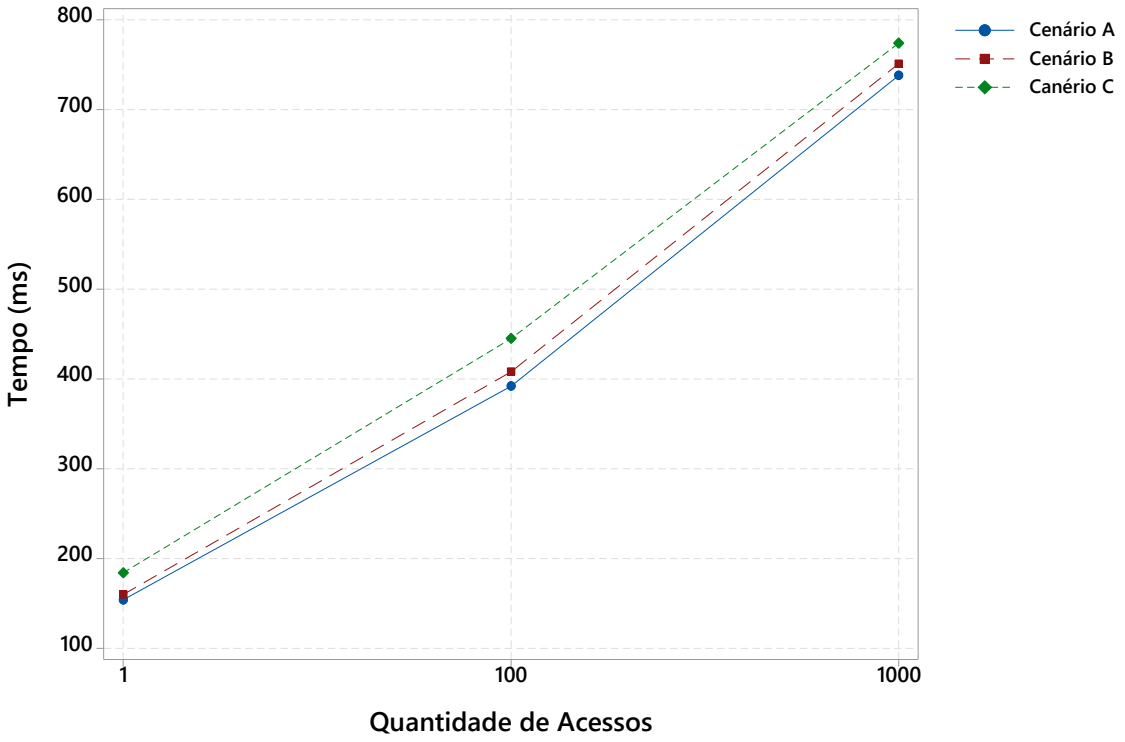

Figura 8 – Resultados para cenários assíncronos em *multi-cloud*

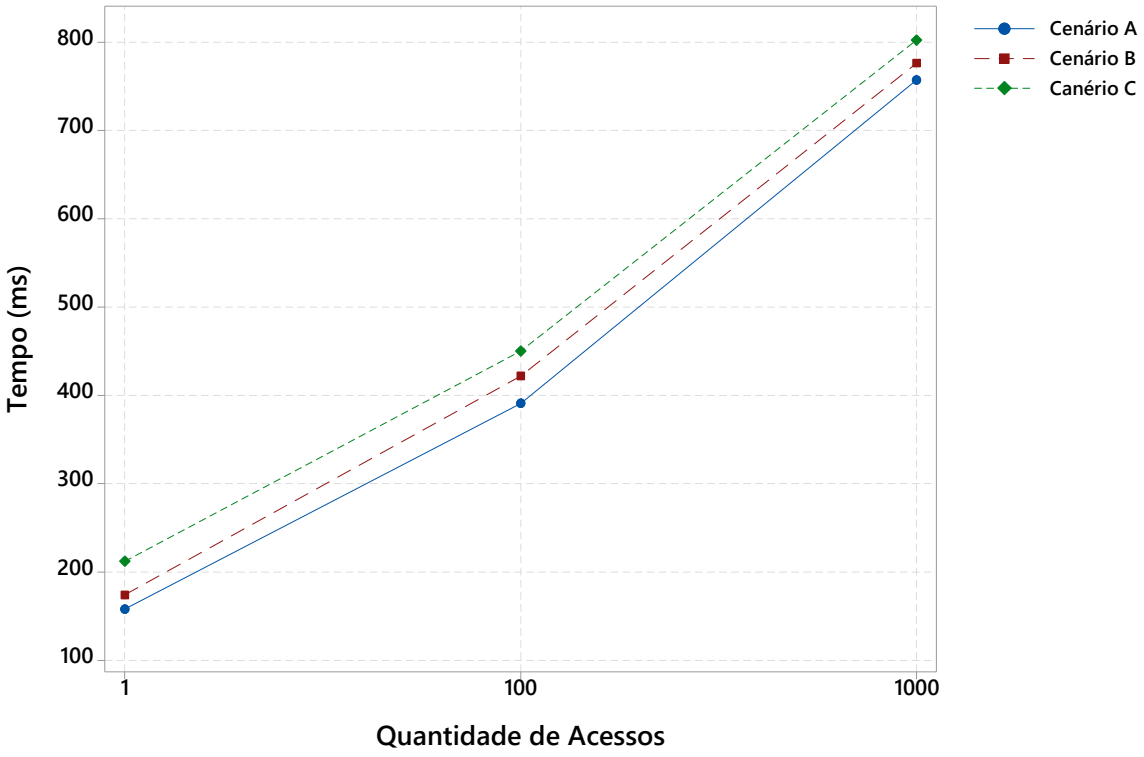

Figura 9 – Resultados para cenários assíncronos *Single-Cloud*

5.2.2.1 Comunicação Assíncrona *single-cloud*

Neste cenário, onde não há comunicação entre os serviços, sendo assim, a aplicação não possui nenhuma comunicação com o servidor de comunicação assíncrono para obter algum resultado. No subcenário A, quando a aplicação havia 1 cliente simultâneo foram necessários 154 ms para retornar uma resposta ao cliente. No subcenário B, quando havia 100 acessos simultâneos, o tempo médio de resposta foi de 392 ms, uma diferença de aproximadamente 154.55% se comparado com o subcenário A.

Já no subcenário C, onde teve-se 1000 acessos simultâneos, foram necessários em média 738 ms para retornar resposta ao cliente, ou seja, uma diferença de aproximadamente 379.22% se comparado com o subcenário A, e 392% ao ser comparado com o subcenário B.

No cenário 2, onde tem-se a comunicação entre 2 microsserviços, o subcenário A levou em média 160 ms para responder. O subcenário B, levou em média 408 ms para responder, uma diferença de 155% se comparado com o subcenário A. Já no subcenário C, a aplicação levou em média 774 ms para retornar uma resposta, levando 388% de tempo a mais se comparado com o subcenário A, e aproximadamente 89.70% a mais que o subcenário B.

Tabela 12 – Comparação entre os cenários 01 e 02

Comparação de Cenários 01 e 02	Tempo	Percentual
Subcenário A	6 ms	3.90%
Subcenário B	16 ms	4.08%
Subcenário C	13 ms	1.76%

A Tabela 12 apresenta uma comparação entre os cenários 01 e 02 em relação ao experimento 01 utilizando comunicação assíncrona, onde tempos uma diferença percentual de 3.90% ou 6 ms no subcenário A. 16 ms ou 4.08% de diferença no tempo médio no subcenário B. Já no subcenário C, a diferença foi ainda menor, onde teve-se 1.76% ou 13 ms de diferença no tempo médio de reposta entre esses cenários.

No cenário 3, tem-se a comunicação entre os 3 microsserviços da aplicação. A aplicação no subcenário A levou em média 184 ms para responder, já quando testado o subcenário B, a aplicação levou em média 455 ms para retornar uma resposta, uma diferença de aproximadamente 147.28% se comparado com o subcenário A. 744 ms foi o tempo médio que a aplicação levou para responder no subcenário C, isso equivale aproximadamente 320.65% a mais de tempo se comparado com o subcenário A, e aproximadamente 70.10% a mais de tempo quando comparado com o subcenário B.

Na Tabela 13 apresenta uma comparação entre os resultados obtidos nos cenários 01 e 03 no ambiente *single-cloud* utilizando comunicação assíncrona. Ao fazer a comparação do subcenário A, a diferença média para o tempo de resposta foi de 30 ms, ou 19.48%. No subcenário B, a diferença foi de 53 ms, sendo equivalente a 13.52% do tempo percentual.

Tabela 13 – Comparação entre os cenários 01 e 03

Comparação de Cenários 01 e 03	Tempo	Percentual
Subcenário A	30 ms	19.48%
Subcenário B	53 ms	13.52%
Subcenário C	36 ms	4.87%

Partindo para o último subcenário, a diferença foi de 4.87%, ou 36 ms de tempo a mais para a aplicação responder.

Tabela 14 – Comparação entre os cenários 02 e 03

Comparação de Cenários 02 e 03	Tempo	Percentual
Subcenário A	24 ms	15%
Subcenário B	37 ms	9.07%
Subcenário C	23 ms	3.06%

A Tabela 14 apresenta uma comparação entre os resultados obtidos nos cenários 02 e 03, onde no subcenário A, teve a diferença média de tempo de 24 ms ou 15% do valor percentual. No subcenário B, a diferença foi de 37 ms no tempo médio para resposta, isso equivale a 9.07% na diferença de tempo. No subcenário C, essa diferença cai para 3.06%, sendo em média 23 ms para obter resposta.

5.2.2.2 Comunicação Assíncrona *multi-cloud*

No cenário 01, não se tem comunicação entre os microsserviços, no subcenário A, a aplicação levou em média 158 ms para obter resposta, no subcenário B, o tempo médio de resposta aumentou para 391 ms, uma diferença de 147.46% para o subcenário A. No subcenário C, o tempo de resposta média foi de 757 ms, uma diferença equivalente a 379.11% se comparado com o subcenário A e aproximadamente 93.61% se comparado com o subcenário B.

No cenário 02, tem-se a comunicação entre dois microsserviços, sendo que esses estão em provedores diferentes. No subcenário A, a aplicação levou em média 174 ms para responder. No subcenário B, o tempo médio de resposta foi de 422 ms, uma diferença de 142.52% ao se comparar com o resultado obtido no subcenário B. Já no subcenário C, a aplicação demorou em média 776 ms para retornar uma resposta ao cliente. Uma diferença percentual de tempo de aproximadamente 345.98% se comparado com o subcenário A e 75.57% com o subcenário B.

Tabela 15 – Comparação entre os cenários 01 e 02

Comparação de Cenários 01 e 02	Tempo	Percentual
Subcenário A	16 ms	10.13%
Subcenário B	31 ms	13.04%
Subcenário C	19 ms	2.51%

A Tabela 15 apresenta um comparativo entre os resultados obtidos nos cenários 01 e 02, no subcenário A, teve-se uma diferença média de tempo de 16 ms, ou 10.13% percentual. Já na comparação do subcenário B, o tempo de diferença média foi de 31 ms ou 13.04%. No último subcenário teve-se uma média diferencial de tempo de 19 ms, percentualmente sendo equivalente a aproximadamente 2.51%.

No cenário 03, tem-se a comunicação entre todos os microsserviços, serviços esses que estão em nuvens diferentes. No subcenário A, a aplicação levou em média 212 ms para responder, já no subcenário B, a aplicação levou em média 450 ms para responder, aproximadamente 112.26% na média de tempo a mais que o subcenário anterior. No subcenário C, a média de tempo de resposta foi de 802 ms, 278.30% de tempo a mais que o subcenário A e 78.22% a mais que o subcenário B.

Tabela 16 – Comparação entre os cenários 01 e 03

Comparação de Cenários 01 e 03	Tempo	Percentual
Subcenário A	54 ms	34.17%
Subcenário B	59 ms	15.09%
Subcenário C	45 ms	5.94%

Observando a comparação entre os resultados obtidos nos cenários 01 e 03 dispostos na Tabela 16, pode-se notar que no subcenário A, a diferença média de tempo entre os cenários 01 e 03 foi de 54 ms ou 34.17%. No subcenário B, a diferença percentual de tempo médio cai para 15.09%, ou 59 ms. No subcenário C, essa diferença foi de 45 ms, sendo equivalente a 5.94% do tempo médio de resposta.

Tabela 17 – Comparação entre os cenários 02 e 03

Comparação de Cenários 02 e 03	Tempo	Percentual
Subcenário A	38 ms	21.85%
Subcenário B	28 ms	6.63%
Subcenário C	26 ms	3.35%

A Tabela 17 apresenta um comparativo entre os resultados obtidos nos cenário 02 e 03, onde no subcenário A, a diferença do tempo médio de resposta entre os cenários foi de 38 ms, ou 21.85%, no subcenário B, essa diferença de tempo foi de 28 ms, ou 6.63%. No último subcenário, a diferença foi de 26 ms, ou 3.35%.

5.3 Análise dos Resultados com mesmo tipo de comunicação

De acordo com os resultados obtidos na sessão 5.2, esta seção faz uma análise sobre as aplicações levando em consideração apenas os ambientes em que elas estão alocadas, ou seja, é uma análise individual de cada tipo de comunicação, fazendo uma análise de como elas se comportaram estando distribuídos ou não em múltiplas nuvens. Na Subseção 5.3.1

apresenta uma análise sobre a aplicação com a comunicação síncrona e a Subseção 5.3.2 apresenta uma análise sobre a aplicação utilizando comunicação assíncrona.

5.3.1 Análise de resultados com comunicação Síncrona

Esta subseção faz um comparativo entre os resultados obtidos em todos os cenários, levando em consideração a aplicação síncrona alocada em uma ou em múltiplas nuvens. A seguir faremos a análise desses cenários.

A Tabela 18 apresenta uma comparação entre os tipos de distribuição da aplicação síncrona. É notável que a diferença de tempo de resposta das aplicações é bastante baixa, e isso era o esperado, pois a aplicação usa apenas 1 microsserviço, e tanto a aplicação alocada em múltiplas nuvens quanto a aplicação baseada em uma única nuvem estão alocadas na AWS.

Tabela 18 – Comparação do cenário A para aplicação síncrona

Subcenários	Single Cloud	Multi Cloud	Diferença em ms	Diferença Percentual
Subcenário A	153	156	3	1.96%
Subcenário B	391	394	3	0.76%
Subcenário C	757	758	1	0.13%

Ainda de acordo com a Tabela 18, a diferença miníma acontece pois as aplicação são parecidas. E essa diferença miníma ocorre apenas por uma variação miníma no tempo de rede, que nos subcenários teve uma variação de 3 ms (subcenário A e B) a 1 ms (subcenário A), um tempo de variação que dificilmente seria perceptível pelo usuário final.

A Tabela 19 apresenta uma comparação apresenta dos resultados obtidos no cenário 2, onde existe a comunicação entre 2 microsserviços, e ao contrário do cenário anterior, pode-se destacar algumas diferenças de tempo entre as aplicações. Um exemplo disso é o subcenário A, quando a aplicação tem apenas 1 acesso simultâneo na aplicação, a diferença percentual média foi de 19.88% (ou 32 ms), uma diferença de tempo maior que nos outros subcenários.

Tabela 19 – Comparação do cenário B para a aplicação síncrona

Subcenários	Single Cloud	Multi Cloud	Diferença em ms	Diferença Percentual
Subcenário A	161	193	32	19.88%
Subcenário B	410	431	21	5.12%
Subcenário C	787	799	12	1.52%

De acordo com Carrera et al. (2003) uma explicação para esse fato é a forma como o nosso servidor de aplicação (no nosso caso, o utilizado foi o Tomcat) lida com as requisições, é claro que a medida com que a quantidade de acessos aumentam, o esforço da aplicação como um todo tendem a aumentar, porém, em momentos em que a aplicação não

tem uma grande quantidade de acessos a tendencia é entrar em um "estado de espera"até que a memória seja liberada para realizar o processamento de novas requisições.

Em termos gerais, tem-se uma baixa diferença de tempo entre os serviços, mesmo no pior caso (32 ms), dependendo do tipo de aplicação, pode ser uma diferença bastante aceitável tendo em vista a utilização de múltiplas nuvens, e ficou mais interessante a medida que os números de acessos aumentaram.

No cenário 03 tem-se uma comunicação entre todos os serviços usando comunicação síncrona, a Tabela 20 apresenta a comparação entre os resultados para esse tipo de comunicação. Um resultado interessante é no subcenário B, onde tendo 100 acessos simultâneos, a diferença média ente as aplicações distribuídas ou não em múltiplas nuvens foi de apenas 5 ms. Nos outros subcenários, a diferença média foi de 28 ms.

Tabela 20 – Comparação do cenário C da aplicação síncrona distribuída ou não em múltiplas nuvens

Subcenários	Single Cloud	Multi Cloud	Diferença em ms	Diferença Percentual
Subcenário A	184	212	28	15.22%
Subcenário B	445	450	5	1.12%
Subcenário C	774	802	28	4.91%

Novamente pode-se notar que a diferença de tempo entre o tipo de distribuição não foi tão considerável, mesmo no cenário onde se tem 1000 acessos simultâneos.

5.3.2 Análise de Resultados com comunicação Assíncrona

Da mesma forma que a subseção passada, esta subseção fará uma análise dos cenários, porém, desta vez, trataremos da aplicação baseada em comunicação assíncrona, estando distribuída ou não em múltiplas nuvens.Nas próximas subseções faremos uma análise de cada cenário.

A Tabela 21 apresenta um comparativo entre a aplicação baseada em comunicação assíncrona quando está distribuída ou não em múltiplas nuvens. Neste cenário (cenário 01), a aplicação não se comunica com outros microsserviços, então não é esperado grande diferenças entre os tipos de distribuição, sendo assim, olhando para o cenário onde tivemos a maior diferença de tempo (subcenário C), uma diferença de 19 ms, é aceitável, tendo em vista as variações de tempo sofrida pela rede.

Tabela 21 – Comparação do cenário A da aplicação assíncrona distribuída ou não em múltiplas nuvens

Subcenários	Single Cloud	Multi Cloud	Diferença em ms	Diferença Percentual
Subcenário A	154	158	4	2.59%
Subcenário B	392	391	1	0.26%
Subcenário C	738	757	19	2.57%

Nesse canário temos a comunicação entre dois microsserviços assíncronos (cenário 02). A Tabela 22 apresenta um comparativo entre as aplicações baseada ou não em múltiplas

nuvens. Inicialmente é notável a baixa variação entre a diferença dos valores levando em consideração os subcenários. O subcenário com a maior diferença percentual foi o subcenário C, onde temos uma diferença de 25 ms no tempo médio de resposta. Novamente vemos o ambiente *multi-cloud* precisando de um pouco mais de tempo para responder, porém, mais uma vez esse tempo não foi tão considerável (25 ms).

Tabela 22 – Comparação do cenário B para a aplicação assíncrona

Subcenários	Single Cloud	Multi Cloud	Diferença em ms	Diferença Percentual
Subcenário A	160	174	14	8.75%
Subcenário B	408	422	14	3.43%
Subcenário C	751	776	25	3.33%

A Tabela 23 apresenta um comparativo no cenário onde a aplicação baseada ou não em múltiplas nuvens no momento em que fazia comunicação entre 3 microsserviços (cenário 03). Mais uma vez não observou-se uma grande diferença entre os resultados obtidos das aplicações, sendo que os piores cenários foram aqueles onde levamos em consideração a diferença em ms, o subcenário A com 28 ms, seguido do cenário C onde a diferença média foi de 23 ms. O subcenário B teve uma diferença bem interessante, de apenas 5 ms entre os tipos de distribuição.

Tabela 23 – Comparação do cenário C para aplicação baseada em comunicação assíncrona

Subcenários	Single Cloud	Multi Cloud	Diferença em ms	Diferença Percentual
Subcenário A	184	212	28	15.21%
Subcenário B	445	450	5	1.12%
Subcenário C	801	824	23	2.87%

5.4 Análise dos Resultados entre os tipos de comunicação

Nessa seção faz-se uma análise comparativa entre os resultados obtidos para os dois tipos de comunicação síncrono e assíncrono, levando em consideração cada experimento executado, estando distribuídos ou não em múltiplas nuvens.

5.4.1 Aplicações Single Cloud

A Tabela 24 apresenta uma comparação entre os resultados obtidos do cenário 01 levando em consideração os dois tipos de aplicação, tendo em vista que a aplicação não realiza comunicações entre microsserviços, a diferença de tempo baixa entre eles eram esperados.

No cenário 02, tem-se a comunicação entre dois microsserviços, os resultados foram bastante interessantes (Tabela 25), no subcenário A, a diferença foi de apenas 1 ms, um valor bastante aceitável. No subcenário B também teve-se uma pequena diferença (2 ms),

Tabela 24 – Comparação do cenário 01 entre as comunicações síncrona e assíncrona

Subcenários	Síncrono	Assíncrono	Diferença em ms	Diferença percentual
Subcenário A	153	154	1	0.65%
Subcenário B	391	392	1	0.26%
Subcenário C	757	738	19	2.50%

mas nesse caso, a aplicação assíncrona teve um desempenho melhor. No subcenário C, onde se tem 1000 acesso simultâneos, a diferença média no tempo de resposta foi de 36 ms, e a aplicação baseada assíncrona novamente teve uma resultado melhor.

Tabela 25 – Comparação do cenário 02 entre as comunicações síncrona e assíncrona

Subcenários	Síncrono	Assíncrono	Diferença em ms	Diferença percentual
Subcenário A	161	160	1	0.62%
Subcenário B	410	408	2	0.49%
Subcenário C	787	751	36	4.57%

Esse resultado é bastante interessante, pois mostra que mesmo trabalhando de forma assíncrona, obteve-se um resultado bastante satisfatório no momento em que a aplicação sofreu um estresse maior. No momento em que a aplicação está sofrendo bastante estresse o tempo de processamento tende a aumentar e o fato da comunicação assíncrona aguardar uma disponibilidade aceitável da Aplicação para mandar para o processamento pode ter sido um dos motivos para ter esse resultado, ao comparar a aplicação síncrona.

A Tabela 26 faz uma comparação entre os resultados obtidos para o cenário 03. Novamente tem-se uma pequena diferença entre os valores, sendo que no subcenário 01 essa diferença foi de apenas 3 ms, tendo a aplicação síncrona com o melhor desempenho fato que também aconteceu no subcenário B, onde a diferença foi de 7 ms. No último subcenário a diferença foi de 27 ms, onde a aplicação assíncrona teve o melhor resultado.

Tabela 26 – Comparação do cenário 03 entre as comunicações síncrona e assíncrona

Subcenários	Síncrono	Assíncrono	Diferença em ms	Diferença percentual
Subcenário A	181	184	3	1.66%
Subcenário B	438	445	7	1.60%
Subcenário C	801	774	27	3.37%

Novamente, pode-se observar que a aplicação baseada em comunicação assíncrona teve o melhor desempenho quando a aplicação foi mais estressada, a qual leva ao mesmo ponto do cenário anterior, onde utilizou-se o processamento por demanda disponibilizado pela tecnologia utilizado na comunicação assíncrona. No outro subcenário a diferença foi pequena, sendo uma diferença quase desprezível.

Por fim, uma comparação é realizada entre os resultados *Single-Cloud* envolvendo os dois tipos de comunicação, podemos observar que a diferença entre média entre eles foi muito pequena, sendo o pior caso no momento em que a diferença foi de 36 ms. No

geral, teve-se a comunicação síncrona tendo desempenho melhor quando tinha 1 e 100 acessos, porém, a medida que o número de acessos foi aumentando o tempo de resposta da aplicação assíncrona melhorou em relação a aplicação síncrona. A primeira vista, isso mostra o quanto a aplicação assíncrona é estável ao numero de acessos em relação a aplicação síncrona.

5.4.2 Aplicações *Multi-Cloud*

A Tabela 27 apresenta uma comparação entre os resultados do Cenário 01 dos dois tipos de comunicação quando a aplicação está alocada em *multi-cloud*. É possível observar a pequena diferença de tempo entre as aplicações, e novamente era um resultado esperado, pois não havia comunicação entre os microsserviços.

Tabela 27 – Comparação do cenário 01 Multi Cloud entre as comunicações síncrona e assíncrona

Subcenários	Síncrono	Assíncrono	Diferença em ms	Diferença percentual
Subcenário A	156	158	2	1.28%
Subcenário B	394	391	3	0.76%
Subcenário C	758	757	1	0.13%

A Tabela 28 apresenta um comparativo entre os resultados obtidos para o cenário 02, onde as aplicações fazem comunicação com dois microsserviços em ambiente *multi-cloud*. Nesse cenário, a aplicação baseada em comunicação assíncrona teve melhor desempenho em todos os cenários, tendo maior destaque no subcenário A, onde teve uma diferença percentual bastante interessante (9.84%).

Com esses resultados, pode-se analisar que em momentos que os microsserviços necessitam fazer comunicações com outras nuvens, a aplicação síncrona foi bem mais custoso, seja pelo fato da comunicação ser feita por HTTPs, onde é necessário alguns passos para realizar a comunicação, ou até mesmo pela localização centralizada de comunicação fornecida pela comunicação assíncrona.

Tabela 28 – Comparação do cenário 02 Multi Cloud entre as comunicações síncrona e assíncrona

Subcenários	Síncrono	Assíncrono	Diferença em ms	Diferença percentual
Subcenário A	193	174	19	9.84%
Subcenário B	431	422	9	2.09%
Subcenário C	799	776	23	2.88%

Observando os resultados da Tabela 29, no cenário 03, onde foi realizada a comunicação entre os 3 microsserviços, a aplicação assíncrona novamente teve um melhor desempenho, destacando-se o subcenário B, onde essa diferença foi de 28 ms. Como em todos os outros cenários, mais uma vez não tivemos grandes diferenças entre os resultados nos subcenários.

Tabela 29 – Comparação do cenário 03 Multi Cloud entre as comunicações síncrona e assíncrona

Subcenários	Síncrono	Assíncrono	Diferença em ms	Diferença percentual
Subcenário A	219	212	7	3.19%
Subcenário B	478	450	28	5.86%
Subcenário C	824	802	22	2.67%

Neste contexto, pode-se dizer que dificilmente um cliente final notaria uma diferença de 28 ms por exemplo, mas claro, isso serve para o nosso cenário, em cenário em que temos um numero maior de microsserviços, um número maior de usuário simultâneos e até mesmo mais dados no banco de dados, esse tempo poderá aumentar.

Quando comparado com o ambiente *single-cloud*, em *multi-cloud* a diferença média de tempo foi mais disperso e a aplicação assíncrona também teve melhores resultados se comparado a aplicação síncrona. Esse é um resultado bastante interessante, pois além de um desempenho linear a medida com que os número foi aumentando, a aplicação assíncrona apresentou resultados melhores mesmo em situações onde tem 1 ou 100 requisição.

É evidente que a diferença entre eles foi pequena, na qual se teve no pior caso 28 ms de diferença. Porém, observa-se novamente que utilizar uma arquitetura assíncrona como centralizador para as mensagens se torna mais robusto a medida que o número de requisições aumentam, e por não precisar fazer re-tentativas de requisições em situações em que a instância está ocupada, a comunicação assíncrona se tona mais assertiva.

6 Conclusão

A Internet está no centro de uma revolução, onde a computação em nuvem tem recursos globalmente conectados e facilmente compartilhados, facilitando o surgimento de diversos novos provedores de nuvem. Esse fato abre espaço para diversas novas soluções tecnológicas, dentre eles a distribuição de *software* em ambiente *multi-cloud*.

Uma das formas utilizadas para usufruir dos benefícios de ambientes *multi-cloud* é utilizando a arquitetura de microsserviços no desenvolvimento de *software*. Porém, esses serviços apesar de serem especializados, dependendo da necessidade, é preciso estabelecer comunicação entre si e apesar de existirem diversas formas de comunicação síncronas e assíncronas, mas pouco se sabe sobre o desempenho desses tipos de comunicação quando distribuído em ambiente *multi-cloud*.

Com o intuito de descobrir qual tipo de comunicação tem melhor desempenho em ambientes *cloud*, esse trabalho teve como objetivo construir duas versões de um mesmo *software*, um utilizando comunicação síncrona para comunicação e outro utilizando comunicação assíncrona, para realizar experimentos de funcionalidades e efetuar comparações entre eles e assim contribuir com o desenvolvimento de aplicações para ambientes *single-cloud* e *multi-cloud* através das conclusões obtidas a partir dos experimentos.

6.1 Contribuições da pesquisa

Através das comparações realizadas entre os resultados obtidos para a aplicação síncrona distribuída ou não em ambiente *multi-cloud*, notou-se uma diferença bastante aceitável entre as aplicações, onde teve-se no pior caso, uma diferença de 28 ms entre as aplicações, e isso, foi uma constatação interessante, pois em diversos tipos de sistemas, 28 ms a mais de diferença não seria um grande problema para continuar oferecendo uma boa experiência aos usuários (ou sistemas terceiros).

Na comparação *single-cloud* e *multi-cloud* da aplicação assíncrona, da mesma forma que na aplicação síncrona, teve-se resultados bastantes constantes, assim como na síncrona, o pior cenário também teve uma diferença de 28 ms. Nesse experimento também teve-se cenários onde a diferença foi extremamente baixa, chegando a 1 ms. Esses resultados mostraram mais uma vez o quão viável é a utilização de arquiteturas distribuídas em *multi-cloud*.

Quando comparado os dois cenários em ambiente *single-cloud*, foi observado pequenas diferença entre eles, onde no pior caso a diferença foi de 36 ms no tempo médio (nesse caso, a comunicação assíncrona teve um melhor desempenho). Notou-se que a aplicação que utiliza comunicação assíncrona teve uma melhor regularidade, sendo que a medida

que o número de requisições foram aumentando, o tempo médio de diferença entre as aplicações síncrona e assíncrona também foi aumentando.

Na comparação entre os tipos de comunicação distribuídos em *multi-cloud*, apresentou-se uma maior variação entre os resultados, porém, no pior caso a diferença foi de 28 ms. Apesar de um pouco mais tímido que nos experimentos com *single-cloud*, a aplicação com comunicação assíncrona se mostrou mais resiliente a medida que o numero de acessos foram aumentando.

Com todas essas análises chega-se a conclusão que as aplicações síncronas e assíncronas tem desempenhos semelhantes com os cenários experimentados, entretanto, enquanto a aplicação síncrona teve melhores resultados que a assíncrona em ambientes *single-cloud*, onde se tem poucas requisições, a aplicação assíncrona conseguiu se destacar em ambientes de maior estresse. Já no cenário *multi-cloud* a aplicação assíncrona teve um desempenho melhor, mesmo em subcenários onde continha poucos acessos. Outro ponto interessante é que se comparado com a a aplicação assíncrona, a comunicação síncrona tem a tendência de aumentar a diferença do tempo médio de resposta a medida que o número de acessos aumentam.

Sendo assim, em arquiteturas que faz sentindo usar qualquer um dos dois tipos de comunicação, a comunicação assíncrona se mostrou mais robusta e escalável, ela tende a manter um padrão no tempo de resposta a medida que o numero de solicitações aumentam, fazendo a aplicação e consequentemente o banco de dados lidarem melhor com as requisições pelo fato de que com a comunicação assíncrona, a aplicação só pode receber dados no momento em que ela tiver livre para processar.

A aplicação com comunicação síncrona, apesar de não ter conseguido desempenhos superiores a comunicação assíncrona, provou que sua utilização é viável em ambientes *single-cloud* e *multi-cloud*. Porém, nesse caso, o engenheiro deve ter cuidados maiores com escalabilidade, ou seja, a aplicação deve ser escalável o suficiente para se auto replicar a medida que o instância seja estressada.

É claro que aplicação utilizando comunicação assíncrona também deve ter essa capacidade, porém, as chances de perda de requisições por parte da aplicação assíncrona são baixíssimas, em casos onde a instância esteja totalmente estressada, fato que não acontece com a aplicação síncrona.

Levando em consideração o paradigma funcional para o desenvolvimento de aplicações distribuídas, clojure se mostrou uma ferramente muito interessante, onde além de oferecer todos os benefícios da programação funcional, ele se mostrou robusto pois não apresentou falhas proporcionadas pela linguagem de programação, outro ponto interessante foi que a aplicação não apresentou indisponibilidade em nenhum dos cenários realizados, se mostrando assim propício para o desenvolvimento de aplicações baseada em microsserviços distribuídos.

6.2 Limitações

Embora os resultados propostos aqui sirvam como base para outros sistemas, uma das grandes limitações deste trabalho foi quanto a memória disponível para os microsserviços. Foi utilizado configurações minímas nos serviços por motivos dos custos cobrados pelos provedores de nuvens quando utilizado configurações mais robustas.

6.3 Trabalhos Futuros

Neste trabalho foi realizada uma análise de desempenho em aplicações baseadas em microsserviços distribuídas ou não em múltiplas nuvens e levando em consideração o seu tipo de comunicação. Um trabalho futuro é fazer uma analise parecida com a deste trabalho, porém, distribuindo os microsserviços globalmente, neste trabalho eles estavam divididos em 1 país.

Outro ponto interessante a ser estudado é sobre a escalabilidade (auto replicação) das instâncias a ser estressados, verificando o quanto os tipos de comunicações influenciam nos na auto replicação dessas instâncias e verificando também o quando de dados ou requisições são perdidos nesses cenários.

Além disso, alguns outros experimentos devem ser considerados, dentre eles levar em consideração os horários das requisições, sabemos que em determinados momentos, a rede tem seus horários de pico, mas o quanto isso poderia impactar no tempo de resposta dos microsserviços?

Referências

ABDELMAWGOUD, A.; JAMSHIDI, M.; BENAVIDEZ, P. Distributed estimation in multimissile cyber-physical systems with time delay. *IEEE Systems Journal*, IEEE, v. 14, n. 1, p. 1491–1502, 2020. Citado na página 14.

ALEXANDER, H. L. State estimation for distributed systems with sensing delay. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Data Structures and Target Classification*. [S.l.], 1991. v. 1470, p. 103–111. Citado na página 14.

BALALAIE, A. et al. Microservices migration patterns. *Software: Practice and Experience*, Wiley Online Library, v. 48, n. 11, p. 2019–2042, 2018. Citado na página 17.

BENLIAN, A. et al. The transformative value of cloud computing: a decoupling, platformization, and recombination theoretical framework. *Journal of management information systems*, Taylor & Francis, v. 35, n. 3, p. 719–739, 2018. Citado na página 13.

CARRERA, D. et al. Complete instrumentation requirements for performance analysis of web based technologies. In: IEEE. *2003 IEEE International Symposium on Performance Analysis of Systems and Software. ISPASS 2003.* [S.l.], 2003. p. 166–175. Citado na página 44.

CARVALHO, J.; VIEIRA, D.; TRINTA, F. Dynamic selecting approach for multi-cloud providers. In: SPRINGER. *International Conference on Cloud Computing*. [S.l.], 2018. p. 37–51. Citado na página 20.

CARVALHO, J.; VIEIRA, D.; TRINTA, F. Greedy multi-cloud selection approach to deploy an application based on microservices. In: IEEE. *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. [S.l.], 2019. p. 93–100. Citado 2 vezes nas páginas 13 e 20.

CARVALHO, J. O. de; TRINTA, F.; VIEIRA, D. Pacificclouds: A flexible microservices based architecture for interoperability in multi-cloud environments. In: *CLOSER*. [S.l.: s.n.], 2018. p. 448–455. Citado 2 vezes nas páginas 24 e 26.

CHAND, M. *Top 10 Cloud Service Providers In 2021*. 2021. Disponível em: <https://www.c-sharpcorner.com/article/top-10-cloud-service-providers/>. Citado na página 28.

CLARK, K. Microservices, soa, and apis: Friends or enemies. *Online]. IBM developerWorks. Accessed July*, v. 26, p. 2018, 2016. Citado na página 14.

CUNICO, E. Aplicação java para transferência de arquivos utilizando microsserviços rest e contêiner. Universidade Tecnológica Federal do Paraná, 2018. Citado na página 24.

DAU, H. *Advantages Of Functional Programming*. 2017. Disponível em: <https://woodridgesoftware.com/advantages-functional-programming/>. Citado na página 16.

ETRO, F. The economics of cloud computing. In: *Cloud Technology: Concepts, Methodologies, Tools, and Applications*. [S.l.]: IGI Global, 2015. p. 2135–2148. Citado na página 13.

GALOV, N. *ACloud Adoption Statistics for 2021*. 2021. Disponível em: <https://www-.infoworld.com/article/3114195/the-8-fallacies-of-distributed-computing-are-becoming-irrelevant.html>. Citado na página 13.

HAT, R. *What is a REST API?* 2020. Disponível em: <https://www.redhat.com/en-/topics/api/what-is-a-rest-api>. Citado na página 21.

HONG, X. J.; YANG, H. S.; KIM, Y. H. Performance analysis of restful api and rabbitmq for microservice web application. In: IEEE. *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. [S.l.], 2018. p. 257–259. Citado 2 vezes nas páginas 24 e 26.

HOOTSUITE. *DIGITAL IN 2018: WORLD'S INTERNET USERS PASS THE 4 BILLION MARK*. 2018. Disponível em: <wearesocial.com/blog/2018/01/global-digital-report-2018>. Acesso em: 02 set. 2019. Citado na página 13.

IBM. *Challenges and benefits of the microservice architectural style, Part 1*. 2019. Disponível em: <https://developer.ibm.com/articles/challenges-and-benefits-of-the-microservice-architectural-style-part-1/>. Citado na página 14.

IBM. *What is Multicloud?* 2021. Disponível em: <https://www.ibm.com/cloud/learn-/multicloud>. Citado na página 13.

LIMA, L. R. T. Implementação de uma arquitetura baseada em microserviços. 2015. Citado 2 vezes nas páginas 23 e 26.

LOMBARDI, F.; PIETRO, R. D. Secure virtualization for cloud computing. *Journal of network and computer applications*, Elsevier, v. 34, n. 4, p. 1113–1122, 2011. Citado 2 vezes nas páginas 13 e 19.

MARSTON, S. et al. Cloud computing—the business perspective. *Decision support systems*, Elsevier, v. 51, n. 1, p. 176–189, 2011. Citado na página 14.

MARTINS, A. B. d. J.; JUSTINO, A. C. F. C.; GABRIEL, G. d. C. F. S. comunicação síncrona, assíncrona e multidirecional. *Políticas de Informação na Sociedade em Rede. Guimarães*, v. 10, 2010. Citado na página 21.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National . . . , 2011. Citado 2 vezes nas páginas 19 e 20.

MESSINA, A. et al. The database-is-the-service pattern for microservice architectures. In: SPRINGER. *International Conference on Information Technology in Bio-and Medical Informatics*. [S.l.], 2016. p. 223–233. Citado na página 18.

MESSINA, A. et al. A simplified database pattern for the microservice architecture. In: *The Eighth International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA)*. [S.l.: s.n.], 2016. p. 35–40. Citado na página 18.

MORAVEIS, J. *Message Queues: Even Microservices Want to Chit Chat*. 2019. Disponível em: <hhttps://blog.avenuecode.com/message-queue-even-microservices-want-to-chit-chat>. Citado na página 21.

NEWMAN, S. *Building microservices: designing fine-grained systems*. [S.l.]: "O'Reilly Media, Inc.", 2015. Citado na página 17.

OMOTUNDE, A. et al. Survey of cloud computing issues at implementation level. *Journal of Emerging Trends in Computing and Information Sciences*, v. 4, n. 1, p. 91–96, 2013. Citado na página 13.

PARAISO, F. et al. A federated multi-cloud paas infrastructure. In: IEEE. *2012 IEEE Fifth International Conference on Cloud Computing*. [S.l.], 2012. p. 392–399. Citado na página 13.

SATZGER, B. et al. Winds of change: From vendor lock-in to the meta cloud. *IEEE internet computing*, IEEE, v. 17, n. 1, p. 69–73, 2013. Citado na página 20.

SINGH, V.; PEDDOJU, S. K. Container-based microservice architecture for cloud applications. In: IEEE. *2017 International Conference on Computing, Communication and Automation (ICCCA)*. [S.l.], 2017. p. 847–852. Citado 2 vezes nas páginas 23 e 26.

SOUSA, G.; RUDAMETKIN, W.; DUCHIEN, L. Automated setup of multi-cloud environments for microservices applications. In: IEEE. *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. [S.l.], 2016. p. 327–334. Citado 2 vezes nas páginas 23 e 26.

THOMAS, D. Functional programming–crossing the chasm. *Journal of object technology*, 2009. Citado na página 16.

VILLAMIZAR, M. et al. Infrastructure cost comparison of running web applications in the cloud using aws lambda and monolithic and microservice architectures. In: IEEE. *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. [S.l.], 2016. p. 179–182. Citado 2 vezes nas páginas 23 e 26.

VOIGT, R.; JUNIOR, O. d. O. B. Análise de desempenho de arquitetura soap e rest para comunicação entre sistemas. *Anais SULCOMP*, v. 8, 2017. Citado 2 vezes nas páginas 24 e 26.

WILDE, . J. P. B. K. Guide to remote work (1.2th ed.). v. 1, 2021. Citado na página 21.

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, **Pedro Augusto Alcantara Ribeiro Moraes,** autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação "Uma análise de Desempenho Multi-Cloud de uma Aplicação Baseada em Microsserviços" de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 31 de outubro de 2022.

Assinatura