

Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação

Avaliação de Desempenho e Eficiência Energética em Cadeias de Funções de Rede

Gabriel Holanda de Araújo

Número de Ordem PPGCC: M001

Picos-PI, Março de 2025

Gabriel Holanda de Araújo

Avaliação de Desempenho e Eficiência Energética em Cadeias de Funções de Rede

Defesa de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Universidade Federal do Piauí – UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação

Orientador: Prof. Dr. Francisco Airton Pereira da Silva

Coorientador: Prof. Dr. Rayner Gomes Sousa

Picos-PI

Março de 2025

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Sistema de Bibliotecas UFPI - SIBi/UFPI
Biblioteca Setorial do CCN

A663a Araújo, Gabriel Holanda de.
Avaliação de desempenho e consumo energético em cadeias de funções de rede / Gabriel Holanda de Araújo. -- 2024.
118 f.

Dissertação (Mestrado) - Universidade Federal do Piauí. Centro de Ciências da Natureza. Programa de Pós-Graduação em Ciências da Computação, Picos, 2024.
"Orientador: Prof. Dr. Francisco Airton Pereira da Silva. Coorientador: Prof. Dr. Rayner Gomes Sousa".

1. Arquitetura de *software*. 2. Rede de Petri. 3. *Network Function Virtualization - NFV*. 4. *Service Function Chain - SFC*. I. Silva, Francisco Airton Pereira da. II. Sousa, Rayner Gomes. II. Titulo.

CDD 005.133

Bibliotecária: Caryne Maria da Silva Gomes - CRB3/1461

Gabriel Holanda de Araújo

Avaliação de Desempenho e Eficiência Energética em Cadeias de Funções de Rede

Defesa de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Documento assinado digitalmente
gov.br FRANCISCO AIRTON PEREIRA DA SILVA
Data: 28/03/2025 14:48:43-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Francisco Airton Pereira da Silva
Orientador

Documento assinado digitalmente
gov.br RAYNER GOMES SOUSA
Data: 27/03/2025 21:24:43-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Rayner Gomes Sousa
Coorientador

Documento assinado digitalmente
gov.br ANDRE CASTELO BRANCO SOARES
Data: 26/03/2025 18:46:35-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. André Castelo Branco Soares
Membro da Banca

Documento assinado digitalmente
gov.br LUIZ FERNANDO BITTENCOURT
Data: 27/03/2025 10:48:13-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Luiz Fernando Bittencourt
Membro da Banca

Documento assinado digitalmente
gov.br ALBERTO EGON SCHAEFFER FILHO
Data: 26/03/2025 15:15:05-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Alberto Egon Schaeffer Filho
Membro da Banca

Picos-PI
Março de 2025

Aos meus pais Iraci de Holanda Gonçalves e Paulo Antônio de Araújo, minha esposa Nádia de Sousa Moura Gonçalves e meu filho Rafael de Moura Holanda por sempre estarem comigo em todos os momentos.

Agradecimentos

Primeiramente quero agradecer a Deus por ter me dado saúde, força e coragem para nunca desistir, mesmo diante de todas as dificuldades. Agradeço aos meus pais, Iraci de Holanda Gonçalves e Paulo Antônio de Araújo, por todo o apoio e por terem me dado a base para me tornar a pessoa que sou hoje. Agradeço a minha esposa Nádia de Sousa Moura Gonçalves e ao meu filho Rafael de Moura Holanda por todo o apoio e por sempre me proporcionarem ótimos momentos. À toda a minha família que sempre me deram o suporte e os conselhos necessários para me tornar alguém melhor. Quero agradecer especialmente a minha madrinha Lurdes, minha irmã Amanda, meus tios e tias, meus primos e primas meus avós. Ao meu orientador Dr. Francisco Airton e ao meu coorientador Rayner Gomes pela orientação, confiança, e paciência, sem eles nada disso seria possível. Também agradeço à todos os outros professores da UFPI foram fundamentais para o meu aprendizado, como também foram fontes de inspiração e valores que com certeza me moldaram para melhor. À todos os amigos que acompanharam de perto minha jornada na UFPI, em especial Carlos Vitor, Lucas Vinícius, Luis Guilherme e Israel Cardoso. Eles tornaram meu dia a dia na universidade muito mais agradável e me ajudaram a superar todos os problemas acadêmicos. Agradeço a CAPES pelo apoio financeiro para realização deste trabalho de pesquisa. À todos que estiveram comigo durante essa importante fase da minha vida, meus mais sinceros agradecimentos.

"Tudo que aprendemos nos engrandece de alguma forma.
Um dia partiremos e quando esse momento chegar,
o conhecimento de uma vida irá conosco, ou
o deixaremos como nosso legado"
(Jorge Rafael Loiola de Macêdo)

Resumo

A arquitetura de microsserviços utiliza a habilidade de virtualização para adotar uma abordagem de arquitetura de *software* baseada na construção de aplicações compostas por diversos módulos interdependentes e fracamente acoplados. A tecnologia de NFV (*Network Function Virtualization*) é um paradigma que busca melhorar a flexibilidade e a eficiência dos serviços de rede. A NFV se concentra na virtualização de funções de rede, permitindo que sejam executadas em *hardware* genérico, o que separa *software* e *hardware* e permite mais implementações ágeis e escaláveis. Através da NFV surge a SFC (*Service Function Chain*), que consiste em uma sequência de VNFs (*Virtual Network Functions*), como *gateways*, *firewalls* e roteadores, organizados de maneira ordenada. As VNFs podem ser alocadas dentro de contêineres que se comunicam em uma arquitetura de microsserviços. No entanto, um desafio crítico enfrentado pela SFC é a sua sensibilidade à latência: o atraso aumenta à medida que a cadeia de VNFs aumenta, o que pode comprometer a qualidade do serviço. Minimizar esta latência é crucial para garantir a eficiência dos sistemas SFC. Além disso, as arquiteturas de microsserviços podem resultar em grandes custos de energia para alcançarem os níveis necessários de desempenho e disponibilidade. Este estudo visa o planejamento de arquiteturas SFC energicamente eficientes, utilizando modelos analíticos, como *Stochastic Petri Network* (SPN), para avaliar o desempenho de diferentes arquiteturas. Um modelo de SPN é proposto para para avaliar o desempenho de diferentes arquiteturas de SFC em um contexto de microsserviços. Além disso, uma revisão sistemática é realizada sobre o consumo energético em arquiteturas de microsserviços, identificando lacunas de pesquisa e tendências na área. Os resultados demonstram que a arquitetura paralela pode trazer ganhos de desempenho e menor consumo energético em relação às arquiteturas sequencial e híbrida. A otimização da SFC aplicada a microsserviços é fundamental para garantir a eficiência dos serviços. O objetivo é auxiliar administradores a planejar e avaliar sistemas SFC com microsserviços, focando em desempenho e eficiência energética.

Palavras-chaves: SFC; microsserviços; rede de Petri; NFV; VNF.

Abstract

Microservices architecture uses the ability of virtualization to adopt a software architecture approach based on building applications composed of several interdependent and loosely coupled modules. NFV (Network Function Virtualization) technology is a paradigm that seeks to improve the flexibility and efficiency of network services. NFV focuses on the virtualization of network functions, allowing them to be executed on generic hardware, which separates software and hardware and allows for more agile and scalable implementations. NFV creates a Service Function Chain (SFC), which consists of a sequence of VNFs (Virtual Network Functions), such as gateways, firewalls and routers, organized in an orderly manner. VNFs can be placed inside containers that communicate in a microservices architecture. However, a critical challenge faced by SFC is its sensitivity to latency: the delay increases as the chain of VNFs grows, which can compromise the quality of service. Minimizing this latency is crucial to ensure the efficiency of SFC systems. In addition, microservices architectures can result in large energy costs to achieve the required levels of performance and availability. This study aims at designing energy-efficient SFC architectures, using analytical models, such as *Stochastic Petri Network* (SPN), to evaluate the performance of different architectures. A SPN model is proposed to evaluate the performance of different SFC architectures in a microservices context. In addition, a systematic review on energy consumption in microservices architectures is performed, identifying research gaps and trends in the area. The results demonstrate that parallel architecture can bring performance gains and lower energy consumption compared to sequential and hybrid architectures. Optimizing SFC applied to microservices is essential to ensure service efficiency. The goal is to help administrators plan and evaluate SFC systems with microservices, focusing on performance and energy efficiency.

Keywords: SFC; microsservices; petri net; NFV; VNF.

Lista de ilustrações

Figura 1 – Metodologia de desenvolvimento do modelo analítico. 10
Figura 2 – Componentes SPNs. 13
Figura 3 – Exemplo genérico de gráfico de efeito dos fatores. 15
Figura 4 – Gráficos de interação genéricos. 16
Figura 5 – Fluxograma da metodologia aplicada na revisão sistemática. 21
Figura 6 – Fluxograma que demonstra as etapas do processo de filtragem. 23
Figura 7 – Diagrama de Venn com a classificação dos contextos mapeados. 25
Figura 8 – Classificação das métricas. 28
Figura 9 – Taxonomia de métricas. 29
Figura 10 – Classificação geral dos métodos de avaliação. 33
Figura 11 – Classificação dos tipos de arquitetura. 37
Figura 12 – Classificação dos tipos de orquestrador. 39
Figura 13 – Classificação do modelo de implantação. 40
Figura 14 – Arquitetura x modelo de implantação. 42
Figura 15 – Arquitetura x orquestrador. 43
Figura 16 – Método de avaliação x arquitetura. 44
Figura 17 – Método de Avaliação x Modelo de Implantação. 45
Figura 18 – Métricas macro x arquitetura. 46
Figura 19 – Métricas macro x método de avaliação. 47
Figura 20 – Métricas macro x modelo de implantação. 48
Figura 21 – Métricas macro x orquestrador. 49
Figura 22 – Orquestrador x modelo de implantação. 50
Figura 23 – Arquiteturas SFC sequenciais, paralelas e híbridas. 60
Figura 24 – Submodelo SPN para um VNF. 62
Figura 25 – Modelo SPN da arquitetura SFC. 63
Figura 26 – Exemplo de modelo SPN. 65
Figura 27 – Impacto dos fatores mais impactantes para MTTA. 68
Figura 28 – Interações entre fatores mais impactantes para o MTTA. 69
Figura 29 – Solicitações concluídas em arquiteturas sequenciais, paralelas e híbridas. 70
Figura 30 – MTTA de arquiteturas sequenciais, paralelas e híbridas. 71
Figura 31 – CDF de arquiteturas sequenciais, paralelas e híbridas. 72
Figura 32 – Variação de TSW e EXVNF1 na configuração sequencial. 73
Figura 33 – Infraestrutura de organização sequencial. 75
Figura 34 – Infraestrutura de organização paralela. 75
Figura 35 – Infraestrutura de organização híbrida. 76
Figura 36 – Gráfico de barras com erros nos resultados do Teste T. 77

Figura 37 – Consumo energético nas arquiteturas sequenciais, paralelas e híbridas com 1000 VNFs. 83

Lista de tabelas

Tabela 1 – Trabalhos relacionados. 18
Tabela 2 – Critérios de exclusão direta 22
Tabela 3 – Categorização 24
Tabela 4 – Trabalhos Relacionados. 58
Tabela 5 – Parâmetros do modelo usados em estudos de caso. 65
Tabela 6 – Parâmetros de falha e sucesso do modelo usados no estudo de caso. . . 65
Tabela 7 – Fatores e configurações. 66
Tabela 8 – Combinações de fatores. 67
Tabela 9 – Resultados do Teste T. 76

Lista de abreviaturas e siglas

TI	Tecnologia da Informação
DoE	*Design of Experiments*
SPN	*Stochastic Petri Net*
PN	*Petri Net*
NFV	*Network Function Virtualization*
NFVI	*Network Functions Virtualization Infrastructure*
SFC	*Service Function Chain*
VNF	*Virtual Network Functions*
MTTA	*Mean Time To Absorption*
ETSI	*Industry Specification Group*
NF	*Network Function*
SF	*Service Function*
VM	*Virtual Machine*
NSucssess	Número de Sucessos
NFail	Número de Elementos Falhas
AMS	Arquitetura de Microsserviços
IoT	*Internet of Things*
CTMC	*Continuous-time Markov Chain*
SLA	*Service Level Agreement*
GNS3	*Graphic Network Simulator 3*
NS-3	*Network Simulator 3*
CDF	*Cumulative Distribution Functions*
CNP	*Colorful Petri Net*

DNI	*Network Infrastructure*
SDN	*Software Defined Networks*
QPN	*Queuing Petri Nets*
IMS	*IP Multimedia Subsystem*
RBD	*Reliability Block Diagram*
SHARPE	*Symbolic Hierarchical Automated Reliability and Performance Evaluator*
SRN	*Stochastic Reward Networks*
NIC	*Network Interface Card*
NQA	*Network Queue Assessment*
MRT	*Mean Response Time*
IP	*Internet Protocol*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
API	*Application Programming Interface*
J	*Joules*
MANO	*Management and Orchestration*
EMS	*Element Management System*
SCF	*Service Classification Function*
SFF	*Service Function Forwarder*
SFP	*Service Function Path*
CTF	*Chain Termination Function*
VNFFGs	*VNF Forwarding Graphs*

Sumário

1 INTRODUÇÃO 3
1.1 Problema 5
1.2 Objetivos 6
1.3 Escopo 7
1.4 Metodologia 8
1.5 Estrutura do Trabalho 9

2 REFERENCIAL TEÓRICO 11
2.1 *Network Functions Virtualization* (NFV) E *Service Function Chain* (SFC) 11
2.2 *Stochastic Petri Net* (SPN) 13
2.3 Análise de Sensibilidade 14

3 REVISÃO SISTEMÁTICA DE LITERATURA EM CONSUMO ENERGÉTICO EM ARQUITETURAS DE MICROSSERVIÇO 17
3.1 Trabalhos Relacionados 17
3.2 Metodologia 21
3.3 Resultados da Classificação Geral 25
3.4 Resultados de Classificação de Correlação Cruzada 41
3.5 Desafios e Direções de Pesquisa 50
3.6 Considerações Finais 53

4 MODELO DE ENCADEAMENTO DE FUNÇÕES DE SERVIÇO DE REDE: UM ESTUDO DE DESEMPENHO APLICADO EM MICROSSERVIÇOS 55
4.1 Contextualização 55
4.2 Trabalhos Relacionados 56
4.3 Arquitetura 60
4.4 Modelo 61
4.5 Parâmetros do Modelo e Análise de Sensibilidade 65
4.6 Resultados 68
4.7 Validação do Modelo SPN 74
4.8 Considerações Finais 76

5 AVALIAÇÃO DE CONSUMO ENERGÉTICO EM DIFERENTES ARQUITETURAS SFC DE MICROSSERVIÇOS 79

5.1 ***Network Simulator 3*** **79**
5.2 **Construção das Arquiteturas de SFC** **80**
5.3 **Resultados** **82**

6 **CONCLUSÃO** **85**

REFERÊNCIAS **87**

1 Introdução

A computação em nuvem é um paradigma que permite o acesso sob demanda a um conjunto de recursos computacionais configuráveis partilhados (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com o mínimo esforço de gestão ou de interação com o fornecedor de serviços (YANAMALA, 2024). A computação em nuvem tem se expandido como uma das principais abordagens de virtualização e está se tornando cada vez mais presente na vida cotidiana das pessoas. O foco da virtualização é a consolidação para a redução de custos, buscando tornar os sistemas mais eficientes e melhorar o desempenho.

A computação em nuvem tem evoluído rapidamente nos últimos anos e se tornou uma das principais tendências tecnológicas da atualidade. Desde sua concepção, a computação em nuvem tem sido considerada uma maneira de democratizar o acesso a recursos computacionais, permitindo que empresas e indivíduos possam utilizar infraestruturas de Tecnologia da Informação (TI) de alta qualidade sem precisar investir em *hardware* e *software* próprios. Provedores como *Amazon*, *Google*, *Salesforce*, *IBM*, *Microsoft* e *Sun Microsystems* começaram a estabelecer novos *datacenters* para hospedar aplicativos de computação em nuvem em locais ao redor do mundo para fornecer redundância e garantir confiabilidade em caso de falhas nos *sites* (YANAMALA, 2024; SHAH et al., 2024). Ao longo do tempo, a computação em nuvem tornou-se cada vez mais sofisticada, oferecendo recursos avançados de virtualização, automação e gerenciamento de dados. Hoje, a computação em nuvem é amplamente utilizada em diversos setores, desde o comércio eletrônico até a pesquisa científica.

Segundo a previsão da Statista, os serviços de Nuvem Pública geraram aproximadamente US$ 631,84 bilhões em todo o mundo em 2023 e continuarão a crescer a uma taxa alta de 18,49% até 2029. Além disso, o mercado de *Software* como Serviço representa a maior fatia, abrangendo 45,58% das receitas mundiais. (STATISTA, 2024). Os microsserviços compõem uma arquitetura de software que se destaca por se basearem na virtualização e no conceito de modularização. A virtualização é uma das técnicas utilizadas para tornar a infraestrutura dos sistemas mais eficiente e a modularização consiste na divisão de um sistema em módulos mais simples e independentes. Os microsserviços, combinados a esses conceitos, permitem a criação de sistemas mais flexíveis, escaláveis e fáceis de manter (DRAGONI et al., 2017; THÖNES, 2015). O termo microsserviço refere-se a um processo independente que implementa uma dada funcionalidade completamente, podendo funcionar autonomamente de maneira isolada sem auxílio de outro tipo de serviço, diferente das arquiteturas monolíticas.

Os aplicativos com arquiteturas monolíticas tornavam os sistemas engessados devido ao alto acoplamento entre suas funções, dificultando alterações em funcionalidades, bem como desperdiçando recursos escalando funcionalidades não necessárias (AL-DEBAGY; MARTINEK, 2018). Na arquitetura monolítica nenhuma funcionalidade do sistema existe e opera por si só. Os principais benefícios da arquitetura de microsserviços são a escalabilidade e a resiliência.. Os principais desafios da arquitetura de microsserviços incluem a dificuldade na refatoração e na comunicação distribuída. Dependendo do contexto, é necessário encontrar um equilíbrio entre as vantagens das arquiteturas monolíticas e as de microsserviços, considerando o número de usuários e o tempo de resposta. Os aplicativos baseados em arquiteturas de microsserviços utilizam serviços individuais e interconectados, permitindo às organizações realizar mudanças específicas em determinados microsserviços. Graças à modularização, isso resulta em um alto nível de agilidade e velocidade de entrega. (AL-DEBAGY; MARTINEK, 2018).

As arquiteturas de microsserviços têm se destacado como uma forma eficiente e flexível de desenvolver e gerenciar aplicativos em nuvem (GOS; ZABIEROWSKI, 2020). O processo de utilização de microsserviços no desenvolvimento de aplicações permite que estes microsserviços sejam distribuídos em diferentes provedores de nuvem de forma independente, porém gerando uma impressão que estão implantados em um único serviço. Dessa forma, o uso de microsserviços ganha força no mercado e se torna cada vez mais presente em grandes corporações. (NARDIN et al., 2021; LIU et al., 2019).

Os microsserviços são usados em aplicações de grandes organizações como: a *Netflix*, a *Amazon* e o *EBay*, que usam com êxito a técnica de dividir e conquistar para particionar funcionalmente suas aplicações monolíticas em unidades atômicas menores, onde cada uma executa uma única função. Essas empresas também têm utilizado com sucesso as arquiteturas de microsserviços para criar sistemas mais escaláveis, modulares e fáceis de manter. Ao contrário das arquiteturas monolíticas, que tendem a ser engessadas e difíceis de modificar, as arquiteturas de microsserviços permitem que as empresas realizem mudanças específicas em determinados microsserviços, sem afetar o funcionamento do sistema como um todo. Isso resulta em um alto nível de agilidade e velocidade de entrega, o que é essencial em um mercado cada vez mais competitivo. De acordo com a previsão do Statista, o mercado global de software de virtualização é estimado entre $40 e $62 bilhões e deverá atingir entre $120 e $163 bilhões até 2027 (STATISTA, 2023).

As arquiteturas de microsserviços podem ser aplicadas nos mais diversos cenários de virtualização, como na NFV (*Network Function Virtualization*) que é baseada na tecnologia de virtualização de funções de rede e visa reduzir consideravelmente o grau de acoplamento entre *hardware* e *software* (CHOWDHURY et al., 2019). A NFV permite o compartilhamento, o dimensionamento e a implantação de recursos de dispositivos de rede de maneira flexível por meio da virtualização. No entanto, o NFV também traz a

necessidade de configuração flexível de serviços.

Requisitos flexíveis de configuração de serviço deram origem ao paradigma SFC (*Service Function Chain*). O SFC é uma cadeia ordenada de VNFs (*Virtual Network Functions*). Um grupo de VNFs relacionados processa cada solicitação. Cada VNF pode representar dispositivos de rede como *gateways*, *firewalls* e roteadores (ZHANG et al., 2017). Além disso, o SFC não se limita aos serviços de rede; também é usado em serviços da camada de transporte na rede, aplicativos e multimídia. A aplicação de microsserviços em arquiteturas SFC pode otimizar o desempenho e eficiência de recursos.

1.1 Problema

As arquiteturas de microsserviços também apresentam desafios significativos, especialmente no que diz respeito à eficiência energética. As arquiteturas de microsserviços podem apresentar um enorme problema devido aos altos custos energéticos demandados pelos sistemas para se manterem disponíveis. O consumo de energia tende a ser maior em arquiteturas de microsserviços devido a fatores como a comunicação entre os serviços, monitoramento de serviços e elasticidade (NARDIN et al., 2021; LIU et al., 2019).

Os altos custos energéticos têm um impacto negativo no meio ambiente e nos custos operacionais das empresas. O alto consumo energético em arquiteturas de microsserviços vai além do custo energético em si e pode afetar a qualidade do serviço de todo o sistema, prejudicando a performance e disponibilidade do sistema (CANOSA-REYES et al., 2022; VALERA et al., 2021a). Por isso, é importante estudar e desenvolver soluções que permitam melhorar a eficiência energética das arquiteturas de microsserviços. Além disso, é importante avaliar e validar tais soluções, para isso é importante investigar a fundo fatores fundamentais como quais são as métricas de avaliação de consumo energético em microsserviços.

Esta dissertação apresenta uma revisão sistemática do consumo energético em arquiteturas de microsserviços. A seleção de artigos foi iniciada com 3625 trabalhos seguidos de três processos de refinamento em que apenas 37 artigos foram selecionados para revisão completa. Os artigos restantes foram catalogados e classificados de acordo com alguns critérios, como, por exemplo: foco geral, método de avaliação, tipo de arquitetura, entre outros. A revisão revelou lacunas de pesquisas e tendências na área de consumo de energia em arquiteturas de microsserviços.

Os microsserviços podem ser aplicados a outros tipos de virtualização, como as arquiteturas SFC, que possuem um funcionamento similar. As arquiteturas SFC são formadas por VNFs que podem ser interpretados como serviços que funcionam de forma isolada, mas também se integram a outros VNFs na cadeia formando um ecossistema de serviços, similar aos microsserviços. Além disso, as arquiteturas SFC aplicadas a

microsserviços precisam ser otimizadas para garantir o funcionamento eficiente dos serviços. O estudo da otimização SFC pode ser dividido em diversas categorias, como gasto de energia, uso de recursos e qualidade de serviço (DUAN et al., 2023). As três categorias principais de otimização SFC são custo, confiabilidade e atraso (RUI et al., 2020). A otimização do atraso SFC é uma das principais categorias de estudo em VNFs, uma vez que o SFC é altamente sensível à latência. Dependendo do sistema, o atraso pode ser inaceitável e afetar significativamente a qualidade do serviço. Além disso, o atraso SFC cresce linearmente com o comprimento da cadeia; ou seja, quanto maior o SFC, maior será o atraso (RYGIELSKI; SELIUCHENKO; KOUNEV, 2016; SCHNEIDER et al., 2019).

Uma maneira de reduzir o atraso é reorganizar as VNFs da SFC em diferentes arquiteturas: sequencial, paralela ou híbrida. Além disso, os SFCs são um ramo da virtualização com grande potencial para aplicação em microsserviços, pois possuem arquiteturas compostas por *Virtualized Network Functions* (VNF) encadeadas, formando uma estrutura semelhante à dos microsserviços. No entanto, a condução de experimentos reais em diferentes tipos de arquiteturas SFC é muito cara, especialmente com múltiplos VNFs. Os modelos analíticos são uma forma de avaliar o desempenho do atraso SFC de forma eficiente e econômica. Existem diferentes modelos analíticos, como cadeias de Markov, teoria das filas e SPNs. Em particular, as *Stochastic Petri Nets* (SPNs) (MARSAN, 1995) são modelos matemáticos baseados na teoria das probabilidades usados para modelar e avaliar sistemas complexos. O modelo de SPN foi escolhido porque oferece maior flexibilidade para planejar e avaliar diferentes arquiteturas SFC, permitindo alternar o tipo de arquitetura de maneira mais simples por meio de condições de guarda. Dessa forma, a presente pesquisa tem como questão principal: **Como planejar arquiteturas de SFC aplicadas a microsserviços com eficiência energética?**

1.2 Objetivos

O objetivo geral deste trabalho é auxiliar administradores de infraestruturas computacionais a planejar e avaliar sistemas SFC com microsserviços, considerando aspectos de desempenho e eficiência energética.

Os objetivos específicos deste trabalho são:

- Realizar uma revisão sistemática sobre consumo energético em arquiteturas de microsserviços. A revisão traz uma visão geral, questões em aberto e direcionamentos futuros sobre o consumo energético em microsserviços.

  Objetivo abordado no artigo publicado: Araújo, Gabriel, et al. "Energy consumption in microservices architectures: a systematic literature review."IEEE Access (2024). (Capítulo 3).

- Conceber um modelo que permita análise de desempenho (tempo de resposta) de sistemas de SFC aplicados a microsserviços. O modelo permite alteração, remoção ou modificação de VNFs, variação de quantidade de requisições, variação da arquitetura SFC em sequencial, paralela ou híbrida; O modelo foi validado por meio de simulação pelo simulador GNS3.

  Realizado - artigo publicado: Araújo, Gabriel, et al. "Network service function chaining: a performance study varying topologies."Cluster Computing 28.1 (2025): 7. (Capítulo 4).

- Aplicar a arquitetura proposta no modelo SPN no simulador NS-3. A arquitetura replicará as diferentes arquiteturas SFC no ambiente de simulação. A métrica de consumo energético será aplicada no NS-3 para avaliar o uso de recursos.

  Objetivo abordado no Capítulo 5 (conteúdo ainda não publicado).

1.3 Escopo

O escopo desse trabalho engloba estrutura, características, comportamento e funcionamento de diferentes arquiteturas de SFC aplicadas a microsserviços. Alguns assuntos estão fora do contexto deste trabalho e, portanto, não foram abordados neste trabalho para mantê-lo enxuto e objetivo. Alguns desses assuntos estão listados abaixo:

- Orquestração de Contêineres: não foi considerada na modelagem. Arquiteturas de microsserviços são baseadas em contêineres, os quais precisam ser orquestrados e gerenciados por ferramentas como o *Kubernetes* ou o *Docker Swarm*, por exemplo. Contudo, essa complexidade é abstraída no modelo, ficando a critério do arquiteto do sistema qual tecnologia utilizar. Esse ponto pode ser explorado em trabalhos futuros, com um detalhamento mais aprofundado dos orquestradores utilizados.

- Capacidade Computacional: não foi abordada na modelagem. Com o advento da computação em nuvem, tornou-se possível acessar recursos computacionais quase ilimitados sob demanda, algo comum em arquiteturas de microsserviços. Dessa forma, não consideramos o aspecto de capacidade computacional para tratamento das requisições ou adição de VNFs. A capacidade computacional pode ser melhor abordada em trabalhos futuros, principalmente com experimentos de carga de trabalho.

- Segurança da Informação: não foi levada em conta na modelagem. A modelagem de problemas que dizem respeito à segurança é, normalmente, bastante complexa. Um dos principais desafios é lidar com invasões, um evento aparentemente aleatório em um modelo matemático que requer taxas e distribuições matemáticas. Embora a

modelagem de segurança não seja impossível, seria mais adequado elaborar um modelo focado nesse aspecto. Uma nova modelagem considerando ataques de segurança poderia ser desenvolvido para tratar problemas neste aspecto.

1.4 Metodologia

O objetivo principal deste trabalho é desenvolver um modelo para avaliar arquiteturas de SFC aplicadas à microsserviços. A Figura 1 apresenta um fluxograma que resume a estratégia usada neste trabalho de forma geral.

- **Entendimento da Aplicação:** É importante entender o funcionamento da aplicação, nesse caso uma arquitetura SFC aplicada a microsserviços. Dessa forma, foi realizado um mapeamento sistemático da literatura para descobrir *insights* sobre consumo energético em arquiteturas de microsserviços.

- **Definição das Métricas:** As métricas de interesse devem ser identificadas, levando em consideração as informações que o modelo pode fornecer para diagnosticar o desempenho do sistema. Neste trabalho, as métricas selecionadas podem ser importantes na percepção do usuário final e úteis para administradores de sistema, são elas: *Mean Time To Absorption* (MTTA) que avalia o tempo médio até que as requisições finalizem e sejam absorvidas pelo sistema; *Cumulative Distribution Functions* (CDF) que analisa a probabilidade de todas as requisições serem processadas em um determinado período de tempo; número de sucessos (NSucssess) e número de falhas (NFail) das requisições; e o consumo de energia das arquiteturas de SFC.

- **Definição dos Parâmetros:** Os parâmetros que serão inseridos no modelo são definidos. Esses parâmetros definem o comportamento e capacidade de recursos para os componentes. Neste trabalho, os parâmetros adicionados foram a quantidade de requisições e os tempos de transmissão e execução dos VNFs.

- **Geração da Modelagem Analítica:** Um modelo de desempenho usando Rede de Petri Estocásticas é desenvolvido aqui. Nesta parte, o modelo é construído considerando as métricas e parâmetros definidos, e os resultados esperados. A Rede de Petri Estocásticas foi escolhida porque oferece o suporte necessário e suficiente para descrever os diversos comportamentos e especificações do sistema avaliado.

- **Análise de Sensibilidade:** A utilização de técnicas de análise de sensibilidade permite ao designer do sistema identificar quais componentes são mais críticos. A metodologia utilizada foi a do DoE.

- **Avaliação com Casos de Uso** Casos de uso foram criados para verificar a capacidade de geração de resultados do modelo. Os casos de uso foram planejados para comparar o desempenho e redução de atraso nas diferentes arquiteturas de SFC.
- **Validação do Modelo SPN:** O processo de validação foi realizado por meio do simulador GNS3. O GNS3 permitiu simular as mesmas arquiteturas e VNFs com as mesmas cargas de trabalho para comparar com os resultados gerados pelo modelo por meio do teste T.
- **Simulação:** Um experimento sobre o consumo de energia nas diferentes arquiteturas SFC foi realizado no simulador NS-3 para avaliação e aplicação do consumo energético.

1.5 Estrutura do Trabalho

O restante desta dissertação está organizado da seguinte forma: O Capítulo 2 apresenta os conceitos necessários para a compreensão desta dissertação; O Capítulo 3 apresenta uma revisão sistemático de literatura sobre o consumo energético de microsserviços; O Capítulo 4 apresenta um modelo de SPN para avaliação de desempenho em uma arquitetura de SFC em três diferentes arquiteturas de microsserviços. O Capítulo 5 apresenta uma avaliação sobre o consumo energético de arquiteturas SFC de microsserviços. Por fim, o Capítulo 6 conclui esta dissertação.

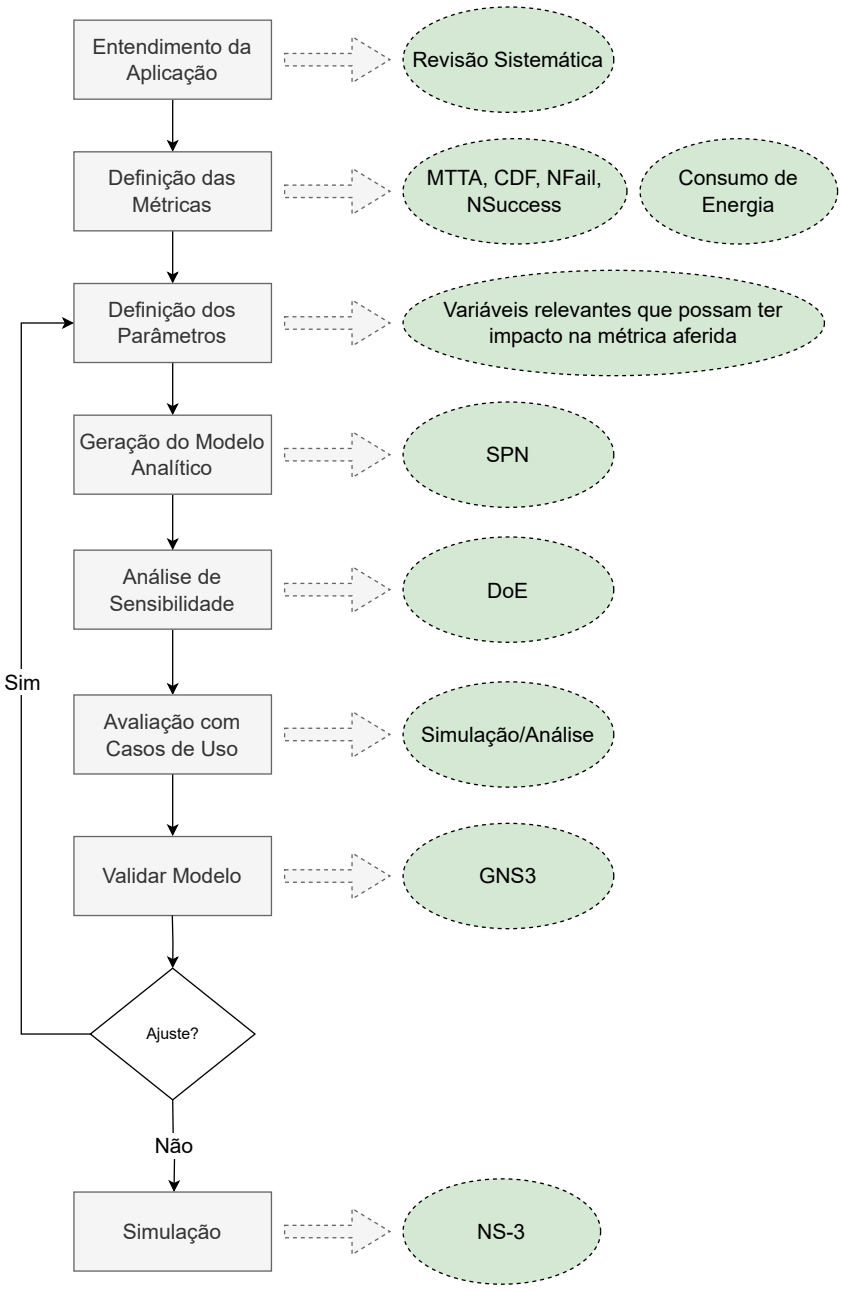


Figura 1 – Metodologia de desenvolvimento do modelo analítico.

2 Referencial Teórico

Este capítulo apresenta conceitos essenciais para a compreensão da proposta e trabalhos relacionados. Informações importantes sobre o funcionamento das *Network Functions Virtualization* (NFVs) e *Service Function Chain* (SFCs). Além disso, são apresentados conceitos sobre Redes de Petri Estocásticas.

2.1 *Network Functions Virtualization* (NFV) E *Service Function Chain* (SFC)

A NFV é uma tecnologia que surgiu para reduzir o acoplamento de hardware e software. A NFV implementa funções de rede em software executado em hardware comum, visando reduzir despesas de capital e operacionais em comparação com hardware personalizado. A NFV é padronizada pelo ETSI (*Industry Specification Group*) (BADULESCU; TRIAY, 2019). Os componentes estruturais da NFV são: Infraestrutura de Virtualização de Funções de Rede, *Network Functions* (NFs) e Gestão e Orquestração de Rede. A infraestrutura é composta por *hardware* (um único computador ou um *cluster* de computação) e *software* de estrutura, que oferece funcionalidades requeridas por NFs, tais como posicionamento de NFs, escalabilidade dinâmica, etc. As funções de rede consideradas para a implementação de *software* em *hosts* de *commodities* variam desde o envio básico de pacotes até funções complexas de *middlebox*, como sistemas de prevenção de intrusão (YI et al., 2018; VEERARAGHAVAN et al., 2017).

A infraestrutura NFV proposta pelo ETSI é composta por três elementos principais: EMS (*Element Management System*), VNF e NFVI (*Network Functions Virtualization Infrastructure*). O EMS gerencia um ou mais VNFs, que são funções de rede virtualizadas substituindo hardware tradicional e podem operar em VMs ou contêineres. Já o NFVI representa a base da arquitetura NFV, incluindo recursos de hardware (computação, armazenamento e rede), camada de virtualização e recursos virtualizados que os VNFs podem utilizar dinamicamente (BENEDICTIS; LIOY, 2017).

Os VNFs são o foco principal dos estudos de desempenho, pois otimizam o desempenho da NFV como um todo. O EMS, por sua vez, deve otimizar a gestão dos VNFs, garantindo uma comunicação eficiente entre eles. O NFVI desempenha um papel essencial na aceleração ao otimizar o caminho de entrega de pacotes, muitas vezes por meio de *switches* virtuais que integram inteligência ao *hipervisor*. A evolução do *software* de *switching* e aprimoramentos na entrega de pacotes são fundamentais para a eficiência da infraestrutura NFV (HUANG; CHOU, 2022).

As NFs que são executadas em *Virtual Machines* (VMs) ou contêineres são chamadas de *Virtual Network Functions* (VNFs). Os componentes MANO (*Management and Orchestration*) apresentam funções de gerenciamento, tais como: gerenciamento de falhas, gerenciamento de configuração, contabilidade, monitoramento de desempenho e segurança, e orquestradores, que gerenciam cadeias de serviços de diferentes NFs. A SFC surgiu pela necessidade de fornecer serviços de rede complexos e flexíveis de maneira eficiente e dinâmica. SFC é definido como uma sequência ordenada de VNFs e o encaminhamento subsequente de fluxos através delas para fornecer serviços de ponta a ponta (KAUR; MANGAT; KUMAR, 2020).

O SFC define uma arquitetura composta por elementos como a *Service Classification Function* (SCF), *Service Function* (SF), *Service Function Forwarder* (SFF) e o SFC *Proxy*, que se comunicam por meio da encapsulação de SFC. No plano de dados, o SCF classifica pacotes e define a sequência de funções de serviço, enquanto o SF executa funções específicas, podendo ou não reconhecer o encapsulamento SFC. O SFF encaminha pacotes entre SFs conforme as regras definidas, e o SFC *Proxy* atua como intermediário para SFs incompatíveis com o encapsulamento. O caminho percorrido pelos pacotes entre SFs e SFFs forma o *Service Function Path* (SFP), e a cadeia é encerrada no *Chain Termination Function* (CTF) (KAUR; MANGAT; KUMAR, 2020).

O plano de controle gerencia o fluxo de tráfego dentro do SFC e o ajusta dinamicamente, garantindo eficiência e adaptação às condições da rede. Ele interage com o plano de dados por meio de interfaces que definem regras de classificação, atualização de estados dos SFs e encaminhamento de pacotes. A integração do SFC com SDN e NFV amplia sua flexibilidade, reduzindo custos operacionais e de infraestrutura. O controlador SDN centraliza o controle das VNFs, aplicando regras dinâmicas para otimizar a rede, enquanto o plano de orquestração define estratégias globais para a alocação eficiente das VNFs e do tráfego (SANTOS et al., 2022).

A arquitetura ETSI NFV foi expandida com a adição do orquestrador de SFC, responsável por gerenciar *VNF Forwarding Graphs* (VNFFGs) e definir políticas para otimização do tráfego. Esse componente interage com o NFV e o controlador SDN para criar e atualizar cadeias de serviço de forma dinâmica. Além disso, o SFC permite escalabilidade automática com base no uso de *CPU* e memória, garantindo a adaptação eficiente dos serviços de rede a diferentes demandas, como o crescente tráfego da IoT. Essa abordagem melhora a eficiência das redes e amplia sua capacidade de atender a novos requisitos de forma flexível e automatizada (LIU et al., 2017).

2.2 *Stochastic Petri Net* (SPN)

As redes de Petri são uma ferramenta de modelagem gráfica e matemática que pode ser usada em diversos sistemas. São uma ferramenta para descrever e estudar sistemas de processamento de informações que se caracterizam como concorrentes, assíncronos, distribuídos, paralelos ou não determinísticos. Como uma ferramenta gráfica, PNs podem ser usados como um meio de comunicação visual semelhante aos fluxos gráficos, diagramas de blocos e redes. Além disso, essas redes utilizam *tokens* para simular atividades dinâmicas e concorrentes de sistemas. É possível configurar equações de estado, equações algébricas e outros modelos matemáticos que regem o comportamento de sistemas. De acordo com o trabalho inicial de Petri, diversas representações e extensões foram propostas, permitindo descrições mais claras e representando características de sistemas que não foram identificadas no início (MOLLOY, 1982; MARSAN et al., 1998; TRIVEDI, 2008; MARSAN, 1990).

Em um modelo SPN, os lugares são representados por círculos e as transições são representadas como retângulos vazios (transições temporizadas). Os arcos (arestas) conectam lugares a transições e vice-versa. Os *tokens* ou símbolos (pequenos círculos preenchidos) podem residir em lugares, que denotam o estado (ou seja, marcação) de SPN. Um arco inibidor é um arco especial que representa um pequeno círculo branco em uma borda, em vez de uma seta, e geralmente são usados para desativar as transições se houver *tokens* presentes em um lugar.

O comportamento de SPN é descrito como um fluxo de *token*. Os *tokens* são produzidos e destruídos de acordo com os disparos (ativação da transição) de transição. As mudanças imediatas representam atividades imediatas e são mais frequentes. A prioridade de disparo é maior do que a das transições temporais. Essas transições também podem conter uma condição de guarda e um usuário pode especificar uma prioridade de disparo diferente entre outras transições imediatas. Há também funções de vigilância em SPNs, que são expressões booleanas que controlam o disparo de uma transição, indicando uma condição em relação à marcação da rede. Se a função de guarda de uma transição produzir um valor verdadeiro, ela dispara, caso contrário, a função é desabilitada (MARSAN et al., 1998). A Figura 2 apresenta os componentes de um modelo SPN.

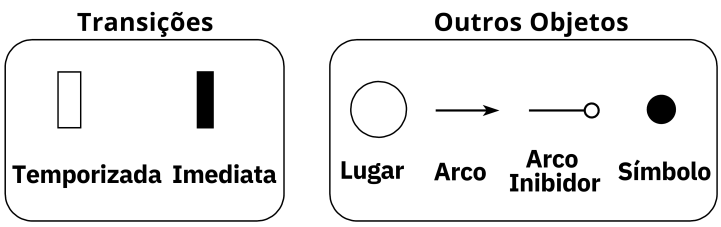


Figura 2 – Componentes SPNs.

Uma desvantagem das SPNs é que geralmente utilizam distribuições exponenciais para modelar tempos de transição. Isso pode ser uma limitação, pois alguns sistemas podem exigir distribuições diferentes (como uniformes, normais ou outras), que não são diretamente suportadas ou exigem extensões da teoria (THONG; AMEEDEEN, 2015).

2.3 Análise de Sensibilidade

A análise de sensibilidade analisa o impacto de dados de entrada específicos nos dados de saída para identificar falhas em sistemas computacionais. Em seguida, técnicas são usadas para aperfeiçoar esses sistemas em diversos cenários (CAMPOLONGO; TARANTOLA; SALTELLI, 1999) Em geral, os designers de sistemas utilizam análises de sensibilidade para avaliar o grau de sensibilidade de uma métrica em relação às alterações no modelo (SANTOS et al., 2021). Há alguns métodos para analisar a sensibilidade, como o *Design of Experiments* (DoE) ou a Diferença Percentual. A análise de regressão, a análise de perturbação, a simulação de Monte Carlo, a análise diferencial paramétrica, a análise de correlação e a diferença percentual estão disponíveis.

A escolha do método é desafiadora, pois é necessário considerar os recursos computacionais disponíveis e as características dos problemas abordados. A análise de sensibilidade, de certa forma, pode proporcionar a segurança necessária e encaminhar os resultados dentro da perspectiva pré-estabelecida pelos administradores do sistema. Neste trabalho, utilizamos técnicas de análise de sensibilidade baseadas no DoE.

Design of Experiments

O DoE é um conjunto de técnicas estatísticas que aperfeiçoam a compreensão do produto ou processo em questão. O DoE é uma técnica bastante eficaz para explorar novos processos e ter uma visão mais aprofundada dos processos existentes, seguida de uma otimização desses processos para atingir um desempenho de nível mundial. Na literatura especializada (FEITOSA et al., 2021; COSTA et al., 2016; SANTOS et al., 2021), são encontradas categorias de gráficos que são frequentemente usadas em experimentos com a abordagem DoE. O gráfico de efeito dos fatores, representado por barras dispostas em ordem crescente, enfatiza o impacto relativo de cada fator. Quanto maior a barra, maior o seu impacto, permitindo uma compreensão clara das influências de cada fator. A Figura 3 apresenta um gráfico do efeito dos fatores. O gráfico apresenta três fatores: A, B e C. O fator C é o fator de maior impacto.

O gráfico de efeito dos fatores nos permite identificar qual interação de fatores tem um impacto mais relevante no processo de otimização ou design do estudo, indicando onde a atenção deve ser direcionada. A compreensão das interações entre os fatores permite selecionar a melhor combinação de medidas, identificando padrões de efeitos cumulativos

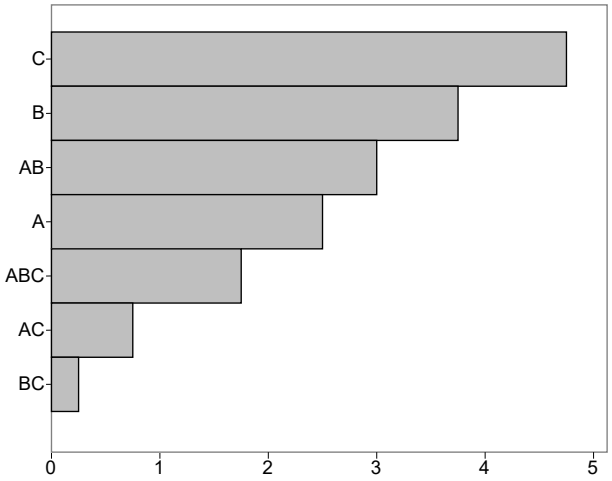


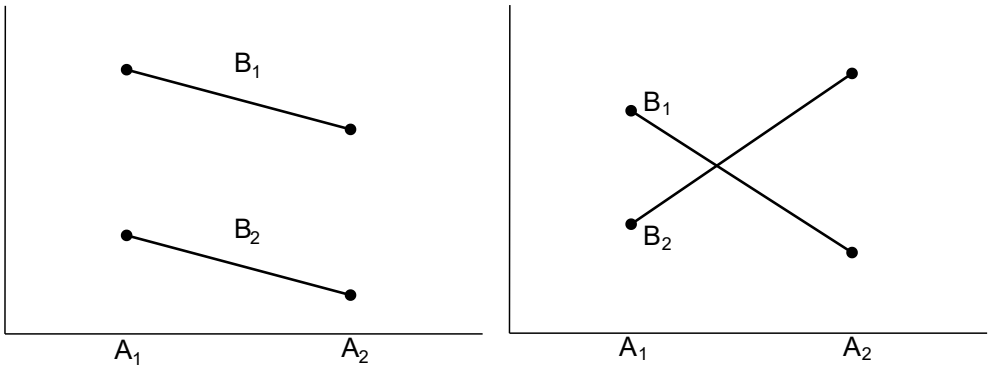
Figura 3 – Exemplo genérico de gráfico de efeito dos fatores.

ou degradantes entre os fatores. A interação entre os fatores A e B pode ser estimada usando a Equação 2.1.

$$I_{A,B} = \frac{1}{2}(E_{A,B(+1)} - E_{A,B(-1)}) \tag{2.1}$$

A Figura 4 (a) ilustra um exemplo em que não há interação entre os fatores, uma vez que as linhas são paralelas. A Figura 4 (b) ilustra um exemplo de interação entre fatores, pois as linhas se intersectam. O objetivo dos gráficos de interação é estabelecer relações entre fatores. Uma interação ocorre quando a influência de um fator no resultado é ampliada (ou reduzida) pela variação nos níveis de outro fator. Se as linhas no gráfico forem paralelas, isso significa que os fatores não estão relacionados. Se as linhas não forem paralelas, isso indica uma interação significativa entre os fatores em questão. A variação de uma métrica para o fator A no nível A1 é maior do que a variação no nível A2. As alterações nos níveis do fator A em uma métrica dada indicam uma dependência entre os níveis do fator A e os níveis do fator B. O gráfico {A,B(+1)} representa o efeito do fator A no nível mais elevado do fator B, enquanto o {A,B(-1)} representa o efeito do fator A no nível mais baixo do fator B.

O DoE foi escolhido por ser um método robusto e sistemático para realizar uma análise de sensibilidade, permitindo explorar a influência de múltiplos fatores simultaneamente e avaliar suas interações de maneira eficiente. Diferente de abordagens tradicionais, como a variação de um fator por vez, o DoE proporciona uma visão mais abrangente do impacto das variáveis de entrada sobre as saídas do sistema, reduzindo o número de experimentos necessários e otimizando o tempo e os recursos computacionais.



(a) Baixa interação entre fatores. (b) Alta interação entre fatores.

Figura 4 – Gráficos de interação genéricos.

3 Revisão Sistemática de Literatura em Consumo Energético em Arquiteturas de Microsserviço

Este capítulo apresenta uma revisão sistemática da literatura sobre consumo energético de microsserviços. O capítulo aborda o consumo energético em microsserviços sob várias perspectivas, como métricas, tipos de arquitetura e métodos de avaliação.

3.1 Trabalhos Relacionados

Esta seção apresenta uma amostra de trabalhos relacionados com abordagens ou contextos semelhantes a este trabalho. Alguns trabalhos recentes analisaram a avaliação de desempenho de arquiteturas de microsserviços utilizando simulações através da ferramenta *CloudSim* (GHOLIPOUR; ARIANYAN; BUYYA, 2020) (KHAN et al., 2021a). Outros trabalhos utilizaram a métrica de elasticidade para avaliar o gerenciamento de energia(SAENGKAENPETCH; ASWAKUL, 2021) (LIN et al., 2018a). Por meio de modelagens, alguns outros trabalhos realizaram a verificação da confiabilidade (LI et al., 2021) (LYU et al., 2020a). Diversos estudos têm mostrado que as arquiteturas de microsserviços podem demandar altos custos energéticos para se manterem disponíveis, como, por exemplo: escalabilidade (KHAIRY et al., 2022), desempenho (BOUAOUDA; AFDEL; ABOUNACER, 2022a), disponibilidade (JARWAR; ALI; CHONG, 2019), confiabilidade (LI et al., 2021), escalabilidade (VALERA et al., 2021a) e sustentabilidade (KUMAR et al., 2019a). Tais estudos trazem à tona questões importantes como os objetivos e os diferentes tipos de arquitetura de consumo energético em microsserviços.

A Tabela 1 apresenta os trabalhos relacionados. Os trabalhos foram classificados considerando o contexto, os critérios e o tipo de estudo. O critério de classificação dos trabalhos relacionados nomeado como **contexto**, refere-se ao tema central abordado nos trabalhos. Entre os trabalhos relacionados, o contexto principal envolve a arquitetura de microsserviços e o seu uso prático. A coluna apresentada como **critérios**, enuncia os parâmetros utilizados nos trabalhos relacionados. Estes critérios dizem respeito à classificação geral dos resultados encontrados nas obras. Os critérios principais incluem os seguintes aspectos: contexto, segurança, desafios, métricas, métodos e ferramentas. O critério de classificação nomeado como **tipo de estudo**, permite identificar os trabalhos relacionados que são do tipo mapeamento sistemático, revisão sistemática ou *surveys*.

Vural *et al.* (VURAL; KOYUNCU; GUNEY, 2017) abordam um estudo de mapea-

Tabela 1 – Trabalhos relacionados.

Trabalhos	**Ano**	**Contexto**	**Critério**	**Tipo de Estudo**
(VURAL; KOYUNCU; GUNEY, 2017)	2017	Microsserviço	Contexto/prática motivações/ferramentas	Mapeamento sistemático
(GHOFRANI; LÜBKE, 2018)	2018	Estado da prática e desafios de coleta em microsserviços	Desafios/Soluções	*Survey*
(VIGGIATO et al., 2018)	2018	Microsserviços na prática	Contexto	*Survey*
(SOLDANI; TAMBURRI; HEUVEL, 2018)	2018	Aplicação industrial de microsserviços	Contexto	Revisão sistemática
(YU et al., 2019)	2019	Comunicação de serviços de microsserviços	Problemas de segurança	*Survey*
(FRANCESCO; LAGO; MALAVOLTA, 2019)	2019	Arquitetura com microsserviços	Contexto/problemas	Mapeamento sistemático
(WASEEM; LIANG; SHAHIN, 2020)	2020	Arquitetura de microsserviços em *DevOps*	Métodos/padrões de *design*/ atributos/ferramentas de qualidade	Mapeamento sistemático
(HANNOUSSE; YAHIOUCHE, 2021a)	2021	Segurança em arquiteturas de microsserviços	Ameaças à segurança/ técnicas e ferramentas	Mapeamento sistemático
(JAWADDI; JOHARI; ISMAIL, 2022)	2022	Escalonamento automático de microsserviços com perspectiva de verificação formal	Técnicas de escalonamento automático/ métricas/desafios	Revisão sistemática
Este capítulo	2023	Consumo de energia em microsserviços	Contexto/Arquiteturas/ Métodos/métricas avaliados	Revisão Sistemática

mento sistemático sobre microsserviços. O objetivo é descobrir as tendências atuais no que diz respeito a microsserviços. Ghofrani *et al.* (GHOFRANI; LÜBKE, 2018) apresentam os resultados de uma pesquisa empírica para avaliar o estado atual da prática e identificar os desafios enfrentados na arquitetura de microsserviços. Embora diversos estudos de mapeamento tenham sido realizados para investigar os desafios relacionados aos microsserviços na literatura, a maioria deles não aborda claramente os desafios envolvidos na concepção, desenvolvimento e manutenção de sistemas baseados em microsserviços no que diz respeito ao consumo de energia. Portanto, ainda há uma lacuna na compreensão desses desafios e na forma como eles podem afetar o desempenho e a eficiência energética desses sistemas.

Viggiato *et al.* (VIGGIATO et al., 2018) relatam os resultados de uma pesquisa com 122 profissionais envolvidos em projetos de microsserviços. Os autores examinam como a indústria está aplicando essa arquitetura e se a percepção dos profissionais sobre as vantagens e desafios dos microsserviços corresponde àquela descrita na literatura existente. Soldani *et al.* (SOLDANI; TAMBURRI; HEUVEL, 2018) propõe realizar uma análise

sistemática da literatura não publicada (literatura cinza) da indústria sobre microsserviços, a fim de identificar as objeções técnicas/operacionais e os ganhos do estilo arquitetônico baseado em microsserviços. Yu *et al.* (YU et al., 2019) apresentam uma pesquisa sobre os riscos de segurança que ameaçam aplicações baseadas em microsserviços em nuvem. Devido à natureza da arquitetura de microsserviços, que consiste em serviços e comunicação frequente, destacam-se problemas de segurança na comunicação entre serviços em quatro aspectos: contêineres, dados, permissões e rede. Os autores propõem uma solução ideal para questões de segurança na comunicação de serviços em aplicações baseadas em microsserviços em nuvem. Di Francesco *et al.* (FRANCESCO; LAGO; MALAVOLTA, 2019) buscam identificar, categorizar e avaliar o estado da arte da arquitetura com microsserviços sob as seguintes perspectivas: tendências de publicação, áreas de pesquisa e potencial de adoção.

Segundo os autores, aspectos da arquitetura com microsserviços ainda são inexplorados. Waseem *et al.* (WASEEM; LIANG; SHAHIN, 2020) visa realizar uma análise sistemática da literatura sobre a utilização da arquitetura de microsserviços (AMS) em *DevOps*, visando identificar, classificar e avaliar o estado da arte dessa aplicação. Hannousse *et al.* (HANNOUSSE; YAHIOUCHE, 2021a) propõe fornecer um guia útil para desenvolvedores sobre as ameaças reconhecidas em microsserviços e como elas podem ser detectadas, mitigadas ou evitadas. O objetivo é identificar possíveis lacunas de pesquisa na proteção de microsserviços arquitetura (AMS). Jawaddi *et al.* (JAWADDI; JOHARI; ISMAIL, 2022) apresentam uma revisão complementar da literatura existente com enfoque no dimensionamento automático com perspectivas de verificação. O estudo destaca as contribuições recentes em microsserviços, verificação e escalonamento automático publicados entre 2017 e 2022.

O trabalho de Ghofrani *et al.* (GHOFRANI; LÜBKE, 2018) apresenta um *survey* sobre os desafios da arquitetura de microsserviços e discute os principais desafios enfrentados na adoção de uma arquitetura de microsserviços, como a complexidade da integração de serviços, a gestão de dados distribuídos, a manutenção e o monitoramento de múltiplos serviços. O artigo de Viggiato *et al.* (VIGGIATO et al., 2018) apresenta um estudo de survey sobre a adoção e prática de arquiteturas de microsserviços e aborda os principais motivos para a adoção de microsserviços, os principais desafios enfrentados durante a implementação e as estratégias adotadas para lidar com esses desafios. Yu *et al.* (YU et al., 2019) apresenta um *survey* sobre questões de segurança na comunicação de serviços de aplicações névoa habilitadas por microsserviços.

O estudo de Vural *et al.* (VURAL; KOYUNCU; GUNEY, 2017) apresenta um estudo de mapeamento sistemático e discute os principais motivos para a adoção de microsserviços, as características e benefícios de arquiteturas de microsserviços, bem como as principais ferramentas e tecnologias utilizadas na implementação de microsserviços.A pesquisa de Soldani *et al.* (SOLDANI; TAMBURRI; HEUVEL, 2018) aborda mapeamento

sistemático, onde apresenta os principais desafios enfrentadas na adoção de microsserviços, incluindo a complexidade de gerenciamento de várias instâncias de serviços, a necessidade de uma infraestrutura de comunicação robusta e a dificuldade de manter a coesão entre serviços. O artigo de Di Francesco *et al.* (FRANCESCO; LAGO; MALAVOLTA, 2019) apresenta um estudo de mapeamento sistemático sobre as tendências de pesquisa em microsserviços, incluindo a evolução da pesquisa ao longo do tempo, as principais áreas de pesquisa e as principais metodologias e técnicas utilizadas na pesquisa em microsserviços.

O estudo de Waseem *et al.* (WASEEM; LIANG; SHAHIN, 2020) apresenta um mapeamento sistemático sobre a adoção da arquitetura de microsserviços na prática do *DevOps*. O estudo também aborda os principais desafios e benefícios da adoção de microsserviços em *DevOps*, bem como as principais ferramentas e técnicas utilizadas para a implementar. O trabalho de Hannousse *et al.* (HANNOUSSE; YAHIOUCHE, 2021a) apresenta um estudo de mapeamento sistemático sobre a segurança em microsserviços e arquiteturas baseadas em microsserviços. O estudo de Jawaddi *et al.* (JAWADDI; JOHARI; ISMAIL, 2022) evidencia as principais abordagens para o auto-escalonamento de microsserviços, bem como as ferramentas e técnicas de verificação formal utilizadas para garantir a sua eficácia.

Vural *et al.* em (VURAL; KOYUNCU; GUNEY, 2017) e (FRANCESCO; LAGO; MALAVOLTA, 2019) se concentram nas arquiteturas de microsserviços, procurando identificar tendências, áreas de pesquisa e potencial de adoção. Já os trabalhos de Ghofrani *et al.* (GHOFRANI; LÜBKE, 2018) e Viggiato *et al.* (VIGGIATO et al., 2018) visam estudar os microsserviços, na prática, buscando compreender os desafios enfrentados. Este trabalho é único em sua abordagem sobre o consumo de energia nas arquiteturas de microsserviços.

Os trabalhos de Vural *et al.* (VURAL; KOYUNCU; GUNEY, 2017), Viggiato *et al.* (VIGGIATO et al., 2018), Soldani *et al.* (SOLDANI; TAMBURRI; HEUVEL, 2018) e Di Francesco *et al.* (FRANCESCO; LAGO; MALAVOLTA, 2019) abordam o contexto no qual os microsserviços são utilizados. Já, Yu *et al.* (YU et al., 2019) e Hannousse *et al.* (HANNOUSSE; YAHIOUCHE, 2021a) discutem a questão da segurança na arquitetura de microsserviços. Ghofrani *et al.* (GHOFRANI; LÜBKE, 2018) e Jawaddi *et al.* (JAWADDI; JOHARI; ISMAIL, 2022) examinam os desafios relacionados aos microsserviços. Jawaddi *et al.* (JAWADDI; JOHARI; ISMAIL, 2022) também apresenta as métricas utilizadas neste campo. Waseem *et al.* (WASEEM; LIANG; SHAHIN, 2020) descreve os métodos utilizados em seu estudo. Vural *et al.* (VURAL; KOYUNCU; GUNEY, 2017), Waseem *et al.* (WASEEM; LIANG; SHAHIN, 2020) e Hannousse *et al.* (HANNOUSSE; YAHIOUCHE, 2021a) também apresentam as ferramentas empregadas em seus respectivos trabalhos. Este trabalho é o único em sua abordagem ao examinar as arquiteturas de nuvem nas quais os microsserviços são utilizados.

3.2 Metodologia

A metodologia de revisão sistemática utilizada neste capítulo tem como inspiração o estudo de (HANNOUSSE; YAHIOUCHE, 2021b). Conforme ilustrado na Figura 5, a pesquisa apresenta as seguintes etapas: definição do escopo, busca inicial, filtragem, classificação e extração de informação. No escopo do capítulo, foram elaboradas questões norteadoras das buscas a fim de obter-se um resultado mais preciso diante do tema a ser estudado. Ao final, mostramos os resultados obtidos.



Figura 5 – Fluxograma da metodologia aplicada na revisão sistemática.

Definição de Escopo:

A primeira etapa é dedicada à definição do escopo que contém as questões que norteiam a pesquisa. As perguntas definem um limite para o escopo do trabalho, ou seja, um limite de área de estudo a ser mapeada. Posteriormente, as strings de busca são utilizadas para a escolha dos artigos. Vale ressaltar que, antes da elaboração das questões, houve um consenso para analisar temas úteis para orientar a pesquisa sobre "*energy in microservice*". Portanto, este trabalho propõe três questões com o intuito de delimitar o escopo do trabalho:

- Q1: quais são os diferentes objetivos relacionados ao consumo energético em microsserviços? O objetivo desta questão é identificar o propósito real das pesquisas quanto ao consumo energético em microsserviços.
- Q2: quais são as diferentes arquiteturas relacionados ao consumo energético em microsserviços? O objetivo desta questão é identificar nas pesquisas, quais tipos de arquiteturas estão sendo utilizadas para lidar com consumo energético em microsserviços.
- Q3: quais são as diferentes métricas de avaliação relacionados ao consumo energético em microsserviços? O objetivo desta questão é observar as variadas métricas utilizadas em pesquisas para avaliação de consumo energético em microsserviços.

Busca:

A segunda etapa da pesquisa trata da busca. Dentro desta etapa foi aplicado um termo de busca em cinco motores de Busca. Logo em sequência, foi realizada a coleta dos

artigos retornados. Para gerar a *string* de busca, utilizamos palavras-chave que tivessem maior relação com o tema "*Energy in microservices*". Após analisar os termos, identificamos que seria necessário usar a *string* "*energy*" como primordial para a busca, enquanto as *strings* "*microservices*" e "*container*" poderiam ser usadas isoladamente, mas uma dessas duas deveria estar acompanhada de "*energy*". A *string* de busca foi ajustada de acordo com algumas bibliotecas: *IEEE Xplore*, *Elsevier*, *Springer*, *ACM* e *Science Direct*. Após a pesquisa e obtenção dos artigos, foi feito um processo de filtragem que pode ser visto na Figura 6. Vale lembrar também que a *string* de busca foi aplicada utilizando a opção de busca avançada disponibilizada pelas motores de busca. Nesse caso em especial, a busca foi ajustada para encontrar a *string* no título e no resumo. A *string* de busca utilizada foi *energy AND (microsservices OR container)*.

Filtragem:

A terceira etapa é sobre a filtragem. A estratégia adotada também considerou critérios de exclusão direta presentes na Tabela 2. A maneira adotada para inclusão dos artigos foi verificar se os artigos não estavam nos critérios de exclusão utilizados. Ou seja, se o artigo não atender aos motivos de exclusão direta, automaticamente o artigo é incluído e avança para as próximas fases de filtragem. A Figura 6 apresenta o fluxograma utilizado para aplicar os critérios de exclusão, e ilustra todas as fases de filtragem executadas nos 3625 artigos levantados. O fluxograma mostra o número de artigos presentes e o número de artigos excluídos com seus respectivos motivos de exclusão em cada fase da filtragem.

Tabela 2 – Critérios de exclusão direta

Critérios de exclusão direta
O trabalho não está em inglês.
Trabalho com menos de seis páginas.
Artigos não científicos.
Ser uma Pesquisa/Mapeamento/Revisão.
Trabalho não é uma questão de consumo de energia em microsserviços.

Na primeira parte de filtragem houve a remoção dos artigos através da análise dos títulos dos trabalhos. A remoção dos artigos foi baseada em alguns critérios de exclusão definidos, como: título não informativo, fora do escopo, duplicados, não é artigo: removendo assim 3436 artigos. A segunda parte foi observar os resumos dos artigos, para ver se estavam no escopo do trabalho, excluindo um total de 139 obras que não estavam no escopo do trabalho. A terceira parte foi analisar todos os 50 artigos restantes por meio da leitura completa dos mesmos, com base na utilização dos critérios de exclusão direta decididos inicialmente (presentes na tabela 2), restando assim 37 artigos. Vale ressaltar que trabalhos fora de escopo foram encontrados em diferentes fases, pois título e resumos

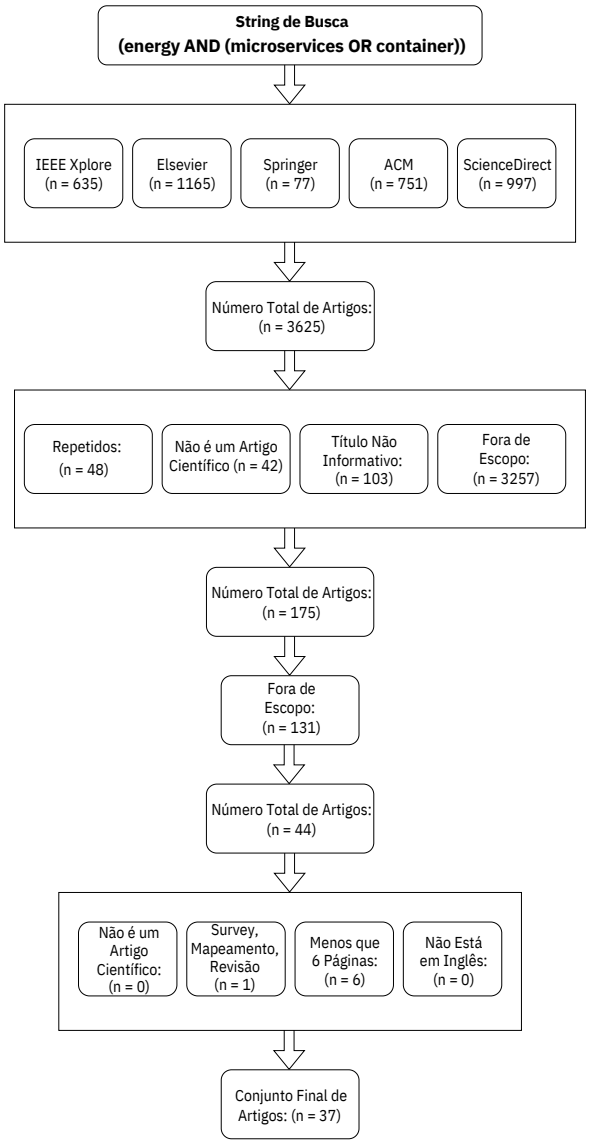


Figura 6 – Fluxograma que demonstra as etapas do processo de filtragem.

foram analisados em etapas diferentes. A etapa seguinte é a classificação, onde os trabalhos restantes foram classificados de acordo com contexto, métricas macro, método de avaliação, tipo de arquitetura, orquestrador e modelo de implantação de cada estudo.

Classificação:

A quarta etapa aborda a classificação, onde foi feita a categorização a fim de extrair informações mais precisas e identificar lacunas de pesquisa. A etapa de classificação dos trabalhos foi dividida em duas fases. A primeira fase consiste na definição dos critérios de classificação, e a segunda trata das classificações desses critérios. Na primeira fase foram definidos seis critérios: contexto, métricas macro, método de avaliação, tipo de arquitetura, orquestrador e modelo de implantação. Na segunda fase, foram criadas classificações presentes para cada categorização. Todas as categorizações utilizadas podem ser vistas

na Tabela 3 e foram estabelecidas com base nos artigos mapeados em definições prévias realizadas pelos autores.

Tabela 3 – Categorização

Classificação Geral	**Classificação**
Contexto	Data Center
	IoT
	Não Informado
Métricas Macro	Desempenho
	Elasticidade
	Escalabilidade
	Confiabilidade
	Sustentabilidade
	Disponibilidade
Método de Avaliação	Simulação
	Mensuração
	Modelagem
Tipo de arquitetura	Computação em Nuvem
	Computação em Névoa
	Computação em Nuvem e Névoa
Orquestrador	Kubernetes
	Docker Swarm
	Não Informado
Implantação	Privado
	Público
	Privado e Público
	Privado e Público e Híbrido
	Não Informado

Extração de Informações:

A quinta etapa trata da extração de informações. Neste estudo, a forma de extrair informações acerca dos artigos foi através da leitura direta dos trabalhos a serem mapeados. O conjunto final de artigos foi lido completamente e analisado, extraindo métricas usadas no trabalho, método de avaliação utilizado, bem como tipo de arquitetura, ferramentas, orquestrador e modelo de implantação. As obras foram classificadas a fim de se obter os principais objetivos de cada uma, para um melhor entendimento do âmbito em que estavam inseridas.

3.3 Resultados da Classificação Geral

Esta seção apresenta os resultados obtidos. Os resultados estão organizados conforme as classificações mencionadas anteriormente.

Contexto

A Figura 7 ilustra o diagrama de Venn que apresenta a classificação dos contextos em que a arquitetura de microsserviços combinada com contêineres é utilizada para otimizar o consumo de energia. Esse diagrama fornece uma visão geral dos diferentes contextos em que essa tecnologia foi aplicada na busca por soluções mais eficientes no gerenciamento de recursos energéticos. A análise do contexto em que microsserviços e contêineres são utilizados é crucial, pois cada ambiente tem suas características e requisitos únicos. Os resultados mostram que 83,8% dos trabalhos informam o contexto utilizado, enquanto 16,2% não fornecem esta informação, pois apresentam propostas de eficiência energética para sistemas genéricos de microsserviços. A análise dos dados mostra dois contextos principais: Data Centers e a *Internet of Things* (IoT).

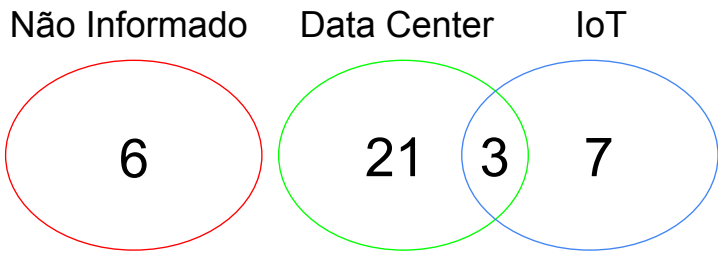


Figura 7 – Diagrama de Venn com a classificação dos contextos mapeados.

O trabalho Gholipoura *et al.* (GHOLIPOUR; ARIANYAN; BUYYA, 2020) propõe uma solução proposta para *Virtual Machine* (VM) conjunta e gerenciamento de recursos de contêiner em data centers em nuvem. Os resultados obtidos mostram reduções notáveis no consumo de energia. O trabalho de Silva de Souza *et al.* (SOUZA et al., 2020) destaca que a otimização da operação de *software* é crucial para tornar a computação em nuvem verde e acessível devido aos impactos financeiros e ambientais crescentes dos data centers. No entanto, a complexidade da operação do software aumentou devido ao espaço de configuração ampliado e à maior heterogeneidade das arquiteturas de data center e nuvem atuais. O trabalho de Ferreira *et al.* (FERREIRA et al., 2022) apresenta um *middleware* para gerenciamento de energia baseado no conceito de computação em borda para residências inteligentes. Já o trabalho de Raza *et al.* (RAZA et al., 2021) visa estabelecer diretrizes e identificar fatores críticos para a implementação de serviços em contêineres em dispositivos IoT com recursos limitados.

Data Center

Um *data center* é uma instalação que consiste em servidores, armazenamento e dispositivos de rede, sistemas de distribuição de energia e sistemas de refrigeração (BARI et al., 2013). Os *data centers* são geralmente utilizados por organizações para armazenar uma grande quantidade de dados e fornecer serviços de processamento de dados. As arquiteturas baseadas em microsserviços oferecem geralmente uma série de vantagens para os Data Centers, podem ajudar a reduzir a complexidade do gerenciamento de múltiplas aplicações, facilitam a escala de aplicações conforme necessário, aumentando a disponibilidade das aplicações e proporcionando maior flexibilidade. O trabalho de Khairy *et al.* (KHAIRY et al., 2022) propõe uma técnica chamada SIMR, que permite que múltiplas solicitações sejam processadas em um único ciclo de processamento. A abordagem reduz o número de instruções executadas pelo processador e minimiza o tráfego de rede, resultando em uma redução significativa no consumo de energia do data center. O trabalho de Silva de Souza *et al.* (SOUZA et al., 2020) descreve o desenvolvimento de uma ferramenta chamada *Containergy*, que permite a medição de energia e desempenho de cargas de trabalho em contêineres em *data centers*.

O trabalho de Lin *et al.* (LIN et al., 2018b) apresenta um método para a alocação e implantação eficiente de contêineres em data centers, visando a redução do consumo de energia. O trabalho de Kaur *et al.* (KAUR et al., 2017) propõe uma estratégia de colocação de contêineres chamada EnLoB para data centers. A estratégia considera tanto o equilíbrio de carga quanto a eficiência energética durante a colocação de contêineres em servidores em um data center. O objetivo é reduzir o consumo de energia e melhorar o desempenho dos servidores. O trabalho de Patel *et al.* (PATEL; PATRA; SAHOO, 2020) apresenta um algoritmo genético eficiente em termos de energia para consolidação de contêineres em sistemas em nuvem. O objetivo é reduzir o consumo de energia em data centers que utilizam contêineres para virtualização de recursos. Os estudos como o de Bouaouda *et al.* (BOUAOUDA; AFDEL; ABOUNACER, 2022b) e Khan *et al.* (KHAN et al., 2021b) estudam a melhor maneira de reduzir o consumo de energia em Data Centers para garantir o desempenho sem desperdiçar recursos.

IoT

A IoT é uma tecnologia que está revolucionando como interagimos e controlamos o ambiente físico. A IoT consiste em dispositivos inteligentes que se comunicam entre si, permitindo que esses dispositivos coletem e troquem dados (AL-SARAWI et al., 2017). A utilização de microsserviços na IoT pode auxiliar no rastreamento e monitoramento do consumo de energia, permitindo tomadas de decisões mais eficientes e sustentáveis do ponto de vista energético. O trabalho de Lennick *et al.* (LENNICK; AZIM; LISCANO, 2021a) apresenta uma arquitetura baseada em microsserviços para avaliação de desempenho e

consumo de energia de distribuições do Linux hospedadas em contêineres *Docker* em dispositivos de IoT.

O trabalho de Jawar *et al.* (JARWAR; ALI; CHONG, 2019) destaca que, na infraestrutura de gerenciamento de dados de energia suportada pela IoT, objetos de terminais de geração e consumo de energia em edifícios produzem uma abundância de dados. Os autores propõem o uso de modelagem de microsserviços de processamento de dados para aumentar a disponibilidade de dados e oferecer capacidades de serviços por meio de microsserviços para o gerenciamento da eficiência energética.

Métricas Macro

Os trabalhos utilizaram uma quantidade de métricas relevantes, portanto dividimos em dois grupos: métricas macro e métricas micro. Métricas macro são grupos de métricas gerais e métricas micro são métricas que pertencem a algum grupo de métricas macro. Os microsserviços são cada vez mais utilizados devido à sua alta flexibilidade. No entanto, é importante monitorar a qualidade e o seu desempenho. Ao estabelecer métricas, é possível avaliar o comportamento, identificar possíveis problemas e planejar melhorias para os serviços. A Figura 8 apresenta as métricas macro dos artigos mapeados. A partir da revisão, foram categorizados cinco tipos de métricas: desempenho, presente em 59,5% dos artigos, elasticidade, presente em 29,7% dos artigos, confiabilidade, presente em 5,4% dos artigos, sustentabilidade, presente em 2,7% dos artigos, e disponibilidade, presente em 2,7% dos artigos.

A Figura 9 apresenta uma taxonomia das métricas com métricas macro e métricas micro. O uso da arquitetura de microsserviços e de contêineres é amplamente reconhecido como uma forma de melhorar o desempenho de sistemas de *software*. No entanto, é importante medir a eficiência do sistema, especialmente quanto ao consumo de energia de forma geral. A análise de desempenho é um exemplo disso. Pois é fundamental para identificar pontos de melhoria e otimizar o funcionamento do sistema. O estudo de Bouaouda *et al.* (BOUAOUDA; AFDEL; ABOUNACER, 2022b) aborda o consumo de energia em um *data center* usando o simulador *CloudSim*, com ênfase na colocação de contêineres nos *hosts*, comparando o desempenho de dois algoritmos para minimizar o consumo de energia em sistemas de nuvem.

A elasticidade é uma métrica que se concentra em como ajustar eficientemente o uso de recursos conforme a variação da demanda de trabalho, podendo assim melhorar a eficiência energética. Os trabalhos Fontana de Nardin *et al.* (Fontana de Nardin et al., 2021) e Ferreira *et al.* (FERREIRA et al., 2022) exploram como melhorar o consumo de energia ao manipular periodicamente a quantidade mais adequada de recursos para executar uma aplicação. Ambos os trabalhos têm como foco prover um modelo para gerenciar de maneira eficiente os recursos e minimizar o consumo de energia. O trabalho Valera *et al.* (VALERA

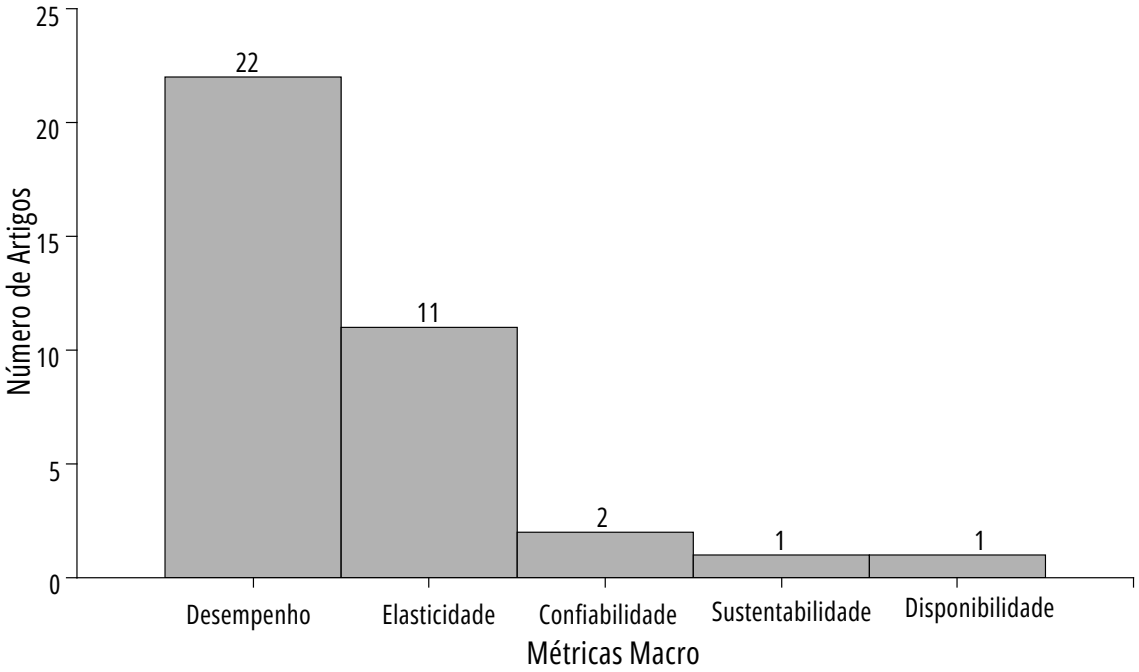


Figura 8 – Classificação das métricas.

et al., 2020) apresenta um simulador capaz de implantar e gerenciar qualquer tipo de rede e dispositivos heterogêneos para executar aplicativos distribuídos baseados em serviços ou microsserviços.

A confiabilidade é um fator importante a ser explorado. Uma falha em um único microsserviço ou contêiner pode afetar a disponibilidade e a confiabilidade do sistema na totalidade. O uso de microsserviços e contêineres pode aumentar a complexidade do sistema, o que pode dificultar garantir a confiabilidade do consumo de energia. O trabalho Lyu *et al.* (LYU et al., 2020b) propõe uma modelo tolerante a k-falhas para melhorar a confiabilidade do sistema de gerenciamento de energia.

A sustentabilidade no consumo de energia é uma precaução em sistemas baseados em microsserviços e contêineres, o aumento de consumo energético nesses sistemas é devido ao grande número de máquinas virtuais e contêineres em execuções nesses sistemas. O trabalho Kumar *et al.* (KUMAR et al., 2019b) apresenta um esquema para classificação e agendamento de trabalhos multi-indexados com consciência energética renovável, utilizando contêineres como serviço para maximizar a eficiência energética e manter a sustentabilidade de data centers. Essa abordagem visa minimizar o consumo de energia em data centers, responsáveis por grande parte do consumo mundial de energia elétrica. Ao utilizar contêineres como serviço, é possível otimizar o uso de recursos energéticos e reduzir o impacto ambiental desses sistemas. Dessa forma, a sustentabilidade no consumo de energia torna-se uma preocupação essencial para garantir a eficiência e a viabilidade de

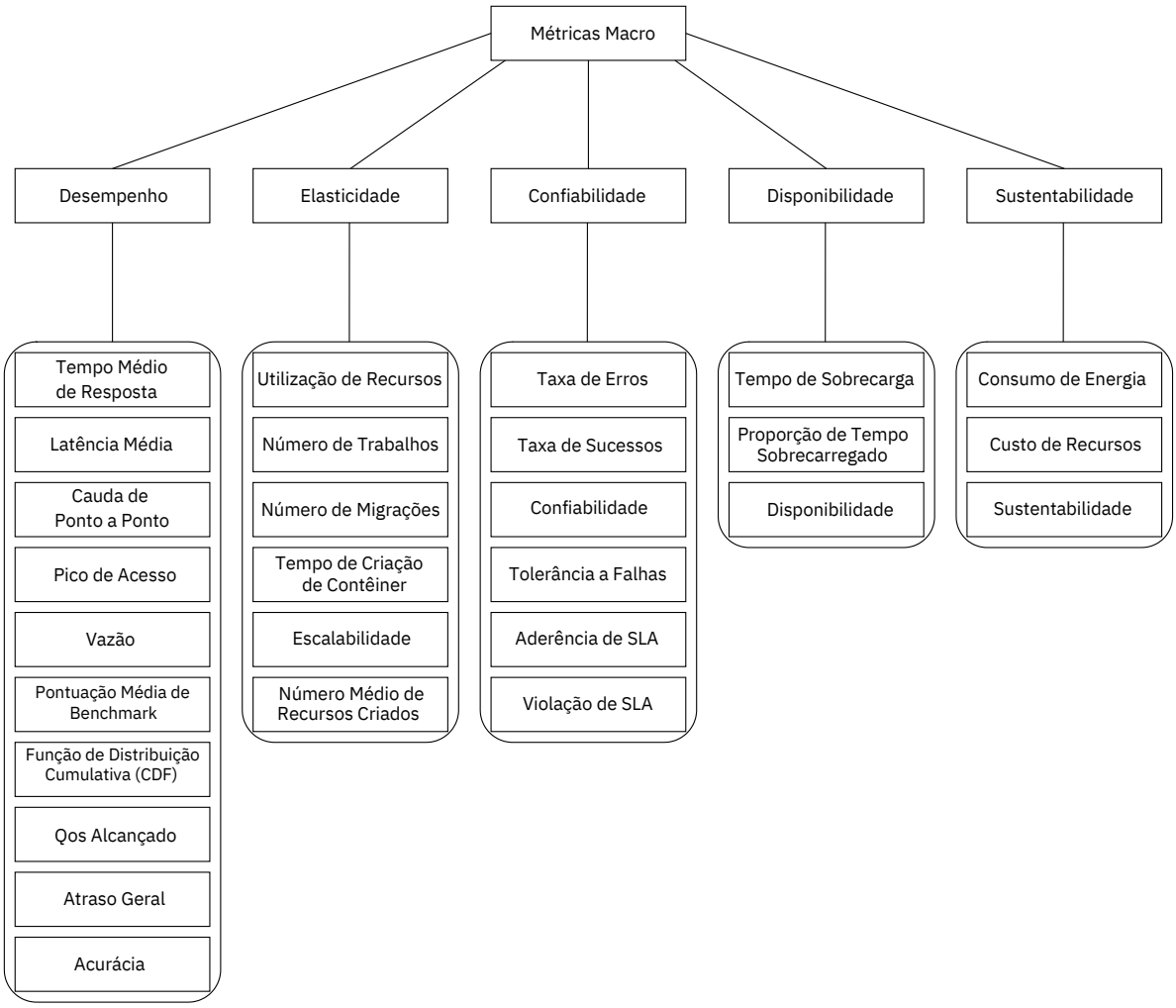


Figura 9 – Taxonomia de métricas.

sistemas baseados em microsserviços e contêineres no longo prazo.

A disponibilidade é um fator crítico para garantir o desempenho e a confiabilidade de um sistema baseado em microsserviços e contêineres. No entanto, pode ser afetada pelo consumo de energia, especialmente em sistemas que possuem alta demanda de processamento. O trabalho Jawar *et al.* (JARWAR; ALI; CHONG, 2019) sugere o uso de modelagem de microsserviços de processamento de dados para aumentar a disponibilidade de dados e expor capacidades de serviços com microsserviços para gerenciamento de eficiência energética. A seguir, as macros métricas são discutidas de forma mais detalhada.

Desempenho:

O uso de microsserviços em data centers permite o gerenciamento eficiente de recursos. A melhoria do desempenho é crucial para garantir que a aplicação possa lidar com cargas de trabalho em constante mudança, garantindo níveis de tempo de resposta

e vazão desejados. Os artigos analisados mostram que uma grande parte deles têm se concentrado nessas abordagens para garantir um desempenho ideal. O trabalho de Ranjan *et al.* (RANJAN et al., 2020) aborda a redução do consumo de energia em data centers em nuvem de grande escala, usando técnicas de virtualização baseadas em contêineres para otimizar o agendamento de fluxos de trabalho. As métricas utilizadas para avaliar o desempenho incluem o consumo de energia, o tempo de execução do fluxo de trabalho e a utilização de recursos. Os autores propõem um algoritmo de agendamento baseado em nuvem que considera a energia consumida pelo hardware subjacente e minimiza o consumo de energia durante a execução do fluxo de trabalho.

O trabalho de Mendes *et al.* propõe um algoritmo avançado de escalonamento de sobre-assinatura, com uma extensão para orquestrador de contêineres. As métricas utilizadas para avaliar o sistema incluem o tempo de resposta, a utilização de CPU, o consumo de energia e a eficiência energética. O objetivo é melhorar a utilização de recursos, a fim de atingir níveis onde a eficiência energética é maximizada. O estudo de Piraghaj *et al.* (PIRAGHAJ et al., 2015) propõe um framework e um algoritmo para a consolidação eficiente de contêineres em data centers em nuvem, visando a redução do consumo de energia. As métricas utilizadas para avaliar o desempenho do algoritmo incluem a utilização de CPU, o consumo de energia e o número de máquinas físicas utilizadas. O objetivo é minimizar o número de máquinas físicas necessárias para executar um conjunto de contêineres, a fim de reduzir o consumo de energia.

Elasticidade:

Na arquitetura de microsserviços, melhorar a elasticidade é um princípio fundamental, por permitir que o sistema automaticamente ajuste sua capacidade de processamento, armazenamento e rede conforme as mudanças na carga. Isso é crucial em ambientes distribuídos, onde a elasticidade é essencial para garantir a disponibilidade e desempenho adequado do sistema, especialmente durante picos de tráfego e demanda. O trabalho de Canosa-Reyes *et al.* (CANOSA-REYES et al., 2022) apresenta estratégias de alocação de tarefas online para melhorar a qualidade de serviços, baseadas em dois ou três níveis de políticas de escalonamento, incluindo seleção de contêineres, distribuição de capacidade e alocação sensível à contenção.

A utilização de contêineres e ferramentas de gerenciamento de contêineres, como o *Docker* e o *Kubernetes*, simplifica a implementação da elasticidade em microsserviços, permitindo o escalonamento e gerenciamento eficientes dos recursos de uma aplicação. O trabalho de Lin *et al.* (LIN et al., 2018b), por exemplo, aborda a questão da alocação de recursos com eficiência energética para a virtualização baseada em contêineres em um data center. Os trabalhos classificados como elasticidade abordam questões sobre o gerenciamento de recursos e como minimizar o consumo de energia, visando fornecer

soluções de melhoria. Dessa forma, os trabalhos relacionados buscam abordar questões como a alocação eficiente de recursos, a escalabilidade, a resiliência e a capacidade de recuperação em ambientes de nuvem. Além disso, há uma preocupação em como minimizar o consumo de energia para garantir que as soluções propostas sejam sustentáveis e ecologicamente corretas.

O trabalho de Fontana de Nardin *et al.* (Fontana de Nardin et al., 2021) apresenta um modelo leve de elasticidade proativa, cujo diferencial está em melhorar o consumo de energia ao manipular periodicamente a quantidade mais adequada de recursos para executar uma aplicação enquanto mantém ou ainda melhora o desempenho de aplicações. O estudo de Valera *et al.* (VALERA et al., 2021b) concentra-se na identificação de candidatos a *host* ideais para execução de microsserviços em uma rede descentralizada, aplicando operações de escalonamento de tempo de execução (migração ou duplicação) para reduzir o consumo de energia. Outro trabalho de Valera *et al.* (VALERA et al., 2020) apresenta *DRACEO*, um simulador inteligente para implantar métodos de economia de energia em redes baseadas em microsserviços. A abordagem proposta permite que os gerentes de data center tomem decisões mais informadas em relação ao uso de energia, considerando a elasticidade do sistema e a alocação de recursos.

Confiabilidade:

A complexidade da tecnologia está aumentando, e os sistemas precisam ser cada vez mais confiáveis para lidar com essa evolução. A garantia da integridade dos dados é importante para implementar técnicas de tolerância a falhas e disponibilizar dados para recuperação em caso de falha, a fim de melhorar a confiabilidade dos microsserviços e dos contêineres. A escalabilidade automática de recursos pode ser uma abordagem eficaz para maximizar a confiabilidade e melhorar o tempo de resposta médio em caso de picos repentinos de tráfego. A avaliação da confiabilidade dos sistemas também deve considerar técnicas de gerenciamento de energia.

Os dois trabalhos classificados como confiabilidade focam em prover modelos que auxiliam na implementação de técnicas para melhorar a alta utilização de recursos. O trabalho de Li *et al.* (LI et al., 2021) aplica a teoria discreta de Markov e os resultados desse modelo de solução são usados para ajustar a redundância em tempo real para obter a mais alta confiabilidade para atender às restrições de recursos. O trabalho de Lyu *et al.* (LYU et al., 2020b) propõe uma arquitetura baseada em microsserviços para um sistema de gerenciamento de energia, para melhorar a confiabilidade do sistema. A arquitetura é projetada para permitir que diferentes módulos funcionem de forma independente, reduzindo a complexidade do sistema e aumentando a sua flexibilidade.

Sustentabilidade:

A arquitetura baseada em microsserviços é amplamente utilizada para promover a sustentabilidade em data centers. Por meio do uso de contêineres para agrupar aplicativos e recursos, a eficiência é aprimorada e o desperdício de recursos é minimizado. Essa abordagem permite que os microsserviços sejam mais flexíveis, escaláveis e independentes da plataforma, permitindo a utilização de infraestrutura de nuvem mais eficiente. O trabalho de Kumar *et al.* (KUMAR et al., 2019b) propõe um esquema de classificação de trabalhos multi-indexados e de gerenciamento de contêineres baseado em energia renovável para aumentar a sustentabilidade de *Data Centers* em nuvem. O objetivo é mitigar os problemas relacionados à energia por meio da seleção e consolidação adequadas de servidores. A proposta visa utilizar recursos de energia renovável para tornar os Data Centers mais sustentáveis e reduzir sua pegada de carbono.

Disponibilidade:

A capacidade de gerenciar e monitorar recursos é diretamente dependente da qualidade da coleta, processamento e utilização eficiente dos dados. A arquitetura de microsserviços é composta por serviços que trabalham em conjunto para realizar uma tarefa específica, mas são implementados e mantidos de maneira autônoma. O uso de microsserviços na infraestrutura de gerenciamento de dados na IoT, é fundamental para garantir a disponibilidade de dados. O processamento de dados em microsserviços é fundamental para assegurar que cada serviço tenha acesso e utilize os dados necessários para realizar suas funções, independentemente das condições de outras tarefas, para a tomada de decisões acertadas. O trabalho Jawar *et al.* (JARWAR; ALI; CHONG, 2019) propõe um modelo de microsserviços para melhorar a disponibilidade de dados para serviços de gerenciamento de eficiência energética em edifícios. O modelo visa fornecer dados precisos e atualizados sobre o desempenho energético dos edifícios, permitindo a tomada de decisões informadas para melhorar a eficiência energética. Ao dividir o sistema em microsserviços independentes, o modelo aumenta a disponibilidade dos dados e reduz a probabilidade de interrupções no serviço de gerenciamento de energia.

Métodos de Avaliação

Os métodos utilizados para avaliar um trabalho são fundamentais para validar uma proposta e garantir que ela seja aceita. Além disso, os métodos de avaliação permitem qualificar a contribuição do trabalho para o campo de estudo em questão e identificar possíveis lacunas que precisam ser preenchidas. A Figura 10 mostra o resultado geral obtido pelos métodos utilizados nos trabalhos analisados, divididos em três categorias: simulação, mensuração e modelagem.

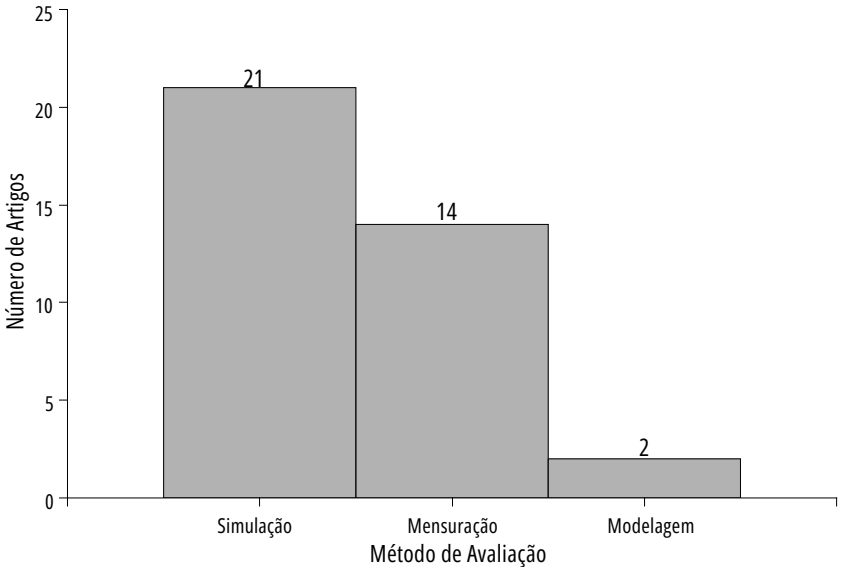


Figura 10 – Classificação geral dos métodos de avaliação.

A simulação é uma técnica que permite avaliar hipóteses e desempenho de sistemas em cenários sem a necessidade de realizar experimentos caros ou perigosos no mundo real. Essa abordagem oferece flexibilidade e controle na coleta de dados, além de permitir a exploração de sistemas complexos antes da implementação. No entanto, é importante notar que possíveis erros na modelagem podem levar a resultados insatisfatórios ou fora da realidade, comprometendo a confiabilidade da simulação. Portanto, é necessário ter cautela e cuidado na etapa de modelagem para garantir a precisão e confiabilidade dos resultados obtidos. Entre todos os trabalhos analisados, 56,8% deles optaram pelo modelo de simulação.

A mensuração provê dados mais precisos e realistas, pois reflete o comportamento real dos sistemas. A mensuração proporciona aos pesquisadores monitorar o desempenho de um sistema ao longo do tempo e avaliar diferentes condições, servindo para identificar diferenças e tendências. Todavia, para fazer mensuração, é preciso ter o sistema ou protótipo, que nem sempre existe, além de ser necessário coletar e processar dados, o que pode exigir equipamentos especializados e um tempo de amostragem prolongado, tornando a mensuração mais cara e demorada que a simulação. Além disso, a mensuração é mais difícil de escalar que a simulação. A mensuração foi o segundo método mais utilizado nos artigos classificados, 37,8% deles optaram por esse método.

Algumas avaliações de desempenho utilizam a modelagem para avaliar os seus sistemas, pois esta permite aos avaliadores simplificar e abstrair sistemas complexos, facilitando a compreensão e análise desses sistemas. Apenas 5,4% dos artigos analisados usaram modelagem.

Simulação

A classificação dos artigos como simulação mostrou que 10% não especificaram os tipos de simuladores utilizados. Esse fato pode ser resultante da não importância atribuída pelos autores à especificação dos simuladores ou da escolha por não compartilhar detalhes da metodologia, considerando-a como irrelevante para o escopo do estudo.

CloudSim é uma biblioteca de simulação de nuvem *open-source* que permite aos pesquisadores modelar e simular sistemas de nuvem computacional. Os artigos que especificaram o tipo de simulador adotado, 55% deles utilizaram *CloudSim*. Khan *et al.* (KHAN et al., 2021b) demonstram, por meio de experimentos, como diferentes políticas de alocação, combinadas com várias abordagens de migração, afetam a eficiência de energia e o desempenho de data centers. Os experimentos utilizam dados de carga de trabalho do *Google*, para simular um *cluster* de data center, utilizando o simulador *CloudSim*, com 12.583 *hosts* e aproximadamente um milhão de tarefas que pertenciam a três tipos diferentes de carga de trabalho.

O estudo realizado por Singh *et al.* (SINGH; AUJLA; BALI, 2021) apresenta uma proposta de método de balanceamento de carga baseado em contêineres para melhorar a eficiência energética em um ambiente de computação de borda definido por software. Para avaliar o desempenho do método proposto, os autores empregaram o *CloudSim*, um simulador de nuvem amplamente utilizado na literatura. O trabalho de Al-Moalmi *et al.* (AL-MOALMI et al., 2021) propõe um sistema de otimização baseado em algoritmo de baleias para a colocação de contêineres em data centers visando maximizar a eficiência energética. O sistema foi avaliado por meio de simulações usando o *CloudSim*, uma plataforma de simulação em nuvem.

O *DRACeo* é um simulador flexível e acessível, desenvolvido para gerenciar o modelo de implantação e escalonamento de aplicações baseadas em microsserviços em diferentes arquiteturas de rede, tanto não centralizadas, centralizadas e híbridas. Valera *et al.* (VALERA et al., 2020) usaram o *DRACeo* para simular o modelo de implantação e agendamento de microsserviços através de uma rede. Entre artigos que especificaram o tipo de simulador usado, 10% utilizaram o *DRACeo*.

Linpack é uma biblioteca de programas para executar cálculos numéricos de álgebra linear em computadores. O *Linpack* é usado como *benchmark* para testar e medir o desempenho de sistemas de computação distribuída. A ferramenta *Linpack*, foi desenvolvida para ser usada em computadores de alto desempenho, especialmente aqueles que usam arquiteturas de processadores paralelos. Os trabalhos que especificaram a simulação, 5% deles utiliza o *Linpack*. Juan *et al.* (LUO et al., 2019a), por exemplo, utilizam o *Linpack* para comparar o desempenho de máquinas virtuais e contêineres em condições de simultaneidade.

MATLAB 2015b é um software com um ambiente de desenvolvimento interativo para cálculo numérico, visualização e programação. O *MATLAB* inclui uma ampla variedade de ferramentas para análise de dados, processamento de sinais e imagens, modelagem e simulação. *MATLAB 2015b* foi utilizado em 5% dos artigos classificados com simulação. O trabalho de Lyu *et al.* (LYU et al., 2020b), por exemplo, implementam um sistema de gerenciamento de energia (MS-EMS) utilizando o*MATLAB 2015b* como uma das ferramentas de desenvolvimento.

O artigo (KHAIRY et al., 2022) propõe uma nova técnica de processamento de microsserviços em centros de dados, chamada *SIMR*. A técnica consiste em agrupar múltiplas solicitações de serviço em um único conjunto de instruções, reduzindo a sobrecarga do processador e melhorando a eficiência energética. O *SIMR* é avaliado em comparação com outras técnicas de processamento em termos de desempenho e consumo de energia. Os resultados mostram que o *SIMR* pode melhorar a eficiência energética do processamento de microsserviços em centros de dados sem comprometer o desempenho. A pesquisa (KHAIRY et al., 2022) menciona três ferramentas de simulação: *BookSim*, *SimTec* e *Accel*. O BookSim está presente em 5% dos artigos, sendo utilizado para simular os microsserviços e as arquiteturas de servidor, e o *SimTec* aparece 5% dos artigos utilizado para modelar as características de energia e latência da rede. Em 5% dos artigos, o *Accel* foi utilizado para avaliar o desempenho do acelerador de hardware implementado para o processamento de solicitações múltiplas em uma única instrução.

Mensuração

Os artigos classificados como mensuração, 60% deles optaram por não informar as ferramentas utilizadas para mensurar. O motivo não foi especificado pelos autores, o que sugere que a omissão da ferramenta seja irrelevante para o foco da pesquisa. Em 20% dos artigos foi utilizado o dispositivo *Raspberry Pi* para mensuração. O *Raspberry Pi* é uma placa de computador de baixo custo e tamanho reduzido que pode ser usada para medir e coletar dados. Esse dispositivo possui uma variedade de interfaces de entrada e saída, que permitem conectar sensores e dispositivos externos para coletar dados.

O trabalho Lennick *et al.* (LENNICK; AZIM; LISCANO, 2021b) apresenta uma arquitetura baseada em microsserviços para análise de desempenho e consumo de energia de distribuições *Linux Docker host* da Internet das Coisas, utilizando o *Raspberry Pi* como sistema de teste. A ferramenta *Grid'5000 testbed* é utilizada em 20% trabalhos, e fornece um ambiente de teste distribuído e escalável para avaliar o desempenho de sistemas de computação distribuída e *grid computing*. Xu *et al.* (XU; TOOSI; BUYYA, 2019) realizam uma avaliação no *Grid'5000 testbed* usando recursos do *cluster* de Lyon para a carga de trabalho da web *Wikipedia*, para avaliar políticas de escalonamento propostas em um protótipo de sistema.

O trabalho de *Tacskin et al.* (TAŞKIN; TAŞKIN et al., 2021) propõe o uso de virtualização baseada em contêineres em aplicações de IoT. O estudo é focado em medir a eficiência de desempenho e o consumo de energia do sistema proposto em comparação com outros métodos de virtualização. O trabalho de Brondolin (BRONDOLIN; SARDELLI; SANTAMBROGIO, 2018) apresenta uma ferramenta chamada *DEEP-Mon* que tem como objetivo realizar o monitoramento de energia em infraestruturas baseadas em contêineres de maneira dinâmica e eficiente. A avaliação da ferramenta foi realizada em um *cluster* de contêineres, onde foram executados diferentes *workloads*.

Modelagem

A utilização de modelos matemáticos é uma estratégia eficiente para avaliar a eficiência e desempenho de sistemas tecnológicos, ao oferecer uma maneira rápida e econômica de obter resultados precisos. Os trabalhos mostraram dois tipos distintos de modelos utilizados na avaliação do desempenho de sistemas: *Markov Chain* e *K-Fault*. Os modelos podem determinar a confiabilidade e tolerância a falhas dos sistemas computacionais, garantindo assim sua disponibilidade e capacidade de funcionamento adequada.

Modelo de *Markov Chain* é uma técnica estatística utilizada para modelar sistemas dinâmicos e estocásticos. Li *et al.* (LI et al., 2021) apresenta uma proposta de modelo de otimização de confiabilidade baseado na teoria discreta de Markov, no qual é utilizado para maximizar a confiabilidade do sistema enquanto se mantém nos limites superiores estabelecidos para os recursos computacionais. *K-Fault* é um modelo de disponibilidade utilizado para avaliar a tolerância a falhas de sistemas computacionais. O modelo é baseado na ideia de que o sistema pode tolerar até k falhas simultâneas sem afetar sua funcionalidade. Lyu *et al.* (LYU et al., 2020b) utiliza esse modelo para melhorar a tolerância a falhas em sistemas de gerenciamento de serviços.

Tipos de Arquitetura

O gráfico de barras presente na Figura 11 apresenta os tipos de arquitetura utilizados pelos artigos estudados. A partir dos dados mostrados é possível observar uma relação entre o número de artigos com o tipo de arquitetura. Os dados mostram o quanto a utilização da arquitetura computação em nuvem é superior às demais. A computação em nuvem aparece em 30 obras (81,1%), computação em borda em 5 obras (13,5%) e computação híbrida em 2 obras (5,4%). O que induz a perceber que arquiteturas como borda e híbrida foram menos utilizadas dentre as obras estudadas.

As arquiteturas de nuvem abordam as principais dificuldades em torno do processamento de dados em grande escala (VARIA, 2008). A arquitetura em nuvem é um conjunto

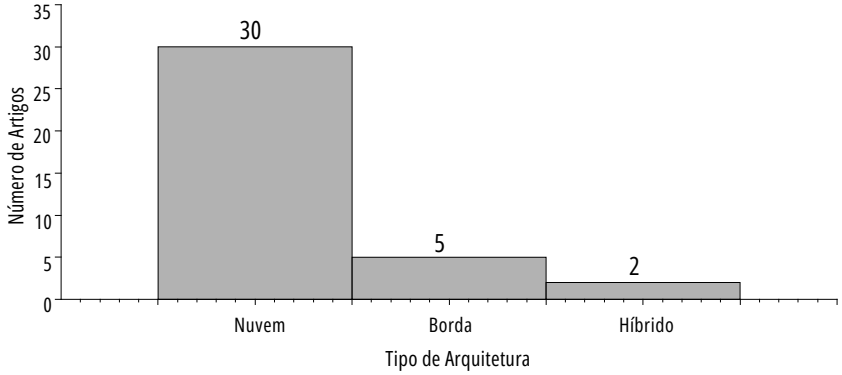


Figura 11 – Classificação dos tipos de arquitetura.

de práticas e padrões para projetar e implementar aplicações e serviços em ambientes de nuvem. computação em nuvem, computação em névoa e computação em borda são três tipos diferentes de arquitetura de computação distribuída que permitem aos usuários acessar recursos de computação de forma flexível.

A computação em nuvem é um serviço que utiliza pagamento baseado ao uso, que permite obter espaço de armazenamento em rede e recursos do computador (HUTH; CEBULA, 2011). A arquitetura de nuvem pode ter benefícios como: escalabilidade, flexibilidade e economia de custos. Com computação em nuvem é possível ajustar facilmente sua capacidade de computação conforme as necessidades, sem se preocupar com a manutenção e atualização de hardware. Assim, é possível pagar somente o que é usado, sem precisar investir em hardware e infraestrutura caros.

O estudo de Lennick *et al.* (LENNICK; AZIM; LISCANO, 2021a) utiliza computação em nuvem como base para analisar o desempenho e o consumo de energia de distribuições Linux executadas em dispositivos da IoT que funcionam como hospedeiros do *Docker*. No trabalho de Rajan *et al.* (RANJAN et al., 2020) a computação em nuvem é utilizada para fornecer os recursos necessários e suportar a execução eficiente de um algoritmo genético de consolidação de contêineres. A consolidação de contêineres tem o objetivo de otimizar a utilização dos recursos da nuvem, como capacidade de processamento e armazenamento, buscando eficiência energética e melhor desempenho. O estudo de Gholipour *et al.* (GHOLIPOUR; ARIANYAN; BUYYA, 2020) propõe um novo método de gerenciamento de recursos na computação em nuvem que combina máquinas virtuais e migração de contêineres simultaneamente. Os resultados mostram reduções significativas no consumo de energia, violações de *Service Level Agreement* (SLA) e número de migrações em comparação com os algoritmos atuais. O trabalho de Kaur *et al.* (KAUR et al., 2019) utiliza da arquitetura de nuvem para lidar com um Data Center na nuvem. A obra propõe uma investigação acerca do impacto conjunto da consolidação de recursos e balanceamento de carga na redução dos índices de utilização de energia.

A arquitetura computação em borda é uma abordagem de rede que visa aproximar a computação o mais próximo possível da origem dos dados, visando reduzir a latência e o uso de largura de banda (CLOUDFLARE, 2023). Essa arquitetura é uma extensão da computação em nuvem, na qual os recursos de computação são colocados em proximidade dos dispositivos finais, como câmeras, sensores e dispositivos IoT. Devido à sua capacidade de reduzir a latência e melhorar a eficiência da rede, a computação em borda é especialmente adequado para aplicações que requerem processamento de dados em tempo real. Um exemplo disso é o trabalho de Kaur *et al.* (KAUR et al., 2017), que utiliza computação em borda para auxiliar no processamento de dados e serviços para usuários finais na borda da rede. O estudo propõe uma arquitetura que utiliza contêineres como serviço para seleção e escalonamento de tarefas na borda da rede. O estudo de Asif *et al.* (ASIF; GHANEM; IRVINE, 2020) explora o uso da tecnologia de conteinerização para programação remota de dispositivos de IoT no setor de energia. A computação em borda desempenha um papel fundamental nesse trabalho, pois os autores propõem a utilização de contêineres para implementar a programação desses dispositivos IoT. A computação de borda permite executar o processamento e a lógica de controle próximos aos dispositivos, reduzindo a latência e a carga na rede central.

Por outro lado, a computação em névoa é uma infraestrutura descentralizada, posicionando os recursos de computação entre a fonte de dados e a nuvem ou outro centro de dados (STACKPATH, 2023). A arquitetura de névoa permite que dispositivos de borda, como sensores IoT e câmeras, processem dados localmente, reduzindo assim a necessidade de enviá-los para centros de dados remotos. Essa abordagem pode melhorar a latência e a eficiência energética. No entanto, é importante destacar que a computação em névoa complementa a computação em nuvem, em vez de substituí-la.

A computação híbrida se trata da junção de duas arquiteturas ao mesmo tempo: nuvem e névoa. A computação híbrida se mostrou útil para modelo de implantação de serviços em contêineres em dispositivos IoT. A obra de Raza *et al.* (RAZA et al., 2021) visou identificar principais fatores no modelo de implantação e serviços em contêineres em dispositivos IoT com recursos limitados. Os recursos limitados citados são os de computação e energia que dificultam a escalabilidade e modelo de implantação de serviços simultâneos. O estudo tem o intuito de avaliar o desempenho e consumo energético de soluções em contêiner em *gateways* de IoT com recursos limitados. Utilizando da arquitetura de névoa (complementar das demais arquiteturas), útil para melhorar latência e eficiência energética.

Orquestrador

A Figura 12 apresenta os resultados das classificações dos orquestradores nos artigos mapeados. O Gráfico de barras mostra que 46,2% dos trabalhos analisados não informam o tipo de orquestrador utilizado. Nos trabalhos que fornecem essa informação, são usados

dois tipos populares de orquestradores de contêineres: *Kubernetes* e *Docker Swarm*.

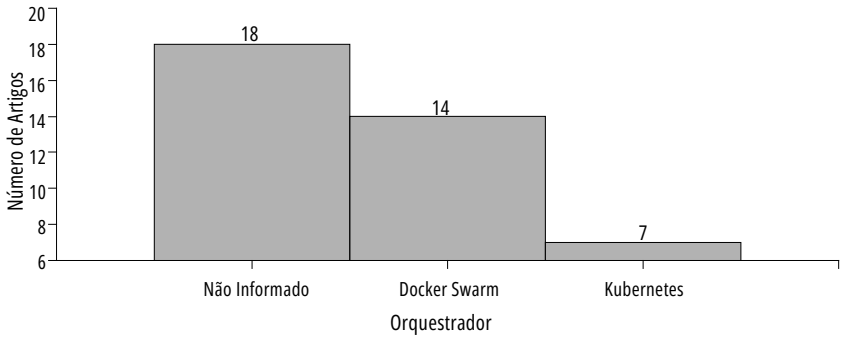


Figura 12 – Classificação dos tipos de orquestrador.

Os orquestradores de contêiner são ferramentas que auxiliam no gerenciamento de execução de contêineres em um sistema. Eles permitem que os usuários criem, implantem e gerenciem aplicações em contêineres de maneira mais eficiente. Além disso, os orquestradores de contêiner também monitoram o uso de recursos e podem escalar as aplicações automaticamente para atender demandas variáveis.

O orquestrador *Kubernetes* é uma ferramenta de código aberto projetada para automatizar o modelo de implantação, o dimensionamento e o gerenciamento dos contêineres em ambientes de suas aplicações. Saengkaenpetch *et al.* (SAENGKAENPETCH; ASWAKUL, 2022) utilizam o *Kubernetes* para gerenciar cargas de trabalho exigidas em contêineres. O *Docker Swarm* é um sistema de gerenciamento de contêineres que permite que os usuários criem e administrem *clusters* de contêineres, além disso, é responsável por escalar aplicações facilmente e garantir maior disponibilidade do serviço. Lennick *et al.* (LENNICK; AZIM; LISCANO, 2021b) utilizam o *Docker* para analisar o desempenho e consumo de energia de distribuições Linux integrado com a IoT.

Kubernetes

Dentre os artigos classificados com orquestrador, 17,9% optaram pelo *Kubernetes*. O *Kubernetes* tem se mostrado eficiente na automação da escalabilidade, implementação e gerenciamento de aplicações em contêineres. O trabalho de Lyu *et al.* (LYU et al., 2020b) utiliza o *Kubernetes* para monitorar a integridade de contêineres em um sistema de gerenciamento de energia. O *Kubernetes* analisa duas sondagens: a sonda de vivacidade, que verifica a integridade e execução do contêiner, e a sonda de prontidão, que determina se o contêiner está preparado para atender às solicitações. O intuito do trabalho é melhorar a confiabilidade do sistema.

Docker Swarm

Observa-se que os artigos categorizados com orquestrador, 35,9% deles utilizam o *Docker Swarm*. Isso indica que a ferramenta tem uma maior aceitação relevante para gerenciar grandes sistemas distribuídos. O estudo de Zhang *et al.* (ZHANG et al., 2019) se concentra em avaliar o desempenho do consumo energético em contêineres. Os autores utilizaram o orquestrador *Docker Swarm* para uma análise experimental, visando medir e comparar uma estratégia proposta pelos autores, com duas estratégias fornecidas pelo *Docker*, a fim de melhorar o consumo de energia.

Modelo de Implantação

A Figura 13 apresenta os resultados da classificação dos tipos de implementação. Como mostrado no gráfico, 83,8% dos artigos mapeados não especificam o tipo de modelo de implantação. Os dados sugerem que essa informação pode ser considerada pouco relevante para os autores dos artigos. O modelo de implantação refere-se ao tipo de nuvem empregada nos trabalhos mapeados. Os tipos de nuvem são divididos em três categorias principais: pública, privada e híbrida. Os tipos apresentados podem ser usados individualmente ou em combinação com um ou dois tipos diferentes de nuvem. Podemos observar isso no gráfico de barras.

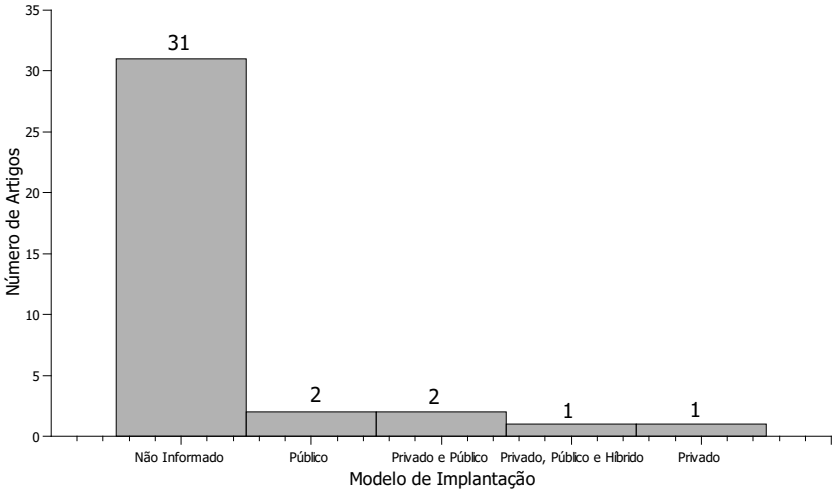


Figura 13 – Classificação do modelo de implantação.

Público

O estudo de Luo *et al.* (LUO et al., 2019b) mostra a arquitetura tradicional de computação em névoa, que se concentra em um *data center* e nós e apresenta desafios para se adaptar ao crescente desenvolvimento da nuvem privada. O modelo de implantação de uma arquitetura *multi-cloud* baseada em contêineres na nuvem pública, no entanto,

proporciona uma solução que melhora a flexibilidade, agilidade e a escalabilidade das operações para o balanceamento de energia. A adoção de microsserviços e contêineres em nuvem pública pode ser uma estratégia eficiente para as organizações que buscam maior flexibilidade e eficiência na construção e execução de aplicativos.

Privado

A nuvem privada representa uma opção viável para as empresas que buscam garantir a privacidade e segurança dos seus dados, oferecendo acesso às tecnologias e serviços equivalentes às disponíveis na nuvem pública, mas com a diferença de que os dados são armazenados em um ambiente exclusivo e controlado pelas próprias organizações. O trabalho de Fontana de Nardin *et al.* (Fontana de Nardin et al., 2021) utiliza a nuvem privada para desenvolver e avaliar um protótipo que executa uma aplicação baseada em microsserviços construída sobre o *Docker Swarm*. No trabalho de Piraghaj *et al.* (PIRAGHAJ et al., 2015), o foco é na melhoria da eficiência energética dos servidores por meio da proposta de um *framework* que consolida contêineres em máquinas virtuais utilizando duas abordagens de modelo de implantação em nuvem. A primeira abordagem, na nuvem privada, visa minimizar o consumo de energia por algoritmos específicos. A segunda abordagem, na nuvem pública, procura reduzir o número total de horas alugadas pelas máquinas virtuais.

Híbrido

O modelo de implantação de uma nuvem híbrida envolve a combinação de recursos de nuvens públicas e privadas para formar um ambiente que aproveite as vantagens de ambos. As empresas podem obter a agilidade e flexibilidade da nuvem pública, juntamente com a segurança e controle da nuvem privada. No trabalho de Asif *et al.* (ASIF; GHANEM; IRVINE, 2020), é analisada a arquitetura de implementação de contêineres para aplicações no setor de energia, especificamente sistemas de automação de subestações de última geração. As vantagens de utilizar contêineres estão na possibilidade de aplicações em diferentes tipos de nuvem, incluindo nuvem híbrida, privada e pública.

3.4 Resultados de Classificação de Correlação Cruzada

Esta presente seção apresenta a frequência de estudos a partir do relacionamento cruzado entre dois critérios. Os gráficos mostram nas interseções dos eixos x e y, o número de trabalhos para cada interseção. Essa abordagem foi utilizada considerando os diferentes critérios ao mesmo tempo. A interpretação desse tipo de gráfico auxilia na identificação de que áreas de pesquisa está sendo discutida e quais são as áreas que apresentam lacunas que possam ser exploradas.

Arquitetura x Modelo de implantação

A Figura 14 apresenta as classificações relacionadas ao cruzamento dos critérios de tipos de arquiteturas e modelo de implantação. No eixo X se encontra o modelo de implantação composta por: privadas; públicas; e privadas e públicas; não informados. Já no eixo Y, apresenta os tipos de arquiteturas utilizados como: computação em nuvem; computação em borda; computação híbrida. No gráfico podemos observar que a arquitetura de computação em nuvem foi a mais aplicada dentre as demais, foram utilizadas modelos de implantação como: privadas; públicas; e privadas e públicas.

A computação em borda apresenta pouca usabilidade quando comparada à computação em nuvem é mencionada em apenas cinco obras. A computação em borda é empregada em um modelo de implantação privada, pública e híbrida enquanto as demais utilizações não são detalhadas. Já a computação em névoa e nuvem, exibe que 50% das obras foram abordadas em implantações públicas, enquanto nas demais, isso não é descrito. A falta de maior detalhamento sobre quais implantações foram usadas, se dá pelo fato de os trabalhos se atentarem a critérios mais relevantes, fazendo com que grande parte do foco seja voltado para outras áreas. O estudo de Bouaouda *et al.* (BOUAOUDA; AFDEL; ABOUNACER, 2022a) por exemplo, propõe-se a prever o consumo de energia de centros de dados em nuvem.

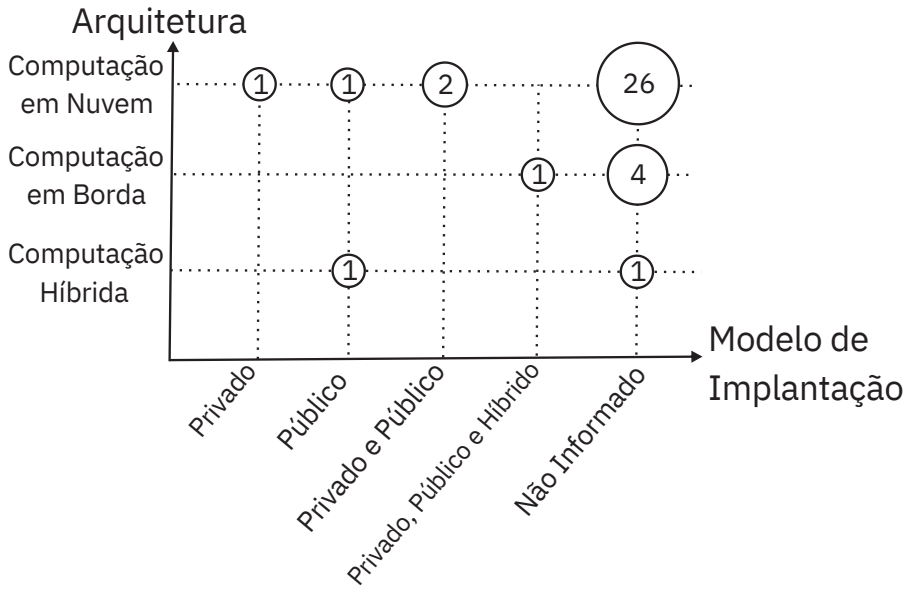


Figura 14 – Arquitetura x modelo de implantação.

Arquitetura x Orquestrador

A Figura 15 exibe o cruzamento das classificações: tipos de arquitetura e orquestradores. Neste gráfico, o eixo X representa os orquestradores operados que são: *Docker*

Swarm; *Kubernetes*; *Docker Swarn* e *Kubernetes*; e não informado. No eixo Y, apresenta os tipos de arquitetura utilizados como: computação em nuvem; computação em borda; computação híbrida. A arquitetura de computação em nuvem foi a mais utilizada nos trabalhos, estando presente em 29 trabalhos selecionados, sendo empregada em todos os orquestradores.

A computação em borda demonstra que sua usabilidade em *Docker Swarn* e *Kubernetes* é semelhante, além da falta de informação acerca de outros trabalhos. A computação em névoa e nuvem, apresenta uma baixa adoção quando comparada as demais arquiteturas. A computação em borda, névoa e nuvem apresentam poucas obras que abordam a sua utilização quando colocadas lado a lado com computação em nuvem que possui um amplo acolhimento na área. A obra de Khan *et al.* (KHAN et al., 2021a) prioriza o uso da computação em nuvem porque é o contexto onde a técnica de consolidação de recursos proposta pelo trabalho é aplicada.

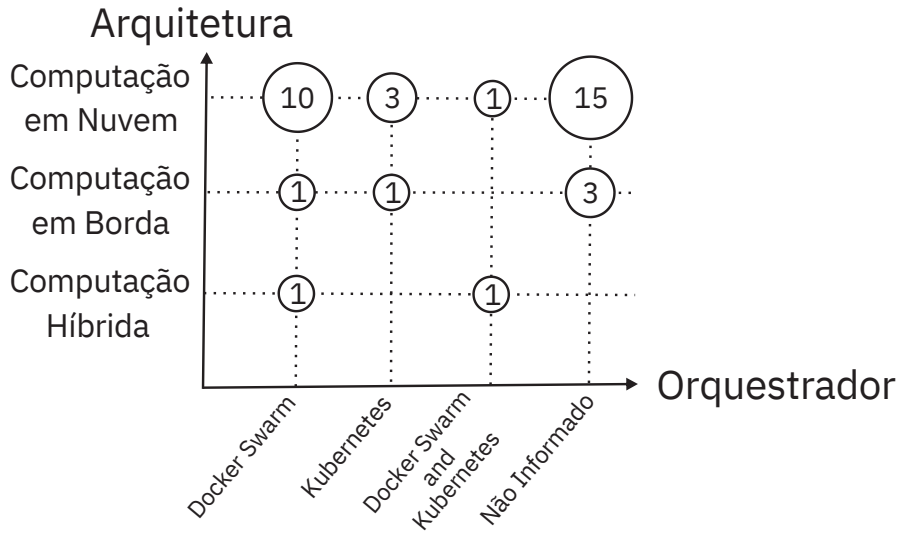


Figura 15 – Arquitetura x orquestrador.

Método de Avaliação x Arquitetura

A Figura 16 demonstra a ligação entre as classificações: método de avaliação e tipos de arquitetura. Interpretando o gráfico no eixo X apresenta os tipos de arquitetura como: computação em nuvem; computação em borda; computação híbrida. Enquanto no eixo Y, é mostrado o método de avaliação empregado: simulação, mensuração e modelagem. Pode ser notado que a simulação foi o método mais usado seguido da mensuração, ambos utilizaram todos os tipos de arquiteturas exibidos. Já a modelagem teve uma baixa presença nos trabalhos e sendo aplicada apenas em computação em nuvem.

O método de avaliação por simulação foi bastante executado nas arquiteturas de computação em nuvem, mas quando observamos sua execução em computação em

borda, e computação em névoa e nuvem pode-se notar uma diferença enorme na queda de utilização comparada a computação em nuvem. O estudo de Canosa-Reyes *et al.* (CANOSA-REYES et al., 2022) utiliza simulação pelo fato de permitir avaliar a técnica proposta em um ambiente controlado, explorar diferentes cenários e coletar dados para análise de desempenho. O mesmo ocorre quando a mensuração é o método de avaliação, na computação em nuvem tem uma abundante quantidade de obras abordando enquanto as demais sofrem uma queda drástica.

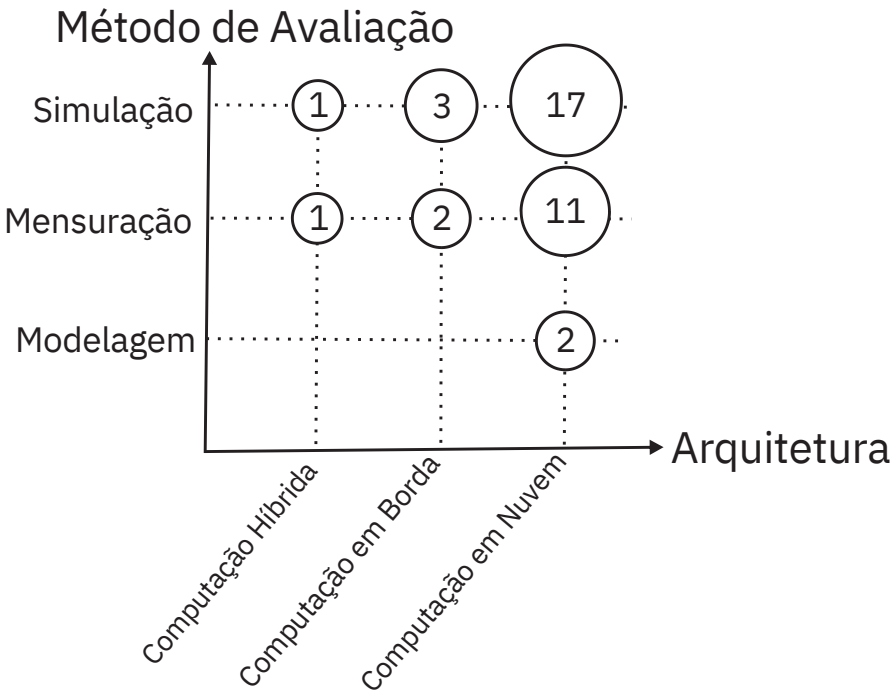


Figura 16 – Método de avaliação x arquitetura.

Método de Avaliação x Modelo de Implantação

A Figura 17 mostra o cruzamento entre as classificações: método de avaliação e implantação. No eixo X localiza-se a modelo de implantação que é composta: privadas; públicas; e privadas e públicas; não informado. Ao mesmo tempo que no eixo Y, está exibido o método de avaliação utilizado: simulação, mensuração e modelagem. Tendo em vista o gráfico, é notável que o método de simulação se destaca perante os demais métodos, sendo o mais abordado na área. A mensuração também possui uma ampla abordagem enquanto a modelagem não tem tanta ênfase.

A simulação como método de avaliação foi bastante utilizada, apesar de não ser bem especificado em quais tipos de implantações foi empregada. As poucas obras que especificaram sua utilização demonstram que foi abordada em diversos tipos de implantações como: privadas; públicas; e privadas e públicas. A mensuração não apresenta

usabilidade em implantações privadas; e privadas e públicas, enquanto modelagem não é informado em nenhum trabalho em qual modelo de implantação foi aplicada.

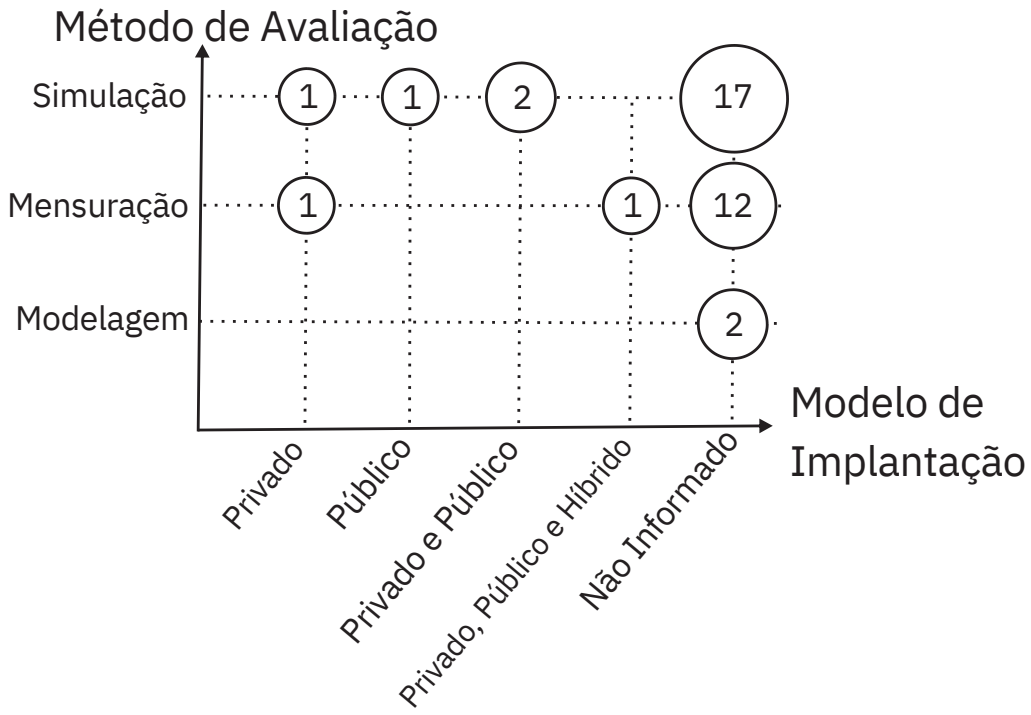


Figura 17 – Método de Avaliação x Modelo de Implantação.

Métricas Macro x Arquitetura

O gráfico presente na Figura 18 apresenta uma ligação entre as classificações: métricas macro e tipos de arquitetura. A partir do gráfico dá para analisar alguns destaques em utilizações de determinadas métricas macro. No eixo X se encontram os tipos de arquitetura como: computação em nuvem; computação em borda; computação híbrida. Já no eixo Y, denota as métricas utilizadas como: desempenho, elasticidade, escalabilidade, confiabilidade, sustentabilidade e disponibilidade. O gráfico apresenta uma maior utilização das métricas: desempenho e elasticidade. Também denota uma maior utilização da arquitetura computação em nuvem onde, quase todas as obras apresentaram o uso da arquitetura de computação em nuvem.

Métricas como desempenho e elasticidade, se destacaram dentre as demais métricas. As obras, em sua maioria, indicam interesses em avaliar desempenho e elasticidade diante do tema: consumo de energia em microsserviços. A obra de Xu *et al.* (XU; TOOSI; BUYYA, 2019) foca em avaliação de desempenho porque o objetivo principal do trabalho é propor uma abordagem integrada para gerenciar tanto a economia de energia quanto os problemas de desempenho. As demais métricas estiveram presentes em algumas obras, mesmo que em menor número, se comparadas às duas anteriores. Diante do gráfico, ficou claro um maior uso da arquitetura computação em nuvem dentre todas as métricas, devido à sua maior

facilidade de uso, especialmente em termos de custo. Tendo um total de 30 obras que demonstraram utilização da arquitetura computação em nuvem. As obras com as outras arquiteturas: computação em borda e computação híbrida, estiveram atreladas somente a obras com as métricas: desempenho, elasticidade e disponibilidade.

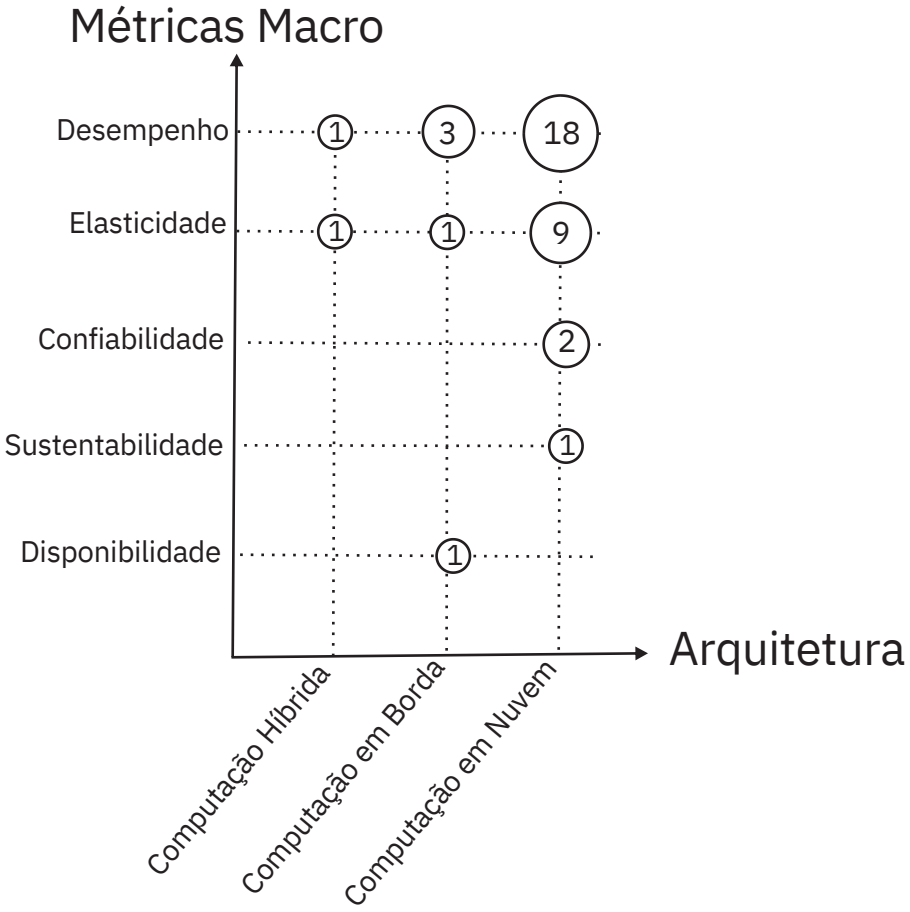


Figura 18 – Métricas macro x arquitetura.

Métricas Macro x Método de Avaliação

A Figura 19 mostra o gráfico com intersecção entre as classificações: métricas e método de avaliação. O eixo X do gráfico apresenta os métodos de avaliação como: modelagem, medição e simulação. Já o eixo Y mostra métricas como: desempenho, elasticidade, escalabilidade, confiabilidade, sustentabilidade e disponibilidade. O gráfico denota um certo espalhamento da utilização de determinadas métricas pelas obras, visto que as métricas de: desempenho e elasticidade, apresentam uma maior utilização se comparada às demais métricas. Ainda é possível notar um certo destaque na utilização de métricas de performance pelas obras mapeadas. O estudo de Zhang *et al.* (ZHANG et al., 2019) utiliza Performance e Mensuração como critérios fundamentais na estratégia de posicionamento de contêineres proposta.

As métricas de desempenho foram as mais utilizadas em relação às demais. Obras com a métrica desempenho estiveram bem atreladas aos métodos de avaliação: medição e simulação. Os trabalhos demonstram interesse nas áreas de mensurar e simular a performance nos ambientes de consumo de energia em microsserviços. Outra métrica que apresentou boa utilização foi a métrica elasticidade, demonstrou uma maior utilização do método de avaliação simulação.

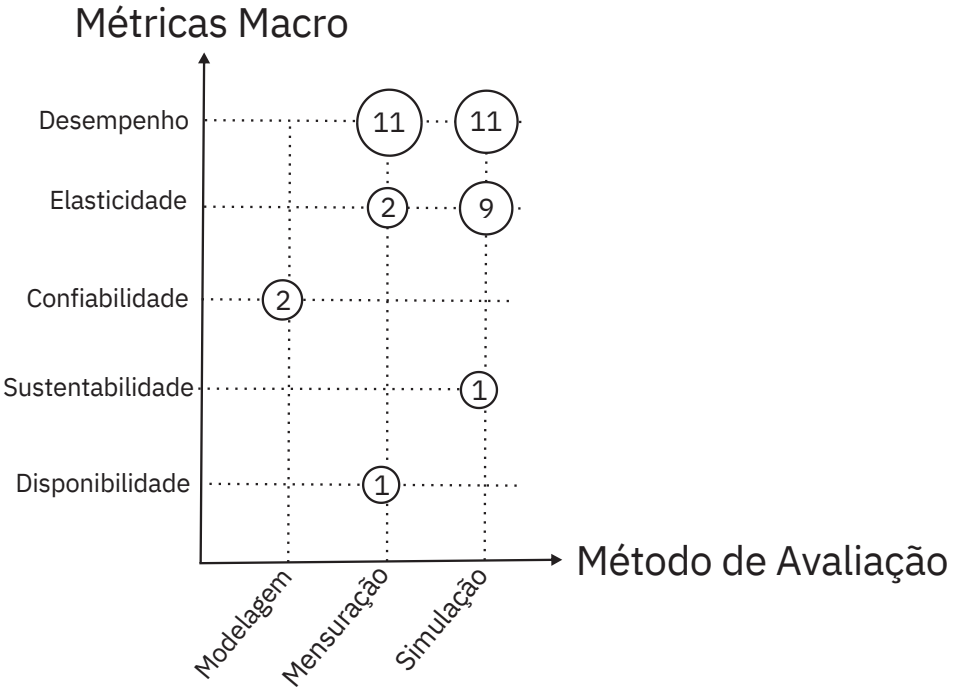


Figura 19 – Métricas macro x método de avaliação.

Métricas Macro x Modelo de Implantação

O gráfico da Figura 20 apresenta a relação cruzada entre as classificações: métricas macro x modelo de implantação. A partir disso, o eixo X do gráfico apresenta tipos de modelo de implantação como: privado, público, privado e público, privado e público e híbrido e não informado. Já o eixo Y demonstra métricas como: desempenho, elasticidade, escalabilidade, confiabilidade, sustentabilidade e disponibilidade. Analisando o gráfico notou-se que ao realizar o cruzamento das classificações, boa parte das obras não deixou explícito o tipo de modelo de implantação utilizado. As métricas de desempenho e elasticidade, mais uma vez, foram as mais utilizadas pelas obras.

As métricas de desempenho mostraram uma superior utilização onde, algumas obras apresentaram o tipo de modelo de implantação utilizada, mas em sua grande maioria não deixaram explícito. O mesmo acontece para as obras com utilização da métrica: elasticidade. As obras com as demais métricas não deixaram claro o tipo de modelo de implantação. Algumas obras com a métrica de performance demonstraram implantações

como: público, privado e público e privado e público e híbrido. Já obras com a métrica de elasticidade apresentaram implantações: privado e público Obras que utilizaram as outras métricas, além de Performance e elasticidade, não deixaram claras o tipo de modelo de implantação utilizadas, sendo classificada como não informado.

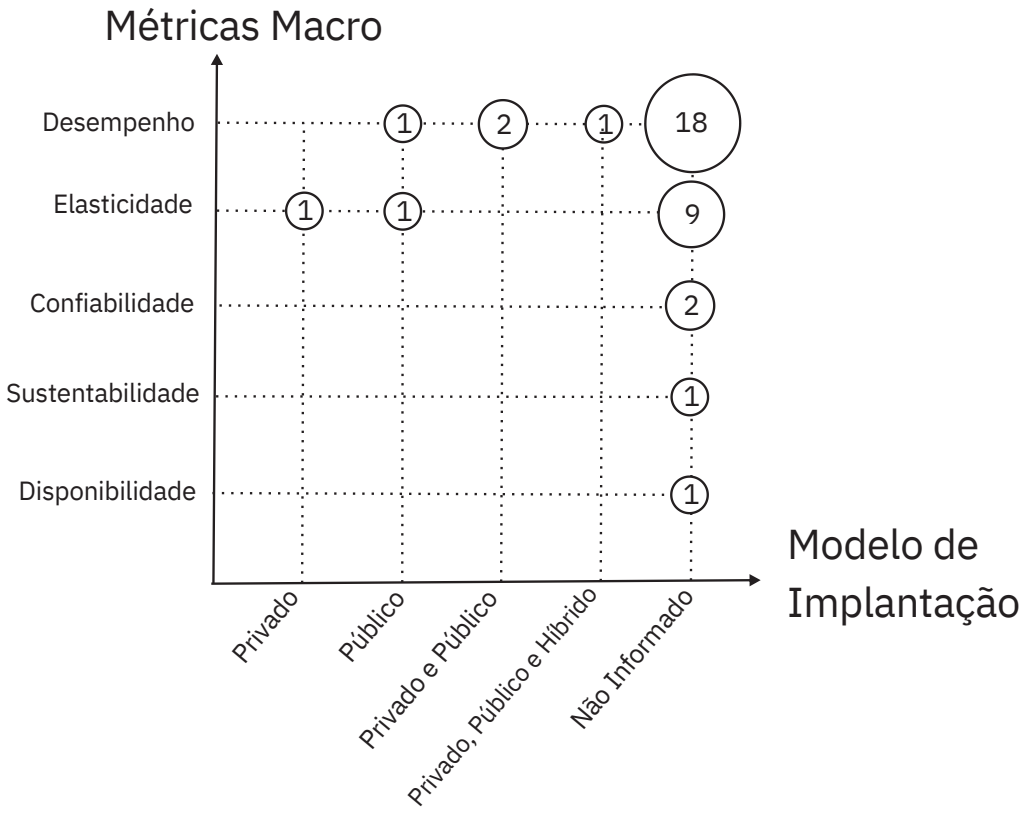


Figura 20 – Métricas macro x modelo de implantação.

Métricas Macro x Orquestrador

A Figura 21 mostra a intersecção entre as classificações: métricas e orquestrador. O eixo X do gráfico mostra orquestradores como: *Kubernetes*, *Docker Swarm* e não informado. Já o eixo Y do gráfico demonstra as métricas utilizadas pelas obras: desempenho, elasticidade, escalabilidade, confiabilidade, sustentabilidade e disponibilidade. O gráfico denota que somente obras com utilização de métricas como: sustentabilidade e disponibilidade, não tiveram algum orquestrador atrelado ou apresentado de maneira explícita pelo estudo. As métricas de Performance e elasticidade tiveram um maior destaque de utilização pelas obras.

Obras com a utilização das métricas de Performance, apresentaram uma maior utilização do orquestrador *Docker Swarm*, algumas com *Kubernetes* e outra parte (mesma quantidade que utilizou *Docker Swarm*) não deixou explícito o tipo de orquestrador utilizado. Já obras com as métricas: elasticidade e escalabilidade, pelo menos alguma obra

apresentou utilização de um orquestrador. Outras obras com a métrica: confiabilidade, todas utilizaram o orquestrador *Kubernetes*. Outras obras que utilizaram as métricas: sustentabilidade e disponibilidade, não demonstraram de maneira clara o orquestrador utilizado.

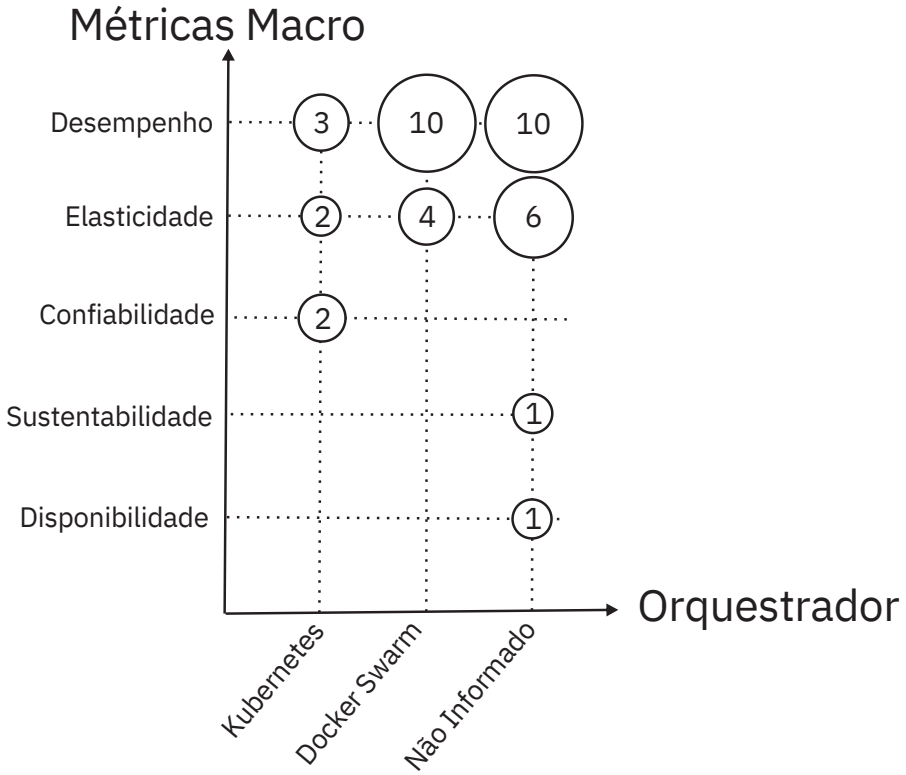


Figura 21 – Métricas macro x orquestrador.

Orquestrador x Modelo de Implantação

A Figura 22 demonstra a relação entre a intersecção das classificações: orquestrador x modelo de implantação. O eixo X do gráfico denota os tipos de modelo de implantação utilizados pelas obras: privado, público, privado e público, privado e público e híbrido e não informado. Já o eixo Y apresenta os orquestradores utilizados pelas obras: *Kubernetes*, *Docker Swarm* e não informado. A interpretação do gráfico apresenta uma concentração elevada de utilização do orquestrador *Docker Swarm* e outra grande concentração de obras que não especificaram qual orquestrador ou tipo de modelo de implantação foi utilizado.

O tipo de orquestrador *Docker Swarm* esteve atrelado a implantações como: público, privado e privado e público. Já a maioria de obras com esse orquestrador, não demonstrou tipo de modelo de implantação. Já o tipo de orquestrador *Kubernetes* esteve ligado a modelo de implantação: privado e público e híbrido. Porém, as demais obras não informaram seu tipo de modelo de implantação. A maioria das obras não deixou explícito seu orquestrador e nem o tipo de modelo de implantação utilizado. Contudo, houve um caso em que uma

obra não demonstrou o orquestrador utilizado, mas especificou um tipo de modelo de implantação: privado e público.

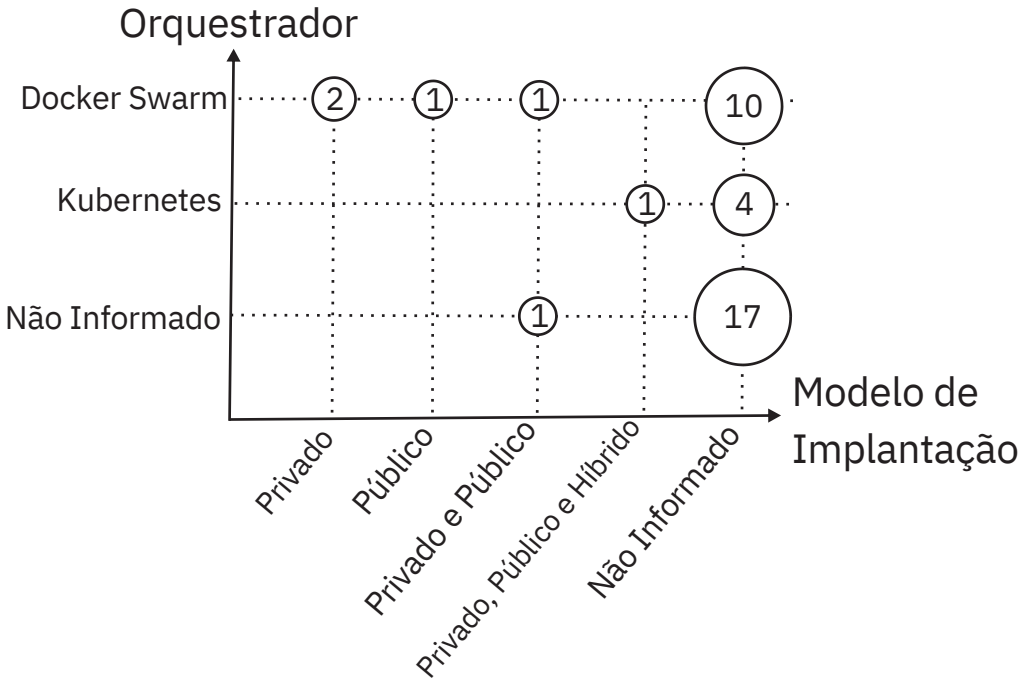


Figura 22 – Orquestrador x modelo de implantação.

3.5 Desafios e Direções de Pesquisa

Esta seção apresenta alguns desafios e direções de pesquisa encontrados com base nos 37 artigos estudados de forma mais aprofundada. A seção está estruturada conforme os principais desafios relacionados a microsserviços e consumo de energia, exploradas neste mapeamento (consumo de energia, gerenciamento de recursos, migração de contêiner e agendamento de tarefas).

Eficiência Energética:

O aumento do número de serviços e a demanda por alta disponibilidade pode resultar em um aumento significativo no consumo de energia. É importante monitorar e otimizar o consumo de energia para tornar a aplicação mais sustentável e alcançar eficiência energética. As máquinas físicas executam orquestradores de contêineres e algoritmos de gerenciamento de recursos continuamente para garantir a qualidade nos serviços, juntamente com aumento do número de servidores e serviços. Esse aumento torna o consumo de energia nos data centers cada vez mais elevado (VALERA et al., 2021b; JARWAR; ALI; CHONG, 2019). O consumo elevado de energia gera ainda outros problemas, como os altos custos dos equipamentos de refrigeração, sobrecargas e os custos energéticos em si. Existem várias

técnicas para mitigar o consumo de energia como balanceamento de carga e agendamento eficiente de tarefas, contudo essas técnicas podem degradar a qualidade do serviço e reduzir o desempenho dos aplicativos em nuvem (CANOSA-REYES et al., 2022; VALERA et al., 2021a). Portanto, novas técnicas de redução de consumo de energia são necessárias para otimizar arquiteturas de microsserviços.

Os principais desafios na eficiência energética de microsserviços incluem o *overhead* gerado pela comunicação entre serviços, que pode aumentar significativamente o consumo de energia devido ao tráfego de rede e serialização de dados. Além disso, a alocação dinâmica de instâncias pode levar a desperdício de recursos quando serviços são escalados de forma ineficiente. A virtualização e conteinerização adicionam camadas de abstração que impactam o consumo energético, enquanto o uso intensivo de CPU e memória para gerenciamento de estado distribuído também contribui para o problema. Por fim, a escolha de linguagens e *frameworks* pode influenciar o consumo energético, exigindo estudos sobre seu impacto em diferentes cargas de trabalho.

Direções futuras de pesquisa incluem a modelagem e simulação do consumo energético de microsserviços em plataformas como NS-3 ou *CloudSim Plus*, permitindo otimizações mais precisas. O desenvolvimento de algoritmos de escalonamento energético pode reduzir o desperdício de recursos ao ativar ou desativar instâncias conforme a demanda. Equilibrar eficiência energética e latência é outro desafio, exigindo técnicas para minimizar *trade-offs*. Além disso, estratégias de computação verde, como a execução de cargas em servidores otimizados para baixo consumo ou em regiões com fontes renováveis, podem reduzir a pegada de carbono. A otimização da rede, incluindo compressão de mensagens e escolha de protocolos mais eficientes, também pode contribuir para reduzir o impacto energético das arquiteturas baseadas em microsserviços.

Gerência de Recursos:

O crescente aumento na produção de dados gera um excesso de requisições nos serviços hospedados em data centers, dificultando o tratamento e causando atrasos no tempo de resposta (VALERA et al., 2021a; SINGH; AUJLA; BALI, 2021). A falta de um gerenciamento eficiente de dados não causa apenas atrasos, mas também descarte de requisições e até mesmo aumento no consumo de energia. Outro aspecto importante é como cada microsserviço gerencia os recursos do data center, como *CPU*, *RAM* e *ROM* de forma que possam lidar com suas requisições e evitar o desperdício de recursos. A computação em nuvem traz à tona crescentes preocupações em relação à elasticidade e balanceamento de cargas de trabalho sensíveis à latência. As arquiteturas de microsserviços permitem gerenciar melhor os recursos por meio da decomposição de serviços. Contudo, o ajuste de elasticidade não é uma tarefa simples e requer fatores, como experiência do usuário, aplicação, arquitetura e modelagem de parâmetros (Fontana de Nardin et al.,

2021; SINGH; AUJLA; BALI, 2021; AL-MOALMI et al., 2021). O gerenciamento dos recursos necessita que seja implementado de forma eficiente e consistente para garantir que a aplicação possa se adaptar às mudanças nas demandas de carga eficientemente.

O gerenciamento eficiente de recursos em microsserviços enfrenta desafios como a alocação dinâmica de instâncias, que pode resultar em desperdício de CPU, memória e energia quando o escalonamento não é otimizado. A sobrecarga da comunicação entre serviços, especialmente em arquiteturas distribuídas, pode impactar a latência e aumentar o consumo de rede. Além disso, a fragmentação dos recursos em múltiplos contêineres ou VMs pode levar a uma utilização ineficiente da infraestrutura, agravada por políticas de balanceamento de carga sub-ótimas. A heterogeneidade dos ambientes de execução, incluindo diferentes provedores de nuvem e arquiteturas híbridas, também dificulta a aplicação de estratégias unificadas de gerenciamento.

Pesquisas futuras devem focar no desenvolvimento de algoritmos inteligentes para escalonamento dinâmico que equilibrem desempenho e eficiência no uso de recursos. Estratégias baseadas em aprendizado de máquina podem prever padrões de carga e otimizar a alocação de microsserviços em tempo real. Técnicas de orquestração avançadas, como compactação de carga e migração eficiente entre nós, podem reduzir a fragmentação e melhorar a utilização dos servidores. Além disso, a integração de métricas energéticas nas plataformas de gerenciamento pode permitir uma abordagem mais sustentável. A otimização de comunicação, por meio de compressão de dados e protocolos mais eficientes, também é essencial para reduzir *overheads* desnecessários e melhorar a performance global do sistema.

Migração de Contêiner:

A migração de contêiner é uma técnica que pode ser usada para controlar as cargas de trabalho de aplicações em nuvem. No entanto, as migrações são custosas em termos de consumo de energia e degradação de desempenho, o que geralmente não é contabilizado em modelos existentes. A alocação e consolidação de recursos com migração podem ser considerados problemas multidimensionais de *bin-packing* onde *bins* são *hosts/VMs* e itens são contêineres, para alocação de contêiner (KHAN et al., 2021a; GHOLIPOUR; ARIANYAN; BUYYA, 2020). Além disso, quando os *hosts* são heterogêneos, a complexidade do problema aumenta por conta do número de tipos de recursos necessários para acomodar *VMs* ou contêineres. A migração mal planejada também pode causar contenção de recursos, menor desempenho e maior custo energético (KHAN et al., 2021a; VALERA et al., 2021b).

Agendamento de Tarefas:

O agendamento eficiente do fluxo de tarefas é um dos desafios mais complicados em uma arquitetura de microsserviços, devido à variação nos fluxos de tráfego gerados

por diversos aplicativos em nuvem. Agendar tarefas de maneira eficiente em ambientes computacionais massivos, dinâmicos, elásticos, como arquiteturas de microsserviços permanece uma tarefa desafiadora (RANJAN et al., 2020; CANOSA-REYES et al., 2022). O agendamento de tarefas apresenta desafios extras, pois as tarefas com prioridades mais altas devem ser executadas antes das tarefas com prioridades mais baixas. Além da questão de prioridade das tarefas, tem a questão de poder computacional necessário para execução sem prejudicar o desempenho dos serviços. O agendamento ineficiente traz ainda a tona problemas de maior consumo de energia e violações de SLA (CANOSA et al., 2018; MENDES; SIMÃO; VEIGA, 2019).

Além dos principais desafios em aberto, algumas métricas essenciais para microsserviços foram pouco exploradas pelos trabalhos relacionados no âmbito do consumo energético. Métricas relacionadas ao desempenho dos microsserviços estão em alta e receberam muito foco de pesquisa, porém métricas fundamentais como confiabilidade, disponibilidade e sustentabilidade carecem de um foco maior e se mostram como oportunidades de pesquisa na área de consumo energético em microsserviços.

3.6 Considerações Finais

Este capítulo apresentou os resultados de uma revisão sistemática sobre consumo energético em arquiteturas de microsserviços. Como contribuição principal, compilamos alguns destaques desse trabalho. A maioria dos trabalhos concentrou seus estudos em métricas de desempenho. O foco em desempenho já era esperado, uma vez que ter um bom desempenho é essencial para qualquer aplicação de microsserviços. Nove trabalhos focaram em métricas de elasticidade, sendo a métrica mais explorada depois da performance. O foco na elasticidade ocorreu, pois o gerenciamento eficiente de recursos é uma boa maneira de reduzir o gasto de energia das aplicações de microsserviços. Poucos trabalhos exploraram métricas de confiabilidade, disponibilidade e sustentabilidade, mostrando-se oportunidades de pesquisa.

Os métodos de avaliação nos microsserviços são a simulação, em segundo lugar a mensuração e em terceiro lugar a modelagem. A maioria dos trabalhos com simulação usaram o simulador *CloudSim* por oferecer recursos para arquiteturas em nuvem. A mensuração também foi um destaque, mas apenas dois trabalhos usaram a modelagem. Apesar da modelagem pouco utilizada, é recomendado explorar tal tipo de avaliação graças ao seu poder de abstração e baixo custo, mesmo tendo uma precisão menor que experimentos práticos.

Em relação ao tipo de arquitetura, a computação em nuvem foi o destaque, as arquiteturas de borda e híbrida foram menos utilizadas nos trabalhos. Além disso, a maioria dos trabalhos se concentraram no tipo de arquitetura e não entraram em detalhes sobre o

modelo de implantação. A maioria dos trabalhos também não detalhou o orquestrador utilizado, contudo, os que detalharam se concentraram mais no *Docker Swarm* que no *Kubernets*. Como todos os trabalhos têm foco em consumo de energia em microsserviços, o contexto mais estudado foi o de data center, alguns também estudaram áreas relacionadas a IoT.

O capítulo enfrentou três ameaças para a validação do estudo. (i) *Perguntas de pesquisa* que podem não ter coberto todos os trabalhos relevantes a respeito do consumo de energia em microsserviços. Todavia, consideramos uma cadeia de pesquisa de texto amplo, além da realização de muitas discussões para validar as questões. (ii) *Viés de seleção*, não temos como garantir que todos os estudos relevantes foram abordados. Alguns estudos relevantes possuem a possibilidade de não terem sido coletados na etapa de busca. Buscamos mitigar essa ameaça o máximo possível seguindo as referências nos estudos relevantes. (iii) *Extração de dados*, classificamos os estudos com base em nosso julgamento. Contudo, é possível que alguns estudos possam ter sido classificados incorretamente. Para mitigar essa risco, seis pesquisadores realizaram o processo de classificação. Este trabalho trouxe sintetizou informações relevantes para entendimento do consumo de energia em microsserviços e mostrou quais os pontos em aberto e direções futuras acerca do tema.

4 Modelo de Encadeamento de funções de serviço de rede: um estudo de desempenho aplicado em microsserviços

Este capítulo apresenta um modelo de SPN para avaliação de desempenho em três diferentes arquiteturas de SFC aplicadas a microsserviços. A revisão sistemática trouxe informações sobre arquitetura e consumo energético em microsserviços para serem utilizados em diferentes arquiteturas de SFC aplicadas a microsserviços. O modelo é avaliado sob diferentes perspectivas por meio de estudos de caso. O modelo é validado por meio de simulação no simulador *Graphical Network Simulator 3* (GNS3).

4.1 Contextualização

O desenvolvimento da tecnologia impulsionou o aumento do tráfego de rede e a geração de novos serviços de rede. Nas redes tradicionais, os serviços de rede eram altamente acoplados ao *hardware*, causando problemas como altos custos de manutenção e atualização de *software*. O surgimento do paradigma NFV e SFC teve como objetivo mitigar este problema (RUI et al., 2020; RAHMAN et al., 2023). Diversos trabalhos avaliaram a otimização do SFC em diferentes categorias utilizando modelos analíticos. Alguns trabalhos avaliaram a otimização de atrasos de SFCs usando redes de Petri (RYGIELSKI; SELIUCHENKO; KOUNEV, 2016; SCHNEIDER et al., 2019; LI et al., 2022a). No entanto, nenhum dos estudos comparou o desempenho de diferentes arquiteturas SFC (sequencial, paralela e híbrida) utilizando modelos analíticos. Neste trabalho, propomos o uso de SPN para modelar a otimização do atraso em arquiteturas SFC sequenciais, paralelas e híbridas aplicadas a microsserviços.

Em resumo, as principais contribuições deste capítulo são:

- O modelo SPN proposto oferece recursos úteis para a configuração e planejamento de arquiteturas SFC, permitindo a configuração de diferentes VNFs e cenários.
- Avaliamos o modelo e comparamos diferentes arquiteturas com base em quatro estudos de caso e diferentes métricas.
- Realizamos uma análise de sensibilidade utilizando o método *Design of Experiments* (DOE). O DOE permitiu identificar os componentes mais impactantes do sistema e suas interações.

- O primeiro estudo de caso comparou o número de solicitações concluídas com sucesso e o número de falhas em cada arquitetura.
- O segundo estudo de caso analisou o *Mean Time To Absorption* (MTTA) de cada solicitação.
- O terceiro estudo de caso analisou as *Cumulative Distribution Functions* (CDF) referente ao MTTA de cada arquitetura.
- O quarto estudo de caso comparou os fatores de maior impacto descobertos no DoE por meio de uma visualização 3D.

O modelo foi validado por meio de simulação no simulador GNS3 (VELIEVA; KOROLKOVA; KULYABOV, 2014). Os resultados mostraram que o modelo SPN proposto corresponde à realidade e é útil para o planejamento de arquiteturas SFC aplicadas a microsserviços.

4.2 Trabalhos Relacionados

Esta seção apresenta trabalhos relacionados que analisaram sistemas baseados em modelos analíticos no contexto de NFV. Artigos que funcionavam no contexto da NFV, mas não utilizavam modelos analíticos, não foram considerados, pois estavam fora do escopo deste trabalho. A tabela 4 compara trabalhos relacionados. Todos os trabalhos em questão utilizam algum modelo analítico (SPN, Cadeia de Markov ou Teoria das Filas). Nosso trabalho utilizou o SPN por sua maior flexibilização e facilidade de implementar diferentes arquiteturas de SFC em um mesmo modelo. Todos os trabalhos adotaram métricas diferentes das utilizadas neste estudo. Nem todos os estudos abordaram especificamente a SFC. Nem todos os estudos validaram seus modelos, ou seja, comparam os resultados do modelo com outro avaliação como uma simulação ou mensuração. Em Mochalov et al., (MOCHALOV et al., 2021), métodos são desenvolvidos para implementar processos de funcionalidade paralela. O trabalho de Mochalov et al. distribui recursos computacionais e de rede entre processos e fornece meios para evitar impasses. Um modelo de Rede de Petri Colorida (CNP) foi desenvolvido e validado por meio de simulações no *CPN Tools*.

O trabalho de (RYGIELSKI; SELIUCHENKO; KOUNEV, 2016) estudou a *Descartes Network Infrastructure* (DNI), uma abordagem de modelagem de desempenho de rede de data center. O estudo aborda como o DNI é usado para suportar novos paradigmas de rede, como *Software Defined Networks* (SDN) e NFV. Além disso, o trabalho mostra como redes baseadas em SDN podem ser modeladas usando DNI e como elas são posteriormente transformadas em *Queuing Petri Nets* (QPN). O trabalho de Rygielski et al. utilizou a ferramenta Iperf para tratamento do tráfego e o simulador *SimQPN* para

validação. No estudo de (SCHNEIDER et al., 2019), um modelo QNP é desenvolvido para analisar o comportamento do VNF. O estudo apresentou um fluxo de trabalho baseado em ferramentas que suporta a especificação de serviços de rede. O *TimeNet* foi utilizado para modelagem e o simulador *OMNeT++* foi utilizado para validação. Schneider et al. também conduziu um estudo de caso, mostrando como os desenvolvedores podem se beneficiar de *insights* analíticos, como prever o impacto de diferentes configurações de serviço.

O trabalho de (FAHMIN et al., 2018) desenvolveu um modelo de enfileiramento SDN com NFV dentro ou fora do controlador. As arquiteturas SDN com NFV são modeladas e analisadas usando um simulador customizado implementado pelos autores (nome não fornecido). Os resultados mostram uma redução no atraso dos pacotes de 68,83%. O estudo de (RUI et al., 2020) apresentou um método de avaliação de confiabilidade SFC e um algoritmo de otimização de confiabilidade. O estudo de Rui et al. analisou a relação de composição e os fatores que influenciam a confiabilidade do SFC usando um modelo de Rede de Petri no simulador Oris. Os resultados mostraram que o algoritmo proposto reduz o custo e o impacto da preempção de recursos na confiabilidade do serviço.

No estudo de (LIU et al., 2022), um modelo CNP é proposto para estabelecer um método de verificação offline que analisa se cada implantação de *Service Function* (SF) no SFC está correta após a implantação online. Liu et al. usaram ferramentas CPN para validar seu modelo por meio de simulação. Os resultados experimentais mostraram que o método proposto pode verificar efetivamente se cada SF no SFC implantado está implantado corretamente. O trabalho de (AI et al., 2022) apresentou um novo método de construção do SFC para evitar possíveis conflitos entre os FS que compõem o SFC. No trabalho de Ai et al., um modelo CNP é desenvolvido e avaliado através de simulação com *CPN Tools*.

Apenas os trabalhos de (MAURO et al., 2017), (MAURO et al., 2017) e (MAURO et al., 2018) não validaram suas propostas. O trabalho (MAURO et al., 2017) propôs uma arquitetura virtualizada de *IP Multimedia Subsystem* (IMS) modelada por uma abordagem hierárquica de três níveis com *Reliability Blocks Diagrams* (RBD), Árvores de Falhas e *Continuous-Time Markov Chains* (CTMC). Di et al. realizou avaliação de disponibilidade e análise de sensibilidade utilizando a ferramenta SHARPE (*Symbolic Hierarchical Automated Reliability and Performance Evaluator*), que fornece um conjunto de ferramentas para análise de confiabilidade de sistemas. O trabalho de (MAURO et al., 2017) apresenta uma arquitetura baseada em SFC para encontrar a configuração ideal que garanta a disponibilidade exigida em sistemas de telecomunicações. Di et al. conduziram um estudo de disponibilidade de SFC avaliado por meio de análises de estado estacionário e de sensibilidade. O estudo utilizou a ferramenta SHARPE para avaliar um modelo híbrido de Redes de Recompensa Estocástica (SRN) e RBD.

O estudo (MAURO et al., 2018) propôs uma avaliação de disponibilidade de uma

Tabela 4 – Trabalhos Relacionados.

Trabalho	**Tipo de Modelo**	**Métricas**	**Arquitetura SFC**	**Validação**
(**RYGIELSKI; SELIU-CHENKO; KOUNEV, 2016**)	Petri net	Vazão	Não	Sim
(**MAURO et al., 2017**)	CTMC, RBD e Fault Trees	Disponibilidade e taxa de falha	Não	Não
(**MAURO et al., 2017**)	SRN e RBD	Disponibilidade, indisponibilidade, taxa de migração e falha	Sim	Não
(**MAURO et al., 2018**)	SRN e RBD	Disponibilidade, indisponibilidade, taxa de reparo e falha	Não	Não
(**FAHMIN et al., 2018**)	Filas	Atraso médio de pacotes	Não	Sim
(**SCHNEIDER et al., 2019**)	Petri net	Rede de Atraso médio	Não	Sim
(**RUI et al., 2020**)	Petri net	Confiabilidade e custo	Sim	Sim
(**MOCHALOV et al., 2021**)	Petri net	Tempo de execução, distribuição de acessos e do tempo	Não	Sim
(**LIU et al., 2022**)	Petri net	Número de nós e arcos conectados	Sim	Sim
(**AI et al., 2022**)	Petri net	Número de nós e arcos conectados	Sim	Sim
(**LI et al., 2022b**)	Filas	Latência e aceitação do serviço	Sim	Sim
(**LI et al., 2022b**)	Filas	Disponibilidade	Não	Sim
Este trabalho	Petri net	Número de sucessos e falhas, MTTA e CDF	Sim	Sim

estrutura IMS virtualizada dividida em duas etapas. O primeiro passo foi uma fase de modelagem através de RBD e SRN. A segunda etapa foi uma análise numérica para caracterizar a configuração ideal do IMS virtualizado para satisfazer uma condição de alta disponibilidade. Di et al. utilizou o SHARPE para realizar as duas etapas do estudo. O trabalho de Li et al. (LI et al., 2022b) propôs um modelo de Rede de Petri para futuras infraestruturas 5G, levando em consideração as particularidades do SFC. O trabalho utiliza simulação de eventos discretos para analisar a eficácia do escalonamento automático e do isolamento de rede no nível de microsserviços. O modelo foi feito em um programa *Python* baseado na representação em Rede de Petri, e a simulação foi realizada no *SimPy*, que também é baseado em *Python*. Os resultados mostram que o escalonamento automático e o isolamento da rede podem aumentar a resiliência da rede quando o tráfego muda abruptamente.

O trabalho de pfitscher et al. (PFITSCHER et al., 2019) apresenta um modelo de filas para atribuir valores de culpa a cada VNF na diminuição do desempenho de um serviço de rede em uma arquitetura SFC. O trabalho também apresenta um algoritmo híbrido baseado em regressão linear e redes neurais que ajusta os parâmetros do modelo de acordo com diferenças no ambiente. Os resultados mostram que o modelo foi capaz de identificar gargalos e quantificar reduções de desempenho, identificando até mesmo a causa raiz das reduções. O trabalho de Heideker et al. (HEIDEKER; KAMIENSKI, 2022) mostra um modelo de filas que usa uma técnica de *Network Queue Assessment* (NQA) para detectar gargalos no SFC. NQA é uma técnica baseada na ocupação de filas de rede e não necessita de *feedback* do usuário final, conhecimento sobre sua arquitetura interna ou qualquer definição de limite. Os resultados de heideker et al. mostram que o NQA alcançou alta precisão na detecção de gargalos.

O trabalho de Li et al. (LI et al., 2022a) apresenta um método de Rede de Petri para modelagem de redes 5G e além das redes de telecomunicações. O trabalho modela uma infraestrutura física, uma infraestrutura virtual, serviços de rede, seus comportamentos e dependências SFC. A *CPN Tools* realizou as simulações, que confirmaram que o modelo proposto teve sucesso na avaliação da resiliência de sistemas complexos de telecomunicações. Este trabalho, por sua vez, propõe um modelo de SPN para otimizar o atraso SFC. Somente os trabalhos de (LI et al., 2022b), (LI et al., 2022a), (HEIDEKER; KAMIENSKI, 2022) e (PFITSCHER et al., 2019) exploraram a otimização de atraso SFC. Nenhum dos trabalhos relacionados realizou análise de sensibilidade.

Este trabalho, por sua vez, propõe um modelo de SPN para otimizar o atraso SFC. O modelo proposto aborda as arquiteturas SFC sequencial, paralela e híbrida, considerando os componentes que podem ou não ser paralelizáveis. Os trabalhos de Li et al. (LI et al., 2022b; LI et al., 2022a) exploraram a otimização de atraso SFC, mas ambos abordam apenas a arquitetura sequencial. O trabalho de Mochalov et al. (MOCHALOV et al., 2021) abordou a arquitetura paralela, mas não o SFC. Somente os trabalhos de heideker et al. (HEIDEKER; KAMIENSKI, 2022) e pfitsche et al. (PFITSCHER et al., 2019) exploraram a otimização de atraso SFC com outros tipos de arquitetura. Portanto, este trabalho foi o único dentre os relacionados que abordou um modelo SPN para otimização do atraso em diferentes arquiteturas SFC aplicadas a microsserviços. O modelo foi desenvolvido no *Mercury* (MACIEL et al., 2017); Realizamos uma análise de sensibilidade utilizando o método DoE e validamos o modelo por simulação utilizando o simulador GNS-3 (*Graphical Network Simulator-3*) (VELIEVA; KOROLKOVA; KULYABOV, 2014).

4.3 Arquitetura

Esta seção apresenta uma visão geral da arquitetura SFC aplicada a microsserviços proposta neste estudo. A arquitetura SFC proposta consiste em quatro componentes que podem ser organizados de forma sequencial, paralela ou híbrida. A Figura 23 apresenta as arquiteturas SFC sequencial, paralela e híbrida. Na arquitetura sequencial (a), o fluxo de solicitações (representado pelo círculo branco Início passa para o *Log* VNF, que registra os dados de tráfego. O *Billing* então soma o tráfego de cada usuário. O *Traffic Analysis* monitora todo o tráfego para gerar relatórios estatísticos. Por fim, a *Cripto* é responsável por realizar a criptografia para fins de segurança. O círculo preto Final representa o final do fluxo, onde o sistema absorve as solicitações. Na arquitetura SFC paralela, todas as VNFs são organizadas em paralelo (b), uma vez que todas as quatro VNFs podem ser paralelizadas. O fluxo funciona de forma semelhante ao da arquitetura sequencial, mas cada componente executa sua tarefa independentemente.. Vale ressaltar que cada VNF independe da outra para desempenhar sua função, cada uma representa um serviço isolado. Todos funcionam separadamente com base em solicitações e, portanto, podem ser paralelizados sem afetar o funcionamento do sistema.

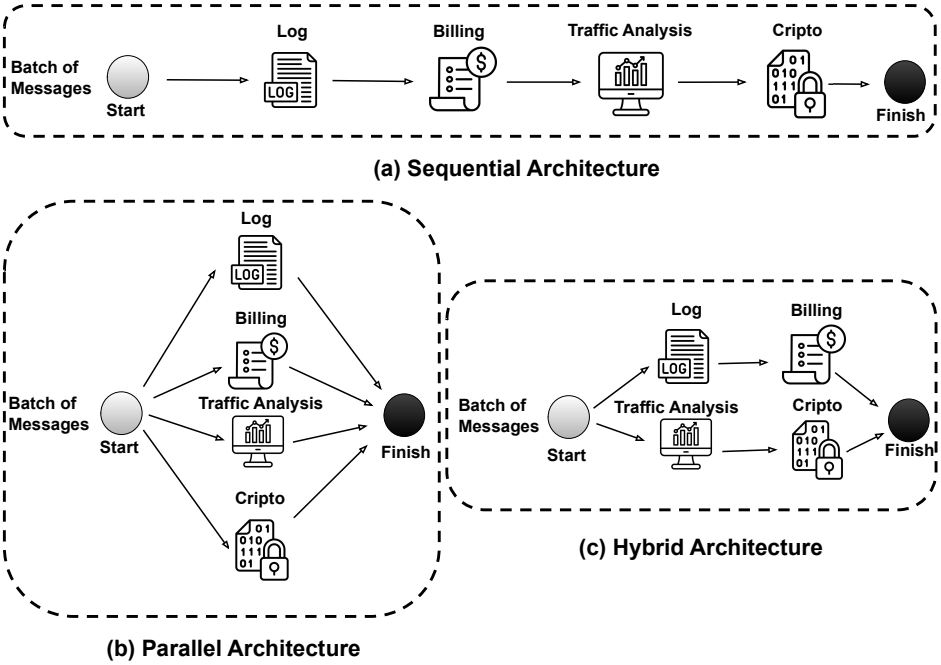


Figura 23 – Arquiteturas SFC sequenciais, paralelas e híbridas.

A arquitetura híbrida SFC (c) combina arquitetura sequencial e paralela. As VNFs podem ser organizadas de ambas as formas na mesma configuração. A arquitetura híbrida é formada por duas arquiteturas sequenciais que funcionam em paralelo. O fluxo passa sequencialmente por *Log* e *Billing* e sequencialmente por *Traffic Analysis* e *Cripto*. As VNFs podem cumprir suas funções e executar o serviço nas três arquiteturas. As arquiteturas

apresentadas são formas de organização dos componentes, e não precisam necessariamente seguir a mesma ordem das Figuras. Portanto, as arquiteturas permitem que as VNFs sejam reorganizadas e até removidas ou adicionadas conforme necessário (para organização paralela, a VNF deve ser paralelizável). Consideramos que cada uma das VNFs está alocada dentro de um contêiner em uma arquitetura de microsserviços. Dessa forma, cada VNF realiza um serviço isolado que se comunica com as demais VNFs de forma sequencial, paralela ou híbrida para realizar múltiplos serviços em conjunto, o que constitui a cadeia de serviço. ETSI [1] fornece alguns casos de uso com NFVs que podem orientar em cenários mais complexos. Porém, optamos por uma arquitetura mais simples, utilizando VNFs genéricas para facilitar o entendimento e demonstrar que o modelo proposto (na Seção seguinte) permite criar uma arquitetura com qualquer VNF.

4.4 Modelo

Esta seção apresenta o modelo SPN proposto para representar a arquitetura apresentada na Seção 4.3. O modelo foi construído na ferramenta *Mercury* (MACIEL et al., 2017). Segundo (RUI et al., 2020; LI et al., 2022a), uma VNF normalmente possui dois tempos associados: um tempo de coleta ou transmissão e um tempo de execução de dados. Além disso, segundo (RUI et al., 2020), cada ação de uma VNF deve associar uma probabilidade de fracasso ou sucesso. As falhas consideradas neste trabalho são genéricas e podem abranger os mais diversos cenários como falta de recursos, falhas de *software*, entre outros. Esses dois aspectos foram considerados na modelagem da arquitetura SFC aplicada a microsserviços. A Figura 24 apresenta um submodelo para representar o comportamento de um SFC sequencial. O local `Pre` (Requisições) representa a chegada de requisições a serem processadas pelo SFC. A variável `K` indica o número de solicitações em `Pre`. Quando a transição temporizada `Tt` (Transição de Transmissão) é acionada, um *token* sai de `Pre` para `Ptq` (Fila de Transmissão). Este disparo indica que a solicitação foi enviada para a fila de transmissão, portanto a transição `Tt` representa o tempo de transmissão do VNF. A transmissão da solicitação VNF tem probabilidade de falha ou sucesso, Ptq é um local intermediário sem tempo associado para aplicação desta probabilidade. Quando o *token* chega em `Ptq`, as transições `T0` (sucesso na transmissão) e `T2` (falha na transmissão) competem pelo disparo, o que ocorre dependendo das probabilidades de sucesso e falha, respectivamente. Se `T0` for disparado, o *token* será enviado para o local `Ptr` (Transmitido), o que indica que o *token* foi transmitido com sucesso. Quando a transição temporizada `Te` (Transição de Execução) é acionada, um token é removido de `Ptr` e é criado em `Pex` (Executado). Este gatilho indica que a solicitação foi feita, portanto a transição `Te` representa o tempo de execução do VNF, ou seja, o tempo que a VNF leva para processar a requisição. A execução da solicitação VNF também apresenta probabilidade de falha ou

[1] <https://www.etsi.org/technologies/nfv/>

sucesso. Quando o *token* chegar em `Pex`, a transição imediata `T1` (sucesso na execução) ou a transição imediata `T3` (falha na execução) será disparada. Se `T1` for disparado, o *token* será enviado para o local `Pab` (Absorção), deixando o *token* executado com sucesso. Se as transições imediatas `T2` ou `T4` forem acionadas, um *token* será enviado de `Ptq` ou `Pex` para o local `Pfa` (Failed), indicando que a transmissão ou a execução falhou. O atraso no modelo é representado por todo o ciclo de vida das solicitações no SFC, desde o processo de transmissão até a tomada de decisão. Quanto mais tempo demorar para completar este ciclo, maior será o atraso. À medida que o SFC aumenta, o tempo de ciclo tende a aumentar e, consequentemente, o atraso aumenta.

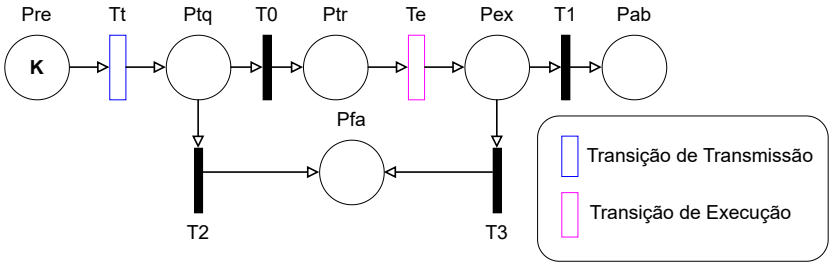


Figura 24 – Submodelo SPN para um VNF.

A Figura 25 apresenta o modelo SPN da arquitetura SFC proposta. O modelo contém quatro estruturas semelhantes ao submodelo apresentado anteriormente, com a adição de um novo local e uma nova transição imediata. Cada uma das quatro estruturas representa um VNF. O número de VNFs no modelo foi escolhido para facilitar a representação dos três tipos de arquiteturas, mas poderia ser facilmente expandido ou reduzido. Um novo local chamado `Pfi` (Concluído) foi adicionado entre `Pex` e `Pab` em cada uma das estruturas. Uma nova transição imediata também foi adicionada logo após `Pfi` em cada uma das estruturas (`T4, T9, T14, T19`); esta transição enviará o *token* para `Pab` seguindo a mesma operação do submodelo. Porém, cada `Pfi` de cada estrutura (`Pfi`, `Pfi2`, `Pfi3`, `Pfi4`) também está vinculado a `Tt` de cada estrutura (`Tt`, `Tt2`, `Tt3`, `Tt4`). Esta configuração foi montada para representar a arquitetura sequencial entre as quatro VNFs. Dessa forma, ao invés de ser concluída em uma VNF, a solicitação será transmitida para a próxima sequencialmente. Este comportamento continuará até que a solicitação chegue em `Pab` através do disparo de `T19` ou até que falhe devido à transmissão ou execução de uma das VNFs. O comportamento sequencial é definido por condições de guarda nas transições temporizadas `Tt2` e `Tt3` e nas transições imediatas `T4`, `T9` e `T14`. A variável `conf` define a arquitetura do sistema. Quando conf é definido como 1, as transições imediatas `T4`, `T9` e `T14` não podem ser disparadas, mas `Tt2` e `Tt3` serão liberadas. Portanto, quando conf for igual a 1, o sistema assume a arquitetura sequencial entre as quatro VNFs.

Três novas transições temporizadas chamadas `Ttp`, `Ttp2` e `Ttp3` (*Parallel Transmission Transition*) foram adicionadas para representar a arquitetura paralela e híbrida. Estas

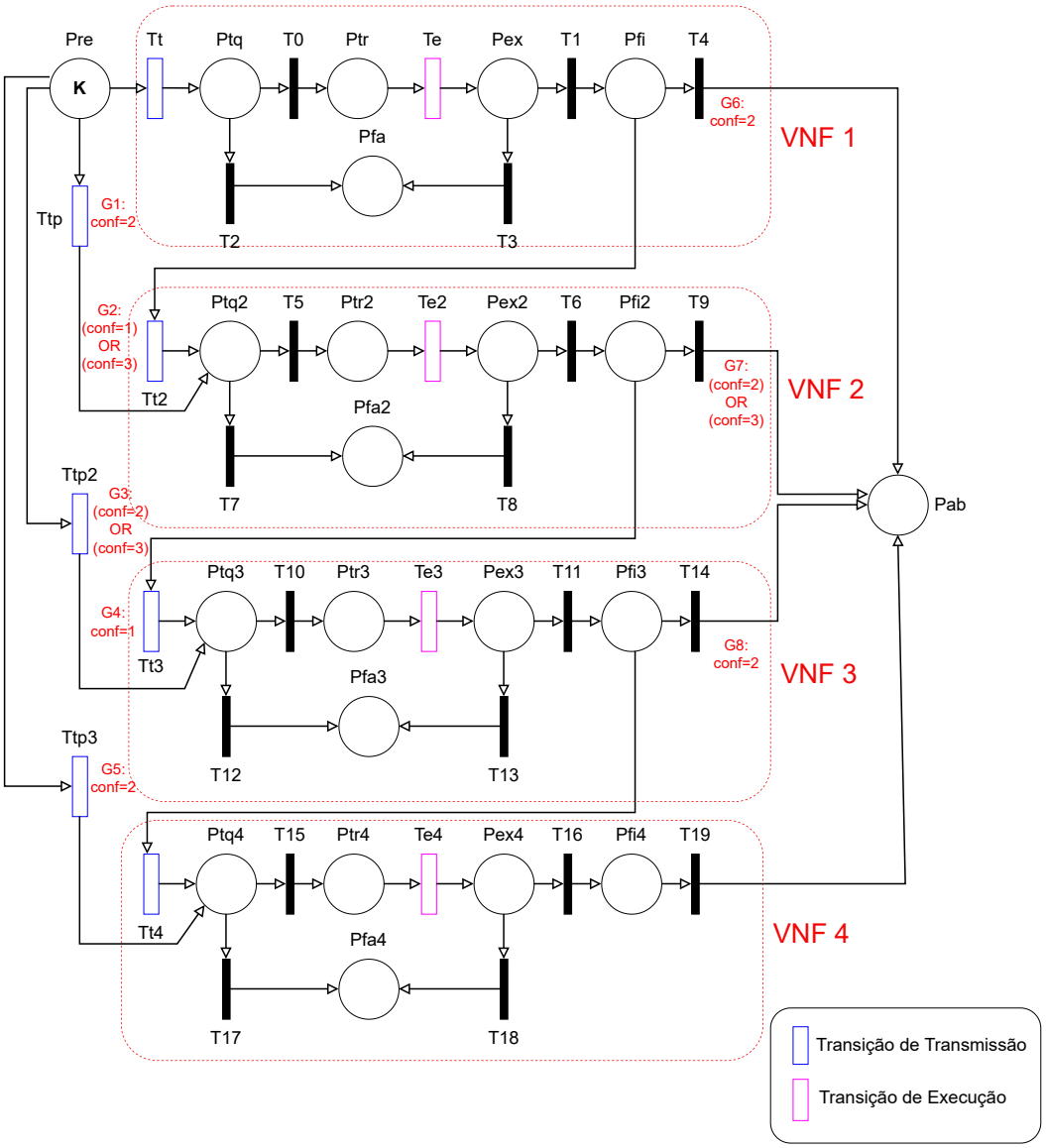


Figura 25 – Modelo SPN da arquitetura SFC.

três novas transições também têm condições de guarda associadas. Quando conf for igual a 1, essas novas transições não poderão ser acionadas e não afetarão o comportamento sequencial. Quando conf é definido como 2, as transições temporizadas `Tt2` e `Tt3` não podem ser acionadas, impedindo a operação sequencial. As transições imediatas `T4`, `T9` e `T14` serão liberadas, permitindo que cada VNF seja completada separadamente. Além disso, as novas transições temporizadas `Ttp`, `Ttp2` e `Ttp3` serão lançadas. `Ttp`, `Ttp2` e `Ttp3` também representam o tempo de transmissão, mas ao invés de serem transmitidos da VNF anterior, serão transmitidos diretamente de `Pre`. Portanto, quando *conf* for igual a 2, o sistema assume uma arquitetura paralela entre as quatro VNFs. Finalmente, quando *conf* é definido como 3, as transições imediatas `T4` e `T14`, e as transições temporizadas `Ttp` e `Ttp3` não podem disparar. As demais transições serão liberadas. Desta forma, as estruturas VNF 1 e VNF 2 assumirão um comportamento sequencial, enquanto as estruturas VNF

3 e VNF 4 também assumirão um comportamento sequencial. Os dois comportamentos sequenciais funcionam em paralelo, formando uma arquitetura híbrida. Portanto, quando *conf* for igual a 3, o sistema assume a arquitetura híbrida entre as quatro VNFs. Para este modelo foram exploradas duas métricas de interesse: número de sucessos e número de falhas. $E_{locationname}$ representa a expectativa estatística de haver *tokens* em "*locationname*", onde $E_{locationname} = (\sum_{i=1}^{n} P(m(Location) = i) \times i)$, sendo n o maior número de *tokens* que *Local* pode conter. Em outras palavras, $E_{locationname}$ indica o valor esperado dos *tokens* naquele local. A equação do número de sucessos representa a esperança estatística dos *tokens* que chegam em `Pab`. Assim, o número de sucessos (*NSuccess*) pode ser obtido pela Equação (4.1). O número de falhas representa a soma das esperanças dos *tokens* de chegar a `Pfa`, `Pfa2`, `Pf3` e `Pfa4`. Portanto, o número de falhas (*NFail*) pode ser obtido pela equação (4.2). O modelo pode ser facilmente ajustado para outros cenários removendo ou adicionando VNFs e excluindo ou adicionando novas estruturas VNF. Dessa forma o modelo é genérico o suficiente para definir arbitrariamente quantos e quais VNFs irão compor a arquitetura. Ao substituir ou adicionar novas VNFs, é necessário alterar as transições temporizadas pelos valores das novas VNFs. Além disso, para configurar o tipo de arquitetura é necessário alterar o valor da variável conf (1 - Sequencial, 2 - Paralela e 3 - Híbrida). Vale ressaltar que cada VNF representa um serviço na arquitetura de microsserviços, mas precisam se comunicar para formar a SFC. Portanto é necessário analisar se o VNF em questão pode ser paralelizável com outro dentro da cadeia de serviço.

$$\mathbf{NSuccess} = E\{\#Pab\} \tag{4.1}$$

$$\mathbf{NFail} = (E\{(\#Pfa)\} + E\{(\#Pfa2)\} + E\{(\#Pfa3)\} + E\{(\#Pfa4)\}) \tag{4.2}$$

Métrica - *Mean Time To Absorption* (MTTA)

O MTTA serve para analisar o tempo médio para que as solicitações do sistema sejam absorvidas. MTTA é o tempo esperado para atingir um impasse de marcação (o *token* não tem mais para onde seguir, chegou ao destino). Na teoria da rede de Petri e da cadeia de Markov, chamamos de absorção de impasse, quando um *token* não tem onde consumir. O estado absorvente pode ser calculado gerando um CTMC associado ou por análise numérica (NELSON, 2013). No presente trabalho utilizamos análise transitória, que gera um CTMC a partir de um SPN e calcula o estado absorvente. Usando a Figura 26 como exemplo, o MTTA é baseado em um conjunto de probabilidades para quando um *token* se move de `Início` para `Fim`. Neste exemplo, a absorção é o resultado do disparo das transições temporizadas `T1` e `T2`. A absorção de K elementos pode ser feita em paralelo dependendo da quantidade de agentes de migração disponíveis (marcação C) (GERMAN,

2000; MARSAN et al., 1998). No modelo proposto, o local que representa o estado de absorção é `Pab`.

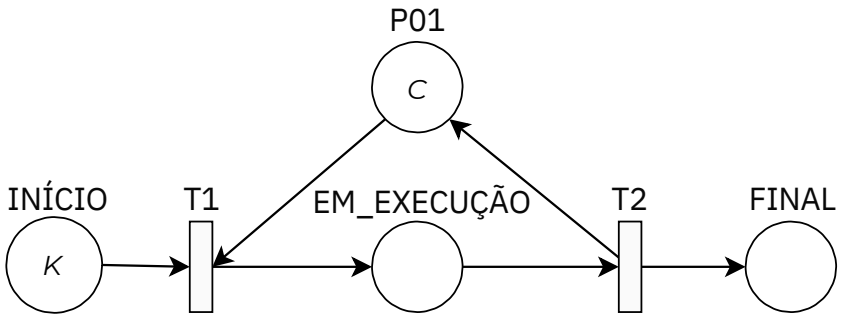


Figura 26 – Exemplo de modelo SPN.

O modelo proposto neste trabalho também calcula o CDF. Com o CDF é possível estimar a probabilidade de terminar a execução antes de um tempo específico $[P(T < t)]$ e a probabilidade de terminar a execução em um intervalo de tempo $[P(t1 < T < t2) = P(T < t2) - P(T \leq t1)]$.

4.5 Parâmetros do Modelo e Análise de Sensibilidade

Nesta seção, mostramos os parâmetros do modelo e o resultado da análise de sensibilidade utilizando o método *Design of Experiments* (DoE). Utilizamos a ferramenta Mercury (MACIEL et al., 2017) para representar o modelo e realizar os experimentos. A tabela 5 lista os parâmetros usados no modelo. A tabela 6 apresenta as taxas de falha e transmissão utilizadas no modelo. Os parâmetros de transmissão e execução foram definidos com base em simulações realizadas no simulador GNS3. Os valores de falha e taxa de execução foram baseados no estudo de Rui et al. (RUI et al., 2020), execuções de modelos e simulações realizadas no simulador GNS3 para obter um impacto significativo. Os valores desses parâmetros são valores médios e fixos. Esses valores devem ser alterados caso as VNFs que compõem a arquitetura sejam alteradas.

Tabela 5 – Parâmetros do modelo usados em estudos de caso.

VNF	**Tempo de Transmissão (ms)**	**Tempo de Execução (ms)**
Log	0,0016	1,3811
Billing	0,0037	1,0232
Traffic Analysis	0,0023	1,0093
Cripto	0,0057	0,9341

Tabela 6 – Parâmetros de falha e sucesso do modelo usados no estudo de caso.

Sucesso na Transmissão	**Falha na Transmissão**	**Sucesso na Execução**	**Falha na Execução**
70%	**30%**	**90%**	**10%**

A tabela 7 apresenta os fatores e níveis utilizados no DoE. Esses fatores foram escolhidos devido a possibilidade de representarem gargalos, uma vez que são os maiores tempos de execução e as probabilidades de sucesso e falha. A tabela 8 mostra as combinações entre os fatores e seus níveis. Os primeiros três fatores, EXVNF1, EXVNF2 e EXVNF3, representam o tempo de execução dos três primeiros VNFs. A variação para os três primeiros fatores foi do valor base (valores das Tabelas 5 e 6) até mais 50% do valor base. Os dois últimos fatores são TSW e ESW, representando a probabilidade de transmissão e execução bem-sucedidas. A variação destes dois fatores foi de 10% até o seus valores base.

Tabela 7 – Fatores e configurações.

Nome do Fator	**Configuração Baixa**	**Configuração Alta**
EXVNF1	1.3811 ms	2.7622 ms
EXVNF2	1.0232 ms	2.0464 ms
EXVNF2	1.0232 ms	2.0186 ms
TSW	0.1%	0.7%
ESW	0.1%	0.9%

A Figura 27 apresenta o gráfico do fator de efeito para o modelo utilizando a métrica MTTA. O gráfico mostra quais fatores e combinações impactam mais o resultado do modelo para as três configurações. O fator TSW foi o mais impactante para o MTTA, pois a probabilidade de transmissão é menor que a de execução, permitindo maiores falhas. O fator EXVNF1 foi o segundo mais relevante para o MTTA, pois possui o maior tempo de execução entre os VNFs. A combinação entre ESW e EXVNF1 foi a mais relevante, representando a interação entre os dois fatores mais impactantes. O gráfico mostra informações sobre o impacto absoluto dos fatores, mas não mostra em que nível o MTTA aumenta ou diminui. Portanto, a seguir, discutimos alguns gráficos de combinações de fatores.

A Figura 28 mostra o impacto das quatro combinações mais impactantes de acordo com o gráfico do fator de efeito. Os gráficos foram organizados da interação mais relevante para a menos relevante. Linhas paralelas significam que não há interação. Quanto menos paralelas as linhas, maior será a força da interação. A Figura 28 (a) mostra a interação entre o Fatores TSW e EXVNF1. TSW e EXVNF1 são as interações mais fortes entre os gráficos; quando TSW está em 0,1 ou EXVNF1 está em 2,7622, o MTTA está baixo em torno de 600. Quando TSW é 0,7 ou EXVNF1 é 1,3811, o MTTA aumenta consideravelmente para cerca de 1000. Portanto, uma alta probabilidade de transmissão bem-sucedida em conjunto com um baixo tempo de execução impacta significativamente o MTTA.

A Figura 28 (b) mostra a interação entre o Fatores ESW e EXVNF2. Os resultados mostram que quando a probabilidade de execução bem-sucedida é de 10%, o sistema não varia muito o MTTA, mesmo quando o tempo de execução do VNF 2 varia. Porém, quando a probabilidade de execução bem-sucedida é de 90%, o MTTA varia consideravelmente

Tabela 8 – Combinações de fatores.

EXVNF1	**EXVNF2**	**EXVNF3**	**TSW**	**ESW**	**MTTA**
1.38 ms	1.02 ms	1 ms	0.1%	0.1%	141.69 ms
1.38 ms	1.02 ms	1 ms	0.1%	0.9%	135.82 ms
1.38 ms	1.02 ms	1 ms	0.7%	0.1%	967.63 ms
1.38 ms	1.02 ms	1 ms	0.7%	0.9%	977.31 ms
1.38 ms	1.02 ms	2.01 ms	0.1%	0.1%	135.46 ms
1.38 ms	1.02 ms	2.01 ms	0.1%	0.9%	138.99 ms
1.38 ms	1.02 ms	2.01 ms	0.7%	0.1%	961.97 ms
1.38 ms	1.02 ms	2.01 ms	0.7%	0.9%	968.05 ms
1.38 ms	2.04 ms	1 ms	0.1%	0.1%	138.82 ms
1.38 ms	2.04 ms	1 ms	0.1%	0.9%	138.6 ms
1.38 ms	2.04 ms	1 ms	0.7%	0.1%	969.78 ms
1.38 ms	2.04 ms	1 ms	0.7%	0.9%	992.48 ms
1.38 ms	2.04 ms	2.01 ms	0.1%	0.1%	135.8 ms
1.38 ms	2.04 ms	2.01 ms	0.1%	0.9%	136.03 ms
1.38 ms	2.04 ms	2.01 ms	0.7%	0.1%	968.81 ms
1.38 ms	2.04 ms	2.01 ms	0.7%	0.9%	994.32 ms
2.76 ms	1.02 ms	1 ms	0.1%	0.1%	269.15 ms
2.76 ms	1.02 ms	1 ms	0.1%	0.9%	282.33 ms
2.76 ms	1.02 ms	1 ms	0.7%	0.1%	1930.59 ms
2.76 ms	1.02 ms	1 ms	0.7%	0.9%	1938.59 ms
2.76 ms	1.02 ms	2.01 ms	0.1%	0.1%	274.48 ms
2.76 ms	1.02 ms	2.01 ms	0.1%	0.9%	275.44 ms
2.76 ms	1.02 ms	2.01 ms	0.7%	0.1%	1927.33 ms
2.76 ms	1.02 ms	2.01 ms	0.7%	0.9%	1940.04 ms
2.76 ms	2.04 ms	1 ms	0.1%	0.1%	274.44 ms
2.76 ms	2.04 ms	1 ms	0.1%	0.9%	280.84 ms
2.76 ms	2.04 ms	1 ms	0.7%	0.1%	1932.37 ms
2.76 ms	2.04 ms	1 ms	0.7%	0.9%	1930.39 ms
2.76 ms	2.04 ms	2.01 ms	0.1%	0.1%	276.29 ms
2.76 ms	2.04 ms	2.01 ms	0.1%	0.9%	274.84 ms
2.76 ms	2.04 ms	2.01 ms	0.7%	0.1%	1929.88 ms
2.76 ms	2.04 ms	2.01 ms	0.7%	0.9%	1940.4 ms

em comparação com EXVNF2. A Figura 28 (c) mostra a interação entre o Fatores ESW e EXVNF1. A relação entre ESW e EXVNF1 não é forte, uma vez que as linhas não se cruzam. Esse comportamento demonstra que os dois fatores praticamente não interagem. Por fim, a Figura 28 (d) apresenta a interação entre o fatores EXVNF1 e EXVNF2. A interação entre EXVNF1 e EXVNF2 também é fraca e não há cruzamento de limites. No entanto, ao contrário da interação entre ESW e EXVNF1, as linhas são menos paralelas e ligeiramente inclinadas. Quando a EXVNF2 está alta em 2,046, o MTTA tende a cair mesmo com uma taxa baixa na EXVNF1.

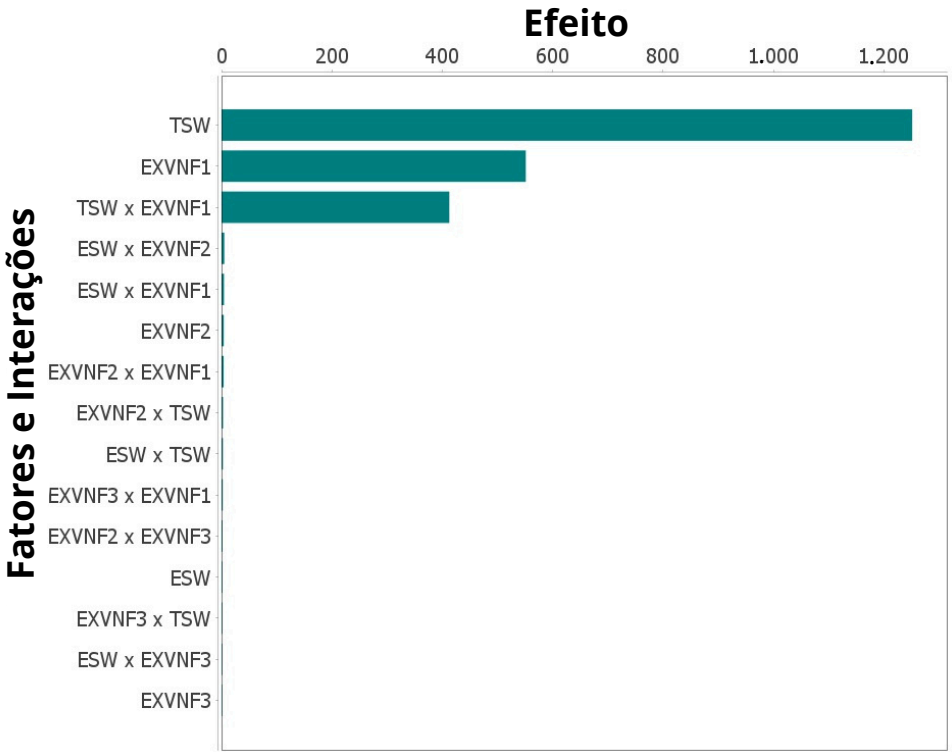


Figura 27 – Impacto dos fatores mais impactantes para MTTA.

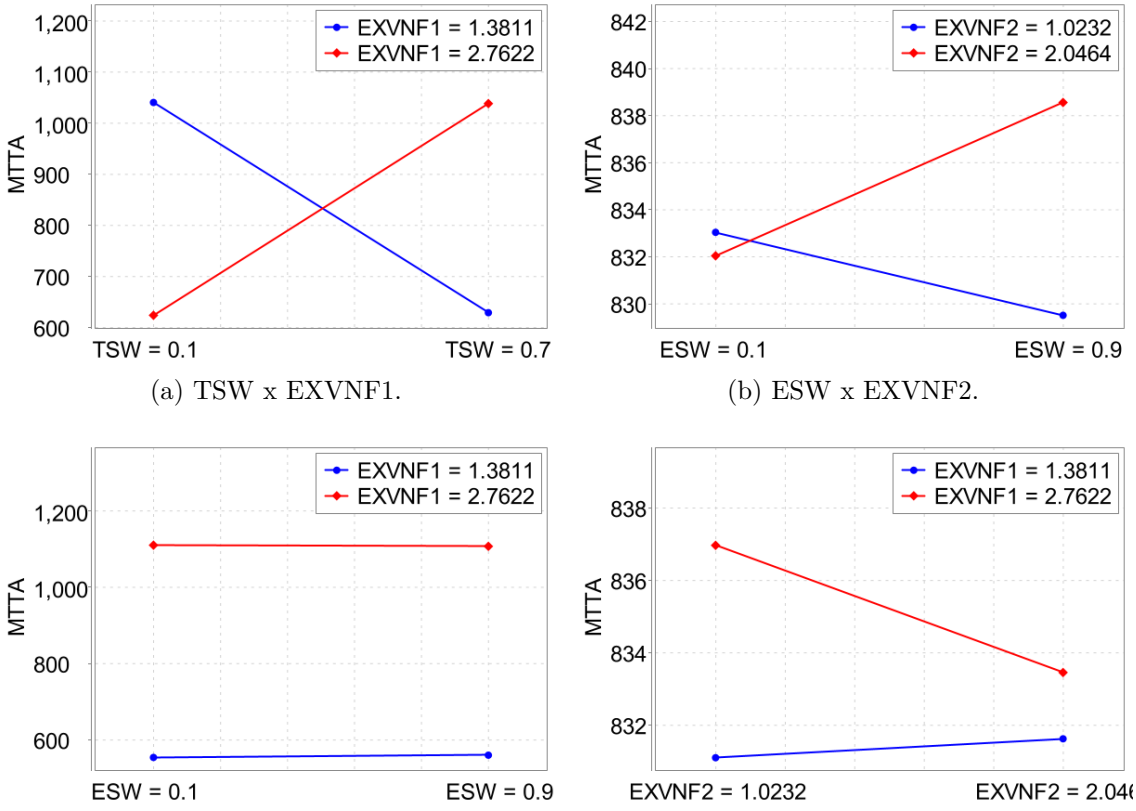
4.6 Resultados

Esta seção apresenta quatro estudos de caso, cada um com um objetivo específico. O foco está na comparação das três arquiteturas usando métricas diferentes. Esses estudos de caso são essenciais para revelar descobertas sobre o funcionamento tanto do modelo teórico quanto do sistema real e explorar novas formas de aplicação do modelo.

Estudo de Caso 1 - Número de Sucessos e Falhas

No primeiro estudo de caso, nos concentramos em comparar o número total de solicitações concluídas com sucesso e aquelas que falharam em três arquiteturas diferentes: sequencial, paralela e híbrida. O objetivo era avaliar como cada arquitetura impacta a eficiência e a confiabilidade ao longo do tempo.

Os resultados deste estudo são ilustrados na Figura 29, onde três gráficos destacam o impacto significativo das arquiteturas na taxa de sucesso e falha em um conjunto de 1.000 solicitações que são um *batch* de mensagens que já estão todas disponíveis para serem processadas. Especificamente, a Figura 29 (a) concentra-se no desempenho da arquitetura sequencial. Nesta configuração observou-se que demora cerca de 1000 segundos para processar todas as solicitações. O número de falhas superou o número de sucessos ao longo do período de execução, iniciando com aproximadamente 300 solicitações e aumentando linearmente até atingir aproximadamente 900.



(a) TSW x EXVNF1.

(b) ESW x EXVNF2.

(c) ESW x EXVNF1.

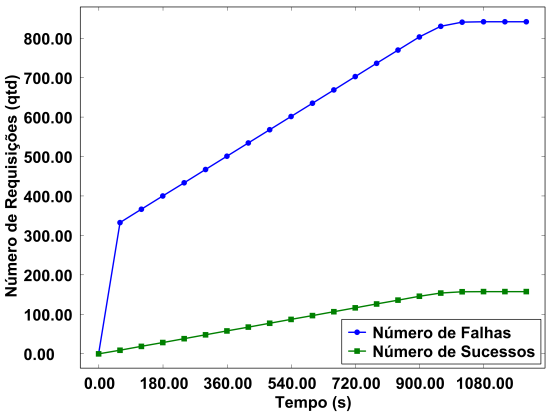
(d) EXVNF1 x EXVNF2.

Figura 28 – Interações entre fatores mais impactantes para o MTTA.

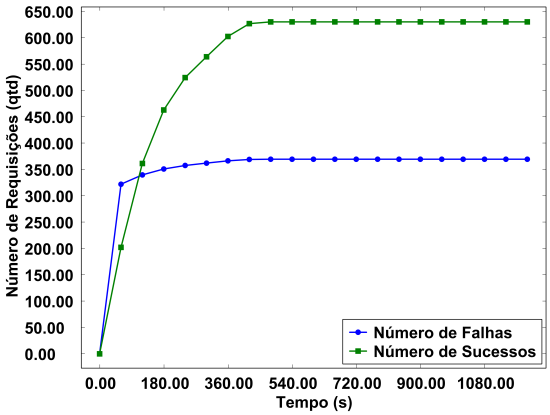
A figura 29 (b) ilustra o desempenho da arquitetura paralela. Nesse caso, o número de solicitações bem-sucedidas supera as falhas, indicando maior probabilidade relativa de sucesso. O sistema pode completar todas as solicitações em aproximadamente 411 segundos, mais de duas vezes mais rápido que a arquitetura sequencial. Além disso, embora o número de falhas também comece em torno de 300 solicitações, ele aumenta menos do que na arquitetura sequencia.

Finalmente, a Figura 29 (c) mostra o desempenho da arquitetura híbrida. Embora o número de sucessos seja inferior ao número de falhas, esta arquitetura apresenta menos falhas do que a sequencial. As 1000 solicitações são processadas em cerca de 500 segundos, tempo menor que o registrado na arquitetura sequencial, mas ainda maior que o da arquitetura paralela. O número de falhas também começa em 300 solicitações e aumenta linearmente, mas em um ritmo mais lento do que na arquitetura sequencial.

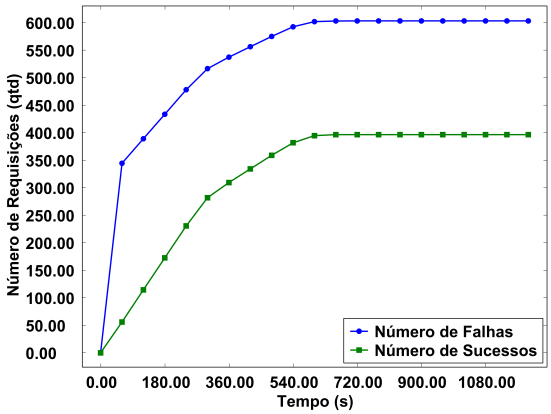
As principais conclusões deste estudo de caso são que cada arquitetura exibiu comportamentos muito diferentes em relação ao número de sucessos e falhas. A arquitetura sequencial apresentou o pior desempenho, com elevado aumento de falhas. Além disso, a arquitetura sequencial demorava muito mais para lidar com todas as solicitações. A arquitetura paralela teve melhor desempenho, com baixo pico de falhas e menor tempo para processar todas as solicitações. A arquitetura híbrida teve desempenho médio, pois, embora



(a) Arquitetura sequencial.



(b) Arquitetura paralela.



(c) Arquitetura híbrida.

Figura 29 – Solicitações concluídas em arquiteturas sequenciais, paralelas e híbridas.

aproveite parcialmente o bom desempenho da arquitetura paralela, também aproveita parcialmente o fraco desempenho da arquitetura sequencial. Embora a arquitetura paralela apresente os melhores resultados, vale ressaltar que nem todas as VNFs podem ser paralelizadas. Portanto, a arquitetura paralela nem sempre é possível, dependendo das VNFs utilizadas; neste caso, a arquitetura híbrida pode ser mais interessante (se possível). A arquitetura sequencial é mais adequada quando nenhuma das VNFs pode ser paralelizada.

Estudo de Caso 2 - MTTA

Neste segundo estudo de caso, procuramos observar o comportamento da métrica MTTA nas três arquiteturas. O MTTA representa o tempo que uma requisição leva para ser absorvida pelo sistema, ou seja, o tempo médio para completar sua execução (GERMAN, 2000; MARSAN et al., 1998). O MTTA determina quando um *token* se move do ponto inicial (`Pre`) para o ponto final (`Pab`) a partir de um conjunto de estimativas de probabilidade.

A Figura 30 apresenta o MTTA em arquiteturas sequenciais, paralelas e híbridas.

A análise foi realizada com 250 requisições até 2.000, com intervalos de 250 requisições em cada uma das arquiteturas. O tempo de absorção tende a aumentar à medida que aumenta o número de solicitações. O resultado da análise do MTTA foi semelhante ao resultado encontrado no estudo de caso 1.

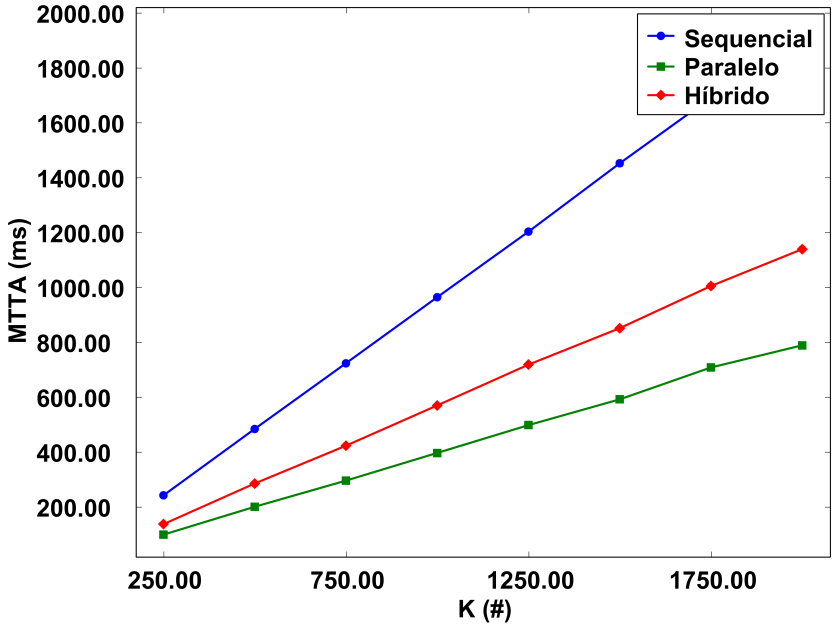


Figura 30 – MTTA de arquiteturas sequenciais, paralelas e híbridas.

A arquitetura sequencial apresentou o maior crescimento gradual no MTTA. Foram necessários aproximadamente 240 segundos para absorver 250 solicitações e, no pico, foram necessários aproximadamente 1.900 segundos para absorver 2.000 solicitações. Na arquitetura sequencial, a solicitação precisa passar por todas as 4 VNFs, portanto o tempo de absorção tende a ser maior. A arquitetura híbrida teve o MTTA médio entre as três arquiteturas: demorou aproximadamente 120 segundos para absorver 250 solicitações e, em seu pico, demorou aproximadamente 1.080 segundos para absorver 2.000 solicitações. A arquitetura híbrida levou cerca de metade do tempo da sequencial para lidar com 250 solicitações e pouco menos da metade do tempo para lidar com 2.000 solicitações. Por fim, a arquitetura paralela demonstrou o menor MTTA entre as arquiteturas. Com 250 solicitações, o desempenho foi semelhante ao da arquitetura híbrida, que levou cerca de 100 segundos para processá-las. Porém, à medida que o número de solicitações aumenta, a arquitetura paralela se afasta da híbrida com desempenho superior, demorando cerca de 700 segundos para absorver 2.000 solicitações. Esse tempo é aproximadamente 380 segundos menor do que o da arquitetura híbrida e aproximadamente 1200 segundos menor do que o da arquitetura sequencial.

Como principais observações para o MTTA, podemos destacar que o desempenho das arquiteturas torna-se ainda mais perceptível quando as comparamos em uma medida de tempo. Uma das principais características buscadas para os sistemas é a redução do atraso

no processamento das solicitações. A forma como as VNFs são organizadas pode impactar significativamente o desempenho e, consequentemente, reduzir os atrasos na resposta às solicitações. Explorar novas formas de arquitetura pode ser a chave para resolver gargalos e otimizar o desempenho.

Estudo de Caso 3 - CDF

No terceiro estudo de caso, procuramos observar o comportamento do CDF do MTTA nas três arquiteturas. O CDF permite calcular a probabilidade de conclusão da execução das solicitações antes de um tempo específico $[P(T < t)]$. O CDF também permite calcular a probabilidade de término da execução em um intervalo de tempo específico $[P(t1 < T < t2) = P(T < t2) - P(T <= t1)]$ (SILVA et al., 2017). A Figura 31 mostra o CDF para arquiteturas sequenciais, paralelas e híbridas. O CDF permite calcular a probabilidade de as solicitações serem absorvidas em uma janela de tempo específica, referindo-se ao MTTA do estudo de caso 2. A análise foi realizada em modo transitório e foi considerada a absorção das três arquiteturas. A análise mostra que a probabilidade de solicitações de modelos paralelos serem absorvidas em 300 segundos é de 100%. A análise da arquitetura híbrida mostra que a probabilidade de 100% das requisições serem absorvidas é de 500 segundos. Finalmente, a probabilidade de 100% das solicitações de arquitetura sequencial serem absorvidas é de 900 segundos.

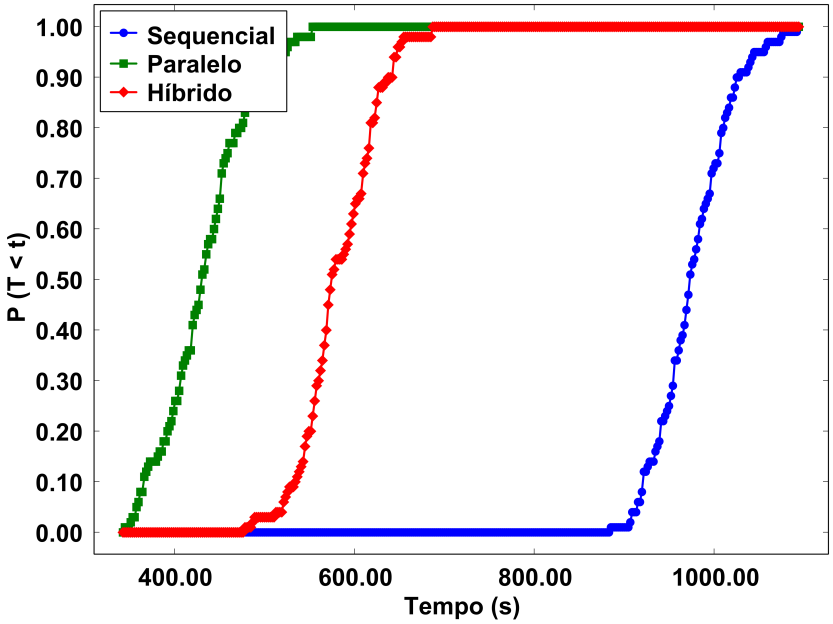


Figura 31 – CDF de arquiteturas sequenciais, paralelas e híbridas.

Os dois estudos anteriores enfatizam que a arquitetura paralela apresenta o melhor desempenho, significativamente diferente das demais arquiteturas. O estudo do CDF demonstra, mais uma vez, a importância de explorar diferentes organizações de VNFs em

arquiteturas distintas, o que pode melhorar significativamente o desempenho. Os ganhos de desempenho destacam também a redução do atraso, que está diretamente ligada ao tempo de conclusão da tarefa. O modelo não só permite a exploração de diferentes arquiteturas e formas de organização de VNFs, mas também permite adicioná-las de forma rápida e eficiente.

Estudo de Caso 4 - TSW x ENVNF1

Esta Seção apresenta o resultado da variação da intensidade do TSW com ENVNF1 para MTTA. TSW x ENVNF1 foi escolhido por ter o maior impacto no MTTA, de acordo com os resultados da análise de sensibilidade. A Figura 32 mostra o impacto da variação da probabilidade de transmissão e do tempo de execução do VNF 1 no MTTA, na configuração sequencial. Neste experimento, ESW foi definido como um intervalo de 0,1 a 0,9, enquanto EXVNF1 foi definido como um intervalo de 0,6905 a 2,7622.

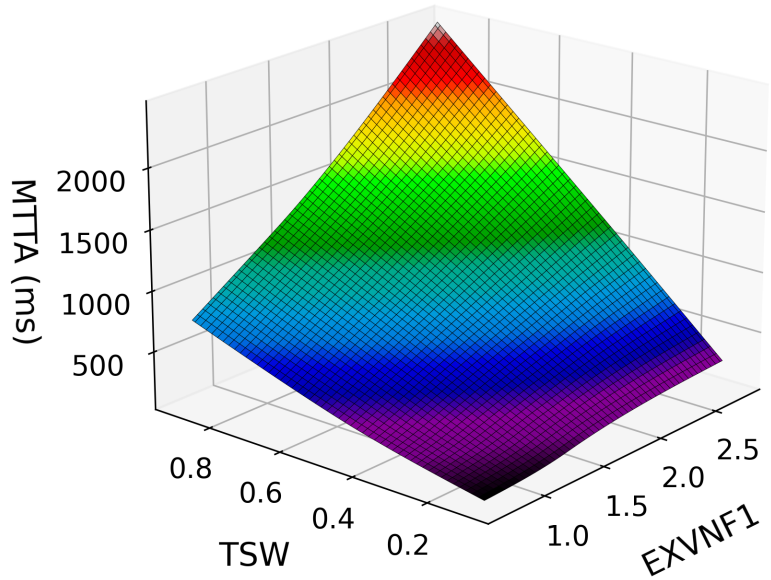


Figura 32 – Variação de TSW e EXVNF1 na configuração sequencial.

O gráfico mostra uma tendência crescente à medida que aumenta a probabilidade de transmissão bem-sucedida e mostra a mesma tendência à medida que o tempo de execução do VNF 1 diminui. Com TSW em 10%, o MTTA fica próximo de 600, mas quando essa probabilidade aumenta para 90%, o MTTA fica próximo de 1000. A variação ENVNF1 tem desempenho semelhante à variação TSW. Porém, quando as variações são realizadas em conjunto, o MTTA ultrapassa 2.000. A variação de TSW e EXVNF1 nas

configurações paralela e híbrida tem comportamento muito semelhante ao da configuração sequencial. Portanto, é importante maximizar a probabilidade de transmissão bem-sucedida de solicitações em conjunto com o tempo de execução VNF com maior tempo de execução, pois é aqui que ocorre a maioria das falhas.

4.7 Validação do Modelo SPN

Esta seção apresenta a validação do modelo SFC proposto. Para validar nosso modelo SPN proposto, comparamos o MTTA calculado pelo modelo com o MTTA coletado através de experimentos de simulação. Usamos as mesmas VNFs apresentadas na arquitetura do modelo (Seção 4.3). Primeiramente, experimentamos obter os valores dos tempos de transmissão e execução para cada uma das VNFs utilizadas. Os tempos de transmissão e execução obtidos com este experimento foram utilizados para alimentar as transições temporizadas do modelo SPN, apresentadas na seção 4.4. Simulamos MTTA em arquiteturas sequenciais, paralelas e híbridas no cenário de validação.

Os experimentos utilizaram três tipos de topologias para simulações relacionadas às organizações sequencial, paralela e híbrida; as infraestruturas estão ilustradas nas Figuras 33, 34 e 35, respectivamente., respetivamente. As Figuras mostram os componentes: (a) Origem: computador que envia a mensagem; (b) Servidores [1-4]: computadores onde as VNFs são implantadas; (c) *Switches* [1-5]: *switches* programáveis, todos acessíveis via protocolo *OpenFlow* e gerenciados pelo Controlador SDN; (d) Controlador SDN: o computador com *OpenDayLight* (BADOTRA; SINGH, 2017), esta máquina desempenha o papel de gerenciador de rede; (d) Destino: ponto final da mensagem. Todos os componentes nas figuras são contêineres *Docker* (DOCKER, 2020). Portanto, os experimentos são emulações cujo tráfego segue fielmente o processo de roteamento de uma rede física. Fonte, Destino e o Controlador SDN são imagens do Ubuntu, e os *switches* são instâncias da imagem *OpenvSwitch* (PFAFF et al., 2015).

Nas figuras existem dois fluxos: em azul, o caminho real da mensagem, e em vermelho, a lógica. Na sequência (Figura 33), deseja-se que a mensagem passe pela VNF 1, VNF 2, VNF 3 e VNF 4, respeitando esta ordem. VNF 1, 2, 3 e 4 são instanciados nos Servidores 1, 2, 3 e 4, respectivamente. Observa-se que a localização das instâncias das VNFs é a mesma em todas as organizações. Como um controlador SDN gerencia os *switches*, o encaminhamento é configurável considerando endereços IP (*Internet Protocol*) e portas de origem e destino.

A Figura 34 ilustra a transmissão da mensagem do dispositivo *Origin* para as quatro funções; a seta vermelha ilustra que a mensagem é encaminhada de forma distinta e não dependente, diferentemente da organização sequencial. Cada VNF possui um tempo indefinido para processar a mensagem, cujo tempo depende do dispositivo, serviço e

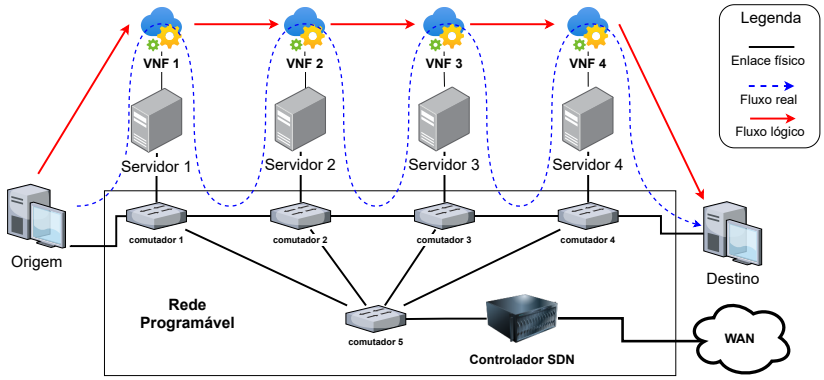


Figura 33 – Infraestrutura de organização sequencial.

sobrecarga da mensagem. Após o processamento de cada função, a mensagem resultante é encaminhada para o dispositivo Destino. O tempo total de processamento é o maior tempo gasto por uma VNF que a mensagem percorreu.

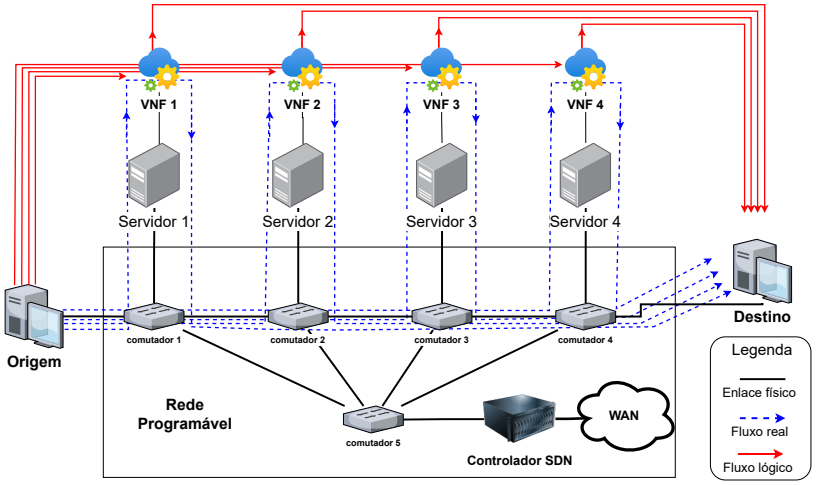


Figura 34 – Infraestrutura de organização paralela.

A figura 35 possui semelhanças entre organização sequencial e paralela; pode-se notar que existe paralelismo entre dois fluxos de serviço, onde uma sequência de duas VNFs forma cada serviço. O tempo total de processamento é o maior para um dos caminhos possíveis. Finalmente, todas as capacidades do dispositivo e do *link* são mantidas durante todos os testes; apenas a configuração do sequenciamento de mensagens é alterada para atender aos aspectos inerentes às organizações investigadas neste trabalho.

Aplicamos o teste T uni-amostral para comparar o MTTA gerado pelo modelo com o MTTA obtido nos experimentos. Todas as amostras apresentaram distribuição normal. Para verificar a significância do Teste T, observamos o valor p. A Tabela 9 e a Figura 36 mostram as médias e desvios padrão obtidos no modelo e simulação, além do valor p em cada uma das arquiteturas. O experimento foi estável porque os resultados não apresentaram valores muito distantes entre si. O valor p está bem acima de 0,05 em todos

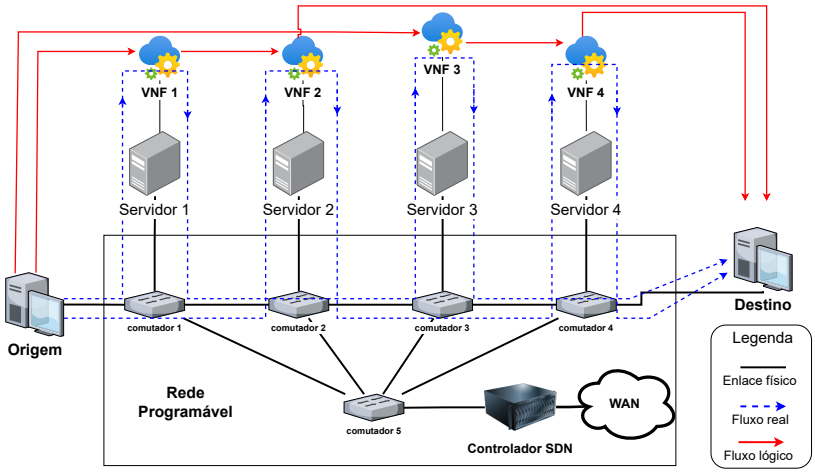


Figura 35 – Infraestrutura de organização híbrida.

os casos. Portanto, não podemos refutar a hipótese nula de igualdade em todos os casos com 95% de confiança. Portanto, os resultados gerados pelo modelo são estatisticamente equivalentes aos resultados da simulação.

Tabela 9 – Resultados do Teste T.

Arquitetura	**Modelo**		**Simulação**		**P**
	Média	**Desvio padrão**	**Média**	**Desvio padrão**	**Valor**
Sequencial	4,4213	0,2389	4,5272	0,2244	0,9580
Paralelo	1,3452	0,0989	1,4125	0,0889	0,9480
Híbrido	2,2529	0,2599	2,3871	0,2273	0,9250

Os resultados sugerem que o modelo é confiável e preciso, dada a semelhança dos resultados obtidos com a simulação. O valor do modelo também está dentro da margem de erro da simulação. O modelo reflete o ambiente real, sendo útil para o planejamento de arquiteturas SFC.

4.8 Considerações Finais

Neste capítulo foi proposto um modelo SPN para avaliar o desempenho de diferentes arquiteturas SFC aplicada a microsserviços. O modelo explora diferentes organizações de VNFs em arquiteturas sequenciais, paralelas e híbridas. O modelo recebe como entrada as probabilidades de sucesso e falha na transmissão e execução das solicitações em cada VNF. Quatro estudos de caso foram utilizados para analisar o comportamento das arquiteturas sob diferentes cargas de trabalho. Esses estudos de caso destacam os ganhos de desempenho e redução de atraso que podem ser obtidos com a mudança de arquitetura. Foi realizada uma análise de sensibilidade utilizando o método DoE para identificar os componentes mais críticos do sistema. O primeiro estudo de caso comparou a capacidade de cada arquitetura

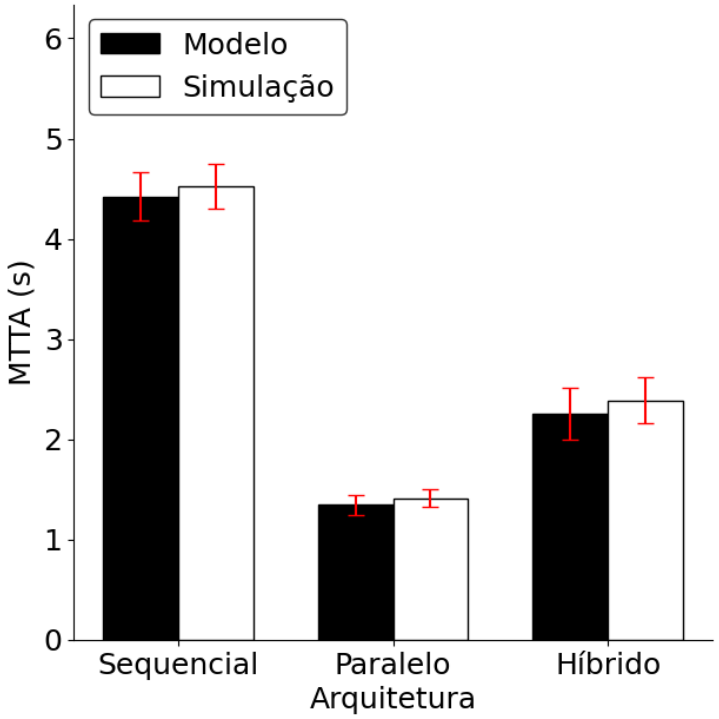


Figura 36 – Gráfico de barras com erros nos resultados do Teste T.

de processar a carga de trabalho, avaliando sucessos e fracassos. O segundo estudo de caso analisou o MTTA, enquanto o terceiro estudo de caso analisou o CDF de cada arquitetura. O quarto estudo de caso explorou as interações dos componentes críticos do sistema com mais profundidade. O trabalho foi validado por meio de simulações no GNS3, utilizando dados reais. Os resultados mostram que o modelo proposto pode auxiliar na criação e otimização de sistemas SFC reais.

5 Avaliação de Consumo Energético em Diferentes Arquiteturas SFC de Microsserviços

Este capítulo apresenta uma avaliação sobre o consumo energético de arquiteturas SFC de microsserviços. O capítulo aborda um experimento realizado no *Network Simulator 3* (NS-3) para comparação das diferentes arquiteturas.

5.1 *Network Simulator 3*

Esta Seção apresenta uma visão geral do *Network Simulator 3*. O NS-3 é uma ferramenta poderosa para simulação de redes de computadores com foco em eventos discretos. O NS-3 é amplamente utilizado para pesquisa de arquiteturas e protocolos de rede, uma vez que possui suporte a uma ampla gama de protocolos como IPv4 (*Internet Protocol 4*), IPv6, TCP (*Transmission Control Protocol*), UDP (*User Datagram Protocol*), Ethernet e Wi-Fi. Além disso, é um simulador com código aberto, o que facilita a atualização frequente para incluir novos protocolos e tecnologias emergentes. O NS-3 é uma evolução do seu antecessor, o NS-2, e foi desenvolvido para ser mais modular, extensível e realista em suas simulações. O *Network Simulator 3* utiliza uma abordagem orientada a objetos e modela componentes de rede, permitindo uma representação mais intuitiva e modular dos diversos elementos de uma rede, como nós, dispositivos, canais e protocolos. O núcleo do simulador é desenvolvido em C++, com *Python* sendo utilizado para criar ligações. Este núcleo contém uma série de bibliotecas que simulam diversos aspectos de uma rede baseada em pacotes, como tráfego, protocolos, propagação de canal, modelos de mobilidade e erros de pacotes (SILVA et al., 2021; RILEY; HENDERSON, 2010; HENDERSON et al., 2008).

No NS-3, cada evento é associado ao seu tempo de execução e a simulação avança executando os eventos na ordem temporal do tempo de simulação. Quando um evento é processado, ele pode gerar zero, um ou mais eventos. À medida que a simulação é executada, os eventos são consumidos, enquanto novos eventos podem ou não ser gerados. A simulação será automaticamente interrompida quando não houver mais eventos na fila ou quando um evento especial de parada for encontrado. O ambiente de simulação foi projetado para permitir seu uso como emulador em tempo real, integrando porções simuladas e físicas na mesma rede. Essa integração é fornecida por um agendador em tempo real e pelo suporte à simulação, o que permite a simulação em *loop*. O simulador deve ser utilizado através de sua *Application Programming Interface* (API) dentro de descrições programáticas de cenários e configurações de usuários, que podem ser desenvolvidas seguindo fluxos de trabalho e processos habituais. A API é usada para produzir saídas em termos de rastros

que são consistentes com aqueles obtidos ao observar o tráfego em redes do mundo real com ferramentas como *Wireshark*, para que possam ser analisados pelas mesmas ferramentas utilizadas para configurações experimentais ou diagnósticos (CAMPANILE et al., 2020; CARNEIRO, 2010; SILVA et al., 2021).

O NS-3 permite a simulação de ambientes de NFV, nos quais as funções de rede são virtualizadas. Dessa forma, é possível simular VNFs integradas às cadeias de serviços (YUAN; CHEN; ZHU, 2024). Além disso, o NS-3 conta com um módulo específico para gerenciamento de energia, possibilitando que os desenvolvedores definam e monitorem o consumo energético dos nós durante a simulação. Esse módulo pode ser configurado para diversos tipos de dispositivos e cenários, podendo incluir diferentes estados de operação. O NS-3 permite o acompanhamento dos níveis de energia dos nós ao longo da simulação, possibilitando a coleta de dados sobre consumo total de energia, duração da bateria e outros aspectos relevantes ao longo do tempo (WU; NABAR; POOVENDRAN, 2012; TAPPARELLO; AYATOLLAHI; HEINZELMAN, 2014).

5.2 Construção das Arquiteturas de SFC

Esta Seção detalha a construção das arquiteturas de SFC no NS-3 para o experimento de consumo energético. O simulador NS-3 é referência para análise de performance para ambientes em nuvem, como a arquitetura de microsserviços e SFC. O NS-3 possui uma comunidade forte, boa documentação e atualizações constantes. O NS-3 foi escolhido porque possui um amplo suporte a tecnologias de virtualização como NFV, SFC e microsserviços. O suporte a consumo de energia e comunicação entre os componentes virtualizados foram fundamentais para a realização do experimento. O NS-3 ainda possui flexibilidade na criação de cenários, suporte a vários protocolos e tecnologias, integração com ferramentas de análise, modelagem detalhada de consumo de energia, entre outros recursos que o destacam de outros simuladores e ferramentas. Além disso, o NS-3 tem sido amplamente utilizado em trabalhos científicos nos mais diversos contextos como IoT (KODALI; KIRTI, 2020), LoRaWAN (KHAN; PORTMANN, 2018) e NFV (EMMANUEL; MICHEL; AGBOR, 2023; MIOZZO et al., 2018).

O NS-3 não possui interface gráfica, dessa forma, as arquiteturas foram construídas por meio de códigos em C++. Um código foi escrito para cada uma das arquiteturas, de forma a representar suas características. O NS-3 oferece uma série de módulos que foram essenciais para a construção dos códigos, como os módulos de núcleo, rede, Internet, ponto-a-ponto, aplicações e energia. O módulo de núcleo (*Core Module*) oferece alguns recursos básicos como *logs* e controle de eventos. O módulo de rede (*Network Module*) oferece estruturas que conseguem gerenciar nós de rede e dispositivos. O módulo de Internet (*Internet Module*) permite a configuração de pilha TCP/IP. O módulo de ponto-a-ponto

(*Point-to-Point Module*) permite configurar *links* ponto-a-ponto entre nós. O módulo de aplicações (*Applications Module*) permite utilizar aplicações como clientes e servidores UDP. O módulo de energia (*Energy Module*) oferece recursos para trabalhar com consumo energético.

Todos os três códigos têm estruturas similares para representar a rede e montar a arquitetura. Dessa forma, é necessário configurar alguns aspectos da rede como criação de nós, configuração de energia, rede e aplicações. As configurações de criação de nós e a configuração de energia é similar nas três demais arquiteturas. Nas arquiteturas de SFC no NS-3, os nós representam contêineres que contêm VNFs que trabalham de forma sequencial, paralela ou híbrida para formar a arquitetura SFC de microsserviços. Além dos nós de VNF, precisam ser criados outros dois nós, um para representar um cliente e outro o servidor para simular o funcionamento da rede. A configuração de energia inicial é apenas para instalar a fonte de energia nos nós.

O *BasicEnergySourceHelper* instala uma fonte de energia em cada nó com energia inicial de 50 J (Joules), enquanto o *energySources* é uma lista que armazena as fontes de energia para vincular ao modelo de consumo que é definido em outra parte do código. A configuração da rede consiste em configurar os *links* ponto-a-ponto, pilha de Internet, criar dispositivos de rede entre os nós e atribuir endereços IP. As configurações de rede na arquitetura sequencial é similar nas três arquiteturas. O servidor é instalado no último nó, com o *UdpEchoServerHelper* que define um servidor UDP que transmite os pacotes recebidos. O cliente é instalado no primeiro nó, em que *UdpEchoClientHelper* define um cliente UDP envia pacotes ao servidor. Além disso, o cliente é configurado para enviar pacotes de 1024 *bytes*, com intervalo de 0,1 segundo entre envios. Os módulos do NS-3 facilitam a configuração básica. Os três códigos são semelhantes nas configurações básicas, mas possuem diferenças nas configurações de modelo de consumo de energia para definir suas características de sequencial, paralelo ou híbrido.

No modelo de consumo de energia sequencial, a classe *SequentialEnergyModel* é modelo de energia que é responsável por simular o consumo em tarefas de processamento e em estado ocioso. O *m_energySource* é a fonte de energia do nó e *m_energyPerTask* é a energia consumida por cada tarefa processada. O *m_idleEnergyPerSecond* é a energia consumida em estado ocioso, e *m_totalEnergyConsumed* é a energia total consumida. O método de simular processamento da classe do modelo de consumo de energia sequencial é o *SimulateProcessing* que processa requisições de forma sequencial, consumindo energia por tarefa e por tempo de espera (*processingDelay*). Este método possui um sistema de espera em que a VNF posterior precisa esperar a VNF anterior terminar para receber a requisição, simulando o funcionamento sequencial. Além disso, a classe *SequentialEnergyModel* possui outros métodos que definem outros comportamentos importantes como agendamento e *logs* de consumo. Dessa forma é possível simular o consumo de energia na arquitetura

sequencial.

Na classe do modelo de consumo de energia paralelo, os atributos são similares ao sequencial, já o método *ProcessRequest* calcula a energia necessária para processar todas as requisições. Se a energia disponível for suficiente, a consome e atualiza o total, senão, exibe uma mensagem de energia insuficiente. A configuração paralela divide o processamento das VNFs de forma paralela pela quantidade de VNFs. Na classe do modelo de consumo de energia híbrido *HybridEnergyModel*, os atributos e funcionamento são similares ao sequencial e paralelo. O *SimulateProcessing* calcula a energia de forma parecida com o sequencial, simulando a espera. Contudo, as VNFs são organizadas em dois grupos sequenciais que estão paralelos entre si. Dessa forma, o processamento dos dois grupos ocorre em paralelo e permite calcular o consumo energético em uma arquitetura híbrida. Os códigos apresentados foram apenas trechos-chaves do funcionamento, um repositório no *Google Drive*[1] contém os códigos completos.

5.3 Resultados

Esta seção apresenta os resultados obtidos no experimento de consumo energético em diferentes arquiteturas de SFC. O experimento foi realizado com 1000 VNFs em cada uma das arquiteturas e uma carga de trabalho 10.000 a 100.000 requisições com intervalos de 10.000. O número maior de requisições e VNFs em relação ao modelo SPN foi escolhido para destacar as diferenças de consumo entre as arquiteturas. A Figura 37 apresenta os resultados do comparativo de consumo energético em Joules entre as arquiteturas de SFC sequencial, paralela e híbrida com 1000 VNFs. Com 10.000 requisições o consumo é muito similar entre as três arquiteturas, porém à medida que a quantidade de requisições aumenta a diferença de consumo se torna evidente. A arquitetura sequencial apresentou o maior consumo com uma variação de 0,005J com 10.000 requisições até 0,455J com 100.000 requisições. A arquitetura híbrida apresentou o segundo maior consumo com uma variação de 0,003J com 10.000 requisições a 0,273J com 100.000 requisições. A arquitetura paralela apresentou o menor consumo com uma variação de 0,001J com 10.000 requisições a 0,091J com 100.000 requisições.

A arquitetura paralela tem um consumo menor porque processa as requisições de maneira paralela e consequentemente, finaliza mais rapidamente, gastando menos energia. A arquitetura sequencial tem um sistema de espera em que a VNF seguinte precisa esperar a VNF anterior terminar de processar a requisição e isso aumenta o tempo de processamento e o consumo de energia. A arquitetura híbrida também possui esse comportamento de espera, mas em menor escala, uma vez que também possui estruturas paralelas. Dessa forma a arquitetura híbrida possui um consumo menor que a sequencial, porém maior que

[1] <https://drive.google.com/drive/folders/11Y_AIaLI6XJCRhIRwSi3RIk1l1xUEvDz?usp=sharing/>

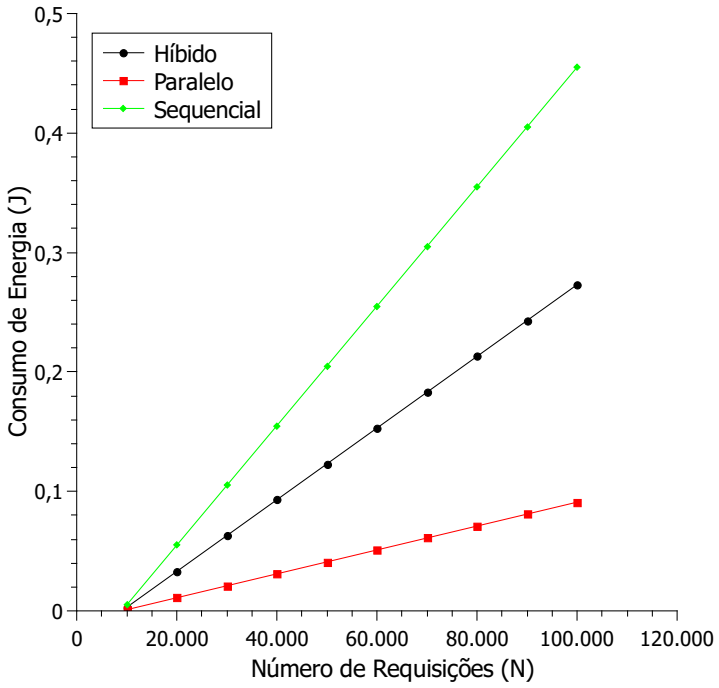


Figura 37 – Consumo energético nas arquiteturas sequenciais, paralelas e híbridas com 1000 VNFs.

a paralela. Esses comportamentos são similares aos comportamentos de desempenho geral apresentados no capítulo anterior, reforçando as diferenças entre as arquiteturas SFC.

Vale ressaltar que o NS-3 possui diferentes modelos de energia, como o *Energy Source* (Fonte de Energia) que representa a energia disponível para um dispositivo, como uma bateria. Ele acompanha o nível de carga ao longo do tempo e pode ser configurado para diferentes perfis de consumo; *Energy Harvester* (Coletor de Energia) em que alguns dispositivos podem recuperar energia de fontes externas, como painéis solares. O NS-3 permite modelar esse comportamento com um coletor de energia que adiciona carga à fonte ao longo do tempo; *Device Energy Model* (Modelo de Consumo de Energia do Dispositivo) que define como cada dispositivo consome energia com base em seu funcionamento, podendo ser dispositivos Wi-Fi, rádio, entre outros; e *Energy Manager* (Gerenciador de Energia) que em alguns casos, é possível implementar políticas de economia de energia, como transição para estados de baixo consumo ou desligamento de dispositivos não utilizados. Todos esses modelos já são prontos para uso nas simulações com o NS-3. Dessa forma, não entramos a fundo na implementação deles, utilizamos uma combinação do *Energy Source* com *Device Energy Model* para avaliar o consumo em dispositivos específicos, os VNFs. Portanto, os resultados são calculados com modelos de energia mais simples e que poderiam sofrer variações com modelos de energia mais robustos e personalizados.

6 Conclusão

Este trabalho propôs uma avaliação de desempenho e consumo energético em arquiteturas SFC de microsserviços. Foi realizada uma revisão sistemática sobre consumo energético em arquiteturas de microsserviços que trouxe uma visão geral, questões em aberto e direcionamentos futuros sobre o consumo energético em microsserviços. A maioria dos trabalhos concentrou seus estudos em métricas de desempenho. O foco em desempenho já era esperado, uma vez que ter um bom desempenho é essencial para qualquer aplicação de microsserviços. Nove trabalhos focaram em métricas de elasticidade, sendo a métrica mais explorada depois da desempenho. O foco na elasticidade ocorreu, pois o gerenciamento eficiente de recursos é uma maneira de reduzir o gasto de energia das aplicações de microsserviços. Poucos trabalhos exploraram métricas de confiabilidade, disponibilidade e sustentabilidade, o que representa oportunidades de pesquisa.

Os métodos de avaliação presentes nos estudos de microsserviços são a simulação, em segundo lugar a mensuração e em terceiro lugar a modelagem. A maioria dos trabalhos com simulação usou o simulador *CloudSim* por oferecer recursos para arquiteturas em nuvem. A mensuração também se destacou, mas apenas dois trabalhos utilizaram modelagem. Apesar da modelagem mal ter sido utilizada, é recomendado explorar tal tipo de avaliação graças ao seu poder de abstração e baixo custo, mesmo tendo uma precisão menor que experimentos práticos.

Em relação ao tipo de arquitetura, a computação em nuvem foi o destaque, as arquiteturas de borda e híbrida foram menos exploradas nos trabalhos. Além disso, a maioria dos trabalhos se concentrou no tipo de arquitetura e não entraram em detalhes sobre o modelo de implantação. A maioria dos trabalhos também não detalhou o orquestrador utilizado, contudo, os que detalharam se concentraram mais no *Docker Swarm* que no *Kubernetes*. Como todos os trabalhos têm foco em consumo de energia em microsserviços, contexto mais estudado foi o de data center; alguns trabalhos também analisaram áreas relacionadas à IoT.

O trabalho enfrentou três ameaças para a validação do estudo. (i) *Perguntas de pesquisa* que podem não ter coberto todos os trabalhos relevantes a respeito do consumo de energia em microsserviços. Todavia, consideramos uma cadeia de pesquisa de texto amplo, além da realização de muitas discussões para validar as questões. (ii) *Viés de seleção*, não temos como garantir que todos os estudos relevantes foram abordados. Alguns estudos relevantes possuem a possibilidade de não terem sido coletados na etapa de busca. Buscamos mitigar essa ameaça o máximo possível seguindo as referências nos estudos relevantes. (iii) *Extração de dados*, classificamos os estudos com base em nosso julgamento.

Contudo, é possível que alguns estudos possam ter sido classificados incorretamente. Para mitigar esse risco, seis pesquisadores realizaram o processo de classificação.

Além da revisão sistemática, foi proposto um modelo de SPN para avaliar o desempenho de diferentes arquiteturas SFC aplicadas a microsserviços. O modelo explora diferentes organizações das VNFs nas arquiteturas sequencial, paralela e híbrida. O modelo calcula as probabilidades de sucesso e falhas na transmissão e execução das requisições por cada VNF. Dessa forma, o modelo SPN é genérico o suficiente para permitir escolhas arbitrárias sobre quais VNFs utilizar, quantas VNFs a cadeia terá, qual será a caraga de trabalho e qual a arquitetura SFC será utilizada. Uma análise de sensibilidade foi realizada por meio do método DoE que apontou os pontos críticos do sistema.

Quatro estudos de caso foram utilizados para analisar o comportamento das arquiteturas sob diferentes cargas de trabalho. Esses estudos de caso ressaltam o ganho de desempenho e redução de atraso que pode ser obtido a partir da mudança de arquitetura. O primeiro estudo de caso avaliou a quantidade de sucessos e falhas, o segundo avaliou o MTTF, o terceiro avaliou o CDF e o quarto avaliou a exploração conjunta dos fatores mais relevantes para o MTTA. Os estudos destacaram as diferenças de desempenho nas três arquiteturas, em que a arquitetura sequencial teve o pior desempenho, a híbrida teve um desempenho médio e a paralela teve o melhor desempenho. O trabalho foi validado por meio da emulação de dados reais no simulador GNS3. As três arquiteturas foram modeladas no GNS3 e os resultados do MTTA e desvio padrão do modelo SPN com os resultados do GNS3 foram comparados por meio do Teste T. O valor P gerado foi superior a 0,9 em todos os casos, mostrando que o modelo SPN é válido.

Por fim, foi realizada uma análise de consumo energético das arquiteturas de SFC sequencial, paralelo e híbrido. A análise foi realizada por meio do simulador NS-3 que permitiu replicar as arquiteturas via linha de comando e calcular o consumo energético. A análise volta a ressaltar os ganhos da arquitetura paralela não só se destacou em desempenho geral, mas também apresentou menor consumo energético. Os resultados mostram que os modelos propostos são capazes de auxiliar na criação e otimização de sistemas reais de SFC. Como trabalhos futuros, pretendemos explorar outros fatores como cargas de trabalho com diferentes variações de chegada. Além disso, pretendemos analisar diferentes métricas de consumo energético como custo e eficiência energética. Novos modelos de energia mais robustos e personalizados no NS-3 podem ser explorados para uma análise mais detalhada de consumo de energia. O modelo também pode ser expandido e deve explorar novos estudos de caso e diferentes cenários.

Referências

AI, Y. et al. Integrated construction of service function chain and business system based on colored petri net. In: IEEE. *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. [S.l.], 2022. p. 56–61. 57, 58

AL-DEBAGY, O.; MARTINEK, P. A comparative review of microservices and monolithic architectures. In: IEEE. *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*. [S.l.], 2018. p. 000149–000154. 4

AL-MOALMI, A. et al. A whale optimization system for energy-efficient container placement in data centers. *Expert Systems with Applications*, Elsevier, v. 164, p. 113719, 2021. 34, 52

AL-SARAWI, S. et al. Internet of things (iot) communication protocols: Review. In: *2017 8th International Conference on Information Technology (ICIT)*. [S.l.: s.n.], 2017. p. 685–690. 26

ASIF, R.; GHANEM, K.; IRVINE, J. Containerization: For over-the-air programming of field deployed internet-of-energy based on cost effective lpwan. In: IEEE. *2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*. [S.l.], 2020. p. 65–70. 38, 41

BADOTRA, S.; SINGH, J. Open daylight as a controller for software defined networking. *International Journal of Advanced Research in Computer Science*, v. 8, n. 5, 2017. 74

BADULESCU, C.; TRIAY, J. Etsi nfv, the pillar for cloud ready ict deployments. *Journal of ICT Standardization*, River Publishers, v. 7, n. 2, p. 141–156, 2019. 11

BARI, M. F. et al. Data center network virtualization: A survey. *IEEE Communications Surveys & Tutorials*, v. 15, n. 2, p. 909–928, 2013. 26

BENEDICTIS, M. D.; LIOY, A. On the establishment of trust in the cloud-based etsi nfv framework. In: IEEE. *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. [S.l.], 2017. p. 280–285. 11

BOUAOUDA, A.; AFDEL, K.; ABOUNACER, R. Forecasting the energy consumption of cloud data centers based on container placement with ant colony optimization and bin packing. In: *2022 5th Conference on Cloud and Internet of Things (CIoT)*. [S.l.: s.n.], 2022. p. 150–157. 17, 42

BOUAOUDA, A.; AFDEL, K.; ABOUNACER, R. Forecasting the energy consumption of cloud data centers based on container placement with ant colony optimization and bin packing. In: *2022 5th Conference on Cloud and Internet of Things (CIoT)*. [S.l.: s.n.], 2022. p. 150–157. 26, 27

BRONDOLIN, R.; SARDELLI, T.; SANTAMBROGIO, M. D. Deep-mon: Dynamic and energy efficient power monitoring for container-based infrastructures. In: *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. [S.l.: s.n.], 2018. p. 676–684. 36

CAMPANILE, L. et al. Computer network simulation with ns-3: A systematic literature review. *Electronics*, MDPI, v. 9, n. 2, p. 272, 2020. 80

CAMPOLONGO, F.; TARANTOLA, S.; SALTELLI, A. Tackling quantitatively large dimensionality problems. *Computer Physics Communication*, Institute Jbr Systems, lnformatics and Safety. Joint Research Centre of the European Commission, v. 117, n. 1, p. 75–85, 1999. 14

CANOSA, R. et al. Energy consumption and quality of service optimization in containerized cloud computing. In: *2018 Ivannikov Ispras Open Conference (ISPRAS)*. [S.l.: s.n.], 2018. p. 47–55. 53

CANOSA-REYES, R. M. et al. Dynamic performance–energy tradeoff consolidation with contention-aware resource provisioning in containerized clouds. *Plos one*, Public Library of Science San Francisco, CA USA, v. 17, n. 1, p. e0261856, 2022. 5, 30, 44, 51, 53

CARNEIRO, G. Ns-3: Network simulator 3. In: *UTM lab meeting April*. [S.l.: s.n.], 2010. v. 20, n. 1, p. 4–5. 80

CHOWDHURY, S. R. et al. Re-architecting nfv ecosystem with microservices: State of the art and research challenges. *IEEE Network*, IEEE, v. 33, n. 3, p. 168–176, 2019. 4

CLOUDFLARE. *What is edge computing? - Cloudflare*. 2023. Urlhttps://www.cloudflare.com/pt-br/learning/serverless/glossary/what-is-edge-computing/. 38

COSTA, I. et al. Availability evaluation and sensitivity analysis of a mobile backend-as-a-service platform. *Quality and Reliability Engineering International*, Wiley Online Library, v. 32, n. 7, p. 2191–2205, 2016. 14

DOCKER, I. Docker. *linea].[Junio de 2017]. Disponible en: https://www. docker. com/what-docker*, 2020. 74

DRAGONI, N. et al. Microservices: yesterday, today, and tomorrow. *Present and ulterior software engineering*, Springer, p. 195–216, 2017. 3

DUAN, Y. et al. Nflcs: An service function chain path optimization strategy based on network-functional layout clustering. *IEEE Transactions on Vehicular Technology*, IEEE, 2023. 6

EMMANUEL, T.; MICHEL, D. D. E.; AGBOR, E. O. B. Virtualization of a 4g evolved packet core network using network function virtualization (nfv) technology with ns3 for enterprise and educational purpose. 2023. 80

FAHMIN, A. et al. Performance modeling and comparison of nfv integrated with sdn: Under or aside? *Journal of Network and Computer Applications*, Elsevier, v. 113, p. 119–129, 2018. 57, 58

FEITOSA, L. et al. Performance evaluation of message routing strategies in the internet of robotic things using the d/m/c/k/fcfs queuing network. *Electronics*, MDPI, v. 10, n. 21, p. 2626, 2021. 14

FERREIRA, L. C. B. C. et al. Edge computing and microservices middleware for home energy management systems. *IEEE Access*, v. 10, p. 109663–109676, 2022. 25, 27

Fontana de Nardin, I. et al. On revisiting energy and performance in microservices applications: A cloud elasticity-driven approach. *Parallel Computing*, v. 108, p. 102858, 2021. ISSN 0167-8191. 27, 31, 41, 52

FRANCESCO, P. D.; LAGO, P.; MALAVOLTA, I. Architecting with microservices: A systematic mapping study. *Journal of Systems and Software*, Elsevier, v. 150, p. 77–97, 2019. 18, 19, 20

GERMAN, R. *Performance analysis of communication systems with non-Markovian stochastic Petri nets*. [S.l.]: John Wiley & Sons, Inc., 2000. 65, 70

GHOFRANI, J.; LÜBKE, D. Challenges of microservices architecture: A survey on the state of the practice. *ZEUS*, v. 2018, p. 1–8, 2018. 18, 19, 20

GHOLIPOUR, N.; ARIANYAN, E.; BUYYA, R. A novel energy-aware resource management technique using joint vm and container consolidation approach for green computing in cloud data centers. *Simulation Modelling Practice and Theory*, Elsevier, v. 104, p. 102127, 2020. 17, 25, 37, 52

GOS, K.; ZABIEROWSKI, W. The comparison of microservice and monolithic architecture. In: IEEE. *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*. [S.l.], 2020. p. 150–153. 4

HANNOUSSE, A.; YAHIOUCHE, S. Securing microservices and microservice architectures: A systematic mapping study. *Computer Science Review*, Elsevier, v. 41, p. 100415, 2021. 18, 19, 20

HANNOUSSE, A.; YAHIOUCHE, S. Securing microservices and microservice architectures: A systematic mapping study. *Elsevier*, p. 1–16, 2021. 21

HEIDEKER, A.; KAMIENSKI, C. Network queuing assessment: A method to detect bottlenecks in service function chaining. *IEEE Transactions on Network and Service Management*, IEEE, v. 19, n. 4, p. 4650–4661, 2022. 59

HENDERSON, T. R. et al. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, v. 14, n. 14, p. 527, 2008. 79

HUANG, Y.-X.; CHOU, J. A survey of nfv network acceleration from etsi perspective. *Electronics*, MDPI, v. 11, n. 9, p. 1457, 2022. 11

HUTH, A.; CEBULA, J. The basics of cloud computing. *United States Computer*, p. 1–4, 2011. 37

JARWAR, M. A.; ALI, S.; CHONG, I. Microservices model to enhance the availability of data for buildings energy efficiency management services. *Energies*, MDPI, v. 12, n. 3, p. 360, 2019. 17, 27, 29, 32, 50

JAWADDI, S. N. A.; JOHARI, M. H.; ISMAIL, A. A review of microservices autoscaling with formal verification perspective. *Software: Practice and Experience*, Wiley Online Library, v. 52, n. 11, p. 2476–2495, 2022. 18, 19, 20

KAUR, K. et al. Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers. *IEEE wireless communications*, IEEE, v. 24, n. 3, p. 48–56, 2017. 26, 38

KAUR, K. et al. Enlob: Energy and load balancing-driven container placement strategy for data centers. In: IEEE. *2019 IEEE Globecom Workshops (GC Wkshps)*. [S.l.], 2019. p. 1–6. 37

KAUR, K.; MANGAT, V.; KUMAR, K. A comprehensive survey of service function chain provisioning approaches in sdn and nfv architecture. *Computer Science Review*, Elsevier, v. 38, p. 100298, 2020. 12

KHAIRY, M. et al. Simr: Single instruction multiple request processing for energy-efficient data center microservices. In: IEEE. *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. [S.l.], 2022. p. 441–463. 17, 26, 35

KHAN, A. A. et al. An energy and performance aware consolidation technique for containerized datacenters. *IEEE Transactions on Cloud Computing*, v. 9, n. 4, p. 1305–1322, 2021. 17, 43, 52

KHAN, A. A. et al. An energy and performance aware consolidation technique for containerized datacenters. *IEEE Transactions on Cloud Computing*, v. 9, n. 4, p. 1305–1322, 2021. 26, 34

KHAN, F. H.; PORTMANN, M. Experimental evaluation of lorawan in ns-3. In: IEEE. *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*. [S.l.], 2018. p. 1–8. 80

KODALI, R. K.; KIRTI, B. Ns-3 model of an iot network. In: IEEE. *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*. [S.l.], 2020. p. 699–702. 80

KUMAR, N. et al. Renewable energy-based multi-indexed job classification and container management scheme for sustainability of cloud data centers. *IEEE Transactions on Industrial Informatics*, v. 15, n. 5, p. 2947–2957, 2019. 17

KUMAR, N. et al. Renewable energy-based multi-indexed job classification and container management scheme for sustainability of cloud data centers. *IEEE Transactions on Industrial Informatics*, v. 15, n. 5, p. 2947–2957, 2019. 28, 32

LENNICK, D.; AZIM, A.; LISCANO, R. A microservice-based architecture for performance and energy benchmarking of docker-host linux distributions on internet-of-things devices. In: IEEE. *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*. [S.l.], 2021. v. 1, p. 705–711. 26, 37

LENNICK, D.; AZIM, A.; LISCANO, R. A microservice-based architecture for performance and energy benchmarking of docker-host linux distributions on internet-of-things devices. In: *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*. [S.l.: s.n.], 2021. v. 1, p. 705–711. 35, 39

LI, R. et al. Petri net-based model for 5g and beyond networks resilience evaluation. In: IEEE. *2022 25th Conference on Innovation in Clouds, Internet and Networks (ICIN)*. [S.l.], 2022. p. 131–135. 55, 59, 61

LI, R. et al. A petri net-based model to study the impact of traffic changes on 5g network resilience. In: RESEARCH PUBLISHING. *32nd European Safety and Reliability Conference (ESREL 2022)*. [S.l.], 2022. p. 3016–3023. 58, 59

LI, Z. et al. Kubernetes-container-cluster-based architecture for an energy management system. *IEEE Access*, IEEE, v. 9, p. 84596–84604, 2021. 17, 31, 36

LIN, C.-C. et al. Energy-efficient core allocation and deployment for container-based virtualization. In: *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. [S.l.: s.n.], 2018. p. 93–101. 17

LIN, C.-C. et al. Energy-efficient core allocation and deployment for container-based virtualization. In: *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. [S.l.: s.n.], 2018. p. 93–101. 26, 30

LIU, J. et al. On dynamic service function chain deployment and readjustment. *IEEE Transactions on Network and Service Management*, IEEE, v. 14, n. 3, p. 543–553, 2017. 12

LIU, M. et al. E3: Energy-efficient microservices on smartnic-accelerated servers. In: *USENIX annual technical conference*. [S.l.: s.n.], 2019. p. 363–378. 4, 5

LIU, Z. et al. Colored petri net reusing for service function chaining validation. In: IEEE. *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. [S.l.], 2022. p. 1531–1535. 57, 58

LUO, J. et al. Container-based fog computing architecture and energy-balancing scheduling algorithm for energy iot. *Future Generation Computer Systems*, v. 97, p. 50–60, 2019. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X1831358X>. 34

LUO, J. et al. Container-based fog computing architecture and energy-balancing scheduling algorithm for energy iot. *Future Generation Computer Systems*, v. 97, p. 50–60, 2019. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X1831358X>. 40

LYU, Z. et al. Microservice-based architecture for an energy management system. *IEEE Systems Journal*, v. 14, n. 4, p. 5061–5072, 2020. 17

LYU, Z. et al. Microservice-based architecture for an energy management system. *IEEE Systems Journal*, v. 14, n. 4, p. 5061–5072, 2020. 28, 31, 35, 36, 39

MACIEL, P. et al. Mercury: Performance and dependability evaluation of systems with exponential, expolynomial, and general distributions. In: IEEE. *2017 IEEE 22nd Pacific Rim international symposium on dependable computing (PRDC)*. [S.l.], 2017. p. 50–57. 59, 61, 65

MARSAN, A. *Modelling with generalized stochastic Petri nets*. [S.l.]: Wiley, 1995. (Wiley series in parallel computing). ISBN 9780471930594. 6

MARSAN, M. A. Stochastic petri nets: an elementary introduction. In: SPRINGER. *Advances in Petri Nets 1989 9*. [S.l.], 1990. p. 1–29. 13

MARSAN, M. A. et al. Modelling with generalized stochastic petri nets. *ACM SIGMETRICS performance evaluation review*, ACM New York, NY, USA, v. 26, n. 2, p. 2, 1998. 13, 65, 70

MAURO, M. D. et al. Availability evaluation of a virtualized ip multimedia subsystem for 5g network architectures. *Safety and Reliability-Theory and Applications*, Taylor & Francis Group, p. 2203–2210, 2017. 57, 58

MAURO, M. D. et al. Ip multimedia subsystem in an nfv environment: Availability evaluation and sensitivity analysis. In: IEEE. *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. [S.l.], 2018. p. 1–6. 57, 58

MAURO, M. D. et al. Service function chaining deployed in an nfv environment: An availability modeling. In: IEEE. *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*. [S.l.], 2017. p. 42–47. 57, 58

MENDES, S.; SIMÃO, J.; VEIGA, L. Oversubscribing micro-clouds with energy-aware containers scheduling. In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. [S.l.: s.n.], 2019. p. 130–137. 53

MIOZZO, M. et al. Sdr and nfv extensions in the ns-3 lte module for 5g rapid prototyping. In: IEEE. *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. [S.l.], 2018. p. 1–6. 80

MOCHALOV, V. et al. Methods for functional parallel processes implementation for network functions virtualization based on coloured petri nets. In: AIP PUBLISHING. *AIP Conference Proceedings*. [S.l.], 2021. v. 2318. 56, 58, 59

MOLLOY. Performance analysis using stochastic petri nets. *IEEE Transactions on computers*, IEEE, v. 100, n. 9, p. 913–917, 1982. 13

NARDIN, I. F. D. et al. On revisiting energy and performance in microservices applications: A cloud elasticity-driven approach. *Parallel Computing*, Elsevier, v. 108, p. 102858, 2021. 4, 5

NELSON, R. *Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling*. [S.l.]: Springer Science & Business Media, 2013. 64

PATEL, D.; PATRA, M. K.; SAHOO, B. Energy efficient genetic algorithm for container consolidation in cloud system. In: *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*. [S.l.: s.n.], 2020. p. 1066–1071. 26

PFAFF, B. et al. The design and implementation of open {vSwitch}. In: *12th USENIX symposium on networked systems design and implementation (NSDI 15)*. [S.l.: s.n.], 2015. p. 117–130. 74

PFITSCHER, R. J. et al. Guiltiness: A practical approach for quantifying virtual network functions performance. *Computer Networks*, Elsevier, v. 161, p. 14–31, 2019. 59

PIRAGHAJ, S. F. et al. A framework and algorithm for energy efficient container consolidation in cloud data centers. In: *2015 IEEE International Conference on Data Science and Data Intensive Systems*. [S.l.: s.n.], 2015. p. 368–375. 30, 41

RAHMAN, A. et al. Impacts of blockchain in software-defined internet of things ecosystem with network function virtualization for smart applications: Present perspectives and future directions. *International Journal of Communication Systems*, Wiley Online Library, p. e5429, 2023. 55

RANJAN, R. et al. Energy-efficient workflow scheduling using container-based virtualization in software-defined data centers. *IEEE Transactions on Industrial Informatics*, v. 16, n. 12, p. 7646–7657, 2020. 30, 37, 53

RAZA, S. M. et al. Empirical performance and energy consumption evaluation of container solutions on resource constrained iot gateways. *Sensors*, MDPI, v. 21, n. 4, p. 1378, 2021. 25, 38

RILEY, G. F.; HENDERSON, T. R. The ns-3 network simulator. In: *Modeling and tools for network simulation*. [S.l.]: Springer, 2010. p. 15–34. 79

RUI, L. et al. Petri net-based reliability assessment and migration optimization strategy of sfc. *IEEE Transactions on Network and Service Management*, IEEE, v. 18, n. 1, p. 167–181, 2020. 6, 55, 57, 58, 61, 65

RYGIELSKI, P.; SELIUCHENKO, M.; KOUNEV, S. Modeling and prediction of software-defined networks performance using queueing petri nets. In: *Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques*. [S.l.: s.n.], 2016. p. 66–75. 6, 55, 56, 58

SAENGKAENPETCH, K.; ASWAKUL, C. Cloud-based smart energy framework for accelerated data analytics with parallel computing of orchestrated containers: Study case of cu-bems. In: *2021 3rd International Conference on Advanced Information Science and System (AISS 2021)*. [S.l.: s.n.], 2021. p. 1–6. 17

SAENGKAENPETCH, K.; ASWAKUL, C. Cloud-based smart energy framework for accelerated data analytics with parallel computing of orchestrated containers: Study case of cu-bems. In: *Proceedings of the 3rd International Conference on Advanced Information Science and System*. New York, NY, USA: Association for Computing Machinery, 2022. ISBN 9781450385862. Disponível em: <https://doi.org/10.1145/3503047.3503088>. 39

SANTOS, G. L. et al. Service function chain placement in distributed scenarios: a systematic review. *Journal of Network and Systems Management*, Springer, v. 30, n. 1, p. 4, 2022. 12

SANTOS, L. et al. Data processing on edge and cloud: a performability evaluation and sensitivity analysis. *Journal of Network and Systems Management*, Springer, v. 29, n. 3, p. 27, 2021. 14

SCHNEIDER, S. et al. Specifying and analyzing virtual network services using queuing petri nets. In: IEEE. *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. [S.l.], 2019. p. 116–124. 6, 55, 57, 58

SHAH, A. et al. Survey on virtualization technique and its role in cloud computing environment. *Advances in Robotic Technology*, 2024. 3

SILVA, F. A. et al. Mobile cloud performance evaluation using stochastic models. *IEEE Transactions on Mobile Computing*, IEEE, v. 17, n. 5, p. 1134–1147, 2017. 72

SILVA, J. C. da et al. A survey of lorawan simulation tools in ns-3. *Journal of Communication and Information Systems*, v. 36, n. 1, p. 17–30, 2021. 79, 80

SINGH, A.; AUJLA, G. S.; BALI, R. S. Container-based load balancing for energy efficiency in software-defined edge computing environment. *Sustainable Computing: Informatics and Systems*, Elsevier, v. 30, p. 100463, 2021. 34, 51, 52

SOLDANI, J.; TAMBURRI, D. A.; HEUVEL, W.-J. V. D. The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software*, Elsevier, v. 146, p. 215–232, 2018. 18, 19, 20

SOUZA, W. Silva-de et al. Containergy—a container-based energy and performance profiling tool for next generation workloads. *Energies*, MDPI, v. 13, n. 9, p. 2162, 2020. 25, 26

STACKPATH. *What is Fog Computing? - StackPath*. 2023. Urlhttps://www.stackpath.com/edge-academy/what-is-fog-computing/. 38

STATISTA. *Virtualization software market size forecast comparison worldwide from 2019 to 2027*. 2023. Urlhttps://www.statista.com/statistics/1139362/virtualization-software-market-forecast/. 4

STATISTA. *Public Cloud: market data and analysis*. 2024. Urlhttps://www.statista.com/study/85676/public-cloud-report/. 3

TAPPARELLO, C.; AYATOLLAHI, H.; HEINZELMAN, W. Extending the energy framework for network simulator 3 (ns-3). *arXiv preprint arXiv:1406.6265*, 2014. 80

TAŞKIN, D.; TAŞKIN, C. et al. Container-based virtualization for bluetooth low energy sensor devices in internet of things applications. *Tehnički vjesnik*, Strojarski fakultet u Slavonskom Brodu; Fakultet elektrotehnike, računarstva . . . , v. 28, n. 1, p. 13–19, 2021. 36

THÖNES, J. Microservices. *IEEE software*, IEEE, v. 32, n. 1, p. 116–116, 2015. 3

THONG, W. J.; AMEEDEEN, M. A survey of petri net tools. In: SPRINGER. *Advanced Computer and Communication Engineering Technology: Proceedings of the 1st International Conference on Communication and Computer Engineering*. [S.l.], 2015. p. 537–551. 14

TRIVEDI, K. S. *Probability & statistics with reliability, queuing and computer science applications*. [S.l.]: John Wiley & Sons, 2008. 13

VALERA, H. H. A. et al. Draceo: A smart simulator to deploy energy saving methods in microservices based networks. In: *2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. [S.l.: s.n.], 2020. p. 94–99. 28, 31, 34

VALERA, H. H. A. et al. An energy saving approach: understanding microservices as multidimensional entities in p2p networks. In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. [S.l.: s.n.], 2021. p. 69–78. 5, 17, 51

VALERA, H. H. A. et al. An energy saving approach: understanding microservices as multidimensional entities in p2p networks. In: *Proceedings of the 36th Annual ACM Symposium on Applied Computing*. [S.l.: s.n.], 2021. p. 69–78. 31, 50, 52

VARIA, J. Cloud architectures. *White Paper of Amazon, jineshvaria. s3. amazonaws. com/public/cloudarchitectures-varia. pdf*, v. 16, 2008. 36

VEERARAGHAVAN, M. et al. Network function virtualization: A survey. *IEICE Transactions on Communications*, The Institute of Electronics, Information and Communication Engineers, v. 100, n. 11, p. 1978–1991, 2017. 11

VELIEVA, T. R.; KOROLKOVA, A. V.; KULYABOV, D. S. Designing installations for verification of the model of active queue management discipline red in the gns3. In: IEEE. *2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. [S.l.], 2014. p. 570–577. 56, 59

VIGGIATO, M. et al. Microservices in practice: A survey study. *arXiv preprint arXiv:1808.04836*, 2018. 18, 19, 20

VURAL, H.; KOYUNCU, M.; GUNEY, S. A systematic literature review on microservices. In: SPRINGER. *Computational Science and Its Applications–ICCSA 2017: 17th International Conference, Trieste, Italy, July 3-6, 2017, Proceedings, Part VI 17*. [S.l.], 2017. p. 203–217. 17, 18, 19, 20

WASEEM, M.; LIANG, P.; SHAHIN, M. A systematic mapping study on microservices architecture in devops. *Journal of Systems and Software*, Elsevier, v. 170, p. 110798, 2020. 18, 19, 20

WU, H.; NABAR, S.; POOVENDRAN, R. An energy framework for the network simulator 3 (ns-3). In: *4th International ICST Conference on Simulation Tools and Techniques*. [S.l.: s.n.], 2012. 80

XU, M.; TOOSI, A. N.; BUYYA, R. ibrownout: An integrated approach for managing energy and brownout in container-based clouds. *IEEE Transactions on Sustainable Computing*, v. 4, n. 1, p. 53–66, 2019. 35, 45

YANAMALA, A. K. Y. Emerging challenges in cloud computing security: A comprehensive review. *International Journal of Advanced Engineering Technologies and Innovations*, v. 1, n. 4, p. 448–479, 2024. 3

YI, B. et al. A comprehensive survey of network function virtualization. *Computer Networks*, Elsevier, v. 133, p. 212–262, 2018. 11

YU, D. et al. A survey on security issues in services communication of microservices-enabled fog applications. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 31, n. 22, p. e4436, 2019. 18, 19, 20

YUAN, J.; CHEN, M.; ZHU, K. A nfv technology with expanded wireless communication emulation function. In: IEEE. *2024 IEEE Wireless Communications and Networking Conference (WCNC)*. [S.l.], 2024. p. 1–6. 80

ZHANG, R. et al. A genetic algorithm-based energy-efficient container placement strategy in caas. *IEEE Access*, IEEE, v. 7, p. 121360–121373, 2019. 40, 46

ZHANG, Y. et al. Parabox: Exploiting parallelism for virtual network functions in service chaining. In: *Proceedings of the Symposium on SDN Research*. [S.l.: s.n.], 2017. p. 143–149. 5