UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA - PPGCA

ALINE DA SILVA MOREIRA

MODELO DE PRECIFICAÇÃO DE CONTÊINERES BASEADO NO CONSUMO ENERGÉTICO

JOINVILLE

2020

ALINE DA SILVA MOREIRA

MODELO DE PRECIFICAÇÃO DE CONTÊINERES BASEADO NO CONSUMO ENERGÉTICO

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Maurício Aronne Pillon

JOINVILLE

2020

Ficha catalográfica elaborada pelo programa de geração automática da

Biblioteca Setorial do CCT/UDESC,

com os dados fornecidos pelo(a) autor(a)

da Silva Moreira, Aline
MODELO DE PRECIFICAÇÃO DE CONTÊINERES BASEADO NO CONSUMO ENERGÉTICO / Aline da Silva Moreira. -- 2020.
68 p.

Orientador: Mauricio Aronne Pillon
Dissertação (mestrado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2020.

1. Modelo de Custo. 2. Contêiner. 3. Nuvem Computacional. 4. Consumo Energético. I. Aronne Pillon, Mauricio . II. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada. III. Título.

ALINE DA SILVA MOREIRA

MODELO DE PRECIFICAÇÃO DE CONTÊINERES BASEADO NO CONSUMO ENERGÉTICO

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada, da Universidade do Estado de Santa Catarina, como requisito parcial para obtenção do título de Mestre em Computação Aplicada, área de concentração em Sistemas de Computação.

BANCA EXAMINADORA

Prof. Dr. Maurício Aronne Pillon
PPGCA - UDESC

Membros:

Prof. Dr. Charles Christian Miers
PPGCA - UDESC

Prof. Dr. Mario Antonio Ribeiro Dantas
PPGCC - UFJF

Joinville, 30 de Novembro de 2020

RESUMO

O acesso sob demanda a recursos virtuais fornecidos sobre infraestruturas de computação em nuvem, revolucionou o desenvolvimento e a execução de aplicativos distribuídos. Os provedores de serviços em nuvem, oferecem recursos e cobram dos locatários de acordo com o conjunto de recursos alocados, multiplicado pela fração do tempo de reserva designado. Esse modelo de precificação é conhecido como *pay-as-you-go* na literatura especializada. Entre os componentes variáveis que compõem o Custo Total de Propriedade (CTP) para modelo de precificação pré-pago, o consumo de energia de um *Data Center* (DC) é um termo dominante. A conteinerização em DC simplifica o gerenciamento administrativo e apoia a execução de aplicativos executados em contêineres. No entanto, a aplicação de modelos tradicionais *pay-as-you-go* orientados por valores de reservas (ou alocações) são imprecisos e levam a valores abaixo do ideal. Este trabalho expõe que, para construir um modelo de precificação preciso, e baseado em energia, deve-se considerar o uso dos recursos computacionais (*e.g.,* CPU, memória, armazenamento, rede) pelos contêineres no modelo pré-pago. Nesse sentido, é proposto um modelo de custo de contêiner orientado pelo consumo de energia, e de acordo com a utilização efetiva de recursos virtuais. O modelo de custo proposto neste trabalho denominado PECN, é amplamente comparado ao serviço Fargate da *Amazon Web Services* (AWS), destacando os benefícios em usar um modelo de precificação baseado em energia. Os valores estimados do modelo de custo Preço Energético de Contêiner em Nuvem (PECN), variam semanalmente entre US $ 2,31 e US $ 10,59, enquanto os valores da AWS Fargate US $ 2,71 e US $ 29,94, representando uma redução de até 35,39%.

Palavras-chaves: Modelo de Custo; Contêiner; Nuvem Computacional; Consumo Energético.

ABSTRACT

The on-demand access to virtual resources provisioned atop cloud computing infrastructures revolutionized the development and execution of distributed applications. Cloud service providers offer resources and charge the tenants according to the set of allocated resources multiplied by the fraction of the designated reservation time. This pricing model is known as pay-as-you-go in the specialized literature. Among the variable components composing the Total Cost of Ownership (TCO) for pay-as-you-go pricing model, the DC energy consumption is a dominant term. Containerized DCs simplify the administrative management and support the execution of applications packaged in containers. However, the application of traditional pay-as-you-go models guided by reservations (or allocations) values are inaccurate and lead to suboptimal values. This research claims that for achieving a precise and fair energy-based pricing model, the pay-as-you-go must consider the utilization values (*e.g.*, CPU, memory, network) of containers. In this sense, a container cost model is proposed, guided by energy consumption, according to the effective virtual resource utilization. The cost model proposed in this research termed PECN, is extensively compared with AWS Fargate, highlighting the benefits of using an energy-based pricing model. The cost model PECN, estimated values vary weekly between US$ 2.31 and US$ 10.59, while AWS Fargate US$ 2.71 and US$ 29.94, representing a reduction of up to 35.39%.

Keywords: Pricing Model; Containers; Cloud Computing; Energy Consumption.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

API *Application Programming Interface*

AWS *Amazon Web Services*

CE *Container Enginer*

CTP Custo Total de Propriedade

DC *Data Center*

EPAVE *Energy-proportional Profiling and Accounting in Virtualized Environments*

GCP *Google Cloud Plataform*

IaaS *Infrastructure as a Service*

MV Máquina Virtual

PECN Preço Energético de Contêiner em Nuvem

PSVE *Proportional-Shared Virtual Energy*

SO Sistema Operacional

TI Tecnologia da Informação

TCO Total Cost of Ownership

UFS *Union File System*

UTS *Unix Timesharing System*

SUMÁRIO

1 INTRODUÇÃO

A virtualização é uma tecnologia em constante evolução, é foco em pesquisas científicas e de aplicações na indústria de *software*. Escalabilidade e flexibilidade são atrativos para o desenvolvimento de aplicações conteinerizadas, visto que a virtualização ocorre a nível de Sistema Operacional (SO). Aplicações conteinerizadas compartilham o núcleo do SO de uma mesma máquina hospedeira, seja esta uma Máquina Virtual (MV), ou não (SHARMA et al., 2016; SOUPPAYA; MORELLO; SCARFONE, 2017). Ademais, é notável que tecnologias de virtualização vem sendo aplicadas em larga escala em ambientes de nuvens computacionais, nas quais o modelo de negócio é baseado na alocação sob demanda de recursos computacionais, e o pagamento, ocorre de acordo com a utilização dos recursos (*pay-as-you-go*).

A estrutura e gerência simplificada de contêineres possibilita a exploração de outra característica de sistemas distribuídos, adaptada aos interesses de clientes de nuvens computacionais, a elasticidade. Com isso, a oferta deste modo de virtualização simplificada, escalável, flexível e elástica passou a fazer parte do catálogo de serviços de provedores de nuvens (*e.g.,* Amazon AWS, Microsoft Azure, Google Cloud, Rackspace, Heroku, dentre outros) (BEGUM; KHAN, 2011).

Devido ao aumento do número de empresas as quais adotam o uso da nuvem computacional, as despesas com nuvem pública tornaram-se uma parcela expressiva do orçamento de Tecnologia da Informação (TI) das organizações (COLUMBUS, 2017). A computação em nuvem possui eficiência de recursos computacionais, facilita a oferta por parte dos provedores de nuvens públicas e a procura de empresas que buscam agilidade e escalabilidade para hospedar seus negócios (HANINI; KAFHALI; SALAH, 2019). O relatório anual *Flexera 2020 State of the Cloud* destaca que as empresas continuam a adotar estratégias da computação em nuvem, e em média utilizam mais de duas nuvens públicas e duas nuvens privadas.

Entretanto, podem ser considerados fatores agravantes na subutilização de recursos, as condições advindas da premissa de recursos sob demanda, e também da complexidade do gerenciamento dos serviços ofertados pelas nuvens computacionais (BITTENCOURT et al., 2018; WU; BUYYA; RAMAMOHANARAO, 2019). Os quais salientam a necessidade do uso de ferramentas e abordagens que otimizem os gastos gerados pelo serviço de modo transparente.

A precificação dos provedores *Infrastructure as a Service* (IaaS) podem diferir em diversos modelos (*e.g.,* baseados em assinatura mensal, licenças, e pagamento conforme o uso conhecido como *pay-as-you-go*) (BINDU; RAMANI; BINDU, 2018). A

Amazon, por exemplo, oferta o modelo *pay-as-you-go* apresentando uma diversidade de opções, entre eles, tem-se cobrança com preço predeterminado por cada hora de recursos utilizados, por tarefa ou por número de chamadas de uma tarefa (WU; BUYYA; RAMAMOHANARAO, 2019). Esse modelo também é usado por outros provedores de serviços em nuvem, incluindo o Microsoft Azure e o Google Cloud Platform. Contabilizam seus custos operacionais e aplicam seus modelos contábeis de precificação de olho na oferta e procura, bem como a maximização dos lucros. Entre os custos operacionais variáveis de uma nuvem computacional, destaca-se o consumo de energia elétrica, sobretudo o dedicado ao *Data Center* (DC) (DANILAK, 2017). Contudo, a precificação do catálogo de serviços de provedores de nuvens, normalmente não é transparente aos clientes. Para clientes, a motivação financeira busca reduzir a configuração de processamento, armazenamento e comunicação dos recursos virtuais solicitados, não necessariamente otimizando a aplicação (estrutura e processamento) para minimizar o consumo de energia (HANINI; KAFHALI; SALAH, 2019). Porém, os componentes de custo do catálogo de serviços são influenciados, entre outros, pelo consumo de energia dos recursos físicos do DC que, por sua vez, são associados às demandas das aplicações virtualizadas em servidores físicos. Portanto, quem gera efetivamente a demanda energética do DC, o cliente, não é motivado a otimizar o consumo de energia em suas aplicações. Afinal, o mesmo desconhece o impacto de sua aplicação na demanda energética e, mesmo que possa estimar, ele não é recompensado financeiramente por esta ação. Dito isso, a pergunta a qual permeia este trabalho de mestrado é: O uso dos recursos computacionais em diferentes ambientes e diferentes tecnologias virtuais impactam no consumo energético produzido pelos servidores em nuvens IaaS?

A virtualização, por si só, contribui diretamente para o consumo energético eficiente, por meio de técnicas de consolidação e isolamento de recursos (COMERFORD, 2015). Na virtualização baseada em hipervisor (MV), os recursos físicos da máquina hospedeira são reservados para cada MV, enquanto a virtualização baseada em SO (contêineres) compartilha os recursos físicos disponíveis entre os contêineres ativos. Desse modo, ambas as virtualizações permitem uma ampla utilização destes recursos. Além dos recursos físicos, contêineres também compartilham o núcleo do SO hospedeiro, se tornando ainda mais atrativos, no que se refere a utilização eficiente dos recursos. Visto que o contêiner não possui todas as camadas de um SO completo, como é o caso de MVs. As tecnologias de virtualização bem como a configuração das aplicações influenciam o comportamento e sobrecarga dos recursos computacionais (*e.g.*, CPU, memória, rede, entre outros). Como por exemplo, a sobrecarga de rede gerada pelo contêiner é proveniente das diferentes configurações de rede que

a aplicação pode utilizar (*e.g.,* Network Address Translation (NAT[1]), bridge[2]) (DANILAK, 2017). Deste modo, a configuração realizada pelos usuários de suas aplicações e a tecnologia de virtualização selecionada, influenciam diretamente no consumo energético.

Neste cenário heterogêneo, no qual existem diversas aplicações conteinerizadas com demandas de diferentes recursos, um modelo de precificação adaptável é essencial(SOUPPAYA; MORELLO; SCARFONE, 2017). O modelo Preço Energético de Contêiner em Nuvem (PECN) contribui com a flexibilidade e transparência no consumo energético para com os clientes, o qual possibilita que estes conheçam o gasto energético ao executar suas aplicações. Consequentemente viabiliza aos clientes desenvolverem aplicações energeticamente conscientes. O consumo de energia é um importante componente dos modelos de custo em nuvens computacionais, além de objeto de pesquisas em trabalhos recentes (HINZ et al., 2018; KURPICZ et al., 2018). Esses modelos estratificam os componentes de ambientes virtualizados com hipervisor, quantificam seu consumo de energia individualizado, e propõem preços de acordo com seu consumo variável ao longo do tempo de execução. Desse modo, quanto maior o consumo de energia da aplicação, maior o custo de energia do DC, consequentemente, maior custo para o cliente. Portanto, reduzir o consumo energético tornou-se um interesse do cliente, uma vez que a recompensa financeira é imediata. No entanto, esses modelos são limitados a componentes e comportamentos associados a técnica de virtualização de hipervisor (MV). Havendo uma lacuna científica nas pesquisas de modelos de precificação para computação em nuvem, guiados pelo consumo de energia para contêineres.

As contribuições desse trabalho são: (i) análise comparativa do consumo de energia por recursos físicos em ambientes *bare metal*, MV e contêiner; (ii) estratificação dos componentes de consumo de energia dos contêineres; e (iii) a descrição de uma proposta de modelo de custo consciente de energia para contêineres. Os resultados preliminares do impacto dos contêineres no consumo de energia, mostram que cada recurso tem um comportamento diferente. Enquanto o consumo de energia da CPU aumenta drasticamente de acordo com a porcentagem de utilização da CPU, a memória e a rede possuem um aumento brando. Por outro lado, o armazenamento tem consumo de energia estável entre 1 GB e 80 GB (intervalo observado) para uso intensivo de E/S. A análise com granularidade fina do consumo energético, possibilitou identificar a estratificação de recursos impactados pelos contêineres. Nesse sentido, o PECN é um modelo de custo ciente do consumo de energia voltado para aplicações em contêineres. O *AWS Fargate* é escolhido para comparar os custos estimados de

[1] IP dinâmico alocado pelo serviço de rede da máquina hospedeira do contêiner, e usa uma subrede Interna

[2] Endereço IP estático de um conjunto de IPs da máquina hospedeira do host.

energia aplicados pela Amazon Web Services (AWS) com o modelo de custo PECN.

O *AWS Fargate* é o provedor de nuvem pública que oferece o modelo de serviço IaaS e opera com um modelo de cobrança *pay-as-you-go*. Assim como em um modelo de cobrança com repartição, os clientes pagam por alocação de recursos por período, segundos neste caso. No entanto, o modelo de custo *AWS Fargate* ignora a utilização do recurso associado ao aplicativo de contêiner contratado, considerando na fatura os custos dos recursos definidos. Desse modo, para comparar com o modelo de custo de energia PECN, considerou-se que os custos de energia do *AWS Fargate* representam 5%, 10% e 15% dos custos de negócios (HINZ et al., 2018) . A plataforma utilizada para experimento foi o GRID5000 Yeti Cluster localizado no site de Grenoble. Os aplicativos de contêiner *AWS Fargate*, descritos na fatura do cliente, foram lançados no Yeti Cluster com monitoramento de energia não intrusivo (wattímetro externo). No Yeti Cluster, o mecanismo de contêiner disponível eram ambientes *Docker* na plataforma *bare metal* ou hipervisor KVM. Os valores estimados do modelo de custo PECN, variam semanalmente entre US $ 2,31 e US $ 10,59, enquanto os valores da *Amazon Web Services* (AWS) Fargate US $ 2,71 e US $ 29,94, representando uma redução de até 35,39%.

A conta *AWS Fargate* era mais barata do que o modelo PECN somente no cenário em 5%. No entanto, se o aplicativo de contêiner for alocado e permanecer por um longo período sem demanda de recursos físicos, o servidor é definido como ocioso. O período total alocado é contabilizado pela *AWS Fargate*, até mesmo o período de inatividade do servidor. De modo que aplicações conteinerizadas são prejudicadas pelo modelo de custo *AWS Fargate*, quando o recurso de alocação é superestimado.

1.1 OBJETIVO

Neste trabalho, propõem-se um modelo de custo para contêineres, guiado pelo consumo energético, de acordo com a efetiva utilização dos recursos virtuais disponibilizados. Para tanto, consideram-se os seguintes objetivos específicos:

- Realizar uma análise do consumo energético de contêineres baseados em Docker;
- Especificar as variáveis e definir a formulação dos custos;
- Representar o consumo real de um contêiner, e especificar seu custo individual e compartilhado; e
- Identificar a influência do gerenciamento dos contêineres no consumo de energia.

1.2 MÉTODO DA PESQUISA

As pesquisas da computação e de suas subáreas possuem três tipos quanto aos objetivos, sendo; (I) pesquisa exploratória: proporcionar maior familiaridade com o problema(explicitá-lo). Pode envolver levantamento bibliográfico, entrevistas com *stakeholders* experientes no problema pesquisado; (II) pesquisa descritiva: descrever as características de determinadas populações ou fenômenos. Uma de suas peculiaridades está na utilização de técnicas padronizadas de coleta de dados, tais como o questionário e a observação sistemática; (III) pesquisa explicativa: identificar os fatores que determinam ou que contribuem para a ocorrência dos fenômenos. É o tipo que mais aprofunda o conhecimento da realidade, visto que explica a razão, e o porquê das coisas. Por esse motivo, é o tipo mais complexo e delicado (GIL, 2008).

Segundo a definição apresentada, este trabalho se encaixa na categoria de pesquisa descritiva, pois são utilizadas ferramentas estatísticas e de análise de dados, a fim de avaliar o consumo dos recursos computacionais. As variáveis de um sistema são classificadas em duas grandes categorias, variáveis qualitativas e as quantitativas (WAZLAWICK, 2017). Outra classificação existente, separa as variáveis em três grupos, sendo (I) independente: variáveis que aplicam influência em outras variáveis, em que o pesquisador realiza alterações para observar possíveis alterações nas variáveis dependentes; (II) dependente: variáveis que dependem diretamente de alguma variável independente, não sendo diretamente alteradas pelo pesquisador; (III) interveniente: variáveis que causam ruídos durante a coleta de dados da pesquisa, sendo variáveis existentes no sistema e não podem ser controladas pelo pesquisador, estas variáveis devem ser reduzidas sempre que possível (RAUEN, 2012). Diante do exposto, o trabalho em questão aborda variáveis qualitativas e quantitativas. Mecanismos de buscas acadêmicos foram utilizados para auxiliar esta pesquisa: o *Springer Link*, *Science Direct*, *ACM Digital Library*, *IEEEXplore*.

Para classificar os principais artigos e autores da área pesquisada, as seguintes palavras chaves foram consideradas nesta pesquisa: IaaS - ambiente aplicado, *virtual machines* - técnica de virtualização, *Contêiner* - técnica de virtualização, Docker - ferramenta específica de contêiner, *Cost* - custos dos serviços ofertados , *Model* - modelos de custo, *Energy accounting* - contabilização da energia, *Power Modeling* - modelagem energética, *Virtualization* A string de busca final utilizada nos mecanismo de busca foi: (IaaS AND "virtual machines"AND (Container OR Docker) AND Cost AND Model AND (Energy OR "Energy accounting"OR "Power Modeling") AND Virtualization AND NOT (iot OR "internet of things") AND NOT mobile AND NOT "Green computing").

1.3 ESTRUTURA DO TEXTO

O presente trabalho está organizado em 7 capítulos. O Capítulo 2, aborda a pergunta problema a qual motivou a proposta deste projeto, bem como sua importância na área estudada, além de apresentar os trabalhos correlatos e as considerações parciais. No Capítulo 3, destaca-se a estrutura, tecnologias e orquestradores de contêineres, bem como o consumo energético em provedores IaaS. No Capítulo 4, são abordados experimentos e análise do consumo energético dos recursos computacionais, além de comparar e quantificar o impacto destes recursos (*i.e.*, CPU, memória, armazenamento e rede) em ambientes de contêineres, MV e *bare metal*. Em seguida, o Capítulo 5, descreve o modelo de precificação proposto. O Capítulo 6, apresenta um estudo de caso para exemplificar a proposta de preços dos recursos computacionais de contêineres com base na tabela de preços pública do serviço AWS *Fargate*, e os resultados parciais. Finalmente o Capítulo 7 apresenta as considerações finais e trabalhos futuros.

2 MODELO DE CUSTO PARA RECURSOS VIRTUALIZADOS

As nuvens computacionais vem sendo adotadas por organizações de diferentes segmentos e tamanhos, devido a flexibilidade e agilidade em que seus clientes podem hospedar seus dados, alocar e executar suas aplicações. Os recursos computacionais são ofertados virtualmente em provedores de nuvem aos clientes. Neste modelo, os recursos computacionais podem ser adquiridos em forma de serviço, o qual possibilita avanços em relação a escalabilidade tecnológica, além de controle de custos. Quanto aos custos, a consolidação dos recursos computacionais influenciou diretamente a redução energética dos DCs. Desde o surgimento do modelo de serviço de nuvem computacional, as organizações vêm migrando, total ou parcialmente, seus serviços para este novo modelo de fornecimento de recursos. Algumas organizações ainda mantêm seus DCs para garantir o controle rígido da política de gerenciamento, ou por conta das limitações de software e hardware legados. Porém, a concentração de recursos físicos em grandes DCs, através do modelo de serviço de nuvem computacional, tem uma melhoria no hardware, reduzindo o consumo global de energia (BAWDEN, 2016). A melhor utilização de recursos dos DCs não superou a crescente demanda por novos aplicativos. Em 2015, o consumo de energia chega a 416,2 terawatts/hora, com previsão de dobrar o consumo a cada quatro anos (BAWDEN, 2016). Nos Estados Unidos, os DCs consumiram mais de 90 terawatts/hora (DANILAK, 2017). Este sistema de abastecimento de energia causa impactos econômicos e ambientais. As quais estas áreas são tópicos para pesquisa científica, ou seja, nuvens verdes, fornecimento de energia renovável, modelos de custo (HU; DENG; WU, 2013; PANDIKUMAR; KABILAN; AMALRAJ, 2012; JAIN et al., 2013; ZHANG et al., 2018; GARG; BUYYA, 2012; SHARMA et al., 2017).

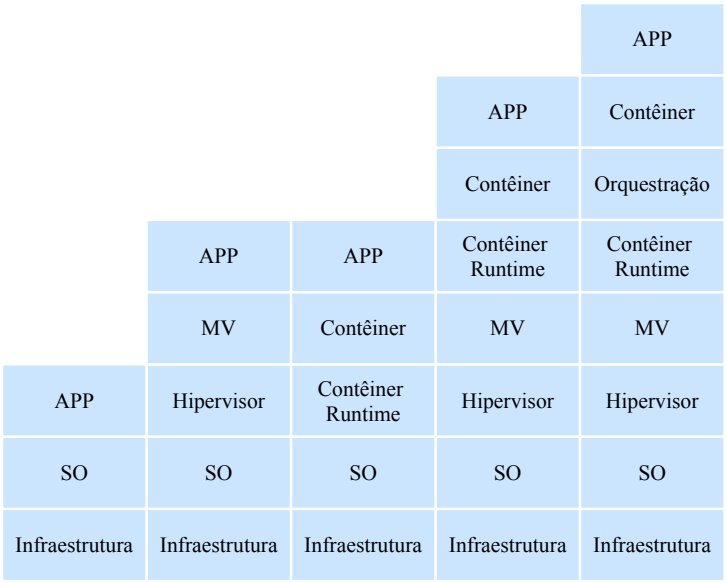
O fornecimento de energia é contabilizado como custo de DC variável na categoria de custo operacional, chegando a 50% desta categoria (GUITART, 2017; COMERFORD, 2015). Por outro lado, o custo do fornecimento de energia é conhecido por fornecedores que estão continuamente em busca de novos métodos, equipamentos ou práticas para melhorar sua eficiência energética de DC. Sob outra perspectiva, os clientes da nuvem não estão cientes do consumo de energia de seus aplicativos, nem são recompensados por otimizá-los. Na visão dos provedores de nuvem, a virtualização de recursos pode amenizar a falta de conhecimento dos clientes sobre o consumo de energia de seus aplicativos.

2.1 RECURSOS VIRTUALIZADOS

Esta tecnologia possibilita a otimização dos recursos computacionais, proporcionando melhorias no uso de energia e simplificação de gerenciamento de um DC (KOMINOS; SEYVET; VANDIKAS, 2017). A virtualização é um conceito amplamente utilizado, existente desde meados dos anos 60. Não obstante, as tecnologias virtuais apresentam diversas melhorias devido à dependência conceitual e aos modelos de preços usados pelos provedores de nuvem. A abordagem de virtualização é amplamente usada em provedores IaaS, fornecendo recursos de infraestrutura aos usuários (*e.g.,* capacidade de processamento de CPU, armazenamento e ambientes virtuais) de forma transparente. Esta tecnologia permite que os provedores IaaS concedam acesso a um conjunto de recursos de computação remotos, configuráveis e compartilhados (*e.g.,* redes, servidores, armazenamento, aplicativos e serviços) essenciais para os serviços fornecidos. Desta forma, os provedores IaaS eficientemente compram e disponibilizam recursos para seus clientes, minimizando os custos de gerenciamento, comunicação e energia (MELL; GRANCE et al., 2011; HAMMADI; MHAMDI, 2014). A virtualização possui diversas tecnologias e formatos de implementações. Duas principais tecnologias amplamente empregadas em provedores de nuvens são a virtualização baseada em hipervisor e em contêiner, possuindo diversos modos de virtualização baseadas em MV e em contêineres. Os modelos de conteinerização (*i,e.,* LXC, Core OS Rocket, Docker), diferem de MV devido ao uso de orquestradores, já que um aplicativo geralmente é composto de vários contêineres. Um exemplo de virtualização baseada em hipervisor é aquela que requer que cada MV carregue um SO completo, duplicando-o a cada MV e, consequentemente, resulta em sobrecarga de desempenho. A virtualização em contêiner é considerada um ambiente virtual leve, baseada em SO e compartilha o núcleo do SO da máquina hospedeira. Os contêineres não geram SO extra para cada nova instância, conforme observado na virtualização com MV, deixando o isolamento a cargo do SO do próprio hospedeiro (SOUPPAYA; MORELLO; SCARFONE, 2017).

O contêiner é uma abstração da camada de aplicação, que empacota o código e as dependências necessárias para seu funcionamento (SILVA; KIRIKOVA; ALKSNIS, 2018). Existem vários ambientes alternativos, a Figura 1 lista apenas alguns desses ambientes adotados por organizações (*e.g., bare metal*, MV, contêiner, contêiner sobre MV, orquestração de contêiner sobre MV). Ao executar vários aplicativos em um ambiente físico, esses aplicativos passam a competir por recursos computacionais, causando problemas de desempenho. O uso do ambiente de contêineres não restringe o uso de ambientes com MVs. Por outro lado, permite o encapsulamento de aplicações em várias camadas (ESTRADA et al., 2014). A adoção de contêineres tem se tornado uma tendência no mercado, sobretudo, em plataformas de serviço de nuvem compu-

Figura 1 – Ambientes Computacionais



Fonte: Adaptado de (PANIZZON et al., 2019)

tacionais. O principal produto baseado na virtualização de recursos em SO é o contêiner Docker[1]. Somente em 2016, a tecnologia de contêiner Docker atingiu *U*$761 milhões, e esse mercado poderia crescer mais de 35 vezes até 2020, o que representaria *U*$27 bilhões (KIM et al., 2018; WU; BUYYA; RAMAMOHANARAO, 2019). A flexibilidade e escalabilidade dos contêineres são atrativos irrefutáveis para sua adoção, mas o principal motivo para este crescimento é a eficiência de recursos (KIM et al., 2018). A transição completa de serviços para computação em nuvem pode resultar em um incremento de eficiência de 4 a 5 no uso de recursos físicos, utilizando apenas o tempo ocioso de recursos físicos em DCs dos provedores (WU; BUYYA; RAMAMOHANARAO, 2019). No caso da virtualização com contêineres, os ganhos podem chegar a 6 vezes, quando comparados a execuções *bare metal*. A implementação de tecnologias de virtualização no mercado é um fato, mas como adequar os fatores econômicos, as leis de oferta, demanda e incentivar os clientes a fazerem o uso consciente dos recursos do DC?

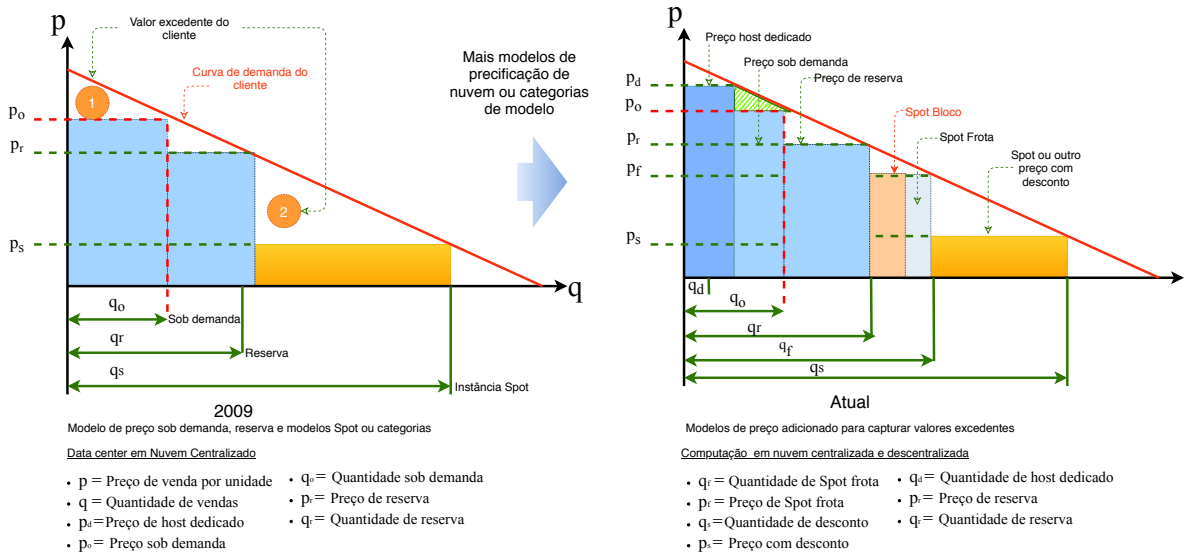
Os provedores de nuvens têm diferentes modelos de preços, cada um com diferentes estratégias (WU; BUYYA; RAMAMOHANARAO, 2019). Os principais modelos são baseados em oferta, mercado e valor. O modelo baseado em valor tem como objetivo a demanda do cliente. O modelo baseado em custos concentra-se na oferta, e o modelo baseado no mercado busca o equilíbrio entre a oferta e a demanda. Em meados de 2009, os modelos de custo e preços de recursos ofertados sob demanda, estão associados aos primeiros modelos de serviços em nuvem(WU; BUYYA; RAMAMOHA-

[1] <http://docker.com>

NARAO, 2019).

Atualmente, a gama de serviços oferecidos por grandes DCs (*i.e.,* AWS, Google, Microsoft Azure), é vasta, incluindo modelos de infraestrutura e formas de alocar, usar e reservar recursos. Portanto, o preço dos recursos não poderia ser diferente. Os fornecedores superaram as barreiras com novos modelos econômicos para oferecer um modelo de custo adequado a uma ampla gama de clientes, aqueles com perfis totalmente diferentes (WU; BUYYA; RAMAMOHANARAO, 2019). A adaptação de novos perfis de clientes e a redução de recursos ociosos em DCs aferem os benefícios desses esforços dos provedores. A nuvem computacional não é um produto para um único nicho de cliente, tem proposta de produto adequado a qualquer modelo de negócio, até mesmo para clientes domésticos. Apesar das melhorias constantes e da gama de novos produtos, os modelos de custo ainda deixam lacunas. Como pode ser observado na Figura 2, apesar de, ao longo dos anos a AWS ter desenhado novos modelos de serviço, ora com foco no mercado, ora em custos operacionais, todos com o objetivo de minimizar lacunas (área branca no diagrama).

Figura 2 – Evolução modelos de Custos AWS.



Fonte: Adaptado de (WU; BUYYA; RAMAMOHANARAO, 2019)

É visível o êxito da AWS na redução dos recursos subutilizados, através da redução das áreas de *gaps* na comparação do esquema da direita (2009), com o da esquerda (2019). O esquema tem em seu *eixo y* o preço do servidor por unidade (p), e em seu *eixo x* o número de vendas por modelo de serviço da AWS (q). O preço dos serviços sob demanda é representado por p_o, enquanto p_r representa o custo por reserva, ainda o p_s representa o preço com desconto, e p_d representa os custos de um hospedeiro dedicado. O q_o representa o número de vendas realizadas por demanda, o

q_r representa a quantidade de vendas por reserva, ainda o q_s representa o número de vendas dos serviços com desconto (*e.g., Spot*), e q_f representa o número de serviços *Spot* por frota[2].

Em dois momentos distintos as lacunas podem ser visualmente identificadas, o primeiro é nos primórdios da computação em nuvem (2009), e segundo em um momento em que a virtualização da computação em nuvem é uma tendência (2019). Em meados de 2009, a AWS concentrava seus recursos em infraestruturas de DCs centralizados, chegando em 2019 com infraestruturas distribuídas geograficamente e totalmente descentralizadas. A oferta e a demanda foram a base do serviço principal em 2009 (instância *Spot*), que proporcionou descontos de até 90% quando comparado ao mesmo servidor no modelo de oferta padrão da AWS. Devido ao modo de sua oferta, as instâncias *Spot* utilizavam recursos ociosos de outros clientes, até que estes recursos fossem solicitados.

Em 2015, a AWS inovou o modelo de oferta de recursos novamente, oferecendo um modelo específico para a execução de cargas de trabalho tolerantes a falhas. No mesmo ano, o portfólio de produtos da AWS incluiu dois outros modelos de preços para o serviço *Spot*, so *Blocos Spot* (*i.e.,* instâncias *Spot* com uma duração especificada), e o *Spot Frota* (*i.e.,* conjunto de Instâncias *Spot* que funciona com base em um determinado critério), aproximando-se de seus clientes e captando novos interessados no serviço, reduzindo os recursos subutilizados (WU; BUYYA; RAMAMOHANARAO, 2019). Os avanços não vieram somente da AWS, mas o mercado aquecido também foi um impulsionador de novas ofertas de concorrentes, *Google Cloud Plataform* (GCP) e *Microsoft Azure*, que apressaram-se em lançar serviços baseados em modelos focados em preços baixo. Devido às instâncias de baixo custo, os clientes ficaram sujeitos a baixa prioridade e disponibilidade, abrindo um novo nicho de mercado. Da mesma forma, na visão dos provedores, o uso da instância *Spot* aumenta sua taxa de utilização do DC, reduzindo seus recursos ociosos .

O esquema apresentado na Figura 2, possibilita uma análise qualitativa do mercado de oferta de recursos de acordo com as demandas nos anos 2009 e 2019. Embora os ganhos após a análise qualitativa sejam evidentes e os resultados fornecidos pelo vasto número de modelos sejam positivos, a perpetuação de lacunas ainda persiste em 2019. Acontece que os interesses, políticas e percepções dos provedores são a base para a evolução dos modelos de custos e pacotes de serviços. No entanto, pouco se considera da informação de posse do cliente, por exemplo, otimização de seus aplicativos para menor consumo de energia ou ações politicamente corretas, como preservação do meio ambiente (HINZ et al., 2018). Recompensar clientes cientes do con-

[2] Conjunto de Instâncias spot, executados com base em critérios especificados (*e.g.,* SO, zona de disponibilidade, plataforma de rede).

sumo de energia pode ser uma forma de motivar um aumento na otimização das aplicações. Por esta razão, este trabalho destaca a necessidade de preencher a lacuna de preço por reserva descrita na Figura 2 através da área hachurada em verde, localizada na imagem identificada como *Atual*.

2.2 DEFINIÇÃO DO PROBLEMA

Os provedores de nuvens computacionais, vem aprimorando ao longo dos anos a oferta dos seus serviços, preenchendo as lacunas de processos subutilizados nas nuvens computacionais, além de buscarem continuamente maximizar a eficiência energética dos DCs (WU; BUYYA; RAMAMOHANARAO, 2019). Porém, do lado do cliente, os esforços dos provedores deixam a desejar nos modelos de custos ofertados. Os modelos de custos ofertados pelos provedores, pecam em não incentivarem economicamente aos seus clientes a desenvolverem suas aplicações conscientes energeticamente. De modo que, esta ação beneficiaria não somente o custo final do cliente, mas também a preservação ambiental. Além de que, esta ação reflete também na redução de Custo Total de Propriedade (CTP) dos provedores, tornando a oferta conveniente não somente ao seu cliente, mas também ao próprio provedor.

É imprescindível que os DCs ofertem modelos de custos, os quais considerem o consumo energético em sua precificação, como foi porposto por (HINZ et al., 2018) em sua pesquisa. Porém, o modelo de (HINZ et al., 2018) contempla apenas ambientes baseados em hipervisor, excluindo contêineres em *bare metal* de seu modelo. Segundo (DIAMANTI, 2019), ambientes baseados em MV suportam um número inferior de contêineres por servidor físico, tornando o modelo de (HINZ et al., 2018) incompleto, pois não contempla a precificação para contêineres sem o uso de hipervisor. Ainda, a tecnologia de contêiner está dentre os cinco principais serviços em nuvem com crescimento acelerado (WEINS, 2020), desse modo é fundamental considerar modelos de custo para ambientes conteinerizados.

Dada a tendência e a procura das organizações por contêineres, e a lacuna científica na precificação desta tecnologia de virtualização. Faz-se necessário um modelo de custo para contêiner, o qual considere o consumo energético em sua modelagem. De modo que os clientes dos DCs, sejam estimulados a desenvolverem suas aplicações energeticamente consciente.

2.3 TRABALHOS RELACIONADOS

A concepção de modelos de custos está vinculada a dois elementos: (i) identificação dos componentes do ambiente a ser modelado energeticamente, bem como a relação entre eles; e (ii) concepção de ferramentas de medição e/ou análise de mé-

tricas para quantificação de uso do componentes (identificados em (i)). Na literatura existem diversas pesquisas quanto ao consumo energético em ambientes computacionais. Esses estudos examinam diferentes ambientes (*e.g., bare metal*, MVs, contêineres), manipulam vários recursos (*e.g.,* processamento, comunicação, memória, armazenamento) além de ter vários propósitos (*e.g.,* ferramenta de monitoração, modelos de custo). A Tabela 1 lista os trabalhos relacionados, cuja preocupação é o consumo de energia, e que lidam com os elementos (i) e (ii) de concepção de modelos de custos. Essas pesquisas possuem características distintas, entretando nenhum dos trabalhos encontrados na literatura atendem a todos os recursos deste trabalho.

Tabela 1 – Modelos e Ferramentas sob a ótica do consumo energético de MV em DC ou Nuvens IaaS.

Autor	Objetivo	Ambiente	Recursos	Escopo
(HINZ et al., 2018)	Análise do custo energético específico da MV e Hipervisor	Nuvens Computacionais IaaS	CPU, tráfego de rede	Precificação
(KURPICZ et al., 2018)	Modelo de custo proporcional de energia em ambientes baseados em provedores de nuvem	Nuvens IaaS	CPU	Precificação
(BHATTACHARYA et al., 2013)	Ferramenta de medição do consumo energético	DC	CPU, Rede, Memória e Armazenamento	Monitoração
(ZAKARYA; GILLAM, 2018)	Relaciona a eficiência energética de cargas de trabalho em função do consumo de CPU	Nuvens	CPU e Memória	Monitoração
(BRONDOLIN; SARDELLI; SANTAMBROGIO, 2018)	Ferramento DEEP-mon monitoramento energético de contêineres	SO	CPU, rede e memória	Monitoração
(LEITNER; CITO; STÖCKLI, 2016)	Modelo de custos de implantação de contêineres	IaaS, PaaS	CPU, Memória e Armazenamento	Precificação
Proposta	Modelo de custo de contêineres baseado no consumo energético	Nuvens Computacionais IaaS	CPU, Rede, Memória e Armazenamento	Precificação

Fonte: Elaborado pela autora (2020).

As pesquisas dos autores (HINZ et al., 2018) e (KURPICZ et al., 2018) propõem modelos de precificação para nuvens IaaS com foco em MVs e guiados pelo consumo energético. O *Proportional-Shared Virtual Energy* (PSVE) baseia seu modelo de precificação na identificação do consumo de energia individualizado por MV através do hipervisor, contabilizado os recursos individual e coletivamente. Os custos coletivos são rateados, proporcionalmente por MV de acordo com o número de CPUs alocadas no período (HINZ et al., 2018). Os benefícios vão não só para o menor custo do cliente e dos fornecedores CTP, mas também para ações de TI verde. Entretanto, PSVE ignora a relevância dos contêineres nas nuvens computacionais e restringe o modelo proposto apenas em MVs. O autor (KURPICZ et al., 2018) propõe o *Energy-proportional*

Profiling and Accounting in Virtualized Environments (EPAVE) um modelo de precificação para contabilizar o consumo dinâmico de uma MV, e o custo estático proporcional da infraestrutura em nuvem, atribuindo custos gerais de energia por MV. O trabalho do autor (KURPICZ et al., 2018) difere de (HINZ et al., 2018) na política da divisão dos custos de energia e ignora o consumo de energia do tráfego de rede. Técnicas de medição de consumo de energia e descrição do impacto da carga de trabalho, com foco em técnicas baseadas em hipervisor (MVs), são descritas nos documentos de pesquisa (BHATTACHARYA et al., 2013) e (ZAKARYA; GILLAM, 2018) Em relação ao impacto da cargas de trabalho no consumo de energia, os autores(ZAKARYA; GILLAM, 2018) afirmam que o impacto na eficiência energética varia de acordo com os modelos de CPU.

Por fim, dois trabalhos se destacam associados a virtualização baseada em SO e aplicativos de contêineres: (i) a ferramenta DEEP-mon, que monitora os recursos de CPU, rede e memória no nível de SO (contêineres) para mensurar o consumo de energia (BRONDOLIN; SARDELLI; SANTAMBROGIO, 2018); e (ii) um modelo de preços para implantação de aplicativos em nuvem baseados em microsserviços (LEITNER; CITO; STÖCKLI, 2016). Uma pesquisa realizada no ano de 2019 mostra que mais de 34% das empresas as quais investem acima de *US*$100 mil em tecnologias, estão aderindo aos ambientes de contêineres em *bare metal*, migrando os aplicativos de MVs para contêineres (DIAMANTI, 2019),. Essas empresas buscam aumentar seu desempenho, reduzir a complexidade da infraestrutura, melhorar a flexibilidade e reduzir os custos. Os contêineres oferecem eficiência e a possibilidade para os operadores fazerem mais com menos, gerenciando aplicativos críticos com alta disponibilidade. O uso de contêineres pelas empresas, varia da modernização de aplicativos legados à análise de *big data* (DIAMANTI, 2019). Tal pesquisa destaca o crescente impacto dos contêineres para as organizações, atestando a lacuna científica no estudo de modelos de precificação de contêineres para provedores de nuvens computacionais orientados pelo consumo de energia e a motivação deste trabalho.

Os provedores de nuvens computacionais vem aprimorando seus serviços oferecidos ao longo dos anos, preenchendo lacunas de processos subutilizados e buscando maximizar a eficiência energética dos seus DCs (WU; BUYYA; RAMAMOHANARAO, 2019). No entanto, do lado do cliente, os esforços dos provedores ficam aquém dos modelos de custo oferecidos. Os modelos de custos ofertados pelos fornecedores falham em não dar benefícios econômicos aos seus clientes para o desenvolvimento de aplicações conscientes energeticamente. Portanto, esta ação beneficiaria não só o custo final do cliente, mas também a preservação ambiental. Esta ação ainda reflete na redução de CTP dos provedores, tornando a oferta conveniente não somente ao seu cliente, mas também ao próprio provedor. O DCs deve ofertar modelos de custos

os quais considerem o consumo de energia em seus preços, conforme proposto em PSVE (HINZ et al., 2018). No entanto, este modelo inclui apenas ambientes baseados em hipervisor, excluindo contêineres em *bare metal* de seu modelo, tendo menor suporte de contêineres por servidor físico em ambientes MV (DIAMANTI, 2019). Tal base teórica torna o modelo PSVE incompleto, pois não inclui a precificação de contêineres sem o uso de hipervisor.

Os provedores de nuvens computacionais oferecem soluções restristas á plataforma de execução, cujos contêineres devem ser provisionados em MV (*e.g.,* Microsoft Azure). Ainda, os provedores precisam ofertar serviços para execução de contêineres com preços pela alocação de recursos, e não por seu uso real (*e.g.,* serviço AWS Fargate). Dada a tendência e demanda das organizações por contêineres, e a lacuna científica na precificação desta tecnologia de virtualização, é necessário propor um modelo de custo para contêineres, o qual considere o consumo de energia em sua modelagem. Para que os clientes dos DCs, sejam incentivados a desenvolver suas aplicações energeticamente conscientes. O estado da arte no consumo consciente de energia reforça a importância do modelo de precificação no contexto IaaS e o monitoramento de energia de aplicativos em contêineres. Identificar também uma lacuna científica em relação ao projeto de modelo de precificação baseado em contêinerização com consciência energética, o foco deste trabalho.

2.4 CONSIDERAÇÕES PARCIAIS

Este capítulo apresentou modelos de serviços ofertados em provedores em nuvem, bem como recursos computacionais virtualizados e ambientes nos quais estes são usados, justificando a motivação deste trabalho. Além de apresentar os desafios e oportunidades atuais nesta área da computação. Ainda, apresentou uma análise de trabalhos presentes na literatura, abordando os modelos de preço, ferramentas de monitoração e o consumo de energia em nuvem computacional. Através desta análise, este trabalho abordou uma lacuna existente na literatura quanto a modelos de precificação para contêineres baseados no consumo de energia. A definição do problema que esta pesquisa pretende resolver também foi abordada neste capítulo.

3 REVISÃO DE LITERATURA

A revisão de literatura visa apresentar os conceitos relevantes a fim de fundamentar os elementos abordados neste trabalho. Na Seção 3.1, é explanada a estrutura de contêineres, além de exemplificar três tecnologias contenerizadas (*i.e.,* Docker, LXC e Rocket) na Seção 3.1. Na Seção 3.2, apresentou-se a introdução de orquetradores e foram descrita três soluções (*i.e.,* kubernetes, Apache Mesos e Docker Swarm) orquestradoras. A Seção 3.3 descreve a relevância do consumo energético na composição de Custo Total de Propriedade (CTP) dos *Data Center*s (DCs).

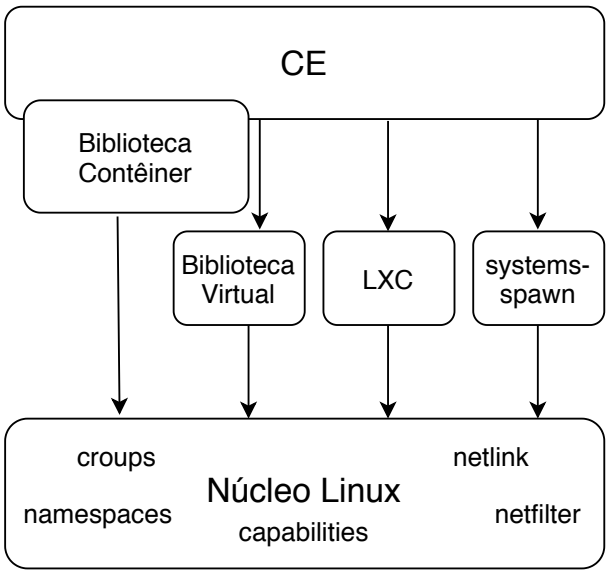
3.1 CONTÊINERES

A tecnologia de virtualização baseada em contêiner não exige o provisionamento de um Sistema Operacional (SO), no entanto têm os recursos de hardware compartilhados com o SO do hospedeiro subjacente, e podem ser executados dentro de uma Máquina Virtual (MV). Os contêineres fornecem agilidade no desenvolvimento e na execução de aplicativos em um ambiente em nuvem, principalmente quando inseridos em uma arquitetura de microsserviços. Aplicações conteinerizadas são encapsuladas e isoladas dos demais processos do SO pelo *Container Enginer* (CE), responsável pelo gerenciamento dos contêineres e de suas imagens (PAHL; LEE, 2015). Por meio de chamadas de sistema, o CE utiliza componentes do núcleo do SO subjacente para gerenciar e alocar recursos. Os componentes do núcleo do SO utilizados por meio de chamadas de sistema em contêineres LXC são o *control Group (cgroups)*, *namespace*, *netlink*, *netfilter*, *capabilities*, *appamor*. Estes componentes podem variar de acordo com a tecnologia de contêiner utilizada. A Figura 3 representa o ambiente isolado dos contêineres LXC. O *cgroups* e o *namespace* são os componentes responsáveis pela criação dos contêineres. Ao *cgroups* cabe o gerenciamento do compartilhamento dos recursos físicos do hospedeiro entre os contêineres instanciados, limitando e isolando a alocação desses recursos (*e.g.,* processador, memória, armazenamento e rede). A função do *namespace* tem um amplo nível de abstração, atuando no isolamento entre grupos de processos e de recursos organizados em subníveis (SULTAN; AHMAD; DIMITRIOU, 2019).

3.1.1 Tecnologias de Contêineres

Existem diversas tecnologias que suportam contêineres, as quais dão suporte na criação e implementação de conteineres. Desse modo, este trabalho explana três dessas tecnologias amplamente conhecidas e implementadas, as quais são: LXC, Core

Figura 3 – Arquitetura Contêiner



Fonte: Adaptado de (HYKES, 2018)

OS Rocket e Docker (MERCL; PAVLIK, 2019). No entanto, cada tecnologia de contêiner possui características as quais se adequam a diferentes necessidades dos usuários. Por este motivo é necessário examinar as tecnologias com o intuito de identificar qual dessas se aplica, ou melhor se qualifica ao cenário, e ao objetivo que se busca atingir (*i.e.,* eficiência energética, desempenho, facilidade, entre outros).

O Docker se destaca em popularidade dentre essas tecnologias abordadas, ainda é a mais utilizada e, destaca-se, sobretudo, pela simplicidade na sua implantação. Inclusive a Amazon ECS oferta um serviço em nuvem computacional AWS Fargate para execução de contêineres com suporte ao Docker. Em quesito de consumo energético, ambas as tecnologias possuem consumo similar. O impacto de diferentes tecnologias de virtualização no consumo energético, é avaliado por (MORABITO, 2015), o qual identifica que tanto o Docker quanto o LXC possuem consumo energético análogo. Já o consumo energético da tecnologia Docker e Rocket são aferidos por (KRETEN; GULDNER; NAUMANN, 2018), no qual avaliam que ambos diferem minimamente no consumo energético. Desse modo, a tecnologia Docker foi escolhida por (KRETEN; GULDNER; NAUMANN, 2018),para dar sequência ao experimento realizado, devido a facilidade de utilização e popularidade. A Subseção 3.1.1.1 aborda características da tecnologia LXC, já a Subseção 3.1.1.2 do Rocket e, a Subseção 3.1.1.3 da tecnologia Docker.

3.1.1.1 LXC

O Linux Containers (LXC) é uma tecnologia de virtualização de código aberto baseada em conteinerização do núcleo do Linux. Esta tecnologia possibilita que contêineres Linux sejam construídos e gerenciados por meio de uma *Application Programming Interface* (API) eficaz, moldável e com ferramentas simples. O LXC e o Docker utilizam recursos em comuns, porém diferem em diversos pontos. O Docker faz uso de camadas fixas, com o intuito de possibilitar a reutilização de projetos. Enquanto o LXC restringe um aplicativo por contêiner, no qual permite a criação de aplicativos únicos e ou múltiplos. Além de possibilitar que contêineres sejam clonados de um subvolume, criando vários contêineres a partir desta clonagem (QIU et al., 2017).

O LXC é utilizado na execução de diversos contêineres Linux isolados dentro de um único host LXC. Não utiliza MV, porém permite o uso de um ambiente virtual com seus próprios recursos de CPU, memória, armazenamento e rede por meio dos recursos de *namespace* (*i.e.,* responsável por fornecer o ambiente virtual ao contêiner) e *cgroups* (controla o uso dos recursos utilizados pelos processos) (CONTAINERS, 2017). Ainda, o LXC facilita a criação de contêineres sem previlégios, permitindo que usuários sem permissão *root* possam criar contêineres .

3.1.1.2 CoreOS Rocket

Rocket é uma tecnologia de código aberto baseada em contêiner integrada pelo CoreOS. Esta tecnologia implementa aplicativos baseados em contêineres virtuais. Porém, o Rocket é estruturado para execução de funções simples, projetado para proporcionar um ambiente heterogêneo, seguro e eficiente (POLVI, 2014).

Heterogêneo visando que os recursos (*i.e.,* CPU,RAM, rede, armazenamento), dos contêineres devem ser integrados, bem estruturados e independentes. Seguro para promover o isolamento dos processos, bem como garantir a confidencialidade de cada conteiner. Eficiente para manter o desempenho das aplicações. Esta tecnologia trabalha com um mecanismo de linha de comando para executar seus contêineres de aplicativos. É composto por descrições de *designs* de imagens, facilitando que os contêineres sejam migrados de maneira simplificada. Porém, sua utilização pode ser complexa devido ao ambiente baseado em linha de comando.

3.1.1.3 Docker

O Docker é uma tecnologia centrada em aplicativos baseados em contêiner, o qual utiliza recursos do núcleo do SO para executar contêineres isolados do restante do sistema. Assim como o *Rocket*, o Docker também implementa aplicativos baseados em contêineres. De acordo com (SANTOS et al., 2018), o Docker ocasiona um au-

mento significante no consumo energético. Especialmente em razão das chamadas de sistema de E/S. Diferente do Rocket, o Docker manipula o isolamento dos recursos do núcleo do sistema através de uma API completa e iterativa (POLVI, 2014). Ao criar um contêiner no Docker, também é gerado um conjunto de *namespace* (pid, net, ipc, mnt, uts) (DOCKER, 2016).

O *PID* é responsável por identificar os processos, e possibilita a reutilização dos ids dos processos. Ao utilizar a configuração de rede no modo *host* para um contêiner, o mesmo compartilha o *namespace net* da máquina hospedeira, no qual a pilha de rede desse contêiner não é isolada do *host* Docker. Desse modo, o contêiner não obtém seu próprio endereço IP alocado. O *IPC*, é o *namespace* responsável por fornecer a separação de segmentos nomeados de memória compartilhada, filas de mensagens e semáforos. A gestão de pontos de montagem dos sistemas de arquivo é realizada pelo *mnt*. Já o *namespace Unix Timesharing System* (UTS) é responsável por definir o domínio visível aos processos em execução e por definir o nome do host.

Para que o SO subjacente possa compartilhar os recursos de hardware disponível com os contêineres existentes, o Docker executa o componente *Control group (cgroups)* . Já que este componente é responsável por disponibilizar e controlar o uso de recursos utilizados pelos processos (NETTO et al., 2016). O Docker utiliza o sistema de arquivos *Union File System* (UFS), o qual é responsável por estabelecer camadas utilizadas para fornecer blocos de construção para contêineres. O ambiente de gerenciamento de contêineres do Docker ainda possibilita a execução simultânea de duas ou mais instâncias, porém exige do administrador o controle manual dessas instâncias.

Com a flexibilidade e a escalabilidade de contêineres, o seu uso por aplicações microsserviço de grande escala (*e.g.,* Uber, Netflix e Amazon (Hassan; Ali; Bahsoon, 2017)) é uma prática comum, tornando o gereciamento manual de centenas de milhares de contêineres impossível. Desse modo, independente da tecnologia utilizada, é necessária a automação do gerenciamento dos contêineres. Assim como ocorre nos provedores *Infrastructure as a Service* (IaaS), os quais utilizam ferramentas para a configuração, manutenção e gerenciamento de contêineres de modo automático. Essas ferramentas denominadas orquestradores, são empregadas na administração de contêineres, e empenham um papel fundamental na otimização dos serviços conteinerizados.

3.2 ORQUESTRADORES DE CONTÊINERES

A orquestração de contêineres serve de apoio automatizado na gerência de contêineres, tornando possível ao administrador selecionar, implantar, monitorar e

controlar dinamicamente a configuração de um vasto conjunto de aplicativos empacotados em contêineres. Tecnologias orquestradoras de contêineres escalonam e administram diferentes aplicações, além de atuarem na detecção e correção de falhas. Ainda, os orquestradores de contêineres aplicam algoritmos de otimização de alocação de recursos com o intuito de atingir a utilização ideal dos recursos compartilhados de um mesmo hospedeiro (PEINL; HOLZSCHUHER; PFITZER, 2016).

Orquestradores proveem informações de monitoração essenciais para *debug* e *profilling* de problemas na implementação de sistemas modulares, (*e.g.,* microsserviços). Se o gerenciamento de contêineres possibilita a melhor utilização dos recursos físicos, a orquestração torna o ambiente de contêineres, efetivamente, em uma plataforma de larga escala e, atua para o mesmo fim, o de melhor utilizar os recursos físicos, porém visto de outro nível de abstração, o de gestão da aplicação. Existem diversas ferramentas orquestradoras de contêineres provenientes de diferentes comunidades e de código aberto. Algumas das mais populares as quais se destacam Docker Swarm (Figura 6), Kubernetes (Figura 4) e Apache Mesos pela popularidade e adoção em provedores IaaS (MELL; GRANCE et al., 2011; HAMMADI; MHAMDI, 2014). A energia consumida pelo gerenciamento de *cluster* e de seu controlador de réplica de contêineres, é minimizada quando implantadas as ferramentas orquestradoras, visto o modo eficiente que trabalham. Além de reduzirem os custos de gerenciamento e comunicação.

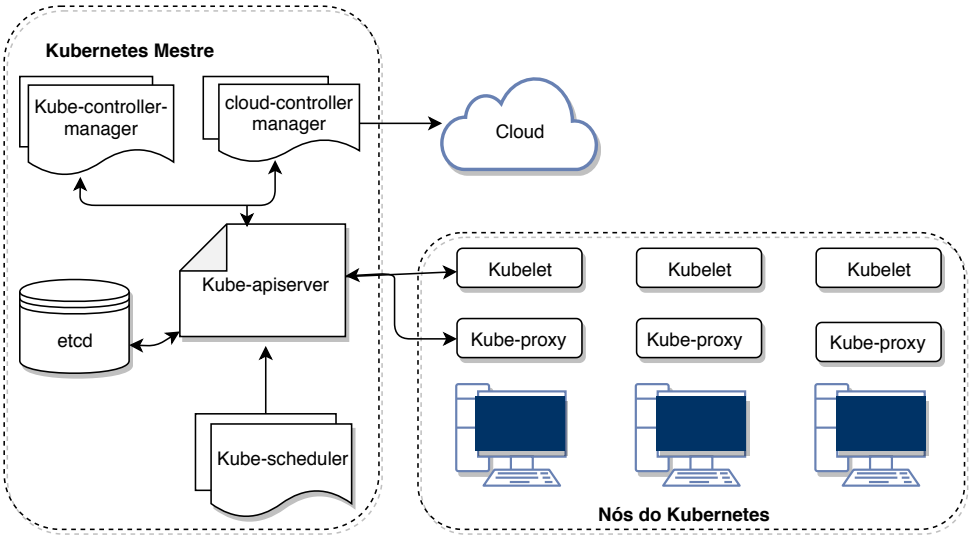
3.2.1 Kubernetes

O Kubernetes é um sistema de código aberto para gerenciar *cluster* de contêineres. Amplamente utilizado em nuvens computacionais (*e.g.,* AWS, Azure), não opera em nível de hardware, mas na camada de orquestração. É uma plataforma de orquestração de contêineres, a qual fornece ferramentas para implantar aplicativos, gerenciar cargas de trabalhos e aplicativos baseados em contêineres. Fornece uma estrutura para execução de sistemas distribuídos de maneira resiliente, e é responsável pelo dimensionamento e *failover* das aplicações.

Ao implantar o Kubernetes, obtém-se um *cluster* constituído por um ou mais conjuntos de máquinas (*i.e.,* chamadas nós), os quais executam aplicativos em contêineres gerenciados pelo Kubernetes. Os nós do *cluster* hospedam os *Pods* (*i.e.,* um conjunto de contêineres dentro de uma mesma máquina, seja ela física ou virtual). Ainda, um *cluster* possui ao menos um nó de trabalho e um nó master. Os nós mestres gerenciam os nós de trabalho e os *Pods* no *cluster*. A Figura 4 apresenta os componentes os quais complementam um *cluster* Kubernetes.

Os componentes do nó mestre são responsáveis pelo gerenciamento e manutenção do *cluster*, e podem ser executados em qualquer máquina do *cluster*. Propor-

Figura 4 – Arquitetura Kubernetes



Fonte: Adaptado de (KUBERNETES, 2020)

cionam o plano de controle, e executam as tomadas de decisões (*i.e.*,agendamentos, detecção de falhas, inicialização de *Pods*, entre outos). Para executar essa funções, o nó mestre conta com os seguintes componentes: (I) *kube-apiserver*, o qual é o *front end* do plano de controle do Kubernetes, este componente é a principal implementação de um servidor de API do Kubernetes. É possível executar diversas instâncias do *kube-apiserver* com equilíbrio de tráfego entre as mesmas. (II) *etcd*, é o repositório dos dados do *cluster*, armazena valores-chaves e de alta disponibilidade. (III) *kube-scheduler*, responsável por monitorar os *Pods* recém criados e designar a estes os nós. (IV) *kube-controller-manager*, responsável por executar os controladores de nó, replicação, pontos finais e conta de serviço e controladores de *token*. (V) *cloud-controller-manager*, é o gerenciador responsável por executar os controladores (*i.e.*, controlador de nó, rota, serviço e de volume), os quais interagem com os provedores de nuvem do controle da nuvem subjacentes.

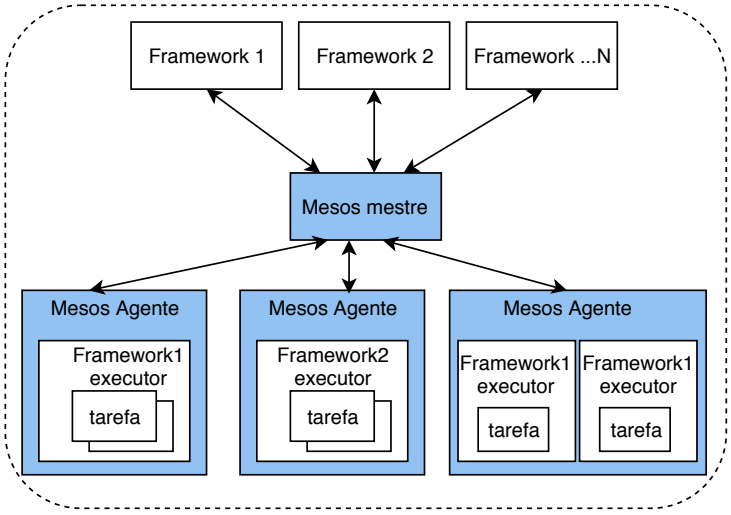
Os nós do Kubernetes são constituídos por três componentes, o *kubelet*, o *kube-proxy* e o *Container Runtime*. O *kubelet* é um agente executado em cada nó, para garantir que os contêineres estejam executando em um *pod*. O *kube-proxy* funciona como um proxy de rede dos nós, permitindo a comunicação de rede com os seus *Pods*. Já o *Container Runtime* é o software responsável pela execução de contêineres.

3.2.2 Apache Mesos

O Apache Mesos é uma plataforma de código aberto, um gerenciador de *cluster* o qual fornece isolamento e compartilhamento de recursos entre aplicativos e estruturas distribuídas. Visa fornecer um núcleo escalável e flexível para implantar e ge-

renciar aplicativos em ambientes em *cluster* de grande escala de maneira eficiente. O Mesos proporciona melhora no uso do *cluster*, e reduz a replicação de dados por estrutura computacional por meio do compartilhamento. Apache Mesos é executado em todas as máquinas do *cluster*, o mesmo realiza a abstração dos recursos e por fornecer uma API para aplicativos de gerenciamento e agendamento de recursos em todo o *cluster*. Os recursos (*i.e.*, CPU, memória, entre outros), são compartilhados de modo refinado entre as estruturas, possibilitando a localização dos dados armazenados em cada máquina. O Mesos utiliza um mecanismo de agendamento distribuído em dois níveis (*i.e.*, chamado de ofertas de recursos), a fim de suprir a demanda dos compartilhamentos requintados das estruturas atuais.

Figura 5 – Arquitetura Apache Mesos



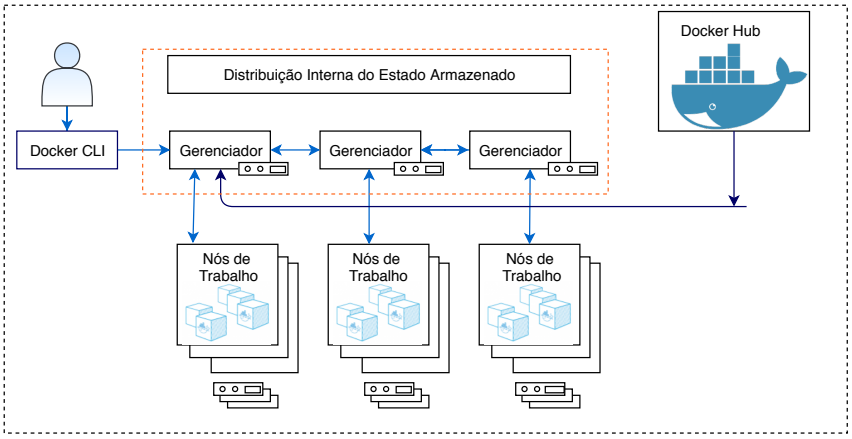
Fonte: Elaborado pela autora (2020).

A Figura 5 representa a arquitetura do apache Mesos, elencando seus principais componentes. O Mesos consiste em um *daemon* mestre o qual gerencia os *daemons* dos agentes em execução em cada nó do *cluster*, e também gerencia as estruturas do Mesos em que executam tarefas nos agentes. Cada estrutura em execução possui um processo planejador registrado no mestre, e um processo executor iniciado no agente para execução das tarefas da estrutura. O mestre utiliza a oferta de recursos para implementar o compartilhamento refinado entre as estruturas. Determina a quantidade de recursos a serem fornecidos a cada estrutura, enquanto que as estruturas decidem quais recursos serão aceitos e quais execuções serão efetuadas. Ao aceitar os recursos fornecidos, a estrutura envia ao mestre a descrição das tarefas que deseja iniciar, e o mestre envia as tarefas para o agente executar.

3.2.3 Docker Swarm

O Docker Swarm, diferente dos orquestradores Kubernetes e Mesos apresentados anteriormente, não é uma ferramenta de código aberto. É uma ferramenta orquestradora desenvolvida pelo Docker, a qual fornece uma solução de orquestração nativa de contêineres em *clusters* (DOCKER, 2016). No entando, também pode ser executado em diversas plataformas. O Docker Swarm é composto por nós de gerente e nós de trabalho apresentados na Figura 6. O nó gerente é responsável por despachar as tarefas, gerenciar e orquestrar os contêineres do *cluster*. Os nós de trabalho executam todas as tarefas recebidas através do nó gerente. O Docker *cli* também apresentado na Figura 6, representa o cliente em nível de usuário o qual interage com o *cluster*. Já o Docker Hub é responsável por suportar os repositórios disponíveis para baixar as imagens para os contêineres.

Figura 6 – Arquitetura Docker Swarm.



Fonte: Adaptado de (DOCKER, 2020).

O Swarm possibilita a utilização de múltiplas máquinas para executar serviços escaláveis com alta disponibilidade e gerenciamento. Ao iniciar uma máquina *Docker host* como nó gerente, um *cluster* é iniciado. Um *cluster* swarm é composto por diversos nós os quais possuem o Docker Engine implantados. Os nós gerentes são responsáveis pelo gerenciamento do *cluster* e a orquestração dos contêineres. Os nós de trabalho recebem e executam as tarefas advindas dos nós gerentes. Possui três estratégias para realizar o escalonamento dos contêineres (*i.e.,* de espelhamento, *binpac* e aleatório). A estratégia de espelhamento define a máquina com o menor número de contêineres em execução. O *binpac*, escolhe a máquina com menos recursos (*i.e.,* CPU e memória) disponíveis. Já a estratégia randômica escolhe a máquina aleatoriamente. O Docker Swarm é considerado uma plataforma de orquestração leve, entretanto o mesmo não garante o agrupamento de contêineres em um mesmo Docker *host*.

Os orquestradores de contêineres apresentados nesta seção fornecem e dis-

ponibilizam de modo eficiente, recursos físicos que são compartilhados com seus usuários. Desse modo, é necessário analisar qual destas tecnologias se adequa ao objetivo buscado para o cenário de implementação (*i.e.,* desempenho, impacto ao meio ambiente, nuvens verdes). Este trabalho, visa identificar o impacto do consumo energético em nuvens computacionais, por este motivo é necessário levantar as métricas que impactam diretamente neste consumo. Desse modo, a Seção 3.3, elenca as métricas a serem compreendidas no processo de identificação do impacto energético.

3.3 CONSUMO ENERGÉTICO

A aplicabilidade de orquestradores e contêineres em ambientes de nuvens computacionais é um fato, sendo que a adoção destas tecnologias ocorrem entre outras características, pelo modo eficiente de utilização dos recursos, impactando diretamente na minimização dos custos em um DC, o qual possui custos variáveis, fixos, diretos e indiretos na manutenção de sua infraestrutura. O consumo energético é um dos principais custos variáveis dentre todos as variáveis presentes no CTP dos DCs, representando até 50% dos custos operacionais (COMERFORD, 2015). Portanto, a precificação dos recursos computacionais alocados sob demanda é impactada diretamente pelo consumo energético da aplicação (HINZ et al., 2018).

Monitorar o consumo energético é fundamental para compreender a relevância do mesmo sob o custo total dos serviços ofertados pelos provedores IaaS. Desse modo, é possível identificar o gasto ponderado do consumo energético, aplicar técnicas de redução e propor um modelo de custo que considera o custo energético na precificação dos recursos computacionais. Tornar o consumo energético individualizado da aplicação em um elemento no modelo de custo motiva o cliente da nuvem computacional, a otimizar sua aplicação em busca de recompensa financeira ou apelo ambiental. A concepção de modelos de custos baseado no consumo de insumos, custos variáveis do CTP, exige a compreensão do impacto dos componentes envolvidos no modelo. Ao analisar um ambiente em que existe a orquestração de contêineres, a identificação dos consumos energéticos de cada contêiner e dos gastos advindos do mecanismo de orquestração é de suma importância. Os principais recursos da computação em nuvem observados que influenciam no gasto de energia em um DC foram destacados por (LI et al., 2017), consistindo em processamento, memória, comunicação e armazenamento.

Processamento de CPU é um recurso computacional influente no consumo de energia. O consumo energético resultante do uso de CPU é considerado uma função exponencial, quanto maior a carga de trabalho processada, maior a energia consumida pelo servidor (TANG; DAI, 2011; VISHWANATH et al., 2015). O número de CPUs do servidor também impacta no consumo energético, possuindo uma relação

proporcional linear (ENOKIDO; TAKIZAWA, 2015; BERGEN et al., 2014).

Memória RAM é um recurso computacional o qual permite acesso aos arquivos armazenados na memória do servidor. Os tamanhos de memória utilizada para E/S, armazenamento, compressão e descompressão de dados, influenciam no consumo de energia (SINGH; NAIK; MAHINTHAN, 2015; LIU; PINTO; LIU, 2015).

Rede é um recurso computacional utilizado para as aplicações comunicarem entre si. O tamanho da vazão de banda, e a quantidade de dados trafegados na rede, impactam no consumo de energia. No qual, quanto maior a vazão de banda e a carga trafegada na rede, maior o consumo de energia para a comunicação (YUAN; GUO; CHEN, 2015; LUO et al., 2015).

Armazenamento é um recurso computacional utilizado para armazenar dados em um servidor. A utilização de operações de E/S de dados do dispositvo de armazenamento impactam no gasto de energia (IZADPANAH et al., 2013).

3.4 CONSIDERAÇÕES PARCIAIS

As tecnologias de virtualização de contêineres possuem influência na oferta sob demanda de provedores IaaS, proporcionando um ambiente mais leve e eficiente aos seus clientes. O Docker, Rocket e LXC são tecnologias de contêineres muito utilizadas pelas organizações e estão em constante desenvolvimento. Porém a utilização de aplicações conteinerizadas em larga escala inviabiliza o controle manual das instâncias, tornando-se necessário o uso de orquestradores realizarem o gerenciamento e escalonamento dos contêineres. Por este motivo, os orquestradores são tecnologias de suma importância e influência para os contêineres. Responsáveis pelo gerenciamento de clusters hospedeiros de contêineres, bem como pelo escalonamento de seus processos conteinerizados. Existem diversas soluções orquestradoras de contêineres, neste capítulo apresentou-se tecnologias populares as quais se destacam *i.e.,* Kubernetes, Apache Mesos e Docker Swarm).

Além da importância das tecnologias de contêineres e orquestradoras as quais viabilizam aos provedores IaaS e seus clientes a facilidade, desempenho e agilidade, um componente de grande influência para os DCs é a energia, a qual representa em até 50% dos custos operacionais que compõem o CTP.

4 ANÁLISE DE CONSUMO ENERGÉTICO

O presente capítulo apresenta análises de experimentos executados neste trabalho de mestrado, abordando o gasto energético da utilização de recursos computacionais (*i.e.,* CPU, memória, armazenamento e rede), em diferentes ambientes (*i.e.,* contêineres, MV e *bare metal*). O objetivo destas análises é estabelecer uma linha de base para comparar e quantificar o impacto destes recursos computacionais em diferentes ambientes (*i.e.,* contêineres, MV e *bare metal*). Os experimentos executados nesta pesquisa são detalhados na Seção 4.1, caracterizando os testes executados e os ambientes explorados. Os resultados obtidos na experimentação foram abordados nas Seções 4.2, 4.3, 4.4 e 4.5.

4.1 PROTOCOLO EXPERIMENTAL

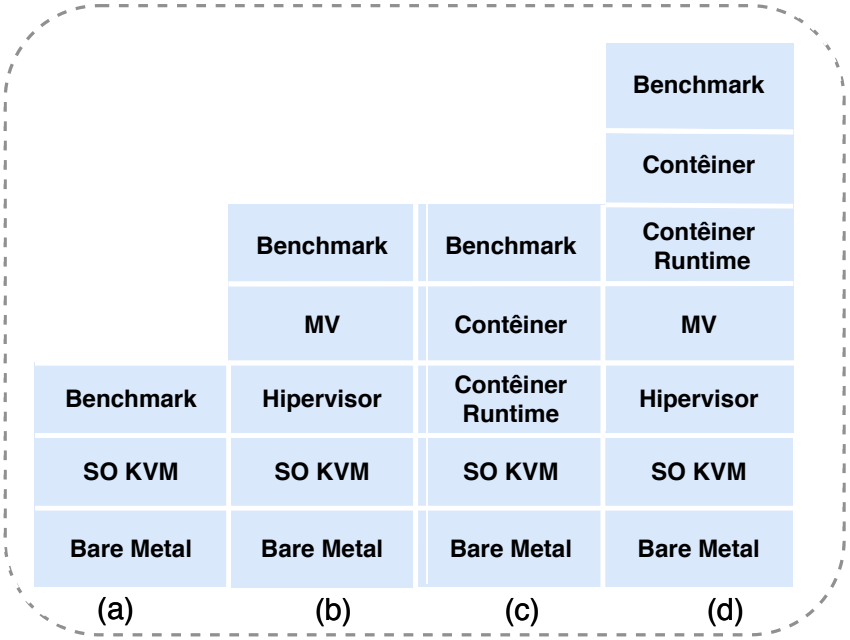
Para a execução dos experimentos dessa proposta, o ambiente experimental utilizado foi a plataforma computacional Grid5000[1], a qual é uma plataforma computacional virtualizada flexível e de larga escala utilizada para experimentos de pesquisas relacionadas a área da ciência da computação. Atualmente, a Grid5000 consiste em oito sites, com os experimentos da proposta executados no site de Grenoble [2]. Dentro do site, o cluster selecionado foi o *yeti*, formado por quatro computacionais *Dell PowerEdge R940*, somando 128 núcleos e 754 GB de memória. O cluster *yeti* é homogêneo, consistindo de nós com processadores *Intel Xeon Gold 6130 (Skylake, 2.10GHz, 4 CPUs/nós, 16 núcleos/CPU)*, e rede *Intel Ethernet Controller X710 for 10GbE SFP+*. Além disso, wattímetros (dispositivos externos não intrusivos) são responsáveis por monitorar o consumo de energia do *yeti*. Na camada de software, os 3 ambientes usados nos experimentos são: (i) instalação *bare metal* do *Debian 10 Linux*; (ii) hipervisor kvm *3.1.0*; e (iii) Docker *19.03*.

A análise do comportamento de consumo de energia dos servidores foi realizada por meio de 4 benchmakrs, em que cada um é responsável por aplicar cargas de trabalho em diferentes recursos do servidor. Os benchmarks são: (*I*) o **StressNG** é responsável por aplicar carga de trabalho configurável á CPU do servidor (KING, 2019); (*II*) o **Stream** realiza cálculos vetoriais que permitem saturar a memória e CPU do servidor. O usuário pode definir a quantidade de memória, porcentagem de CPU ou número de processos a serem alocados (MCCALPIN et al., 1995); (*III*) **Flexible I/O** realiza a sobrecarga de armazenamento do servidor, executando *threads* assíncronas realizando leitura e escrita assíncronas (AXBOE, 2017); e (*IV*) **iPerf3** calcula a vazão

[1] <http://grid5000.fr>
[2] <https://www.grid5000.fr/w/Grenoble:Hardware>

de banda de uma rede, e executa fluxos de comunicação paralela entre o servidor e o cliente (MORTIMER, 2018). Estes benchmarks permitem observar o comportamento dos recursos físicos que influenciam diretamente no consumo de energia do DC (LI et al., 2017), (*i.e.,* CPU, memória, armazenamento e rede). O bjetivo é quantificar o

Figura 7 – Esquema do Ambiente de Experimentação.



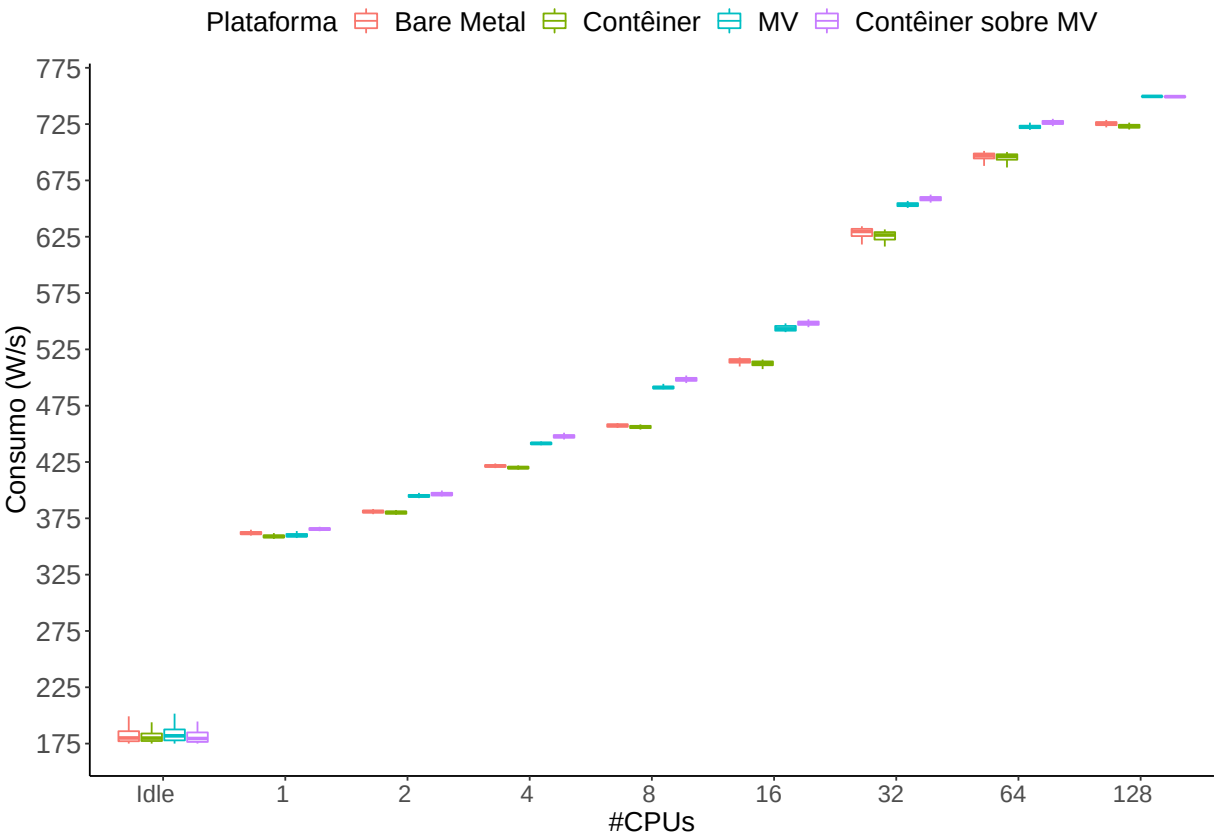
Fonte: Elaborado pela autora (2020).

consumo de energia dos recursos físicos associados a cada recipiente ativo. A linha de base do cenário de experimentação é *bare metal* (Figura 7 (a)). O consumo de energia associado a cada recurso físico na linha de base é comparado com três outros cenários: (i) MV, (ii) contêiner Docker e (iii) contêiner Docker sobre MV, conforme mostrado na Figura 7 (b) - (d). Cada benchmark foi realizado com a mesma carga de trabalho em cada ambiente por 10 vezes cada, para reduzir o possível ruído ao capturar o consumo de energia.

4.2 ANÁLISE DE CPU

O primeiro recurso físico analisado é a CPU (Figura 8), o maior consumo de energia, no pior caso, ~ 750 $Watts/s$ no cluster Yeti. O objetivo da análise da CPU é mostrar o comportamento do consumo de energia de acordo com o aumento no uso da CPU. A linha de base do consumo de energia do servidor, neste caso, é o consumo ocioso. As cargas de trabalho da CPU são alteradas, iniciando observações com consumo de energia do servidor ocioso e aumentando o número de *threads* de 1 até 128 em todos os 4 cenários de (Figura 7). Stress NG (KING, 2019) linha de comando utilizada é *./stress-ng –cpu CPU –timeout TOTAL_TIME;*.

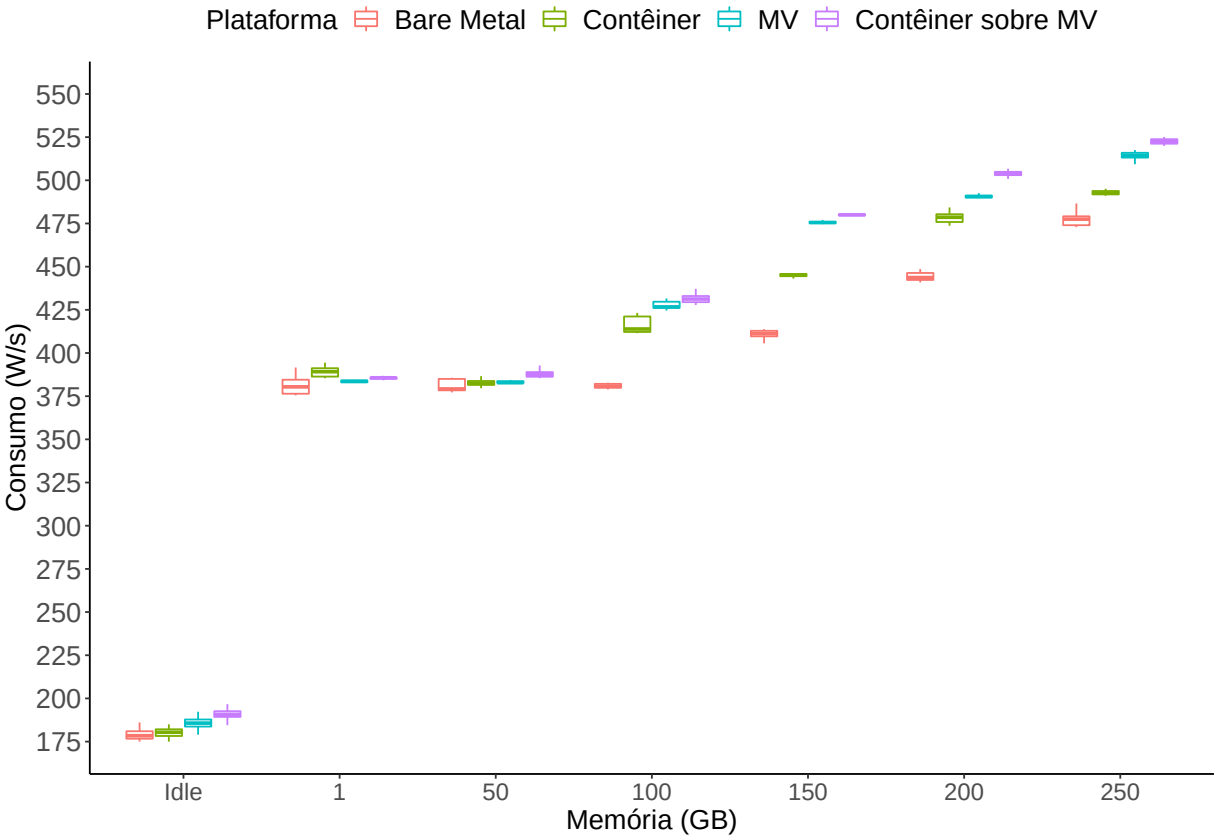
Figura 8 – Consumo de energia de recurso físico de CPU.



Fonte: Elaborado pela autora (2020).

A Figura 8 apresenta os resultados obtidos, no qual o *eixo x* representa o número de *threads* atribuídas ao benchmark, e o *eixo y* representa o consumo de energia do servidor ($Watts/s$). Neste experimento, o impacto do uso da CPU no aumento do consumo de energia pode ser comprovado. Na linha de base do consumo de energia do servidor (servidor ocioso), todos os cenários têm o mesmo comportamento de consumo. Ao comparar o servidor inativo com a utilização da CPU, os resultados mostram um crescimento de 113,29% com 1 CPU, enquanto alcançava 342,86% com 128 CPUs. Esta análise também permite verificar as sobrecargas da plataforma, mostrando que a MV e o contêiner sobre MV consomem mais energia do que *bare metal* e contêiner. O contêiner sobre MV consome um pouco mais que a MV, exceto quando o servidor está saturado, neste caso, ambos têm o mesmo nível de consumo energético. A lacuna de consumo de energia entre as plataformas piora à medida que o uso da CPU aumenta. Este comportamento se deve ao gerenciamento realizado pelo hipervisor KVM. Com apenas 1 CPU, a diferença no consumo entre os ambientes é relativamente insignificante (*e.g.*, menos de 10 $watts$ ou 1,81%). A maior diferença está em 8 CPUs, com 9,26%. Finalmente, a análise do consumo de energia da CPU permite concluir que a CPU é um componente essencial para um modelo de custo.

Figura 9 – Consumo de energia de recurso físico de memória.



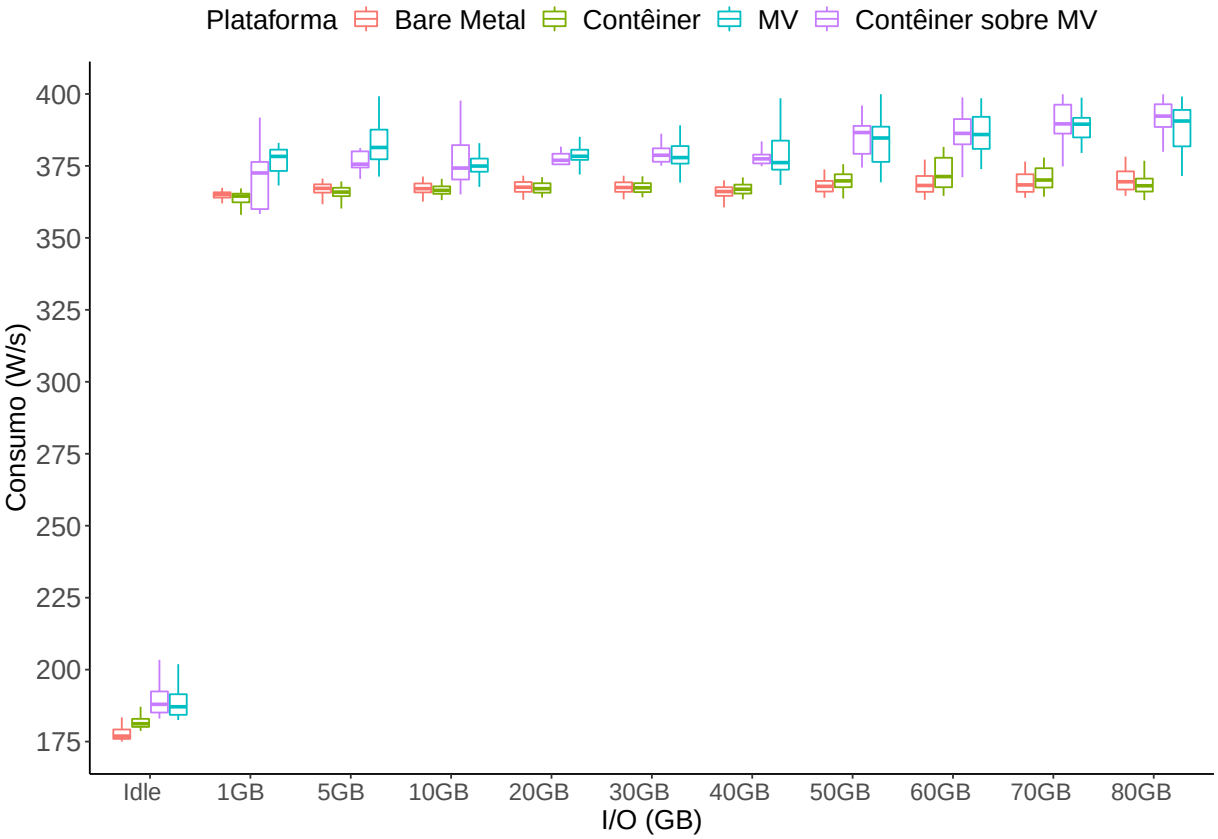
Fonte: Elaborado pela autora (2020).

4.3 ANÁLISE DE MEMÓRIA

A curva de consumo de energia do recurso de memória é semelhante ao recurso de CPU, porém sua faixa de consumo de energia é menor (entre ~ 175 e $sim525$ $Watts/s$). Neste experimento, existem 7 cargas de trabalho de memória que são processadas por apenas 1 *thread* e aplicadas em todos os cenários predefinidos (Figura 9). Stream (MCCALPIN et al., 1995) é o benchmark de memória escolhido para projetar o comportamento do consumo energético da memória. Novamente, a linha de base do consumo de energia é um servidor ocioso. A linha de base é comparada a 6 outras demandas de memória, *e.g.,* 1 GB, 50 GB, 100 GB, 150 GB, 200 GB e 250 GB.

A Figura 9 mostra os dados obtidos no experimento, no qual o *eixo x* corresponde à carga de trabalho de memória aplicada pelo benchmark, enquanto o *eixo y* indica a energia consumida do servidor. O primeiro estágio já pode ser observado com demanda de memória de 1 GB, onde o consumo de energia aumenta para ~ 375 $Watts$ em todos os cenários. No próximo experimento, onde a demanda de memória é de 50 GB, o consumo de energia permaneceu o mesmo. Para os outros experimentos, o consumo de energia aumentou em média 6,95% a cada 50 GB.

Figura 10 – Consumo de energia de recurso físico de armazenamento.

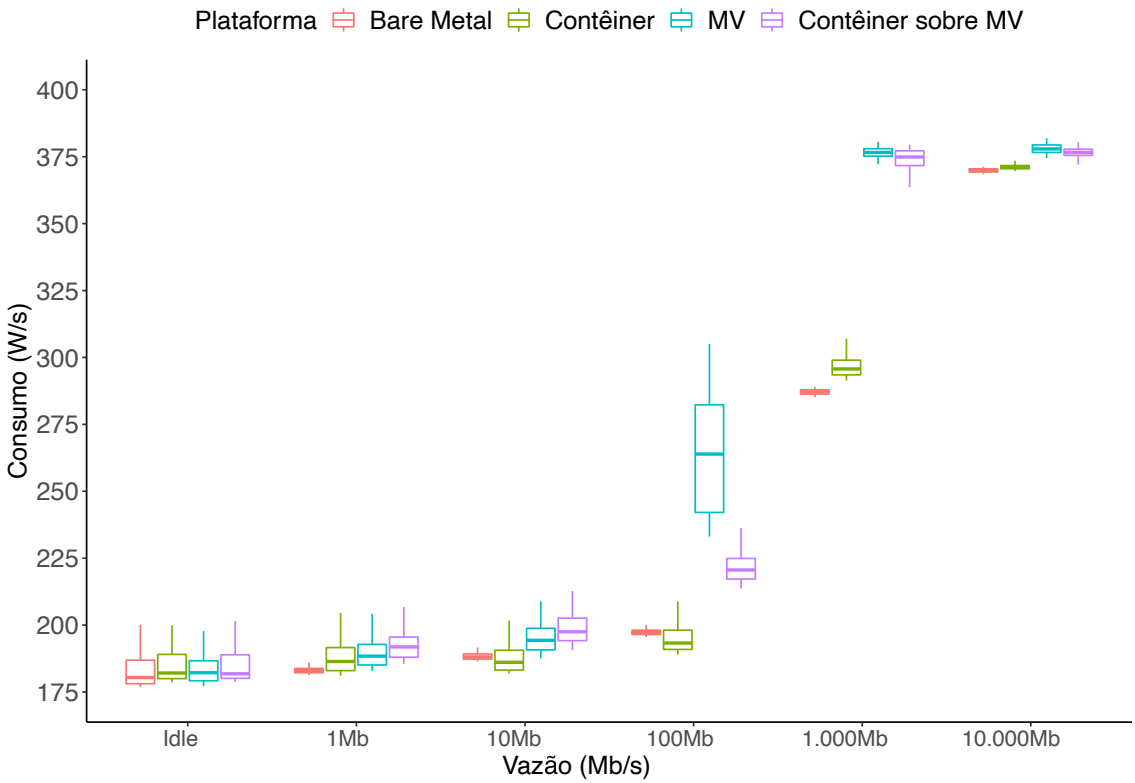


Fonte: Elaborado pela autora (2020).

Ainda, é possível observar que o gerenciamento de memória é pesado para todos os ambientes virtuais. Quando a curva de aumento do cenário *bare metal* é comparada a outras, a lacuna é mais acentuada do que a observada com recurso de CPU. Nem mesmo o contêiner tem o consumo energético compatível com o *bare metal*. No pior caso, ao comparar o consumo de energia entre o servidor ocioso e o contêiner sobre *MV* com 250 GB de demanda de memória, o consumo de energia aumenta para 214%. A análise do consumo de energia, descrita nesta seção, permite afirmar que o gerenciamento da memória é um fator de impacto no consumo de energia. Portanto, um modelo de custo baseado no consumo de energia precisa considerar o recurso de memória.

Para aferir o consumo de memória dos ambientes abordados na Figura 1, utilizou-se o benchmark *Stream* para mensurar o comportamento do consumo energético frente a diferentes cargas de utilização de memória RAM. Os experimentos foram executados utilizando apenas 1 thread, com um conjunto de 7 configurações de cargas de trabalho, em *idle*, com $1GB$, $50GB$, $100GB$, $150GB$, $200GB$ e $250GB$. Os dado obtidos são apresentados na Figura 9, onde o eixo x apresenta a carga de trabalho de memória aplicado pelo benchmark, enquanto o eixo y apresenta o consumo de energia do ser-

Figura 11 – Consumo de energia de recurso físico de Rede.



Fonte: Elaborado pela autora (2020).

vidor.

4.4 ANÁLISE DE ENTRADA E SAÍDA

O objetivo da análise de armazenamento é quantificar o consumo de energia do recurso de armazenamento local, neste caso, o disco rígido. *Flexible IO* (AXBOE, 2017) é o benchmark de armazenamento escolhido para projetar o comportamento de armazenamento de consumo de energia e, o servidor inativo permanece a linha de base. As demandas de armazenamento variam entre 1 GB e 80 GB em todos os casos, o benchmark é processado por apenas 1 *thread*.

A Figura 10 mostra o comportamento do consumo de energia com o armazenamento. No *eixo x*, existem 10 demandas de armazenamento, enquanto o *eixo y* mostra o consumo em $Watts/s$. Os resultados mostram que o consumo de energia de armazenamento é estável de 1 GB a 80 GB para todos os cenários. O coeficiente de variação é de cerca de $8,4\%$. Esse comportamento permite concluir que o aumento da demanda de armazenamento não impacta no consumo de energia. Por outro lado, quando o consumo de energia do servidor ocioso se compara a qualquer demanda de

armazenamento, o consumo de energia de crescimento é superior a 105%. Portanto, é possível afirmar que as demandas de armazenamento precisam ser consideradas para um modelo de custo, no entanto, o nível de demanda de armazenamento não importa.

4.5 ANÁLISE DE REDE

A última análise de impacto de energia no recurso físico é da rede. O gerenciamento da rede varia de acordo com o tipo de rede, ou seja, *host*, *bridge*, *overlay* e *macvlan*. Em ambientes virtualizados, o gerenciamento de rede pode se comportar de forma diferente e o consumo de energia também (MORABITO, 2015). Neste trabalho, todos os cenários são definidos para o tipo de rede *bridge*. O iPerf3 (MORTIMER, 2018) é o benchmark de rede escolhido para projetar o comportamento da rede de tráfego de consumo de energia e, o servidor ocioso permanece novamente a linha de base. O tráfego de rede precisa de dois nós, um nó servidor e outro nó cliente. Os experimentos realizados para medir o consumo de energia no processamento da rede variam de 1 MB, 10 MB, 100 MB, 1 GB e 10 GB, para entender a influência da vazão de banda processada na energia consumida.

A Figura 11 apresenta o consumo de energia observado durante a execução do experimento. No *eixo x* contém as configurações de vazão de banda aplicadas no benchmark, enquanto o *eixo y* representa o consumo de watts por segundo consumido pelo servidor. Os resultados (Figura 11) indicam que há um início com aumento suave, quando a vazão de banda usada pelo aplicativo é inferior a 10 Mbps (28%). Porém, quando o consumo de vazão de banda ultrapassa 100 Mbps, há uma alta no consumo de energia, chegando a 117% se comparado ao consumo do servidor ocioso.

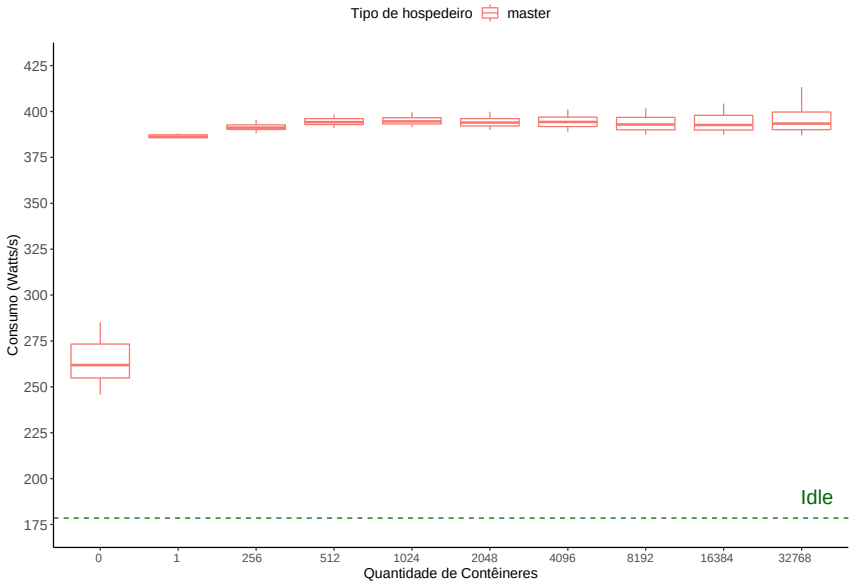
Portanto, é possível concluir que considerar a vazão de banda utilizada é um componente necessário no modelo de custo energético. E deve ser considerado em conjunto com a quantidade de dados transferidos, permitindo assim um refinamento no cálculo do consumo de energia.

4.6 ANÁLISE DE ORQUESTRAÇÃO DE CONTÊINERES

As análises anteriores neste capítulo descrevem o comportamento dos recursos físicos. No entanto, os ambientes de nuvem, compostos de centenas ou milhares de contêineres, são sistemas de gerenciamento dependentes. A arquitetura do sistema de gerenciamento escolhido compreende um único mestre dedicado (Figura 14), e o software de gerenciamento é o Kubernetes (KUBERNETES, 2020).

Embora o Kubernetes seja um serviço remoto, o objetivo deste capítulo é analisar como as demandas do cliente afetam o comportamento do consumo energético

Figura 12 – Consumo de energia do orquestrador do Kubernetes.



Fonte: Elaborado pela autora (2020).

do mestre do Kubernetes. A Figura 12 mostra esse comportamento, o *eixo x* indica o número de contêineres em execução, gerenciados pelo Kubernetes mestre e no *eixo y* mostra o consumo de energia em *Watts/s* produzidos por este gerente.

A ativação do orquestrador corresponde a cerca de 90 *Watts/s* de consumo de energia. A segunda etapa fica evidente quando a primeira demanda do consumidor chega. Entre 1 e 32.768 contêineres, o consumo de energia permanece em torno de 390 *Watts/s*. O consumo energético para manter ativo um orquestrador de gerenciamento de contêineres precisa ser considerado em um modelo de custo, no entanto, a análise de comportamento mostra que esse custo deve ser dividido por todos os contêineres gerenciados.

4.7 CONSIDERAÇÕES PARCIAIS

Este capítulo apresentou a análise dos experimentos executados na presente pesquisa de mestrado, a qual compara a eficiência energética dos ambientes computacionais apresentados na Figura 1. Além de abordar os benchmarks usados nos experimentos para os recursos computacionais de CPU, memória, I/O e rede. O consumo de energia, a tecnologia e o benchmark usados no experimento do orquestrador de contêineres também foi apresentado e analisado. Por meio das análises, é possível concluir que os recursos computacionais apresentados (*i.e.,* CPU, memória, I/O, rede e orquestrador de contêiner), devem ser considerados no modelo de custo energético.

5 MODELO DE CUSTO PARA NUVEM: CONTÊINERES & CONSUMO ENERGÉTICO

Nos Capítulos 2 e 3, discutiu-se a motivação, tecnologias e orquestradores de contêineres, componente energético o qual compõe os custos do CTP de serviços de nuvem, e a tendência no uso de contêineres. A relevância do gasto energético na composição de custos de um CTP em nuvem IaaS é indiscutível e, a necessidade de considerar esse gasto na modelagem da precificação de modelos de serviços apoiados em contêineres também se torna essencial (BRONDOLIN; SARDELLI; SANTAMBROGIO, 2018). No Capítulo 4, é abordada a análise da experimentação realizada nesta pesquisa em relação ao consumo de energia de recursos físicos, no qual o servidor dedicado tem seus recursos saturados. O qual confirma a relevância desta proposta ao considerar o impacto do consumo energético ao utilizar os recursos computacionais. A principal contribuição desse trabalho é a descrição da proposta de modelo de custos para nuvens IaaS, o qual este modelo considera o consumo energético de contêineres. A precificação de serviços da nuvem, baseada no consumo de insumos, incentiva aos clientes da nuvem otimizarem suas aplicações, pois são recompensados financeiramente por isso. A Tabela 2 descreve os elementos que compõem o preço dos recursos computacionais usados por contêineres em um provedor de nuvem. Desse modo, é necessário analisá-los para um melhor entendimento do modelo aqui proposto. Por sua vez, P_w representa o preço cobrado pela distribuidora de energia elétrica do provedor de nuvem, a qual tributa o provedor por watts consumidos e de acordo com a região em que a energia é fornecida. W_i descreve o consumo de watts produzido por um servidor, para manter o mesmo ligado em *idle* (*e.g.,*, ocioso, sem aplicações em andamento, apenas disponível para que contêineres sejam executados). C_i é o custo mínimo de energia do servidor, a despesa de energia que o provedor tem para mantê-lo ativo e disponível para execução de contêineres. Cada máquina possui um C_i específico para estar ligada, os fatores de variância deste elemento são as características específicas de cada hardware (*i.e.,* tecnologias utilizadas, fontes de alimentação, processadores, placas, drives, armazenamento dentre outros). Por este motivo, o custo de C_i muda dependendo da plataforma em que os contêineres são executados. Assim, cada servidor específico possui um custo fixo em inatividade, e deve ser distribuído através do rateio proporcional entre clientes e contêineres alocados no servidor. Portanto, a Equação 5.1 define o custo para manter o servidor ativo.

$$C_i = P_w.W_i \tag{5.1}$$

No entanto, para a parcela proporcional de C_i, é necessário o rateio de W_i. Esta repartição consiste em 5 elementos: (i) Tx_u, uma taxa pela utilização de recursos computacionais do servidor, usada para quantificar a porcentagem dos recursos alocados

Tabela 2 – Notação relacionada ao consumo energético de contêiner em um período de tempo.

Notação	Descrição
P_w	Preço por watts consumido.
W_i	Consumo de Watts do servidor de contêineres em *idle*.
C_i	Custo de energia para manter o servidor ativo em *idle*.
Tx_u	Taxa de utilização dos recursos computacionais disponíveis em um servidor.
Tx_c	Taxa de complemento de recursos computacionais disponíveis em um servidor.
$\|Cont_u\|$	Quantidade de contêineres por clientes em um servidor.
G_p	Custo do provedor por manter recursos disponibilizados e não alocados.
$\|Cont_s\|$	Quantidade de contêineres por servidor.
$f(x)$	Função para análise contínua da taxa de utilização de CPU.
$f_2(x)$	Função para análise do custo da utilização de CPU.
P_{cpu}	Preço do consumo energético de CPU.
W_{cpu}	Watts consumido pelo recurso de CPU do contêiner.
C_{cpu}	Consumo de energia relacionado ao uso da CPU física.
$U_{cpu}(t,c)$	Utilização de CPU de um contêiner específico c.
W_{mem}	Watts consumidos pelo recurso de memória do contêiner.
C_{mem}	Custo de energia relacionado ao uso de memória.
(t,c)	Dado instante t de um contêiner específico c .
$U_{mem}(t,c)$	Utilização de memória de contêiner.
fvc	Fator de variação do consumo por contêineres alocados no servidor.
C_{rede}	Custo de energia relacionado às operações de rede.
$U_{rede}(t,c)$	Utilização de rede de um contêiner.
W_{rede}	Watts consumidos pelo recurso de rede do contêiner.
P_{disk}	Preço de armazenamento.
Q_{disk}	Quantidade de armazenamento utilizada .
W_{orq}	Watts consumidos pelo orquestrador de contêineres.
C_{orq}	Custo de energia relacionado ao orquestrador de contêineres.
$Cost$	Custo de energia de um contêiner em um período de tempo.
C_{total}	Custo de energia relacionado a utilização de recursos por contêineres de um dado cliente..

Fonte: Elaborado pela autora (2020).

por clientes que executam contêineres em um servidor; (ii) Tx_c, uma taxa para complementar os recursos computacionais do servidor, aplicada para isentar o cliente de incorrer em um custo maior do que o alocado. Esta taxa equaliza o percentual de alocação de recursos, atribuindo ao provedor de nuvem a responsabilidade de gerenciar e alocar clientes nos servidores. De modo que, um único cliente alocado em um servidor não reservar os recursos do servidor como um todo, o consumo de W_i não deve ser atribuído 100% para este cliente, pois não corresponde com sua utilização. Assim, o provedor paga por um percentual que chamamos de Tx_c, por ser o responsável por este gerenciamento. Portanto, no caso da taxa Tx_u atribuída aos clientes alocados na plataforma for inferior a Tx_c, o rateio dos custos é feito entre clientes e também entre o provedor; (iii) outro elemento que compõe W_i é G_p, um custo de gerenciamento atribuído ao provedor devido à sua responsabilidade de dimensionar e gerenciar os recursos computacionais do servidor. Já que manter uma máquina específica dispo-

nível é considerado um custo fixo; (iv) $|Cont_u|$ é o elemento o qual contabiliza e atribui o número de contêineres alocados por cada cliente do servidor; E, finalmente, (v) $|Cont_s|$ contabiliza e identifica por unidade o número de contêineres alocados no servidor. A Equação 5.2 trata desse rateio.

$$W_i = \begin{cases} Tx_u < Tx_c, & \text{então} \\ |Cont_u|.\left(\frac{G_p.\left[100+(Tx_u-Tx_c)\right]}{|Cont_s|}\right), & \text{se não} \\ |Cont_u|.\left(\frac{G_p}{|Cont_s|}\right) \end{cases} \tag{5.2}$$

A Equação 5.2 verifica se a taxa de utilização Tx_u, é menor que a taxa de complemento Tx_c. Assim, o rateio deste custo ocorre entre: o número de contêineres por clientes ($|Cont_u|$); o número de contêineres por servidor ($|Cont_s|$); e entre o provedor de nuvem (G_p). E se, Tx_u for maior que Tx_c, então este custo é distribuído apenas entre $|Cont_u|$ e $|Cont_s|$.

A partir da identificação do custo do servidor em *idle*, é necessário entender os custos de recurso utilizado pelos contêineres (*e.g.,* CPU, memória, rede, armazenamento e de orquestração). O recurso de CPU impacta diretamente sobre o consumo energético de um DC, além de ser o recurso que demanda mais energia do servidor (HSU et al., 2014). A Figura 13 mostra a função $f(x)$, a qual apresenta a porcentagem de utilização da CPU no *eixo y*, em um determinado período representado no *eixo x*.

Para converter o gráfico de utilidade em um gráfico de valores, é necessário realizar a precificação discreta do uso de CPU da função $f(x)$ em um intervalo de tempo $T1 \leq t \leq T2$, no qual $T1$ é o tempo inicial, e $T2$ é o tempo final. E então multiplicar este intervalo pelo custo de 1 CPU, resultando na função $f_2(x)$ (mostrado na Figura 13).

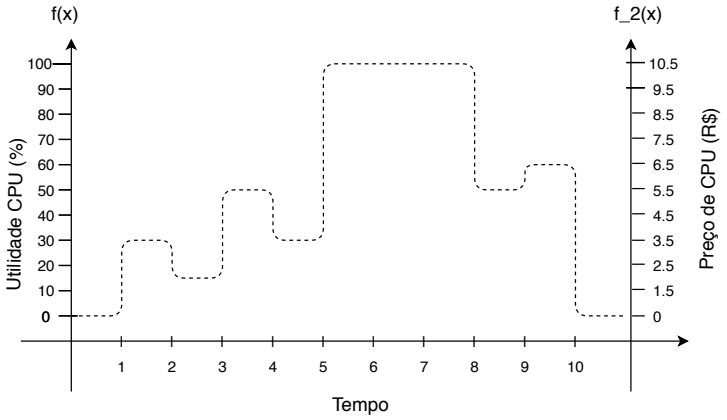
Conforme demonstrado na Figura 13, o *eixo x* à esquerda expõe a função $f(x)$ da utilização da CPU, e o *eixo y* à direita a função $f_2(x)$ do preço da CPU em um instante t no eixo x. Este gráfico resulta na Equação 5.3.

$$f_2(x) = f(x).P_{cpu} \tag{5.3}$$

Por sua vez, o preço de CPU é dado pelo elemento P_{cpu}, o qual representa o preço pago à distribuidora de energia pelo consumo da CPU do servidor. Este elemento consiste em W_{cpu} elemento o qual identifica a quantidade de watts consumido pelo uso de CPU. E composto também pelos elementos P_w e C_i, ou seja pelo preço do watts e o custo do servidor em *idle*. Dado pela Equação 5.4

$$P_{cpu} = (W_{cpu}.P_w) - C_i \tag{5.4}$$

Figura 13 – Precificação de CPU



Fonte: Elaborado pela autora (2020).

A Equação 5.4 permite a formulação do custo total de CPU expresso por C_{cpu}. É composto pelo elemento $U_{cpu}(t,c)$, o qual equivaleà utilidade de CPU em um dado momento t de um contêiner específico (c), definido pela soma do intervalo de tempo $T_1 < t < T2$, e pela soma dos contêineres alocados por servidor. Portanto, o custo total da CPU é expresso na Equação 5.5

$$C_{cpu} = \left(\sum_{t=T_1}^{T_2} \sum_{c=1}^{|Cont_u|} U_{cpu}(t,c) \right).P_{cpu} \tag{5.5}$$

Seguindo a precificação dos recursos influentes energeticamente na execução de contêineres, tem-se a memória, recurso essencial em aplicações e sistemas de contêineres. O elemento W_{mem} reflete a energia do consumo de memória, identificando a quantidade de watts consumidos pelo recurso de memória. Além disso, o $U_{mem}(t,c)$ equivale a utilidade de memória em um dado instante t de um contêiner específico (c). Por meio da soma do intervalo de tempo $T_1 < t < T2$, e da soma dos contêineres alocados por servidor $|Cont_u|$. Assim, é possível identificar o custo energético da memória representado pelo elemento C_{men}, abordado na Equação 5.6, que expõe o preço da memória na execução do contêiner.

$$C_{mem} = \left(\sum_{t=T_1}^{T_2} \sum_{c=1}^{|Cont_u|} U_{mem}(t,c) \right).\left(\left[W_{mem}.P_w \right] - C_i \right) \tag{5.6}$$

A rede é outro recurso computacional energeticamente influente. O consumo de energia deste recurso varia de acordo com a vazão de banda da rede utilizada. Uma análise do elemento W_{rede}, responsável por identificar o consumo em watts que uma aplicação possui ao utilizar o recurso de rede, destaca que este recurso possui variação pouco significativa quanto a quantidade de contêineres ativos na máquina, mantendo uma variação significativa apenas em relação à vazão da rede utilizada. Portanto, para calcular o elemento C_{rede}, o qual representa o custo energético de rede,

é necessário analisar 2 elementos além de W_{rede}: (i) o $U_{rede}(t,c)$; e (ii) P_{rede}. O elemento $U_{rede}(t,c)$ responsável por identificar a utilidade da rede em um dado instante t para um contêiner específico (c), ou seja, o tamanho da vazão de banda utilizada na execução do contêiner em um determinado período, através da soma do intervalo de tempo, bem como da soma dos contêineres do mesmo servidor. E P_{rede} é composto por fvc, que se refere a um fator de variação de preço dependendo do consumo dos contêineres existentes no mesmo servidor. Portanto P_{rede} é detalhado na Equação 5.8, enquanto C_{rede} é expresso pela Equação 5.7 .

$$C_{rede} = \Big(\sum_{t=T_1}^{T_2} \sum_{c=1}^{|Cont_u|} U_{rede}(t,c) \Big).P_{rede} \tag{5.7}$$

$$P_{rede} = \Big(\Big[(W_{rede}.P_w) - C_i \Big].fvc \Big) \tag{5.8}$$

O elemento fvc é aplicado para obter P_{rede}, atende a condição abaixo expressa pela Equação 5.9. Neste contexto fvc estima a variação de consumo energético (*i.e.,* mesmo que não seja muito significativo, este trabalho entende que é relevante atribuí-la ao cliente), de acordo com o número de contêineres em execução no servidor. Assim, caso o servidor possui apenas um cliente executando seus contêineres, o fvc é 1. No entanto, caso houver mais de um cliente executando contêineres no servidor, o número de contêineres por cliente é dividida pelo número total de contêineres alocados na máquina, e então adicionado 1.

$$fvc = \begin{cases} \text{se} \quad |cont_u| = |cont_s| \quad \text{então fvc é 1} \\ \\ \text{senão} \quad \frac{|cont_u|}{|Cont_s|} + 1 \end{cases} \tag{5.9}$$

O próximo recurso computacional a ser analisado energeticamente é o recurso de armazenamento. A análise mostra que o recurso é constante em termos de consumo energético quando há apenas um contêiner executando no servidor, independente do tamanho da entrada e saída de disco. No entanto, ao executar mais de um contêiner por servidor, o consumo de energia do armazenamento é variável. Assim, o custo de armazenamento dado pelo elemento C_{disk} (descrito na Equação 5.10), é composto por U_{disk}, o qual refere-se ao tamanho de armazenamento usado, e por W_{disk} representando os watts consumidos pelo recurso de armazenamento.

$$C_{disk} = \begin{cases} \text{se,} & |Cont_s| > 1, \\ \\ \text{então,} & \big(\sum_{t=T_1}^{T_2} \sum_{c=1}^{|Cont_u|} U_{disk}(t,c) \big). \\ & \big([W_{disk}.P_w - C_i] \big), \\ \\ \text{senão,} & \big(\sum_{t=T_1}^{T_2} U_{disk}(t) \big). \\ & \big([W_{disk}.P_w - C_i] \big) \end{cases} \tag{5.10}$$

Caso o número de contêineres seja superior a 1, o custo de armazenamento é obtido por dois somatórios. Um somatório considera o uso do disco por contêineres solicitados pelo cliente (c), enquanto o outro somatório considera o uso do disco em um determinado intervalo de tempo $T1 \leq t \leq T2$. Caso contrário, o custo é obtido apenas pelo somatório de tempo para o uso de armazenamento do contêiner alocado no servidor. Além do consumo de energia dos recursos computacionais em conteineres ativos, existe também o consumo energético do gerenciamento de contêiner. As tecnologias de orquestração de contêineres realizam o gerenciamneto desses contêineres (*i.e.,* Kubernetes, Swarm, Mesos, dentre outros). Portanto, as despesas de orquestraçào deve, ser contabilizadas e atribuídas aos contêineres gerenciados.

Por fim, a análise do consumo de energia é realizada no recurso computacional de orquestração de contêineres. Esta análise indica que o nó mestre não é influenciado pelo número de contêineres gerenciados. Por este motivo, é necessário que o rateio de energia dessa gestão ocorra entre todos os contêineres orquestrados. O preço da energia da orquestração envolve as taxas de utilização e compreensão ou seja, Tx_u e Tx_c respectivamente. Além disso, o elemento W_{orq} representa os watts consumidos pelo orquestrador, definindo o custo da orquestração do contêiner (C_{orq}). Desse modo, C_{orq} é apresentado na Equação 5.11.

$$C_{orq} = \begin{cases} \text{se,} & Tx_u < Tx_c, \\ \text{então,} & |Cont_u|.\left(\frac{\left[(W_{orq}.P_w)-C_i\right].\left[\frac{100+(Tx_u-Tx_c)}{100}\right]}{|Cont_s|}\right), \\ \text{senão,} & |Cont_u|.\left(\frac{\left[(W_{orq}.P_w)-C_i\right]}{|Cont_s|}\right) \end{cases} \tag{5.11}$$

A Equação 5.11 determina se a taxa de utilização Tx_u é menor que a taxa de complemento Tx_c, então o rateio desse custo ocorre entre o número de contêineres por clientes $|Cont_u|$, e entre a quantidade de contêineres por servidor $|Cont_s|$. Porém, caso Tx_u for maior que Tx_c, o rateio deste custo ocorre somente entre $|Cont_u|$ e $|Cont_s|$. Assim, caso o provedor de nuvem alocar um cliente em uma máquina sem outro usuário, o cliente não pagará o preço total do servidor ativo. Portanto, a Equação 5.12 apresenta o custo do contêiner, no qual C_{total} é o custo do uso de recursos por contêineres para um determinado cliente.

$$C_{total} = C_i + C_{cpu} + C_{mem} + C_{rede} + C_{disk} + C_{orq} \tag{5.12}$$

O modelo de custos proposto neste capítulo é simplificado e estratificado de acordo com os elementos que compõem um ambiente de virtualização de SO em provedores de nuvem IaaS. A validação e calibragem dos componentes do modelo dependem da análise do comportamento de cada componente.

5.1 CONSIDERAÇÕES PARCIAIS

Neste capítulo foi descrito a proposta de um modelo de custo para contêineres executados em provedores IaaS, considerando o gasto energético dos recursos computacionais para compor a precificação do modelo. Os recursos computacionais mensurados neste modelo englobam CPU, memória, I/O e rede, os quais compõem o custo individual por contêiner. Além do custo de plataforma, e de orquestração que compõem o custo final de contêineres.

O modelo proposto neste trabalho corresponde a uma simplificação minimalista de um modelo ideal a ser aplicado em um provedor IaaS. Esta proposta possui alto valor, pois apresenta benefícios financeiros tanto ao provedor quanto ao cliente, e incentiva aplicativos eficientes energeticamente. Além de contribuir para redução do impacto ambiental produzido pelos provedores de nuvens computacionais. O intuito deste modelo é preencher a lacuna existente no custo de contêineres nas nuvens computacionais.

6 ESTUDO DE CASO: *Amazon Web Services* (AWS)

O modelo de serviço em nuvem computacional AWS *Fargate*[1], oferecido pela Amazon ECS, permite a execução de contêineres, sem associação a MV, a servidores ou a aglomerados. A escolha do serviço *Fargate* como estudo de caso deste trabalho ocorreu devido o modelo de precificação de contêineres ofertada pela AWS a seus clientes, pagamento da alocação de recursos sob demanda. Contabilizados em períodos de 1 segundo e 1 hora em instâncias Linux e MS-Windows, respectivamente, os clientes pagam pelo número de CPUs alocadas, quantidade de memória, armazenamento e tamanho dos dados de comunicação usados (VOHRA, 2018).

Na visão do modelo de custo, o serviço *Fargate* preenche lacunas nas demandas de recursos por períodos. Os clientes podem adaptar as demandas de recursos segundo a segundo. Infelizmente, na prática, a otimização do gerenciamento de contêineres é uma tarefa complexa. Além disso, a tarefa de monitoramento fino da utilização de recursos e decisão de agendamento são responsabilidade do cliente. Contêineres são serviços flexíveis e permitem aplicar um modelo de custo flexível como um Preço Energético de Contêiner em Nuvem (PECN).

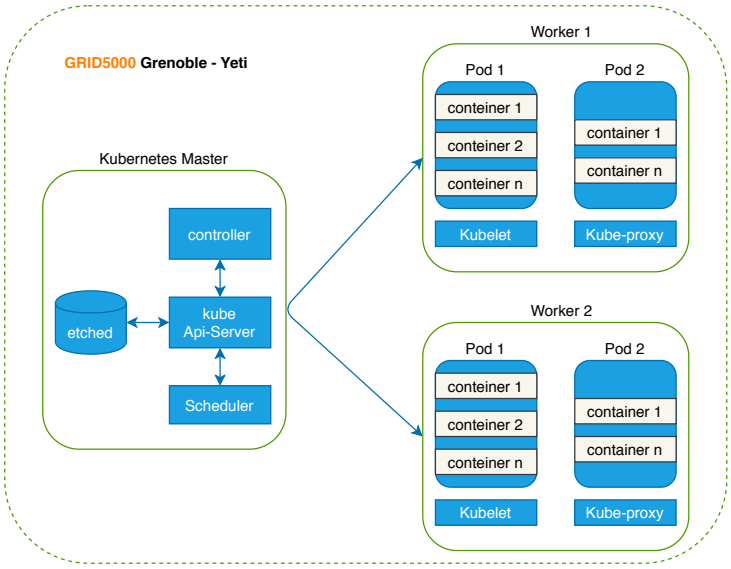
6.1 ANÁLISE DE CONSUMO DE ENERGIA COM CONFIGURAÇÕES DE FARGATE

O Capítulo 4 descreve a análise do consumo de energia de recursos físicos onde o servidor dedicado tem seus recursos saturados. Como a plataforma AWS não permite baixo nível de acesso a recursos físicos do DCs, uma plataforma análoga é construída no cluster do Yeti. O serviço AWS *Fargate* oferece configurações limitadas com suporte (<https://aws.amazon.com/fargate/pricing/>) que são reproduzidas no cluster Yeti. O contêiner e a tecnologia de gerenciamento usados neste cenário foram Docker e Kubernetes, respectivamente. Essas tecnologias foram escolhidas devido à sua popularidade, e o serviço *Fargate* as utiliza para executar os contêineres de seus clientes. O ambiente do experimento consiste em 3 servidores, no qual um é reservado para o orquestrador mestre do Kubernetes, enquanto os outros executam os contêineres do Docker por meio dos nós do Kubernetes (Figura 14).

Atualmente, o preço do AWS *Fargate* é calculado com base nos recursos de CPU e memória para os períodos usados, já as transferências de dados (<https://aws.amazon.com/ec2/pricing/on-demand/#Data_Transfer>), bem como o armazenamento (<https://aws.amazon.com/pt/ebs/pricing/>), são contabilizados adicionalmente. O modelo de custo AWS *Fargate* é privado, portanto, é impossível comparar diretamente

[1] https://aws.amazon.com/pt/fargate/

Figura 14 – Gerenciamento de Contêineres



Fonte: Adaptado de (KUBERNETES, 2020)

os custos do contêiner do orquestrador. Portanto, os próximos experimentos permitirão estimar o consumo de energia para cada componente de custo do PECN de acordo com as configurações do AWS *Fargate*.

6.1.1 Ofertas de serviço AWS Fargate

Para este estudo de caso, o comportamento do consumo de energia da CPU e da memória é analisado de acordo com as ofertas do AWS *Fargate* (Tabela 3). O padrão AWS *Fargate* é baseado na alocação de vCPU e recursos de memória. Este estudo de caso apresenta o consumo de energia por recurso físico individualmente quando o recurso é utilizado ao máximo. Portanto, este experimento consiste em dois benchmarks, Stress NG e Stream, e 21 *cases* compostos de vCPU e memória.

Tabela 3 – AWS Fargate service offers. Adapted from (Amazon Web Services (AWS), 2020b).

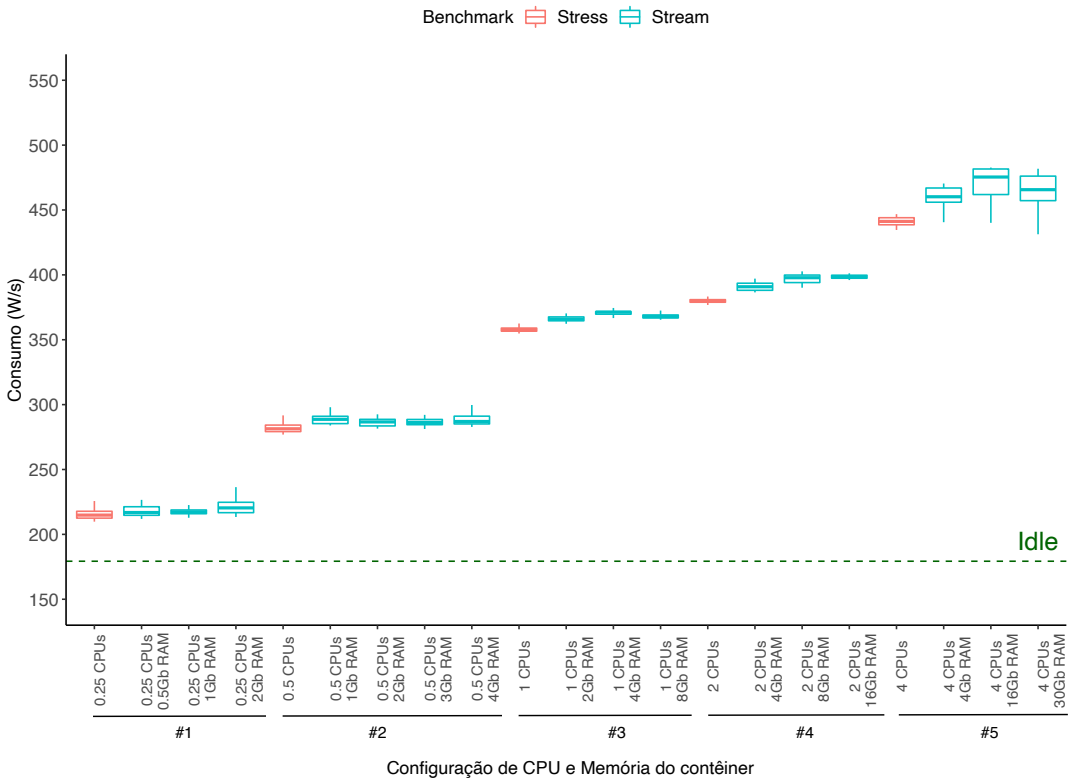
ID	CPU	Valores de Memória
#1	0.25 vCPU	0.5 GB, 1 GB and 2GB
#2	0.5 vCPU	Min. 1GB e Max. 4GB, em incremento de 1 GB
#3	1 vCPU	Min. 2GB e Max. 8GB, em incremento de 1 GB
#4	2 vCPU	Min. 4GB e Max. 16GB, em incremento de 1 GB
#5	4 vCPU	Min. 8GB e Max. 30GB, em incrementos de 1 GB

Fonte: Elaborado pela autora (2020).

O primeiro ID do serviço AWS *Fargate* (#1) tem 4 casos, todos com 0,25 vCPU e, respectivamente, alocação de memória, 0,5 GB, 0,5 GB, 1 GB e 2 GB. O consumo de

energia de vCPU para cada tipo de serviço (ID) da AWS *Fargate* é obtido por alocação mínima de recursos, neste caso 0,25 vCPU e 0,5 GB, porém, utilização máxima de vCPU (benchmark Stress). Por outro lado, o consumo de energia da memória é obtido pela utilização máxima de vCPU e memória em cada ID. A mesma metodologia é aplicada aos IDs #2, #3, #4 e #5.

Figura 15 – Consumo energético de CPU e Memória.



Fonte: Elaborado pela autora (2020).

A Figura 15 mostra o impacto da energia nos recursos físicos (CPU e memória) de acordo com as ofertas do serviço da AWS *Fargate*. Os eixos *x* e *y* mostram todos os 21 estudos de caso e o consumo de energia, respectivamente. A linha verde pontilhada representa o consumo mínimo de energia do servidor, sem carga de trabalho, ou seja, quando um cliente da AWS *Fargate* possui alocação de recursos, porém, seus aplicativos não requerem recursos. Portanto, independentemente do estudo de caso escolhido, é possível observar a lacuna entre a linha ociosa e os estudos de caso. A lacuna no consumo de energia pode chegar a 300 *Watts/s*, porém o AWS *Fargate* não leva isso em conta na cobrança.

6.1.2 Serviços adicionais AWS Fargate

O AWS considera o armazenamento e as transferências de dados como serviços adicionais. Para o serviço de armazenamento (Volumes Amazon EBS), os clientes

têm5 opções de volume, e pagam por GB-mês de armazenamento provisionado, conforme descrito na Tabela reftab:stodfPrice. No estudo de caso, vale ressaltar que o volume do Amazon EBS escolhido é o volume do HDD frio (sc1) por se aproximar do hardware do Yeti.

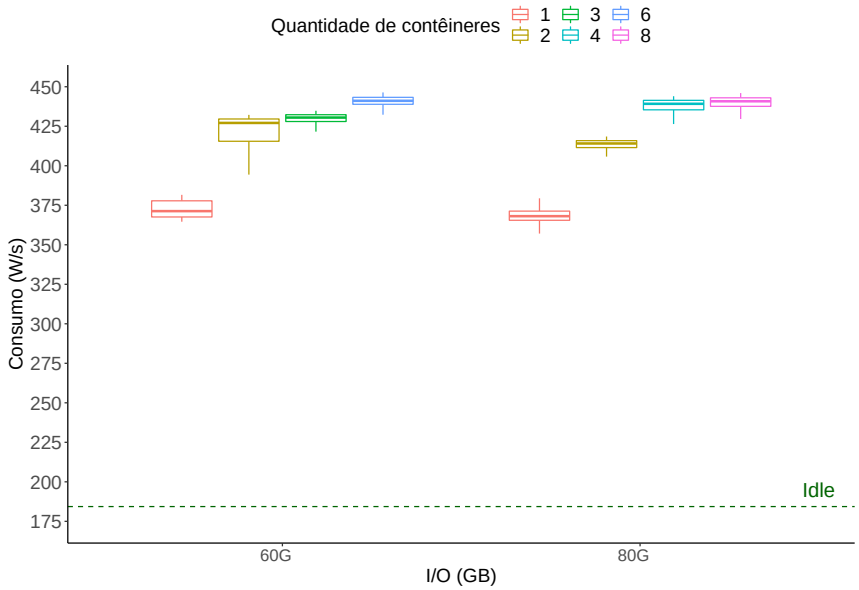
Tabela 4 – Additional AWS offers. Adapted from (Amazon Web Services (AWS), 2020a; Amazon Web Services (AWS), 2020c).

Storage Services	Data transfers OUT
General Purpose SSD (gp2) Volumes	Acima de 1 GB / Mês
Provisioned IOPS SSD (io2) Volumes	Próximo a 9.999 TB / Mês
Provisioned IOPS SSD (io1) Volumes	Próximo a 40 TB / Mês
Throughput Optimized HDD (st1) Volumes	Próximo a 100 TB / Mês
Cold HDD (sc1) Volumes	Maior que 150 TB / Mês

Fonte: Elaborado pela autora (2020).

Um serviço de armazenamento consome energia para permanecer ativo, com ou sem atividade de armazenamento, e consome energia adicional quando as atividades de transação de disco (gravação / leitura) são necessárias. O serviço de volume do Amazon EBS desconsidera as atividades de transação de disco e, consequentemente, a variação do consumo de energia. A Figura 16 representa o consumo de energia de armazenamento considerando as atividades de transação de disco, onde o *eixo y* expõe o consumo de energia de armazenamento (*Watts/s*), e o *eixo x* mostra a quantidade de tamanho de espaço em disco disponível para realizar transações intensivas em disco.

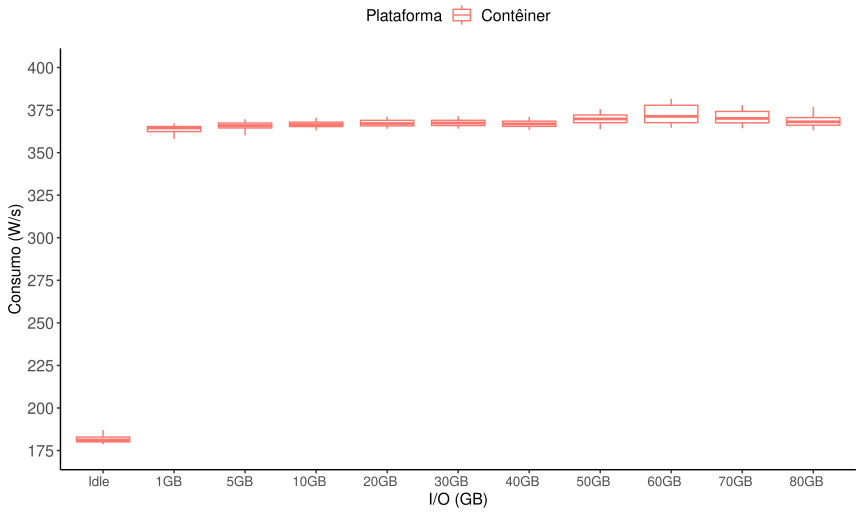
Figura 16 – Consumo energético .



Fonte: Elaborado pela autora (2020).

O benchmark *Flexible IO* (AXBOE, 2017) é escolhido para gerar transações intensivas em disco. As duas últimas opções são o número de contêineres, entre 1 e 8, e o tamanho do espaço em disco, 60 e 80 GB. Os resultados mostram que o comportamento do consumo de energia é semelhante com 60 e 80 GB. Por outro lado, o número de contêineres tem impacto de energia, (*i.e.,* 8 contêineres consomem ~ 70 *Watts/s* a mais do que apenas 1 contêiner). Finalmente, se o consumo de energia de um cliente de transações intensivas em disco, com até 8 contêineres, for comparado a outro cliente, sem transações intensivas em disco (linha verde pontilhada), a lacuna pode chegar a ~ 260 *Watts/s*. A Figura 17, mostra a execução do benchmark *Flexible IO* com um único contêiner, a qual apresentou variância significativa no consumo de $watts/s$ (*eixo y)* apenas ao alterar o estado ocioso (*i.e.,* com o servidor em *idle*), para o estado ativo utilizando 1 GB (*eixo x*) de armazenamento.

Figura 17 – Consumo energético .

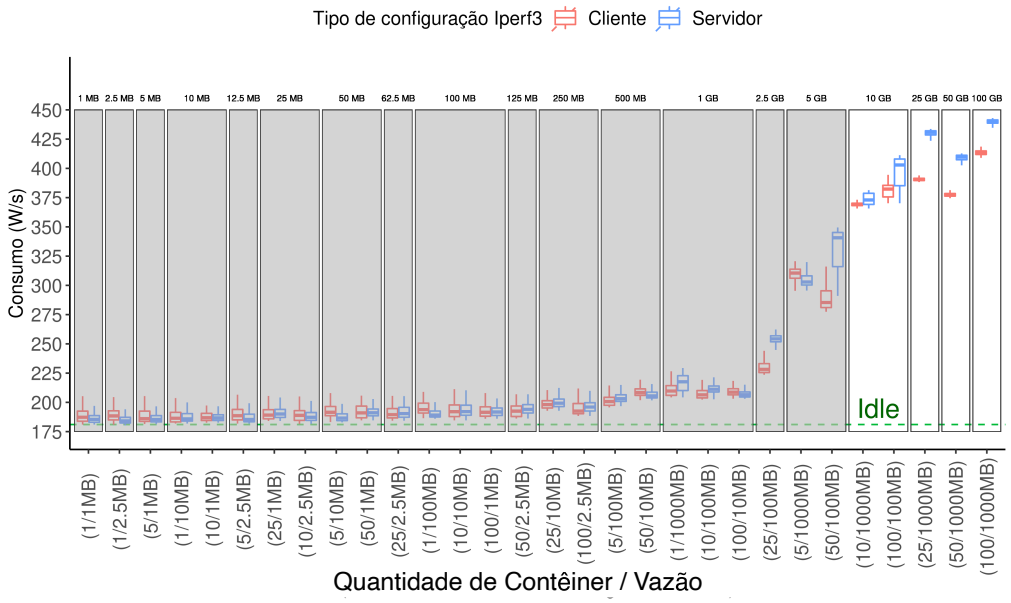


Fonte: Elaborado pela autora (2020).

Para o serviço de transferência de dados, o Amazon EC2 oferece taxas gratuitas para transferências de dados interno da Amazon. As transferências de dados das ofertas de serviço da Amazon são descritas na Tabela 4. O serviço de transferência de dados da Amazon considera o consumo de energia do equipamento de rede como fixo. No entanto, o cliente da aplicação pode ter alta demanda de fluxo de rede em um mês e nenhuma demanda de fluxo de rede em outro. Além disso, a mesma quantidade de dados transferidos tem diferença no consumo de energia por cliente e servidor.

A análise de transferência de dados com base no consumo de energia escolheu o benchmark iPerf3 (MORTIMER, 2018) para gerar fluxos de rede. A Figura 18 apresenta o comportamento do fluxo de rede do cliente e do servidor, onde *eixo x* mostra o número de contêineres / vazão de banda, e *eixo y*, o consumo de energia. A variação da vazão de banda da rede começou em 1 MB exibidos no *eixo x*, realizando

Figura 18 – Consumo energético .



Fonte: Elaborado pela autora (2020).

um aumento gradual até a saturação total da vazão de banda do servidor (10 GB). Porém, para saturar a rede do servidor e identificar seu comportamento com solicitações superiores às disponíveis, os testes exigiram um excedente de 25 GB, 50 GB e 100 GB. Os resultados mostram um aumento gradativo no consumo de energia até 500 MB, com uma diferença máxima de 50 *watts/s* quando comparado ao servidor inativo, obtendo um aumento de 42%. Por outro lado, valores superiores a 1 GB se comportaram de maneira diferente, mostrando um crescimento não linear no consumo de energia, fazendo com que a diferença no consumo de energia resultasse em um máximo de 270 *watts/s* em comparação com o servidor inativo, apresentando um aumento de 157% no consumo. Além disso, é essencial avaliar as mudanças no número de contêineres usando uma rede paralela. Assim, se a aplicação visa a redução do consumo de energia, é oportuno reduzir o número de intercomunicação entre contêineres.

Portanto, os serviços adicionais da AWS *Fargate* sofrem com o consumo de energia variável. É possível caracterizar o impacto da energia do recurso de armazenamento relevante variando o número de contêineres que usam armazenamento. Para o recurso de rede, o impacto da energia é evidente quando o recurso está saturado.

6.1.3 PECN *vs.* AWS Fargate

As seções anteriores descrevem o comportamento do consumo de energia que envolve o gerenciamento de contêineres, um novo modelo de custo com consciência energética (PECN) e o consumo de energia de recursos físicos associados às ofertas do serviço AWS *Fargate*. Esta seção é dedicada à comparação de preços entre PECN e AWS *Fargate*. No primeiro cenário, existem duas configurações, com CPUs de 2 e 4.

Tabela 5 – Preço AWS Fargate

CPUs	Utilidade (%)	Preço (US$/hour)
2	50%	0.08
	100%	0.08
4	50%	0.16
	100%	0.16

Fonte: Elaborado pela autora (2020).

A Tabela 5 confirma que o modelo de custo AWS *Fargate* considera apenas o provisionamento de CPU e memória, esquecendo a porcentagem de utilização de recursos. Em ambos os casos, o preço é o mesmo para 50% e 100% de utilização de recursos da CPU. No entanto, uma CPU (correspondente a 50% da demanda provisionada da CPU) representa 5,57% menos de consumo energético se comparado a 100% considerado pela AWS *Fargate*, ou seja, deve diminuir a conta do cliente. Os contêineres operam na plataforma flexível, portanto, o modelo de custo do contêiner também deve ser flexível. Um modelo de custo flexível pode motivar o cliente a tornar seus aplicativos eficientes, incentivando a economia de energia e práticas de TI verdes.

6.1.4 Precificação individual de recursos de Contêiner

O PECN considera o consumo de energia de 5 componentes (*i.e.,* CPU, memória, armazenamento, rede e orquestração de contêiner) para compor o faturamento do cliente. Por outro lado, como AWS possui um modelo de custo privado, a proporção do custo de energia no faturamento do cliente final não é conhecida. Na literatura, existem relatos de que os custos operacionais de DC têm os custos de energia chegando a 50% (COMERFORD, 2015). Os custos operacionais incluem ar condicionado, entre outros equipamentos que consomem muita energia. Em *Proportional-Shared Virtual Energy* (PSVE) (HINZ et al., 2018), os autores estimam que o consumo de energia pode estar entre 5 e 15% do preço da instância IaaS, e este trabalho seguirá esta estimativa. A linha de base do preço da energia se aplica ao modelo PECN é de um distribuidor elétrico na região norte da Virgínia, onde o DC está instalado, que é US$ 0,0773 por quilowatt-hora.

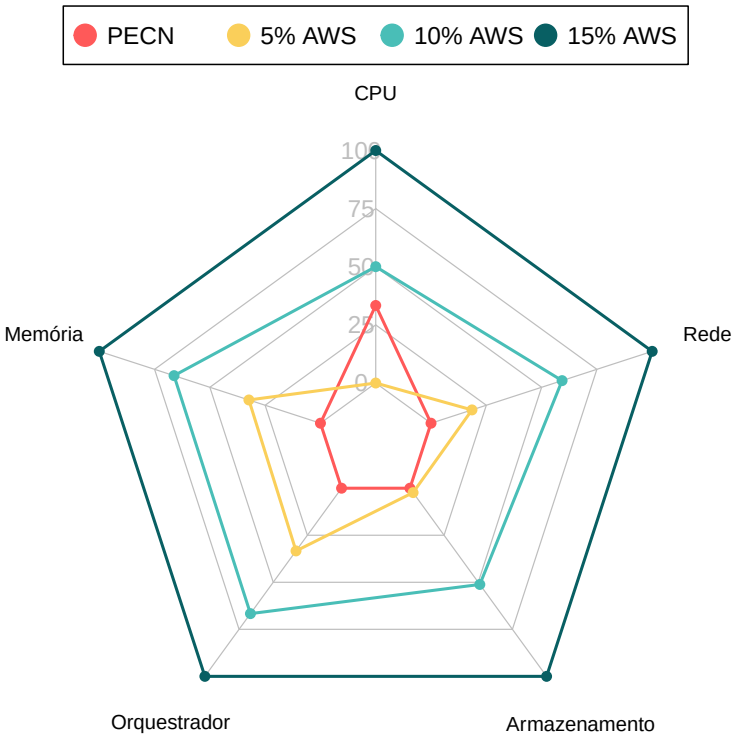


Figura 19 – PECN *vs.* AWS *Fargate*:Preço estimado dos componentes de custo individuais de acordo com o uso de 25% dos recursos.

Fonte: Elaborado pela autora (2020).

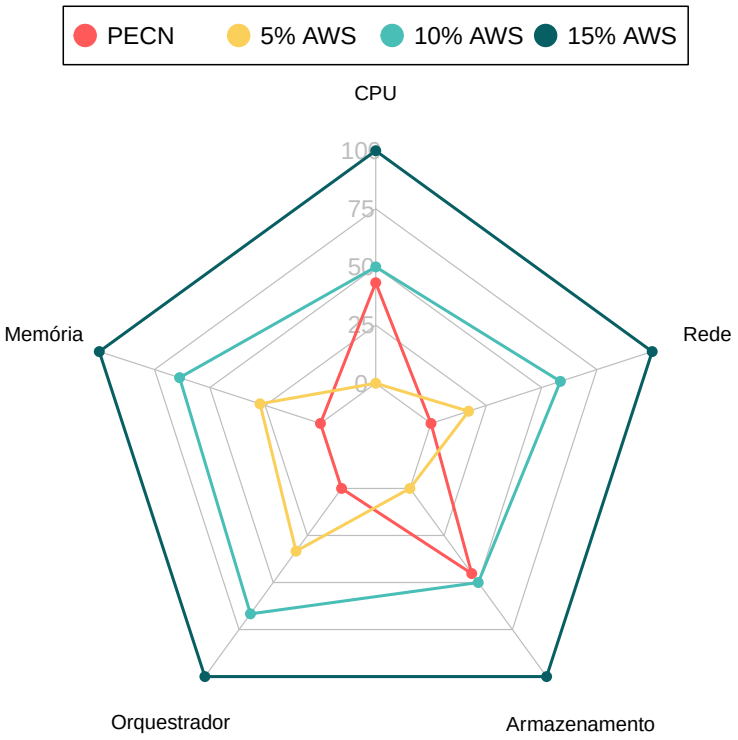


Figura 20 – PECN *vs.* AWS *Fargate*:Preço estimado dos componentes de custo individuais de acordo com o uso de 50% dos recursos.

Fonte: Elaborado pela autora (2020).

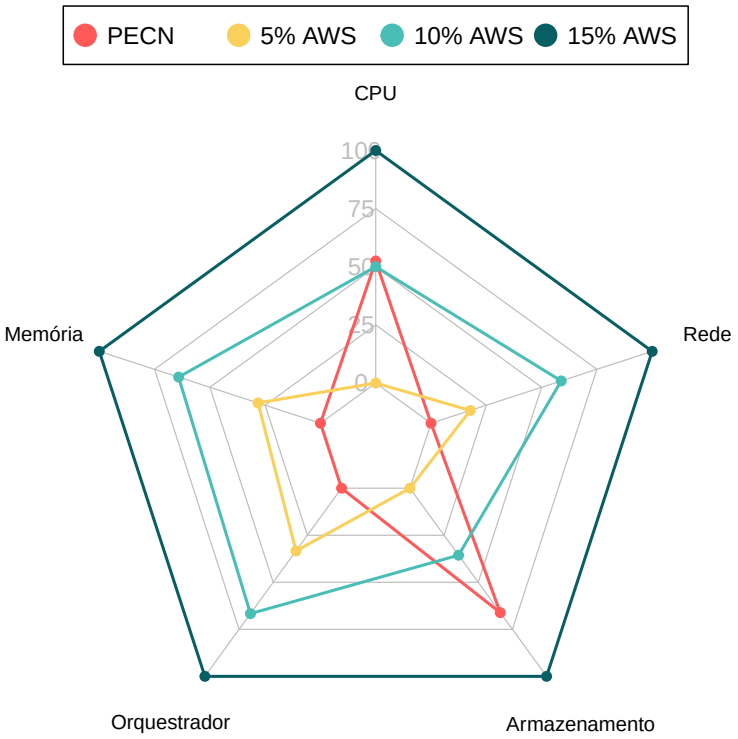


Figura 21 – PECN *vs.* AWS *Fargate*:Preço estimado dos componentes de custo individuais de acordo com o uso de 75% dos recursos.

Fonte: Elaborado pela autora (2020).

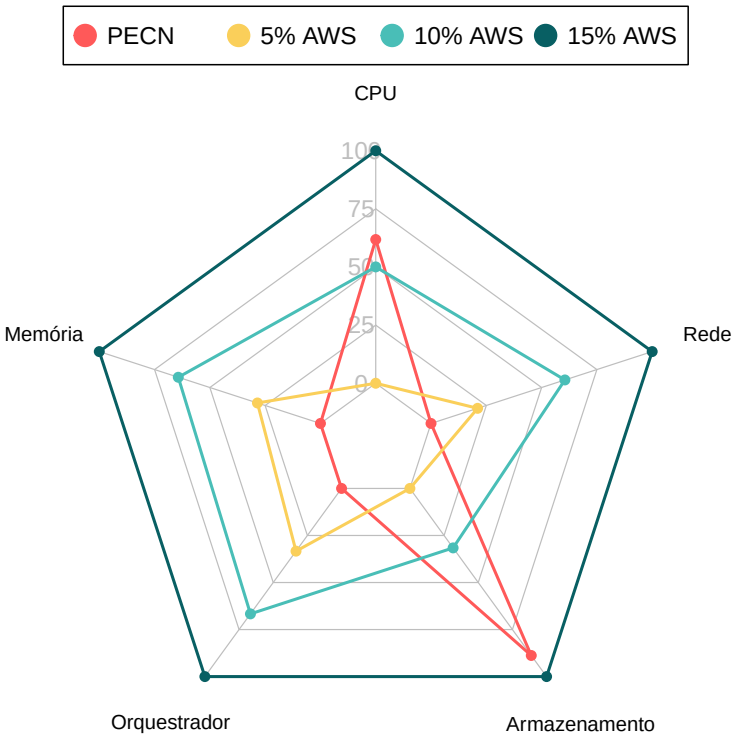


Figura 22 – PECN *vs.* AWS *Fargate*:Preço estimado dos componentes de custo individuais de acordo com o uso de 100% dos recursos.

Fonte: Elaborado pela autora (2020).

Os resultados da estimativa de preços são apresentados nas Figuras 19, 20, 21 e 22. Como PECN considera o uso dos componentes, os resultados mostram o preço estimado para quatro cenários diferentes (25%, 50%, 75% e 100%). Um gráfico de radar tem um eixo proporcional, no qual o valor máximo é 100% e o mínimo é 0%. Portanto, os preços mais altos estão na borda e os mais baixos no centro. Cada eixo representa o preço estimado, com custo de energia de 5%, 10%, 15%, e PECN, para cada componente do modelo de custo, (*i.e.,* CPU, memória, rede, orquestrador e armazenamento). Os cenários são representados por diferentes cores, e linhas sólidas unidas por pontos, formando uma superfície poligonal. A análise dos componentes combinados é feita pelo cálculo da área de superfície do polígono. Quanto menor for a área da superfície, menor será o preço.

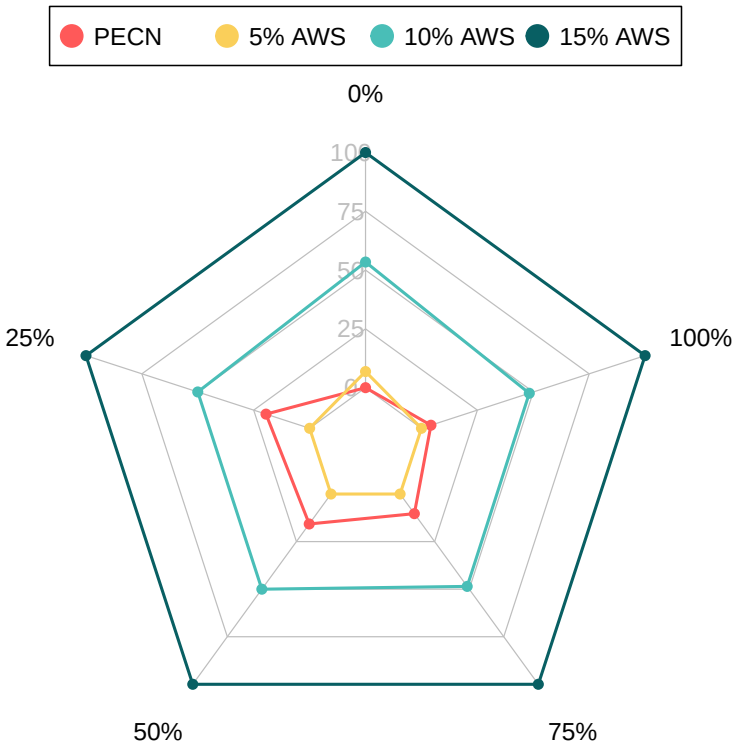
Em todos os cenários (Figuras 19, 20, 21 e 22), a estimativa verde escuro (AWS 15%) é o pior caso, como esperado, (*i.e.,* o preço mais alto). Com os componentes de CPU e armazenamento, a estimativa em verde claro (AWS 10%) é melhor do que PECN, quando as utilizações de CPU e armazenamento são 75% e 100%. No entanto, a área de superfície de PECN permanece menor do que a área de estimada de AWS 10%. Finalmente, comparando a AWS 5% (estimativa laranja) e PECN, PECN tem pelo menos três dos cinco pontos no centro (o melhor resultado), porém superior ao preço estimado AWS 5% em dois cenários (75% e 100%). Como esperado, PECN é melhor quando o uso do componente é baixo e, conforme o uso do componente aumenta, o preço também aumenta.

Além dos preços dos componentes individuais, o preço total considera o custo da energia para manter o servidor ocioso ativo. A Figura 23 mostra um gráfico radar com 5 cenários diferentes, de acordo com o uso de componentes, (*i.e.,* 0% mostra o servidor inativo, e 100% todos os componentes estão saturados). Em todos os cenários, PECN mostra preços totais inferiores a AWS 15% e 10%. A estimativa AWS 5% tem o menor valor para o preço total, exceto quando o servidor está ocioso. Finalmente, é importante ressaltar que o preço total da estimativa de PECN se adapta ao consumo de energia de acordo com o uso dos componentes. Assim, os clientes devem liderar suas aplicações em função do consumo de energia, otimizando recursos físicos e aplicando conceitos verdes de TI.

6.2 CONSIDERAÇÕES PARCIAIS

Este capítulo apresenta um estudo de caso baseado no serviço Fargate da AWS, bem como a análise do consumo energético dos recursos computacionais (*i.e.,* CPU, memória, rede, armazenamento e orquestração de contêiner) usados por aplicações conteinerizadas. Além de abordar as ofertas aplicadas aos recursos computacionais no AWS Fargate. Por fim apresenta a análise dos resultados atingidos, o preço final de

Figura 23 – Custo energético Total de Contêineres - C_{total}



Fonte: Elaborado pela autora (2020).

todos os recursos em conjunto, e um comparativo dos preços de energia obtidos com o modelo PECN e dos preços estimados na AWS Os resultados obtidos com o estudo de caso fortalece a necessidade do modelo de custo proposto neste trabalho. De modo que esta proposta incentiva as práticas de redução energética e TI verde. O modelo proposto neste trabalho, se restringe ao custo energético dos recursos computacionais utilizados na execução de contêineres.

7 CONCLUSÃO

A energia é um dos principais componentes do CTP de DCs e da infraestrutura física em nuvem computacional. Desde o seu surgimento, os modelos de custos evoluíram, ampliando a oferta de serviços e os meios tarifários. Entre outras características, o modelo de serviço em nuvem consolidou-se por sua capacidade de fazer otimização de recursos físicos. Os clientes da nuvem estão cientes da alocação de recursos físicos porque isso afeta suas contas. No entanto, os clientes da nuvem não estão cientes do impacto da energia produzida por seus aplicativos. Quanto maior o consumo de energia da aplicação, maior o custo de energia do DC, consequentemente, maior custo para o cliente. Portanto, reduzir o consumo energético tornou-se um interesse do cliente, uma vez que a recompensa financeira é imediata.

Este trabalho realizou um estudo baseado em trabalhos correlatos, o qual apresentou modelos de precificação presentes na literatura. Porém estes modelos são limitados a componentes e comportamentos associados a técnica de virtualização de hipervisor (MV) (HINZ et al., 2018; KURPICZ et al., 2018). Havendo uma lacuna científica nas pesquisas de modelos de precificação para computação em nuvem, guiados pelo consumo de energia para contêineres. As contribuições realizadas por este trabalho são uma análise comparativa do consumo de energia por recursos físicos em ambientes *bare metal*, MV e contêiner; a estratificação dos componentes de consumo de energia dos contêineres; e a principal contribuição deste trabalho é a descrição da proposta de um modelo de custo para nuvens IaaS consciente de energia para contêineres. Cada recurso tem um comportamento diferente no consumo de energia na execução dos contêineres. O recurso de CPU aumenta o consumo de energia drasticamente de acordo com a porcentagem de utilização da CPU, a memória e a rede possuem um aumento brando. Por outro lado, o armazenamento tem consumo de energia estável entre 1 GB e 80 GB (intervalo observado) para uso intensivo de E/S.

O modelo de precificação de contêineres proposto neste trabalho (PECN), contabiliza o uso dos recursos de alocação, agrega o componente ciente de energia para reconhecer a alocação de recursos físicos Desse modo PECN recompensa os clientes conscientes de energia. A análise de recursos individuais do consumo de energia mostra que um aplicativo que faz uso intensivo de recursos pode consumir 300% vezes mais energia. Os valores estimados do modelo de custo PECN, variam semanalmente entre US $ 2,31 e US $ 10,59, enquanto os valores da AWS Fargate US $ 2,71 e US $ 29,94, representando uma redução de até 35,39%.

As análises realizadas nesta pesquisa, a qual compara as métricas de energia e

de desempenho abordadas no Capítulo 4, confirmam a eficiência do modelo descrito e proposto nesta dissertação. Apesar deste modelo corresponder a uma simplificação minimalista de um modelo ideal a ser aplicado em um provedor IaaS, a proposta deste trabalho possui alto valor, pois apresenta não somente benefícios financeiros tanto ao provedor quanto ao cliente, mas também incentiva o desenvolvimento de aplicativos eficientes energeticamente, reduzindo o impacto ambiental produzido pelos provedores de nuvens computacionais.

Durante o desenvolvimento da proposta deste trabalho, uma publicação foi realizada. O artigo (MOREIRA et al., 2019) apresenta um estudo preliminar sobre a precificação de contêineres em nuvens IaaS com base no consumo energético. Diversos obstáculos foram encontrados quanto a elaboração e execução dos *scripts* utilizados nas experimentações realizadas nesta pesquisa, além de desafios enfrentados para configurar devidamente a plataforma computacional GRID5000 para cada ambiente executado, bem como a captação precisa dos dados de consumo energético de cada experimento.

Em trabalhos futuros, é importante aprofundar os impactos do consumo de energia em outros serviços de computação em nuvem, por exemplo, OpenStack nova, OpenStack Magnos, OpenStack Ironic. Também a extensão deste modelo comparando diversos provedores e diferentes tecnologias de contêineres e de orquestradores, a fim de determinar o comportamento dos custos em diferentes cenários. Ainda, uma tecnologia a ser analisada quanto ao impacto energético em nuvens computacionais é a *serverless*. Assim como a proposta do contêiner, a tecnologia *serverless* propõe uma arquitetura com baixa sobrecarga, escalável e eficiente para executar as inovações tecnológicas em nuvens computacionais. Por este motivo, é importante um estudo comparativo entre os modelos de serviço e de precificação baseados em contêineres e *serverless*.

REFERÊNCIAS

Amazon Web Services (AWS). **Amazon EBS Volumes**. 2020. <https://aws.amazon.com/fargate/pricing/>.

Amazon Web Services (AWS). **AWS Fargate Pricing**. 2020. <https://aws.amazon.com/fargate/pricing/>.

Amazon Web Services (AWS). **Data Transfer**. 2020. <https://aws.amazon.com/ec2/pricing/on-demand/#Data_Transfer>.

AXBOE, J. **1. fio - Flexible I/O tester rev. 3.23¶**. 2017. Disponível em: <https://fio.readthedocs.io/en/latest/fio_doc.html>.

BAWDEN, T. Global warming: Data centres to consume three times as much energy in next decade, experts warn. **The Independent**, v. 23, 2016.

BEGUM, S.; KHAN, M. K. Potential of cloud computing architecture. In: IEEE. **Information and Communication Technologies (ICICT), 2011 Int. Conf. on**. [S.l.], 2011. p. 1–5.

BERGEN, A. et al. Towards software-adaptive green computing based on server power consumption. In: **Proceedings of the 3rd International Workshop on Green and Sustainable Software**. [S.l.: s.n.], 2014. p. 9–16.

BHATTACHARYA, A. A. et al. The need for speed and stability in data center power capping. **Sustainable Computing: Informatics and Systems**, Elsevier, v. 3, n. 3, p. 183–193, 2013.

BINDU, G. H.; RAMANI, K.; BINDU, C. S. Energy aware multi objective genetic algorithm for task scheduling in cloud computing. **International Journal of Internet Protocol Technology**, Inderscience Publishers (IEL), v. 11, n. 4, p. 242–249, 2018.

BITTENCOURT, L. F. et al. Scheduling in distributed systems: A cloud computing perspective. **Computer Science Review**, Elsevier, v. 30, p. 31–54, 2018.

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

COLUMBUS, L. State of cloud adoption and security. **Forbes**, 2017.

COMERFORD, T. How data center operators can avoid energy price hikes this winter. **Data Center Knowledge**, 2015.

CONTAINERS, L. Linuxcontainers. org infrastructure for container projects. **Canonical Ltd,[En línea]. Available: https://linuxcontainers. org/lxc/introduction**, 2017.

DANILAK, R. **Why Energy Is a Big And Rapidly Growing Problem for Data Centers. 2017**. 2017.

DIAMANTI. **Container Adoption Benchmark Survey**. [S.l.], 2019. Disponível em: <https://diamanti.com/wp-content/uploads/2019/06/Diamanti_2019_Container_Survey.pdf>.

DOCKER. **How nodes work**. 2020. <https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/>, journal=docker docs.

DOCKER, I. Docker-build, ship, and run any app, anywhere. **URL: https://www. docker. com**, 2016.

ENOKIDO, T.; TAKIZAWA, M. Power consumption and computation models of virtual machines to perform computation type application processes. In: IEEE. **2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems**. [S.l.], 2015. p. 126–133.

ESTRADA, Z. J. et al. A performance evaluation of sequence alignment software in virtualized environments. In: IEEE. **Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on**. [S.l.], 2014. p. 730–737.

GARG, S.; BUYYA, R. Green cloud computing and environmental sustainability. p. 315–340, 01 2012.

GIL, A. C. **Métodos e técnicas de pesquisa social**. [S.l.]: 6. ed. Ediitora Atlas SA, 2008.

GUITART, J. Toward sustainable data centers: a comprehensive energy management strategy. **Computing**, Springer, v. 99, n. 6, p. 597–615, 2017.

HAMMADI, A.; MHAMDI, L. A survey on architectures and energy efficiency in data center networks. **Computer Communications**, Elsevier, v. 40, p. 1–21, 2014.

HANINI, M.; KAFHALI, S. E.; SALAH, K. Dynamic vm allocation and traffic control to manage qos and energy consumption in cloud computing environment. **International Journal of Computer Applications in Technology**, Inderscience Publishers (IEL), v. 60, n. 4, p. 307–316, 2019.

Hassan, S.; Ali, N.; Bahsoon, R. Microservice ambients: An architectural metamodelling approach for microservice granularity. In: **2017 IEEE International Conference on Software Architecture (ICSA)**. [S.l.: s.n.], 2017. p. 1–10.

HINZ, M. et al. A cost model for iaas clouds based on virtual machine energy consumption. **Journal of Grid Computing**, Springer, v. 16, n. 3, p. 493–512, 2018.

HSU, C.-H. et al. Optimizing energy consumption with task consolidation in clouds. **Information Sciences**, Elsevier, v. 258, p. 452–462, 2014.

HU, J.; DENG, J.; WU, J. A green private cloud architecture with global collaboration. **Telecommunication Systems**, Springer, v. 52, n. 2, p. 1269–1279, 2013.

HYKES, S. Docker 0.9: introducing execution drivers and libcontainer. **Internet: http://blog. docker. com/2014/03/docker-0-9-introducin g-execution-drivers-and-libcontainer/,[10.03. 2014]**, 2018.

IZADPANAH, S. et al. Evaluation of energy consumption and data access time in data fetching in grid-based data-intensive applications. In: IEEE. **Energy Efficient and Green Networking (SSEEGN), 2013 22nd ITC Specialist Seminar on**. [S.l.], 2013. p. 37–42.

JAIN, A. et al. Energy efficient computing- green cloud computing. In: **2013 International Conference on Energy Efficient Technologies for Sustainability**. [S.l.: s.n.], 2013. p. 978–982.

KIM, N. Y. et al. Cf-cloudorch: container fog node-based cloud orchestration for iot networks. **The Journal of Supercomputing**, Springer, v. 74, n. 12, p. 7024–7045, 2018.

KING, C. I. **Stress-ng**. Canonical, 2019. Disponível em: <https://elinux.org/images/5/5c/Lyon-stress-ng-presentation-oct-2019.pdf>.

KOMINOS, C. G.; SEYVET, N.; VANDIKAS, K. Bare-metal, virtual machines and containers in openstack. In: **2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)**. [S.l.: s.n.], 2017. p. 36–43.

KRETEN, S.; GULDNER, A.; NAUMANN, S. An analysis of the energy consumption behavior of scaled, containerized web apps. **Sustainability**, Multidisciplinary Digital Publishing Institute, v. 10, n. 8, p. 2710, 2018.

KUBERNETES. **Kubernetes Components**. 2020. <https://kubernetes.io/docs/concepts/overview/components/>.

KURPICZ, M. et al. Energy-proportional profiling and accounting in heterogeneous virtualized environments. **Sustainable Computing: Informatics and Systems**, Elsevier, v. 18, p. 175–185, 2018.

LEITNER, P.; CITO, J.; STÖCKLI, E. Modelling and managing deployment costs of microservice-based cloud applications. In: ACM. **Proceedings of the 9th International Conference on Utility and Cloud Computing**. [S.l.], 2016. p. 165–174.

LI, Z. et al. A survey on modeling energy consumption of cloud applications: deconstruction, state of the art, and trade-off debates. **IEEE Transactions on Sustainable Computing**, IEEE, v. 2, n. 3, p. 255–274, 2017.

LIU, K.; PINTO, G.; LIU, Y. D. Data-oriented characterization of application-level energy optimization. In: SPRINGER. **International Conference on Fundamental Approaches to Software Engineering**. [S.l.], 2015. p. 316–331.

LUO, C. et al. A holistic energy optimization framework for cloud-assisted mobile computing. **IEEE Wireless Communications**, IEEE, v. 22, n. 3, p. 118–123, 2015.

MCCALPIN, J. D. et al. Memory bandwidth and machine balance in current high performance computers. **IEEE computer society technical committee on computer architecture (TCCA) newsletter**, v. 2, n. 19–25, 1995.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. **Special Publication 800-145**, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011.

MERCL, L.; PAVLIK, J. The comparison of container orchestrators. In: SPRINGER. **Third International Congress on Information and Communication Technology**. [S.l.], 2019. p. 677.

MORABITO, R. Power consumption of virtualization technologies: an empirical investigation. In: IEEE. **2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)**. [S.l.], 2015. p. 522–527.

MOREIRA, A. S. et al. Precificação de contêineres em nuvens iaas com base no consumo energético: um estudo preliminar. In: SBC. **Anais da XIX Escola Regional de Alto Desempenho da Região Sul**. [S.l.], 2019.

MORTIMER, M. **iperf3 documentation**. 2018.

NETTO, H. et al. Replicaçao de máquinas de estado em containers no kubernetes: uma proposta de integraçao. **Anais do XXXIV Simpósio Brasileiro de Redes de Computadores-SBRC**, 2016.

PAHL, C.; LEE, B. Containers and clusters for edge cloud architectures–a technology review. In: IEEE. **2015 3rd international conference on future internet of things and cloud**. [S.l.], 2015. p. 379–386.

PANDIKUMAR, S.; KABILAN, S. P.; AMALRAJ, L. Article: Green it: A study and analysis of environmental impact of social networks and search engines. **International Journal of Computer Applications**, v. 60, n. 6, p. –, December 2012. Full text available.

PANIZZON, G. et al. A taxonomy of container security on computational clouds: concerns and solutions. **Revista de Informática Teórica e Aplicada**, v. 26, n. 1, p. 47–59, 2019.

PEINL, R.; HOLZSCHUHER, F.; PFITZER, F. Docker cluster management for the cloud - survey results and own solution. **J. Grid Comput.**, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 14, n. 2, p. 265–282, jun. 2016. ISSN 1570-7873. Disponível em: <http://dx.doi.org/10.1007/s10723-016-9366-y>.

POLVI, A. Coreos is building a container runtime, rkt. **Retrieved**, v. 4, p. 2015, 2014.

QIU, Y. et al. Lxc container migration in cloudlets under multipath tcp. In: IEEE. **2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)**. [S.l.], 2017. v. 2, p. 31–36.

RAUEN, F. J. Pesquisa científica: discutindo a questão das variáveis. **SIMFOP: Santa Catarina**, 2012.

SANTOS, E. A. et al. How does docker affect energy consumption? evaluating workloads in and out of docker containers. **Journal of Systems and Software**, Elsevier, v. 146, p. 14–25, 2018.

SHARMA, P. et al. Containers and virtual machines at scale: A comparative study. In: ACM. **Proceedings of the 17th International Middleware Conference**. [S.l.], 2016. p. 1.

SHARMA, P. et al. Design and operational analysis of a green data center. **IEEE Internet Computing**, p. 1–1, 2017. ISSN 1089-7801.

SILVA, V. G. da; KIRIKOVA, M.; ALKSNIS, G. Containers for virtualization: An overview. **Applied Computer Systems**, Sciendo, v. 23, n. 1, p. 21–27, 2018.

SINGH, J.; NAIK, K.; MAHINTHAN, V. Impact of developer choices on energy consumption of software on servers. **Procedia Computer Science**, Elsevier, v. 62, p. 385–394, 2015.

SOUPPAYA, M.; MORELLO, J.; SCARFONE, K. Application container security guide. **NIST Special Publication**, v. 800, p. 190, 2017.

SULTAN, S.; AHMAD, I.; DIMITRIOU, T. Container security: Issues, challenges, and the road ahead. **IEEE Access**, IEEE, v. 7, p. 52976–52996, 2019.

TANG, C.-J.; DAI, M.-R. Dynamic computing resource adjustment for enhancing energy efficiency of cloud service data centers. In: IEEE. **2011 IEEE/SICE International Symposium on System Integration (SII)**. [S.l.], 2011. p. 1159–1164.

VISHWANATH, A. et al. Energy consumption comparison of interactive cloud-based and local applications. **IEEE Journal on selected areas in communications**, v. 33, n. 4, p. 616–626, 2015.

VOHRA, D. **Amazon Fargate Quick Start Guide: Learn how to use AWS Fargate to run containers with ease**. [S.l.]: Packt Publishing Ltd, 2018.

WAZLAWICK, R. **Metodologia de pesquisa para ciência da computação**. [S.l.]: Elsevier Brasil, 2017. v. 2.

WEINS, K. **Cloud Computing Trends: 2020 State of the Cloud Report**. 2020. Disponível em: <https://www.flexera.com/blog/industry-trends/trend-of-cloud-computing-2020/>.

WU, C.; BUYYA, R.; RAMAMOHANARAO, K. Cloud pricing models: Taxonomy, survey, and interdisciplinary challenges. **ACM Computing Surveys (CSUR)**, ACM, v. 52, n. 6, p. 108, 2019.

YUAN, P.; GUO, Y.; CHEN, X. Uniport: A uniform programming support framework for mobile cloud computing. In: IEEE. **Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2015 3rd IEEE International Conference on**. [S.l.], 2015. p. 71–80.

ZAKARYA, M.; GILLAM, L. Managing energy, performance and cost in large scale heterogeneous datacenters using migrations. **Future Generation Computer Systems**, Elsevier, 2018.

ZHANG, J. et al. Sustainable green data center: Guaranteeing flow deadlines in chains of virtual network functions with mrouting. **Sustainable Computing: Informatics and Systems**, 2018. ISSN 2210-5379. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2210537917304274>.