Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 29, Num. 02 (2022) 63-83

RESEARCH ARTICLE

Stochastic Models for Planning VLE Moodle Environments based on Containers and Virtual Machines

Modelos Estocásticos para o Planejamento de Ambientes AVA Moodle Baseados em Contêineres e Máquinas Virtuais

Cleyton Gonçalves [1]*, Ermeson Andrade [1], Júlio Mendonça[2], Gustavo Callou[1]

Abstract: Moodle Virtual Learning Environments (VLEs) represent tools of a pedagogical dimension where the teacher uses various resources to stimulate student learning. Content presented in hypertext, audio or video formats can be adopted as a means to facilitate the learning. These platforms tend to produce high processing rates on servers, large volumes of data on the network and, consequently, degrade performance, increase energy consumption and costs. However, to provide efficient sharing of computing resources and at the same time minimize financial costs, these VLE platforms typically run on virtualized infrastructures such as Virtual Machines (VM) or containers, which have advantages and disadvantages. Stochastic models, such as stochastic Petri nets (SPNs), can be used in the modeling and evaluation of such environments. Therefore, this work aims to use analytical modeling through SPNs to assess the performance, energy consumption and cost of environments based on containers and VMs. Metrics such as throughput, response time, energy consumption and cost are collected and analyzed. The results revealed that, for example, a cluster with 10 replicas, occupied at their maximum capacity, can generate a 46.54% reduction in energy consumption if containers are used. Additionally, we validate the accuracy of the analytical models by comparing their results with the results obtained in a real infrastructure.
Keywords: Container — Virtual Machine — Performance Evaluation — Energy Consumption — Cost Evaluation — Stochastic Petri Nets

Resumo: Os ambientes virtuais de aprendizagens (AVA) *Moodle* representam ferramentas de dimensão pedagógica onde o professor utiliza vários recursos para estimular a aprendizagem dos alunos. Os conteúdos apresentados em formatos de hipertextos, áudios ou vídeos podem ser adotados como meios para facilitar a aprendizagem. Essas plataformas tendem a produzir elevadas taxas de processamento nos servidores, grandes volumes de dados na rede e, consequentemente, degradam o desempenho, aumentam o consumo de energia e os custos. Todavia, para fornecer o compartilhamento eficiente de recursos computacionais e ao mesmo tempo minimizar os custos financeiros, essas plataformas de AVA normalmente são executadas em infraestruturas virtualizadas como as máquinas virtuais (VM) ou os contêineres, as quais possuem vantagens e desvantagens. Modelos estocásticos, como as redes de Petri estocásticas (SPNs), podem ser usados na modelagem e avaliação de tais ambientes. Dessa forma, este trabalho utiliza as SPNs para avaliar o desempenho, o consumo de energia e o custo de ambientes baseados em contêineres e VMs. Métricas como vazão, tempo de resposta, consumo de energia e custo são coletadas e analisadas. Os resultados revelaram que, por exemplo, um agrupamento com 10 réplicas, ocupado na sua vazão máxima, pode gerar uma redução de 46,54% na demanda de energia elétrica caso os contêineres sejam usados. Adicionalmente, validamos a acurácia dos modelos analítico comparando os resultados gerados nas análises com os experimentos obtidos na infraestrutura real.
Palavras-Chave: Contêiner — Máquina Virtual — Avaliação de Desempenho — Consumo de Energia — Avaliação de Custo — Redes de Petri Estocásticas

[1] *Departamento de Computação, Universidade Federal Rural de Pernambuco (UFRPE), Recife - Pernambuco, Brasil*
[2] *Coordenação de Informática, Instituto Federal de Alagoas (IFAL), Maceió - Alagoas, Brasil*
*__Corresponding author__: cleyton.goncalves@ufrpe.br
DOI: http://dx.doi.org/10.22456/2175-2745.119196 • **Received:** 09/10/2021 • **Accepted:** 13/04/2022

1. Introdução

Os *data centers* de alta performance possuem infraestruturas para suportar altas taxas de utilização dos recursos em virtude da demanda em larga escala. Esses ambientes são compostos por diferentes componentes computacionais, como servidores convencionais, equipamentos de processamento (ex.: *clusters* com *blades*) e armazenamento de dados (ex.: *storages*), dispositivos de rede (ex.: roteadores e *switches*), alimentadores de energia e sistema de refrigeração. Todos esses componentes consomem uma grande quantidade de energia e custo para manter a operação. Nos EUA, no ano de 2018, por exemplo, o consumo de energia anual dos *data centers* representou cerca de 200 terawatt-hora (tWh), ou seja, 1% da demanda de energia elétrica global [1]. Além disso, a Nature [1] pontuou que as emissões de carbono geradas pela indústria global de TIC representa mais de 2% das emissões globais de carbono.

A pandemia do coronavírus trouxe várias mudanças no cotidiano do mundo em relação à mobilidade da população, devido às determinações governamentais de distanciamento social para conter a disseminação do vírus SARS-CoV-2. Em virtude da necessidade de permanecer mais tempos nas residências, as pessoas passaram a usar Internet com mais frequência para fazer diversas atividades online, como compras em lojas virtuais, consultas médicas via telemedicina, jogos online, serviços de *Streaming* para entretenimento e videoconferências para diversas finalidades, como estudos online na modalidade de ensino a distância (EAD) e trabalhos em *home office*. Esse último levou uma grande parcela dos negócios migrarem as suas atividades para a modalidade de teletrabalho, a qual contribuiu para o crescimento considerável na demanda de Internet das residências através de aplicações nas nuvens [2].

Nos EUA, no ano de 2021, uma pesquisa da Amdocs [2] mostrou que os jogos online tiveram um aumento de 49% em relação ao período pré-pandemia do virus SARS-CoV2, seguidos por serviços de *Streaming* de vídeo com 48% e EAD com 45%. Isso representou um aumento na taxa de utilização dos recursos e na demanda de energia elétrica dos *data centers* [2]. Na educação, os setores públicos e privados também foram afetados em todos os níveis de ensino, migrando a continuidade das atividades de ensino para a modalidade de educação à distância (EAD) através da utilização de plataformas baseadas em AVAs. Como consequências, as instituições acadêmicas tiveram de expandir as capacidades de operações dos seus *data centers* para suportar o aumento na demanda de milhares estudantes que utilizaram recursos digitais como meios de aprendizagem [3].

Um tipo de AVA amplamente adotado no âmbito acadêmico, é o *Modular Object Oriented Dynamic Learning Environment*, ou simplesmente *Moodle* [4]. Todavia, o ajuste incorreto na capacidade da infraestrutura que provisiona tal plataforma pode apresentar problemas de queda no desempenho da operação em virtude do subdimensionamento dos recursos. Por outro lado, o superdimensionamento pode gerar alocação desnecessárias de recursos que poderiam ser destinados a outras finalidades [5]. As operações com *Moodle* normalmente são executadas em infraestruturas virtualizadas baseadas em máquinas virtuais (VM) ou contêineres. No entanto, a VM pode acrescentar uma sobrecarga desnecessária no desempenho geral do sistema operacional hospedeiro, visto que cada instância baseada em VM possui seu próprio sistema operacional (SO) [6]. Por outro lado, a tecnologia contêiner preconiza baixa utilização dos recursos de hardware, visto que dispensa o uso do *hypervisor*, VMs e sistemas operacionais convidados [7], isto é, os contêineres são carregados na própria camada do SO da máquina física. Adicionalmente, a virtualização através das VMs permite o trabalho com diversos SOs em um mesmo ambiente, enquanto o contêiner possui a dependência do SO que ele está rodando. Como cada um desses ambientes virtualizados possuem suas especificidades e devem atender às necessidades dos mais variados negócios, se faz necessário estudar os *trade-offs* relativos ao desempenho, consumo de energia e custo dessas infraestruturas.

A modelagem analítica é uma técnica poderosa que tem sido empregada para representar e analisar o desempenho de diversos tipos de sistemas complexos, incluindo sistemas computacionais, como *data centers* [8–10]. O uso da modelagem analítica pode ajudar a analisar diferentes aspectos referente ao desempenho, a confiabilidade e a eficiência energética no contexto dos sistemas computacionais. Dentre os vários tipos de modelagens analíticas, como cadeia de *Markov*, árvore de decisão, diagrama de blocos de confiabilidade (*Reliability block diagram* - RDB), entre outros, destaca-se as redes de Petri (RdP) e suas extensões [9], as quais correspondem a formalismos amplamente adotados para representar os estados de sistemas através de uma notação formal composta por elementos gráficos. Esses modelos podem ser criados para representar abstrações de sistemas complexos cujos comportamentos podem denotar concorrência, paralelismo, assincronicidade, distribuição, etc [10]. Vale destacar que as RdPs podem ser usadas para encontrar possíveis pontos de gargalos nas infraestruturas sob análise, realizar o planejamento de capacidade e provê informações para os projetistas/administradores na tomada de decisão.

Sendo assim, este artigo propõe uma abordagem integrada com experimentos e modelos baseados em *Stochastic Petri Nets* (SPNs) - em português, redes de Petri estocásticas [9, 10] - para avaliar o desempenho, o consumo de energia e o custo das aplicações virtualizadas em contêineres e VMs. Os modelos SPNs propostos têm como principal objetivo avaliar os *trade-offs* de infraestruturas que utilizam o AVA *Moodle* referente ao desempenho, consumo de energia e custo. Para isso, utilizamos o modelo proposto para obter métricas quantitativas como a vazão, o tempo de resposta e o consumo a energia. Complementarmente, realizamos experimentos em uma infraestrutura real para avaliar uma operação equivalente à plataforma *Moodle* provida por ambientes baseados em contêineres e VMs. Tais experimentos consideram diferentes cenários e cargas de trabalhos. As análises mostram que os modelos apresentam resultados precisos e consistentes

aos valores obtidos pelo sistema real. Adicionalmente, os custos para os cenários avaliados foram calculados com base na tarifa média nacional do kilowatt-hora (KWh) [11]. Os resultados obtidos através da abordagem proposta poderão apoiar as organizações ou quaisquer partes interessadas, como administradores, especialistas, projetistas e gestores, em um melhor planejamento de infraestruturas virtualizadas para mitigar eventuais sobrecargas ou desperdícios de recursos.

O restante deste artigo está organizado como segue. A Seção 2 descreve os principais conceitos usados neste trabalho. A Seção 3 descreve os trabalhos relacionados à avaliação de desempenho e o consumo de energia dos ambientes baseados em contêineres e VMs. A Seção 4 apresenta a metodologia. A Seção 5 apresenta os componentes e detalhes sobre a arquitetura experimental. A Seção 6 descreve o modelo SPN desenvolvido. A Seção 7 apresenta uma discussão dos resultados. Por fim, a Seção 8 apresenta as conclusões e os trabalhos futuros.

2. Fundamentação Teórica

Esta seção apresenta os conceitos fundamentais empregados neste trabalho. A seguir são apresentados conceitos acerca da virtualização, da conteinerização, do AVA, da modelagem usando as redes de Petri estocásticas.

2.1 Ambientes Virtuais de Aprendizagem

Os AVAs são definidos como softwares desenvolvidos para auxiliar os professores na promoção do ensino e da aprendizagem de forma virtual, tendo como principais características a interatividade, a hipertextualidade e a conectividade [12]. As instituições acadêmicas podem adotá-los de acordo com a necessidade do seu contexto educacional para promover cursos nas modalidades presenciais, semipresenciais ou à distância [13]. Os AVAs são considerados como sistemas de ensino e aprendizagem integrados, sendo capazes de promover o interesse do aluno na aprendizagem e complementar a interação em sala de aula com professor [13].

Os AVAs mais conhecidos são o *Moodle*, o TelEduc, o AulaNet, o Claroline, o BlackBoard, o E-Proinfo e o Amadeus. No entanto, entre eles, apenas o *Moodle*, o Claroline e o Amadeus possuem licenças de código aberto [12]. Destaca-se o *Moodle* com cerca de 181 mil instâncias (sites), distribuídas entre 234 países, 35 milhões de cursos e 262 milhões de usuários inscritos. O *Moodle* está entre as principais plataformas de aprendizagens online, desenvolvida sob licença de software livre e de código aberto [14]. Em decorrência dessas vantagens, este trabalho adotará o AVA *Moodle* para modelagem e análise.

2.2 Virtualização

A virtualização refere-se ao conjunto de tecnologias que permite um único computador real hospedar múltiplas máquinas virtuais, onde cada instância significa um SO diferente em execução sobre o hardware subjacente compartilhado [15]. A virtualização funciona no sentido de otimizar a utilização do hardware e minimizar a ociosidade dos recursos computacionais [16]. Ela torna possível a utilização da capacidade total de uma máquina física, distribuindo recursos para múltiplos ambientes, como servidores de aplicação, serviços de armazenamento e serviços de rede de acordo com a necessidade [17]. Dessa forma, com o uso da virtualização, há uma tendência de economia na aquisição de hardware, uma vez que os recursos computacionais podem ser alocados de forma mais aprimorada e econômica [15].

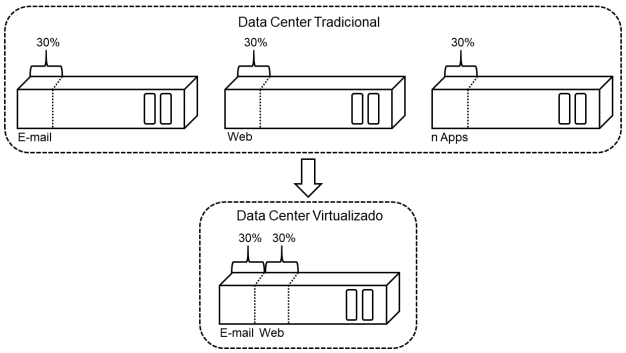

Figura 1. Exemplo de três servidores tradicionais sendo virtualizados para um computador hospedeiro.

A parte superior da Figura 1 apresenta um cenário cuja infraestrutura é composta por três servidores físicos com SOs *Linux Red Hat* e com as mesmas configurações de recursos, mas cada um deles com finalidades específicas. O primeiro é um servidor de e-mail baseado em PHP, o segundo é um servidor web com uma aplicação Java e o terceiro executa diferentes aplicações. Nesse cenário hipotético, a Red Hat [16] afirma que de um modo geral tais serviços consomem cerca de 30% da capacidade de cada uma das máquinas físicas em que os serviços estão instalados. Assim, cerca de 70% dos recursos das mesmas ficam ociosos. Os valores elevados de ociosidade resultam em custos desnecessários que podem prejudicar a manutenção da operação do provedor de serviço. Esse problema pode ser resolvido com a criação de duas VMs que podem ser provisionadas simultaneamente com diferentes SOs, compartilhando o hardware de uma única máquina físico [18]. Dessa forma, as migrações dos serviços de *e-mail* e *web* juntos utilizariam cerca de 60% da capacidade dos recursos. Por consequência, ainda restariam cerca 40% de recursos computacionais da máquina física para eventuais aumentos de demanda das aplicações elásticas provisionadas por instâncias virtualizadas.

Uma máquina virtual pode ser definida como uma réplica eficiente e isolada de um computador real com SO, aplicativos e serviços. Elas podem simular vários computadores ou servidores virtuais hospedados em um determinado computador físico, proporcionando aos usuários as mesmas experiencias de um ou mais computadores reais. As VMs apresentam características como particionamento dos recursos do sistema entre múltiplas máquinas virtuais e a execução de diversos SOs na máquina física, incluindo as características de isolamento de

falhas, segurança no nível de hardware, preservação no desempenho do sistema, otimização na alocação e gerenciamento dos recursos.

Para criar uma VM, é necessária a instalação de um software chamado de *hypervisor*, o qual representa uns dos principais componentes de um sistema virtualizado. Os *hypervisors* representam uma camada de software que dissocia as VMs da máquina física hospedeira e aloca os recursos de hardware dinamicamente ou estaticamente para cada VM criada. Eles possuem implementações baseadas em softwares e hardwares que permitem criar, executar e gerenciar as máquinas virtuais [19]. Este artigo fez a opção por adotar o KVM [20] dada a sua disponibilidade nativa para o *Linux*, o qual integra facilmente as funções de virtualização assistida por hardware ao *kernel*, tornando SO um *hypervisor* do tipo 1.

2.3 Conteinerização

A conteinerização é definida como um mecanismo que empacota arquivos, bibliotecas e dependências de aplicações por uma tecnologia denominada de contêineres, a qual abrange características de portabilidade, escalabilidade, configurabilidade, isolamento e economicidade na utilização dos recursos [21]. Isso permite que as distribuições dos contêineres entre os ambientes de desenvolvimento e operação sejam conduzidas com máxima agilidade nas entregas, sem causar prejuízo a produção [21]. Diferentemente das tecnologias virtualização tradicionais, os contêineres não exigem uma camada de emulação ou um *hypervisor* para serem executados. Em vez disso, eles utilizam uma interface no nível do SO. Essa característica os torna uma tecnologia enxuta tendo em vista que exigem uma sobrecarga limitada para executá-los [22]. Isso permite que um número maior de instâncias sejam executadas na máquina física [22]. Além disso, os contêineres são carregados no espaço do usuário sobre o *kernel* do SO. Tal característica garante uma redução considerável na sobrecarga dos recursos e aumenta o desempenho das tarefas executadas nos ambientes baseados em contêineres, visto que as instâncias são executadas diretamente no SO hospedeiro sem redirecionar as instruções para o *hypervisor* [23].

Os contêineres armazenam apenas os arquivos necessários à execução de um ou mais processos empacotados por uma imagem inicializada isoladamente no SO hospedeiro. A imagem inicializada no contêiner difere de um SO inicializado na VM, visto que uma imagem possui apenas as dependências e os binários necessários para carregar os serviços e os artefatos da aplicação, construído a partir dos códigos fontes. Os contêineres conseguem implantar ambientes praticamente em tempo de execução, visto que a tecnologia dispensa a necessidade do *hypervisor*, o qual é imprescindível para realizar as tarefas de gerenciamento das VMs, desda criação, inicialização, desligamento até a destruição delas [21].

A Figura 2 mostra as diferenças essenciais entre ambientes baseados em VMs (a) e os em contêineres (b). Considerando que um hardware subjacente é utilizado para as duas tecnologias, as camadas de softwares para um ambiente que

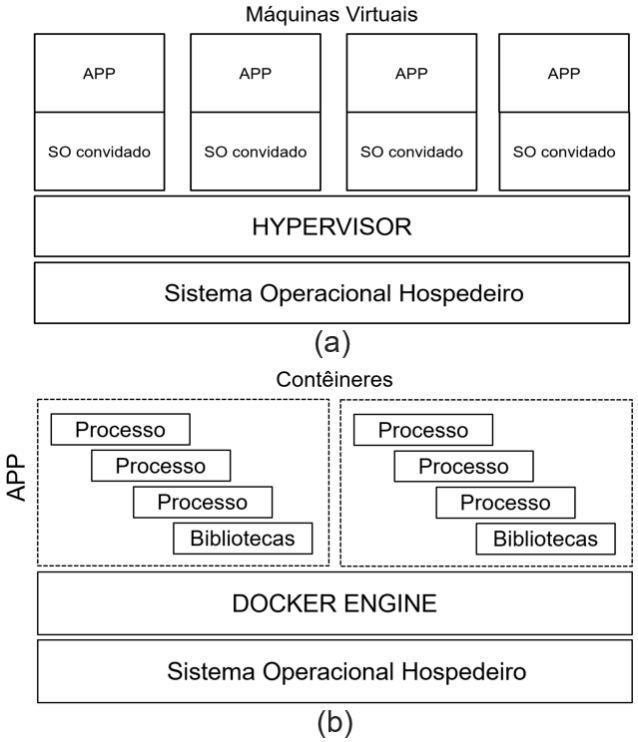

Figura 2. Exemplos de ambientes utilizando a máquinas virtuais (a) e contêineres (b).

adota a virtualização baseada em VMs depende de seus respectivos SOs convidados e um *hypervisor* para redirecionar as instruções entre o hardware subjacente e as VMs. Enquanto, o ambiente que adota a conteinerização é composto pelos contêineres com seus respectivos processos, bibliotecas, dependências e binários da aplicação, o mecanismo de distribuição (*Docker engine*) e o *kernel* do SO hospedeiro. Isso garante mais agilidade em todos os estágios no desenvolvimento de aplicações, desde a construção de ambientes de testes ou homologação até a entrega para o ambiente de produção.

O *Docker Engine* é uma ferramenta capaz de realizar virtualização a nível de sistema operacional, popularmente conhecida como conteinerização. O *Docker* possibilita o empacotamento de aplicações ou ambientes completos dentro um ou mais contêineres, tornando-os portáveis para qualquer outro ambiente baseado na tecnologia. Essa característica permite a criação, o teste e a implantação de aplicações rapidamente, reduzindo drasticamente o tempo de *deployment* [22]. O *Docker* oferece uma API com variadas funcionalidades que podem ser manipuladas via arquivos de especificação ou interface de linha de comando (CLI - *command-line interface*) que permitem acessar, criar, inicializar, parar ou destruir contêineres, bridge de rede, volumes de armazenamento, imagens entre outros. No caso das imagens, os comandos podem construí-las (ou reconstruí-las), renomeá-las, apagá-las, baixá-las ou compartilhá-las em repositórios locais ou remotos, bem como públicos ou privados [21]. Outra funcionalidade do *Doc-*

ker é a criação de imagens a partir de instruções predefinidas em arquivos denominados de *Dockerfiles*. Esse arquivo pode conter instruções relacionadas aos parâmetros de configuração do ambiente, bem como os artefatos da aplicação [21].

As aplicações baseadas no AVA Moodle tendem a receber elevadas quantidades de requisições dada a sua vasta adesão pelas instituições de ensino em virtude de oferecer experiências positivas aos alunos. No entanto, equilibrar a capacidade da infraestrutura com a demanda dos usuários sem causar queda no desempenho ou desperdícios de recursos computacionais e financeiros é uma tarefa complexa. É importante balancear a experiência positiva dos alunos com a redução de excessivos de custos para manter a qualidade na entrega dos serviços, eliminando eventuais prejuízos injustificáveis para os negócios. Assim, as redes de Petri estocásticas podem ser adotadas para auxiliar na modelagem e análise de cenários de acordo determinadas cargas de trabalhos que melhor garantam os trade-offs de desempenho, consumo de energia e custo.

2.4 Modelagem e análise utilizando SPNs

As RdP podem ser definida como um conjunto de formalismos adotados para representar graficamente os estados de sistemas cujos comportamentos são caracterizados por concorrência, paralelismo, assincronicidade, distribuição, determinísticos ou estocásticos [9, 10]. A notação formal das RdP fornece um conjunto de elementos gráficos que constroem modelos de abstrações de sistemas complexos. Como as demostrações matemáticas podem tornar as análises numéricas menos intuitivas, as representações gráficas das RdP facilitam uma melhor compreensão entre as atividades dos componentes do sistema. Contudo, as RdP possuem uma base matemática consolidada por equações de estados e algébricas que servem para demonstrar todo o comportamento do sistema [8, 24]. Além disso, ao longo dos anos, desde a primeira versão das RdP [25] até hoje, elas vêm recebendo várias atualizações através de contribuições propostas por outros pesquisadores, incluindo características de temporização, orientação a objetos, estrutura de dados, entre outras [8].

As RdP são constituídas por 4 elementos gráficos básicos, tais como lugar, transição, arco e *token* (marca). Os lugares são representados por círculos, as transições por retângulos pretos, os arcos por setas e os *tokens* por pontos (ver Figura 3). Os lugares (Figura 3 (a)) são repositórios utilizados para armazenar os *tokens* que, por sua vez, indicam marcações para representar o estado do sistema em determinado instante. As transições (Figura 3 (b)) são ações (ou eventos) realizadas para alterar o estado do sistema. Quando os lugares possuem *tokens* armazenados e arcos conectados às transições, torna-se satisfeita uma precondição de disparo da transição. Dessa forma, os arcos (Figura 3 (c)) representam o fluxo de passagem dos *tokens* pela rede. Após o disparo, alguns lugares terão suas informações alteradas (*tokens*) de uma transição, ou seja, ocorrerá uma pós-condição para que outras transições do modelo sejam habilitadas para disparar os demais eventos do sistema [8–10, 24].

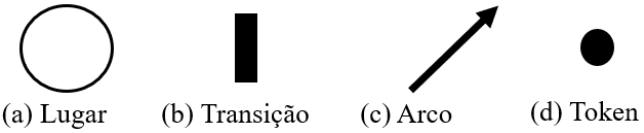

Figura 3. Elementos da rede de Petri.

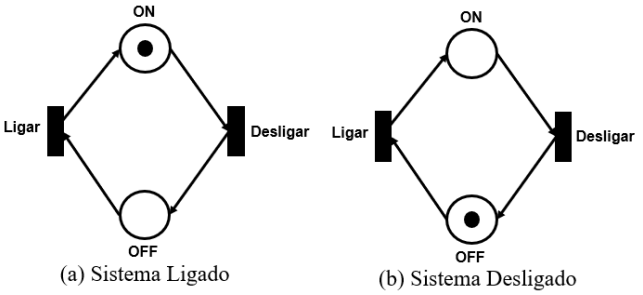

Figura 4. Exemplo de uma rede de Petri

A Figura 4 mostra uma representação gráfica de um modelo que pode atingir dois tipos estados (ON e OFF). Consequentemente, o modelo é composto pelos lugares `ON` e `OFF` que representam os estados do sistema. As transições `Ligar` e `Desligar` representam os eventos que alteram os estados do sistema. A Figura 4 (a) mostra o estado inicial do sistema onde o lugar `ON` contém um `token`. O arco saindo do lugar `ON` em direção à transição `Desligar`, a torna apta para o disparo em virtude do `token` armazenado no lugar `ON` satisfazer a precondição de habilitação dela. O modelo mostrado na Figura 4 (b) representa o outro possível estado atingido pelo sistema quando ocorrer o disparo do `token` pela transição `Desligar` em direção ao Lugar `OFF`. Nesse caso, a transição `Desligar` ficará inabilitada e o `token` armazenado no lugar `OFF` habilitará o disparo da transição `Ligar`, tornando-a apta a retirar um `token` do lugar `OFF` e armazená-lo novamente no lugar `ON`.

A SPN (*Stochastic Petri Net*) é uma variante das redes de Petri que propõem a incorporação de tempo associados aos atrasos das transições. Dentre os modelos estocásticos, as SPNs possuem uma destacada adesão em vários campos do conhecimento devido a sua extensa aplicabilidade nos contextos da avaliação de desempenho, da disponibilidade e da dependabilidade [9]. Com as SPNs, tornou-se possível a atribuição de tempos às transições para que fosse possível modelar eventos temporizados intervalares, determinísticos, não determinísticos ou estocásticos [24]. Nessas redes, destacam-se as transições imediatas e as exponenciais. As imediatas são caracterizadas por tempos de disparos igual a zero ou nulo, ao passo que as exponenciais são caracterizadas por tempos de disparo atribuídos à variáveis aleatórias com distribuição exponencial [10]. A Figura 5 (a) corresponde à transição estocástica cujo elemento gráfico é representado por um retângulo branco. Nesta transição é possível atribuir tempos exponencialmente distribuídos relacionados a cada evento do sistema. Ao passo que, a Figura 5 (b) corresponde à transição imediata cujo elemento gráfico é representado por

(a) Transição Estocástica (b) Transição Imediata

Figura 5. Elementos da SPN

um retângulo preto fino. Essa transição não possui o atributo de tempo associado ao disparo de transições. Porém, possui prioridade sobre as transições temporizadas para realizar os disparos [8–10, 24].

3. Trabalhos Relacionados

Esta seção apresenta alguns trabalhos relacionados que têm sido desenvolvidos para avaliação de desempenho, custo e consumo de energia de infraestruturas computacionais baseadas em contêineres e VMs. Primeiramente, iremos apresentar os relacionados ao consumo de energia. Em seguida, abordaremos os trabalhos relacionados à avaliação de desempenho. Por fim, serão apresentados trabalhos relacionados aos dois temas citados anteriormente. Estes trabalhos foram explorados com o auxílio de outras técnicas importantes, como experimentações, metodologias, heurísticas baseadas em algoritmos, modelos e validações. Apesar de boa parte dos trabalhos relacionados terem pontos de convergência com um ou mais temas este artigo, eles não cobrem na totalidade os temas explorados neste estudo.

3.1 Consumo de energia

Fieni et al. [26] desenvolveram um software medidor de energia chamado *Smartwatts*. O medidor pode calibrar automaticamente os modelos de potência da CPU (*Central Processing Unit*) e da memória RAM (*Random Access Memory*), explorando os dados das medições de energia oriundos do software RAPL, o qual converte os registros dos contadores de desempenho da CPU e da memória RAM para valores de potência. Nos experimentos, o *Smartwatts* conseguiu realizar medições com erros limitadas a 5 *Watts* para a CPU e 1 *Watt* para a memória RAM, considerando ciclos de *clocks* de 2 Hz. Com esses resultados, foi possível estimar potências com erros inferiores a 3 *Watts* para a CPU e 0,5 *Watts* para a memória RAM.

Phung et al. [27] propuseram um medidor virtual chamado de cWatts++, o qual foi capaz de estimar o consumo de energia através de algoritmos que implementam funções matemáticas usando dados oriundos dos contadores de eventos gerados pelos recursos do computador. Um modelo matemático chamado de raplModel monitorou os eventos relacionados ao consumo de energia atribuídos à CPU e a RAM. Os resultados obtidos pela ferramenta desenvolvida foram comparadas com as medições reais do medidor *Cabac Power-Mate*, apresentando um erro menor que 5%.

Zheng at al. [28] adotaram uma serie de algoritmos visando melhor aproveitar as implantações dos contêineres conforme o percentual de utilização da CPU dos nós alocados para um *cluster* criado pelo simulador de nuvem *CloudSim4.0*. A ideia central foi diminuir a frequência de migrações dos contêineres entre os nós para provocar uma queda no consumo de energia. Os experimentos foram realizados por diferentes tipos de algoritmos com finalidades de distribuir a demanda por recursos entre os nós de forma equilibrada. Alguns algoritmos apresentaram uma redução de aproximadamente 2% no consumo de energia. Os resultados mostraram que as VMs demandaram cerca de 28% a mais no consumo de energia. Enquanto, os contêineres demandaram apenas 1,3% a mais no consumo de energia.

Chen et al. [29] utilizaram o simulador de nuvem *CloudSim4.0* para avaliar uma estratégia de distribuição de instâncias baseadas em contêineres dentro de VMs. Os autores desenvolveram um algoritmo que cria uma lista ordenada que prioriza as VMs mais ociosas em detrimento das VMs menos ociosas, observando a taxa de utilização da CPU e da memória RAM. O algoritmo faz varreduras repetitivas em listas de classificação conforme o percentual de utilização da CPU e da Memória tanto das VMs quanto dos contêineres. A estratégia criada aproveita a disponibilidade dos recursos nas VMs para suportar o máximo possível de contêineres sem comprometer o desempenho geral da operação. Os resultados apresentaram uma otimização da utilização dos recursos e reduziu o consumo de energia em 12,8%.

3.2 Avaliação de Desempenho

Lin et al. [30] propuseram algoritmos baseados na heurística gulosa e na programação dinâmica com propósitos diferentes para encontrar a solução ideal na distribuição dos contêineres entre máquinas físicas. Os algoritmos utilizaram o percentual de ociosidade da CPU para realizar tal distribuição. A heurística *Consecutive Allocation* se destacou nos experimentos, levando 0,123 segundo para encontrar o consumo de energia ideal. Enquanto, o algoritmo de programação dinâmica e a heurística gulosa *decelerate* levaram 1,824 e 0,990 segundo, respectivamente. Já a heurística gulosa *decelerate* apresentou melhores resultados para cenários com requisitos de escalabilidade.

Yadav et al. [31] propuseram uma metodologia para comparar o desempenho das VMs *VMWare* e dos contêineres *Docker*. O tempo de execução da memória foi avaliado através de transferências de arquivos com tamanhos variados. O tempo de execução da CPU foi observado através dos cálculos de números primos com intervalos numéricos variados. Os resultados demonstraram que os desempenhos dos contêineres sobre as VMs representaram diferenças médias de 1,1% e 0,69% no tempo de execução da memória e da CPU, respectivamente. Assim, a diferença do tempo de execução entre os contêineres e as VMs sobre os cenários apresentados não indicaram uma superioridade significativa de ambas tecnologias.

Salah et al. [32] abordaram uma avaliação de desempenho utilizando a infraestrutura como serviço (*Infrastructure as a service*) da *Amazon Cloud Plataform* para distribuir serviços

em contêineres e VMs. As duas tecnologias executaram os serviços *Web* dentro dos ambientes da *Amazon EC2* e *Amazon ECS*, respectivamente. O ambiente de experimentação foi estruturado em três cenários com variações de escopos contendo uma, duas e três instâncias de servidores *Apache Web*. A análise de desempenho dos serviços *Web* baseados em contêineres e VMs levaram em consideração as seguintes métricas: vazão, tempo de resposta e utilização da CPU. As cargas de trabalhos foram requisições enviadas simultaneamente aos servidores web durante intervalos de 20 segundos. Os resultados da vazão e do tempo de resposta mostraram melhores desempenhos das VMs avaliadas nos três cenários. Todavia, é importante destacar que os contêineres do ECS operam sobre VMs. Por essa razão, a inferioridade no desempenho dos contêineres está atrelada aos *overheads* adicionais provocados pelas VMs que hospedaram os contêineres.

Bhimani et al. [33] analisaram e compararam o desempenho de soluções *Big Data* provisionadas por contêineres e VMs. As instâncias utilizaram o *framework Apache Spark* que representa uma estrutura de processamento para grandes quantidades de dados em memória, substituindo as alternativas baseadas em disco rígido. Os *clusters* compostos por Nós *Spark* suportaram a execução de várias tarefas paralelas, visando concorrer os recursos computacionais dos núcleos das CPUs, memórias e discos rígidos. Os *benchmarks* tiveram funções de avaliar o desempenho de aplicações baseadas em aprendizagem de máquinas, computação de gráfica e consultas em SQL. Os experimentos observaram o percentual de utilização da CPU, da memória e do disco rígido conforme a classificação de cada *benchmark*. Os experimentos conduzidos nos contêineres apresentaram resultados melhores para a maioria dos *benchmarks* executados. Todavia, observou-se que os testes de *K-Means* realizados no *cluster* das VMs utilizaram 2% da CPU, ao passo que o *cluster* dos contêineres utilizaram 5% da CPU.

Barik et al. [34] apresentaram uma avaliação de desempenho entre virtualização e a conteinerização através do *Oracle Virtual Box* e o *Docker*. O estudo avaliou as duas tecnologias com ênfase na largura de banda de leitura e de escrita nas memórias do tipo: *cache* de nível I, *cache* de nível II e a RAM. O desempenho da rede foi medido através de transferências de arquivos via protocolos TCP *(Transmission Control Protocol)* e UDP *(User Datagram Protocol)*. A vazão em MB/s alcançada na operação de escrita da memória alocada para as duas tecnologias atingiu uma diferença de 149% a favor do contêiner. No entanto, a vazão na operação de leitura entre as duas tecnologias apresentou uma leve diferença 1,73% a favor do contêiner. Os tempos demandados para realizar os testes de requisições HTTP apresentaram diferença em torno de 83,04% a favor do contêiner. Assim como, o tempo demandado para realizar os testes de renderizações das páginas PHP *(Personal Home Page)* apresentaram diferença em torno de 112,83% a favor do contêiner. No entanto, a VM superou o contêiner em 58,01%, quando os testes de requisições HTTP *(Hypertext Transfer Protocol)* incluíram a criptografia com cifra de bloco e chaves simétricas do tipo AES *(Advanced Encryption Standard)* de 256 bits. Nos testes de emissões de certificados SSL, os contêineres emitiram um valor percentual de 126,64% a mais do que as VMs.

3.3 Avaliação de Desempenho e de Consumo de energia

Cordero et al. [23] compararam o desempenho entre o KVM e o *Docker*. O trabalho focou na eficiência energética e na qualidade de serviço QoS *(Quality of Service)* para uma infraestrutura em nuvem. O tempo de resposta foi a principal métrica utilizada para avaliar o QoS da rede entre as duas tecnologias. Os resultados mostraram que os contêineres apresentaram um ganho de 26% no tempo de resposta, se comparados aos resultados das VMs. Com relação ao consumo de energia, os resultados comprovaram que o *Docker* consome menos energia do que o KVM.

Brondolin et al.[35] propuseram uma ferramenta de monitoramento denominada de *DEEP-mon*, a qual é capaz de realizar medições de consumo de energia atribuídas aos contêineres em execução no SO hospedeiro. A ferramenta avalia os *trade-offs* entre a potência e o desempenho de um ambiente implantando pelo *Kubernetes*. Esse estudo adotou duas categorias de *benchmarks* compostos por três tipos de testes, que observaram o consumo de energia conforme cargas de trabalhos geradas por eles. A metodologia avaliou a correlação linear entre o consumo de energia e o desempenho através das amostras obtidas pelas medições dos contadores RAPL (Running Average Power Limit).

Tadesse et al. [36] realizaram uma avaliação de consumo de energia de ambientes baseados em contêineres *Docker*. O objetivo da pesquisa foi apresentar resultados sobre o percentual de utilização das CPUs e do impacto no consumo de energia quando servidores de contêineres são submetidos a elevadas taxas de transferências de arquivos na rede. O trabalho realizou experimentos considerando diferentes cenários de cargas de trabalhos que produziam transferências de dados para exaurir a capacidade total das interfaces de rede dos contêineres. Os resultados mostraram que o aumento na demanda da rede esteve associado ao percentual de utilização da CPU e, por consequência, resultou em um aumento no consumo de energia do computador hospedeiro.

Os trabalhos apresentados anteriormente apresentaram estratégias para avaliar o desempenho e o consumo de energia de contêineres e VMs através de softwares de medição, modelos matemáticos e algoritmos. Os estudos relacionados à avaliação de desempenho apresentaram resultados mais favoráveis aos contêineres quando os *benchmarks* demandam recursos da CPU, da memória e do disco rígido. Todavia, boa parte desses estudos apresentaram medidores baseados em softwares para estimar a potência demandada através de contadores de desempenho, implementados por sensores embutidos nos componentes da placa mãe, como CPU, memórias RAM e Cache. Diferentemente dos trabalhos apresentados, este trabalho apresenta uma abordagem baseada em modelos

Tabela 1. Relação entre a proposta deste artigo e outros trabalhos relacionados

	Avaliação de Desempenho	Avaliação de Consumo de Energia	Avaliação de Custo	Propõe uma Metodologia	Modelos Analíticos	Validação dos Dados	Ambientes em Contêineres	Ambientes em Máquinas Virtuais	Aplicação em Ambiente Real
Este artigo	✓	✓	✓	✓	✓	✓	✓	✓	✓
(FIENI et al., 2020) [26]		✓		✓	✓	✓	✓		
(Phung et al., 2020) [27]		✓			✓	✓	✓		
(Zheng at al., 2020) [28]		✓			✓	✓	✓	✓	
(Chen et al., 2019) [29]		✓			✓	✓	✓	✓	
(Lin et al., 2018) [30]	✓				✓		✓		
(Yadav et al., 2018) [31]	✓			✓			✓	✓	
(Brondolin et al., 2018) [35]	✓	✓					✓		
(Cordero et al., 2017) [23]	✓	✓		✓			✓	✓	
(Salah et al., 2017) [32]	✓						✓	✓	✓
(Bhimani et al., 2017) [33]	✓						✓	✓	
(Tadesse et al., 2017) [36]	✓	✓			✓		✓		
(Barik et al., 2016) [34]	✓						✓	✓	

e experimentos para avaliar o desempenho, o consumo de energia e o custo de ambientes baseados em contêineres e VMs. Além disso, o trabalho propõe modelos estocásticos baseados em redes de Petri para auxiliar o planejamento de capacidade de ambientes virtualizados considerando métricas como vazão, tempo de resposta, consumo de energia e custo.

4. Metodologia

Esta seção descreve a metodologia adotada neste trabalho para avaliar o desempenho, o consumo de energia e o custo de instâncias computacionais baseadas em contêineres e VMs utilizando um AVA. A metodologia faz uso de experimentos e modelos SPNs para avaliação de tais métricas, sendo compostas por nove etapas que orientam o processo, a saber: entendimento da aplicação, seleção das tecnologias, definição das métricas, escolha das ferramentas de monitoramento e coleta, experimentação no ambiente real, geração do modelo analítico, validação do modelo, geração de cenários e análises dos resultados. A seguir, conforme descreve a Figura 6, serão discutidas as etapas e as ferramentas utilizadas para auxiliar na realização de cada uma delas.

• **Entendimento da aplicação:** dada a grande adesão das plataformas baseadas em AVA na educação, esta etapa visa o estudo aprofundado do *Moodle* [12], que é uma das ferramentas de ensino online mais utilizadas para ministrar aulas de cursos oferecidos pelas instituições públicas e privadas. O objetivo deste estudo é compreender as funcionalidades essenciais da ferramenta, bem como todas as etapas de sua implantação, de modo a construir um ambiente similar às plataformas de ensino à distância (EAD).

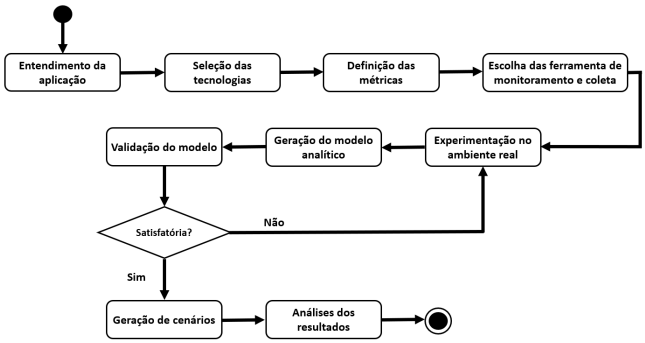

Figura 6. Metodologia Adotada.

• **Seleção das tecnologias:** nesta etapa, são identificadas todas as tecnologias necessárias para construir o ambiente do *Moodle* e distribuí-lo através de instâncias baseadas em contêineres e VMs. Neste momento, são feitos estudos prévios acerca de todas as tecnologias envolvidas no processo de montagem e configuração dos ambientes experimentais. As tecnologias estudadas compreendem medidores de potência elétrica, roteadores, sistemas operacionais, *hypervisors*, *containers engines*, sistemas de orquestração, serviços web, serviços de banco de dados, *proxy* reverso, *load balancers*, *benchmarks*, analisadores de desempenho de recursos computacionais e de tráfego de rede.

• **Definição das métricas:** as análises realizadas neste trabalho consideraram métricas relacionadas ao desempenho do sistema e ao consumo de energia variado no tempo e custo financeiro. No aspecto de desempenho, a vazão em requisições por segundo (rps) e o tempo de resposta são as métricas esco-

lhidas para obter os resultados deste estudo. Para o consumo de energia, a métrica escolhida é o kilowatt-hora variados em períodos mensais e anuais. Essa métrica será referenciada para derivar o custo da energia demandada pelos experimentos observados em cada cenário. Dessa forma, os diferentes cenários permitirão observar os comportamentos dessas métricas durante as operações do *Moodle* distribuídas pelas instâncias baseadas em contêineres e VMs. Adicionalmente, o modelo terá flexibilidade para calcular outras métricas baseadas nas fórmulas da Lei de Little [37] ou customizadas conforme a necessidade do projeto.

• **Escolha das ferramentas de monitoramento e coleta:** para realizar os experimentos em um ambiente real, é necessário utilizar *benchmarks* ou ferramentas que produzam cargas de trabalhos, monitorem o desempenho dos recursos, coletem as amostras referentes à vazão da aplicação e meçam o consumo de energia. Os *benchmarks* e as ferramentas adotadas neste trabalho são descritas a seguir:

a) **Ifstat Network Monitor:** é uma ferramenta de monitoramento de tráfego de rede [38]. O *ifstat* é utilizado para monitorar o desempenho da interface *Gigabit Ethernet* do computador que hospeda as instâncias do Moodle, a fim de identificar eventuais gargalos na rede.

b) **Nigel's Monitor (Nmon):** é uma ferramenta de monitoramento de desempenho para computadores com sistemas operacionais AIX e *Linux/Unix* [39]. O Nmon possui dois modos de funcionamento onde primeiro exibe uma tela com resumos estatísticos das métricas de desempenho do sistema e o segundo funciona em *background* onde as saídas do processo equivalem às amostras compondo as estatísticas de desempenho armazenadas em arquivos de tabulação. Essas amostras podem ser importadas em um arquivo de *Excel* com macros que transformam os dados em gráficos para auxiliar na análise e na compreensão das métricas vinculados aos recursos do computador hospedeiro que provisionou as instâncias do *Moodle*.

c) **Jmeter:** permite medir o desempenho de serviços fornecidas por diversos protocolos da camada de aplicação que utilizam a arquitetura TCP/IP, como aplicações web, banco de dados, e-mail, FTP *(File Transfer Protocol)*, ou LDAP *(Lightweight Directory Access Protocol)* [40]. Para a geração das cargas de trabalhos, os clientes Jmeter podem gerar diversos tipos de requisições, simulando *threads* com vários usuários virtuais. O Jmeter disponibiliza também uma variedade de relatórios (*listeners*) estatísticos compostos por tabelas e gráficos bastantes úteis nas análises dos resultados.

d) **Wattsup Power Meter:** consiste em um equipamento medição de potência elétrica [41]. Utilizando esse equipamento, é possível coletar amostras da potência elétrica demandada pela infraestrutura experimental e assim verificar o consumo de energia de tal infraestrutura.

• **Experimentação no ambiente real:** nesta etapa, uma arquitetura experimental é construída para que os recursos computacionais sejam alocados nos ambientes baseados em contêineres e VMs. Nesse caso, os recursos alocados têm a finalidade de provisionar ambientes com capacidades que podem variar conforme o andamento de cada cenário experimental. Esses cenários compreendem quantidades iguais de instâncias baseadas em contêineres e VMs que, por sua vez, possuem as mesmas configurações de recursos. Nos experimentos são utilizados um conjunto de ferramentas de medição que realizam coletas de amostras contendo as métricas selecionadas. Sendo assim, a capacidade de cada cenário deve suportar cargas de trabalhos suficientes para gerar amostras relacionados às métricas selecionadas e, posteriormente, os resultados são submetidos a análises e comparações em outra etapa.

• **Geração do modelo analítico:** nesta etapa, reúnem-se todas as informações necessárias para iniciar o passo a passo de criação do modelo analítico que representará o comportamento do ambiente sob análise. Neste momento, considerando finalizado o entendimento da aplicação, as seleções das tecnologias, as escolhas das métricas e a experimentação no ambiente real, então o modelo será criado. Uma vez criado tal modelo, o mesmo será capaz de simular cenários com configurações não possíveis (ou difíceis) de implementar na arquitetura experimental em virtude de sua limitação de capacidade. Neste trabalho, mais especificamente, adotamos a modelagem baseada em SPN [10], que permite realizar análises a partir de variações de inúmeros parâmetros, como o tempo de chegada e o número de instâncias do *Moodle* utilizadas em cada cenário. Ainda nesta etapa, foi escolhida a ferramenta Mercury [42] para modelagem e análise dos cenários, visto que a mesma permite calcular as métricas escolhidas na etapa anterior.

• **Validação do modelo:** um modelo é uma representação completa ou parcial de algo do mundo real. Esta etapa verifica se as abstrações implementadas no modelo estão em conformidade e consistentes com sistema real. Para tal fim, a validação confere uma série de métodos matemáticos e estatísticos para checar se o modelo consegue reproduzir o comportamento do mundo real. Portanto, validar um modelo significa verificar se os pressupostos desenhados apresentam resultados aproximados da realidade. Em suma, a validação assegura que as análises reproduzam valores consistentes em relação aos valores de referência. Para que os valores das métricas sejam estatisticamente equivalentes, o sistema real e o modelo SPN devem ser submetidos sob as mesmas condições de reprodutibilidade de experimentos, considerando, por exemplo, igualdade entre os parâmetros de configurações e as cargas de trabalhos. As médias podem ser validadas através de intervalos de confiança com margem de erro predefinida estatisticamente, bem como um teste denominado de T não pareado [43], o qual compara estatisticamente as equivalências entre as médias do sistema real e do modelo. Caso o modelo

não represente adequadamente o sistema, outros ajustes deverão ser realizados com novas rodadas de experimentação no ambiente real, até que o modelo seja refinado.

• **Geração de cenários:** caso a validação do modelo seja concretizada com êxito, as análises poderão ser extrapoladas para outros cenários com escopos maiores e diversificados. De acordo com a exatidão do modelo, é possível prever resultados das métricas de desempenho e de consumo de energia sem a necessidade de executar experimentos no sistema real. Dessa forma, serão criados cenários hipotéticos com variações de configurações definidas nos parâmetros do modelo, como números de requisições, tempo de chegada, números de instâncias (réplicas), tempo de serviço e Kilowatt sobre variações de tempos. Assim, tanto as análises quanto as simulações poderão identificar influências de determinados parâmetros nas métricas de desempenho e de consumo de energia. As métricas devem ser avaliadas com o objetivo de encontrar cenários que indiquem melhorias significativas na operação de toda infraestrutura.

• **Análises dos resultados:** nesta etapa, os dados serão analisados e interpretados por ferramentas estatísticas amplamente utilizadas nos meios acadêmicos, como R [44], RStudio [45], o Minitab [46], entre outras. Os resultados das análises serão demonstrados por gráficos e tabelas que apresentarão as relações mais relevantes entre as métricas. Os cenários baseados em contêineres e VMs terão variações nos parâmetros de configuração para que os resultados sejam objetos de análises de desempenho, de consumo de energia e custo. Dessa forma, os resultados obtidos poderão ser interpretados com mais clareza e precisão, no sentido de minimizar as incertezas nas definições de capacidades dos ambientes reais. Isso poderá assegurar uma maior taxa de acertos nas decisões tomadas pelas partes interessadas, podendo trazer melhorias significativas nas métricas selecionadas. Neste trabalho, os cenários serão variados em escopos como, por exemplo: a) a vazão e o tempo de chegada das requisições; b) o tempo de resposta por números de requisições; c) o tempo de resposta por números de instâncias (réplicas); d) a vazão por números de instâncias; e) o consumo de energia em relação à vazão; f) o custo do consumo de energia em relação à vazão. Importante destacar que caso os dados coletados não forem satisfatórios pelos testes realizados na validação, uma ou mais condições de retornos podem ocorrer entre as etapas experimentação no ambiente real, validação do modelo e a geração do modelo analítico, a fim de obter uma precisão esperada entre os dados do sistema real e do modelo. Portanto, pode ser necessário realizar novas medições no sistema real até o modelo representar adequadamente o comportamento do sistema.

5. Arquitetura Experimental

Esta seção descreve todos recursos utilizados na composição da arquitetura experimental da aplicação *Moodle* provisionada por contêineres e VMs. Esses ambientes foram observados por experimentos submetidos a cinco cenários que variaram de capacidade entre os contêineres e as VMs. Os experimentos forneceram amostras necessárias para os seguimentos das etapas de modelagem e validação que deram consequência a criação dos modelos propostos neste trabalho. A criação dos ambientes baseados em VMs utilizou a ferramenta *hypervisor* KVM, ao passo que os ambientes baseados em contêineres foram criados pelo *Docker Engine*. Sendo assim, tanto as instâncias do *hypervisor* KVM, quanto as instâncias do *Docker* alocaram com igualdade os recursos de CPU, da RAM, da interface de rede e os demais componentes do computador hospedeiro. O objetivo das medições foi assegurar a imparcialidade entre as coletas das amostras relacionados às duas tecnologias.

A Figura 7 apresenta uma visão geral da arquitetura montada para realizar os experimentos. O servidor hospedeiro das VMs/contêineres foi configurado com processador *intel Core* i5 (4 núcleos) de 3.4 Ghz, 8 GB de memória RAM, HD de 1 TB e interface de rede com 1 Gbit/s. Esse computador teve o seu HD particionado de modo a conter duas instalações da distribuição *Linux Ubuntu 16.10*, sendo uma partição SDA1 de 500 GB destinada à criação das VMs através do *KVM* e outra partição SDA2 de 500 GB destinada à criação dos contêineres através do *Docker*. Cada instância da aplicação *Moodle* utilizou o serviço *Apache HTTP Server* [47] em conjunto com o sistema de gerenciamento de banco de dados *MySQL* [48], o qual persistiu os dados em um volume compartilhado no mesmo servidor hospedeiro. Como o *back-end* do *Moodle* é implementado na linguagem PHP [49], foi instalada a extensão PHP no servidor *Apache* para processar as páginas enviadas nessa linguagem. A máquina representando os clientes foi configurada com processador *Intel Dual Core* de 2.0 Ghz, 4 GB de memória RAM, HD de 500 GB e interface de rede com 1 Gbit/s. Esses clientes tiveram instalações do *Jmeter* para gerar as cargas de trabalhos através das requisições *http*, a fim de demandar processamento das instâncias do *Moodle* provisionadas no computador hospedeiro.

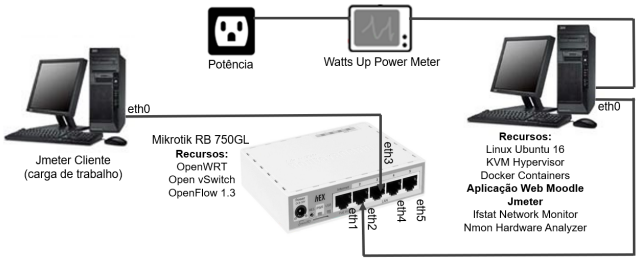

Figura 7. Topologia do ambiente de experimentação.

A topologia foi configurada com um roteador *Mikrotik* 750GL foi utilizado para estabelecer as conexões de rede com suporte a 1 Gbit/s de largura de banda para todas as portas. O computador cliente *Jmeter* fez uma conexão entre sua interface de rede de 1 Gbit/s com a porta *eth3* do roteador, estabelecendo um *link* de 1 Gbit/s. Da mesma forma, o computador hospedeiro fez uma conexão entre sua interface de

rede de 1 Gbit/s com a porta *eth2* do roteador, estabelecendo outro *link* de 1 Gbit/s. As instâncias de contêiner ou VM tiveram as suas interfaces *Ethernet* virtuais configuradas com taxa de transferência correspondente a 100 Mbit/s. Dessa forma, não houve risco de ocorrer gargalos na interface de rede computador hospedeiro em virtude do *link* ter disponível 1 Gbit/s de largura de banda para suportar as demandas dos clientes. As ferramentas de monitoramento realizaram as coletas das amostras no computador hospedeiro, durante o período que o *Jmeter* gerou as requisições destinadas às instâncias do *Moodle* em operação. Ao mesmo tempo, o medidor *Watts UP* [41] fez as coletas das amostras de potências elétricas demandadas pelo computador hospedeiro.

Os experimentos foram executados através de requisições http que representaram 13 interações frequentemente utilizadas pelos alunos do *Moodle*. Isso significou que um aluno representou um conjunto de 13 requisições enviadas para as instâncias do *Moodle*. As cargas de trabalhos foram definidas conforme cinco turmas com diferentes quantidades de alunos que demandaram recursos do computador hospedeiro. Para cada turma, os experimentos forneceram 60 amostras da vazão de requisições por segundo e a potência elétrica demandada. O kiloWatt-hora foi a unidade de potência adotada nas exposições dos resultados de consumo de energia e custo, visto que as concessionárias de energia elétrica adotam essa unidade de energia nos cálculos de faturamento pelos serviços fornecidos aos consumidores.

6. Modelo Analítico

O modelo SPN foi criado para representar eventos do *Moodle* estendidos aos ambientes provisionados por contêineres e VMs. O modelo permite calcular a vazão, o tempo de resposta e o consumo de energia demandada por instâncias baseadas nas duas tecnologias. Ele pode facilitar na criação de ajustes nos parâmetros de entrada do sistema, visando encontrar as configurações mais aderentes com a capacidade da operação e as cargas de trabalhos esperadas. Isso permite estimar os efeitos de eventuais aumentos ou reduções na capacidade da operação no sistema real. Os cenários foram subdivididos em escopos relacionados aos contêineres e VMs. Dessa forma, as duas perspectivas possuem configurações semelhantes para que os resultados sejam analisados sob as mesmas condições.

Neste trabalho, uma carga de trabalho corresponde a um aluno efetuando 13 requisições http pertinentes às funcionalidades mais recorrentes do *Moodle*, como acessar o *front-end* da aplicação, efetuar logon, visualizar um curso, visualizar páginas de atividades, submeter um questionário de atividades, entrar no fórum de discussões, responder a uma discussão, enviar um arquivo de texto, efetuar logout, entre outras. Cada requisição representa um *token* armazenado na constante `Número_de_Requisição`, a qual representa a quantidade de alunos fazendo acessos simultâneos aos recursos do *Moodle*. Consequentemente, o lugar `Clientes` que faz referência aos *tokens* definidos na constante `Número_de_Requisição` para que eles sejam disparados pela transição exponencial `T_Chegada_Fila` cujo *delay* é definido pela constante `Tempo_de_Chegada`. O disparo da transição `T_Chegada_Fila` também está condicionado à presença de *tokens* no lugar `Fila` cujo tamanho é definido pela constante `Tamanho_da_Fila`. Dessa forma, quando ocorre o disparo da transição `T_Chegada_Fila`, os *tokens* são consumidos ao mesmo tempo pelos lugares `Clientes` e `Fila` e, consequentemente, um *token* é gerado no lugar `Processamento_Fila`.

Os *tokens* presentes no lugar `Fila` representam a capacidade disponível do servidor (contêiner ou VM) para enfileirar as requisições. Os *tokens* do lugar `Fila` são decrementados à medida que as requisições de entrada são processadas e enfileiradas no lugar `Processamento_Fila`, o qual habilita o disparo da transição imediata `T_Saída_Fila` que representa o envio das requisições dos usuários para dentro da fila do servidor responsável por processar as requisições dos usuários. A presença de *tokens* nos lugares `Processamento_Fila` e `Docker/KVM` habilitam a transição imediata `T_Saída_Fila`. Quando `T_Saída_Fila` dispara, um *token* é consumido do lugar `Processamento_Fila` e, ao mesmo tempo, um *token* é consumido do lugar `Docker/KVM`. Simultaneamente, um *token* é gerado no lugar `Fila`, devolvendo a capacidade da fila para atender novas requisições. Por consequência, um *token* é gerado no lugar `Req_Processamento`, indicando que uma ou mais instâncias (contêineres ou VMs) estão processando as requisições. O número de *tokens* presentes no lugar `Docker/KVM` define a quantidade de instâncias que são hospedadas no servidor hospedeiro. O lugar `Docker/KVM` faz referência à constante `Número_de_Instâncias`, a qual representa a capacidade do sistema suportar a carga de requisições conforme limites estabelecidos no mecanismo de auto-escalonamento do *cluster*. Quando o lugar `Req_Processamento` termina de processar uma ou mais requisições, o sistema devolve as instâncias novamente para o *cluster* `Docker/KVM`. Esse processo é representado pelo disparo da transição exponencial `T_Serviço`, a qual faz referência à constante `Tempo_de_Serviço` que corresponde ao tempo necessário para que uma instância (um *token*) possa processar uma ou mais requisições conforme as cargas de trabalhos geradas pelas turmas.

O tempo médio para processar uma requisição de um usuário é utilizado como *delay* da transição exponencial `T_Serviço` que representa a conclusão do processamento das requisições dos usuários pelas instâncias do *Moodle* (VMs ou Contêineres). O tempo de serviço corresponde ao atraso gerado para uma instância processar uma carga de requisições simultâneas com determinado tempo de chegada entre elas. Por exemplo, considerando que apenas uma instância de contêiner recebe uma carga de trabalho correspondente a 130 requisições por segundos, simultaneamente, onde cada requisição possui um tempo de chegada equivalente a 100 ms. O processamento dessa operação produziu uma vazão equivalente a 1.95690 req/s. A partir da vazão gerada pela instância pode ser calculado o seu tempo de serviço para pro-

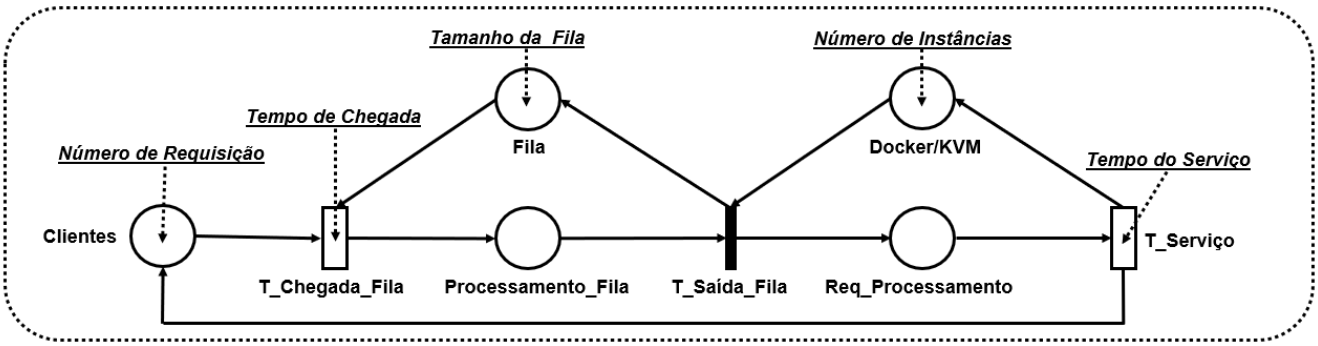

Figura 8. Modelo representando os conjuntos de requisições do Moodle para ambientes baseados em contêineres e VMs.

cessar uma requisição individual em unidades de segundos. Então, o tempo de serviço é calculado entre a razão de uma unidade de segundo e a vazão de requisições por segundos, ou seja, 1/1.95690 req/s que resulta em 0.51101 segundos de processamento para que uma instância de contêiner atenda cada requisição individual dentre as 130 requisições. O tempo de serviço está associado diretamente à capacidade de processamento do sistema responder a uma certa carga de trabalho. O poder computacional dos nós têm influência direta no tempo de serviço de um determinado *cluster* associado a uma aplicação. Supondo que os Nós disponham de recursos suficientes, cada instância atribuída a um agrupamento de contêineres ou VMs tenderá a reduzir o tempo de serviço da aplicação e aumentar a sua taxa da vazão. Por outro lado, o tempo de chegada das requisições influencia diretamente na taxa da vazão e no tempo de resposta de todo o sistema. É importante destacar que o tempo de chegada é um parâmetro atribuído à constante `Tempo_de_Chegada`, o qual serve de referência para a transição `T_Chegada_Fila`.

A Tabela 2 apresenta as transições utilizadas no modelo SPN *Moodle*. O tempo de chegada entre as requisições do *Jmeter* assumiu uma distribuição exponencial, assim como os *tokens* gerados pela transição `T_Chegada_Fila`. A transição temporizada `T_Chegada_Fila` foi definida com a política de disparo *Single Server*, enquanto a transição temporizada `T_Serviço` foi definida com a política de disparo *Infinite Server*. Na semântica *Infinite Server*, todos os *tokens* são disparados pela transição `T_Serviço` na mesma janela de tempo, indicando um comportamento de processamento paralelo das requisições, ao passo que, na semântica *Single Server*, os *tokens* são o disparados sequencialmente em janela de tempo não compartilhada. Esse comportamento indica que as instâncias não operam paralelamente. A transição imediata `T_Saída_Fila` teve o seu peso e prioridade com valores 1. A Tabela 3 apresenta as expressões utilizadas na ferramenta *Mercury* [42] para calcular as métricas da vazão, do tempo de resposta e o do consumo de energia, enquanto a Tabela 4 apresenta os parâmetros utilizados no modelo SPN proposto.

As expressões citadas na Tabela 3 são baseadas nas fórmulas da Lei de Little [37]. A `Vazão` é dada pelo produto do número de *tokens* esperados no lugar `Req_Processamento` multiplicado pela taxa atrelada ao processamento da requisição. A métrica `CT-Tempo_de_Resposta` é obtida pelo resultado do somatório de *tokens* esperados nos lugares `Processamento_Fila` e `Req_Processamento` dividido pela vazão da infraestrutura. A métrica `kiloWatt_por_Tempo` representa o consumo de energia calculado através do produto entre o kiloWatt e variações de tempos definidos na variável $\triangle t$. Os *tokens* esperados no lugar `Req_Processamento` representaram o aumento na demanda de energia elétrica em decorrência das instâncias agregadas no processamento da demanda. A energia consumida foi calculada de acordo com a equação 1, onde a variável *PD* representa a potência (W) demandada pelas instâncias em processamento. O consumo de energia fixo associado à infraestrutura em período de inatividade foi referenciado pela variável PB para indicar a potência base. A soma das variáveis PB e PD representa a potência total (PT) da operação, a qual corresponde à energia fixa consumida pela infraestrutura vinculada a um agrupamento de instâncias sendo demandadas por cargas de trabalhos. A constante 1000 é empregada na conversão da unidade de potência *Watt* para a unidade *kilowatt* (kW). A variável $\triangle t$ representa as variações de tempos com períodos relacionado às horas, dias, meses e anos.

$$E(kWh) = \frac{PB + PD}{1000} \times \triangle t \tag{1}$$

Tabela 2. Atributos das transições utilizadas nos modelos SPN contêineres e VMs.

Transição	Tipo	Semântica	Peso	Prioridade
T_Chegada_Fila	Temporizada	Single Server	-	-
T_Saída_Fila	Imediata	-	1	1
T_Serviço	Temporizada	Infinite Server	-	-

7. Resultados e Discussão

Esta seção apresenta os resultados obtidos a partir dos experimentos realizados na arquitetura experimental e das análises feitas a partir do modelo SPN gerado. Primeiro, é apresentado a validação do modelo e, em seguida, são detalhados os quatro cenários avaliados, a saber: O cenário I analisa o

Tabela 3. Expressões para calcular as métricas nos modelos SPN contêineres e VMs.

Métricas	Expressões
Vazão	((E{#Req_Processamento}) × (1/Tempo_do_Serviço))
Tempo_de_Resposta	((E{#Processamento_Fila})+(E{#Req_Processamento})) /((E{#Req_Processamento})/Tempo_do_Serviço)
kiloWatt_por_Tempo	PB+(((PD/1000) × (Δt)) × (E{#Req_Processamento}))

Tabela 4. Descrição dos parâmetros utilizados nos modelos SPN contêineres e VMs.

Variável	Parâmetro
Número_de_Requisição	Quantidades de requisições por alunos ou turmas
Tempo_de_Chegada	Tempo de chegada da requisição
Número_de_Instâncias	Quantidades de instâncias disponíveis
Tempo_de_Serviço	Tempo de serviço da aplicação
kilowatt	Potência elétrica convertida para KW (PB+PD)/1000

desempenho da vazão em função de alguns tempos de chegados predefinidos em um intervalo. O cenário II mostra os gargalos nos tempos de respostas gerados por sucessivas cargas de trabalhos. O cenário III faz estimativas de capacidade para os agrupamentos de instâncias, visando encontrar os melhores ajustes entre a capacidade e a demanda. Por fim, cenário IV realiza análises de consumo de energia e custo com o objetivo de proporcionar um melhor custo-benefício relação a demanda por energia elétrica.

7.1 Validação do modelo SPN

O processo de validação utilizou os parâmetros e os modelos apresentados na seção anterior. As Tabelas 2, 3 e 4 apresentam as transições, as métricas e os parâmetros adotados neste modelo, respectivamente. A validação foi conduzida com base em cinco cenários experimentais, classificados como turmas de tamanhos diferentes (de 1 a 20 alunos). Os experimentos realizados nos ambientes baseados em contêineres e VMs produziram amostras com 60 médias relativas à vazão das requisições por segundos. As amostras foram analisadas isoladamente conforme os tamanhos das turmas. As cargas de trabalhos definidas para as cinco turmas assumiram os seguintes montantes de requisições: 13, 65, 130, 195 e 260. Neste trabalho, foram consideradas turmas compostas pelas quantidades de alunos, ao invés do montante de requisições. Dessa forma, cada aluno individual representou uma carga de trabalho relativa à 13 requisições pertinentes às funcionalidades frequentemente acessadas no *Moodle*, como acesso à página inicial, efetuar logon, visualizar um curso, visualizar páginas de atividades, submeter um questionário de atividades, entrar no fórum de discussões, responder a uma discussão, enviar um arquivo de texto, efetuar logout, entre outras. Portanto, em virtude deste trabalho adotar os termos turmas e alunos para representar os cenários, os montantes de requisições foram divididos pela quantidade requisição demandada por cada aluno, por exemplo: 13/13=01-Aluno, 65/13=05-Alunos, 130/13=10-Alunos, 195/13=15-Alunos e 260/13=20-Alunos. Assim, a validação teve como referência amostras oriundas de experimentos gerados por turmas compostas pelos seguintes quantitativos de alunos: 01, 05, 10, 15 e 20. Os resultados apresentados nos cenários posteriores também adotaram as denominações de turmas e alunos.

A Tabela 5 apresenta as configurações inseridas nos modelos e nos ambientes baseados em contêineres e VMs. Os cenários avaliados no sistema real adotaram uma instância de contêiner e outra de VM. Sendo assim, o modelo utilizou apenas um *token* atribuído à constante `Número_de_Instâncias` para indicar as instâncias contidas nos lugares `Docker/KVM`, os quais representam os comportamentos de um contêiner ou de uma VM utilizada no sistema real. No *Jmeter*, os tempos de chegadas entre as requisições foram definidos seguindo uma distribuição exponencial entre intervalos aleatórios de 100 milissegundos. O *Jmeter* oferece uma funcionalidade na qual são definidos tempos aleatórios entre as chegadas das requisições. Neste trabalho, foi utilizado um *script* em java que simulou os tempos de chegadas baseados em uma função de distribuição exponencial. No modelo, a constante `Tempo_de_Chegada` fez referência a um atraso de 100 milissegundos atribuído ao parâmetro *delay* da transição `T_Chegada_Fila`. Já o *delay* definido na constante `Tempo_do_Serviço` foi atribuído à transição `T_Serviço` para representar o atraso no processamento das requisições. Considerando que foram observadas cinco turmas com escopos diferentes para cada experimento dos contêineres, os tempos de serviços atribuídos à constante `Tempo_do_Serviço` corresponderam a 0.84854, 0.52849, 0.51101, 0.59981, 0.59119 para turmas compostas por 01, 05, 10, 15 e 20 alunos, respectivamente. Da mesma forma, os experimentos das VMs apresentaram tempos de serviços atribuídos à constante `Tempo_do_Serviço` corresponde a 0.80141, 0.50669, 0.49895, 0.57313, 0.59726, respectivamente. Os *tokens* pertinentes aos tamanhos das cinco turmas foram armazenados na constante `Número_de_Requisição`.

Tabela 5. Configurações dos cenários base para a validação dos modelos contêineres e VMs

Modelo Contêiner		Modelo VM	
Parâmetro	**Valor**	**Parâmetro**	**Valor**
Número_de_Requisição	13 req	Número_de_Requisição	13 req.
Tempo_de_Chegada	100 ms	Tempo_de_Chegada	100 ms
Tempo_de_Serviço	0.8485 seg.	Tempo_de_Serviço	0.8014
Número_de_Instâncias	1 contêiner	Número_de_Instâncias	1 VM

De acordo com os resultados apresentados na Tabela 6, os experimentos realizados no sistema real e nas análises dos modelos apresentaram diferenças significativamente pequenas, no que diz respeito à vazão. Considerando os ambientes baseados em contêineres, observou-se que as turmas contendo 1, 5, 10, 15 e 20 alunos obtiveram os intervalos de confiança correspondentes a [1.1156;1.2413], [1.799;1.9854], [1.8929;2.0209], [1.6223;1.7121], [1.653;1.7299], respectivamente. Ao passo que, o modelo do contêiner obteve as

Tabela 6. Validação do modelo analítico

Alunos	Experimentos no Contêiner		Modelo do Contêiner	Experimentos na VM		Modelo da VM
	Média	IC	Média	Média	IC	Média
01	1.17850	[1.1156;1.2413]	1.12919	1.24780	[1.1917;1.304]	1.20936
05	1.89220	[1.799;1.9854]	1.82472	1.97360	[1.8754;2.0717]	1.90691
10	1.95690	[1.8929;2.0209]	1.92697	2.00420	[1.9247;2.0836]	1.96029
15	1.66720	[1.6223;1.7121]	1.64683	1.74480	[1.6994;1.7902]	1.72798
20	1.69150	[1.653;1.7299]	1.67789	1.67430	[1.6438;1.7048]	1.66626

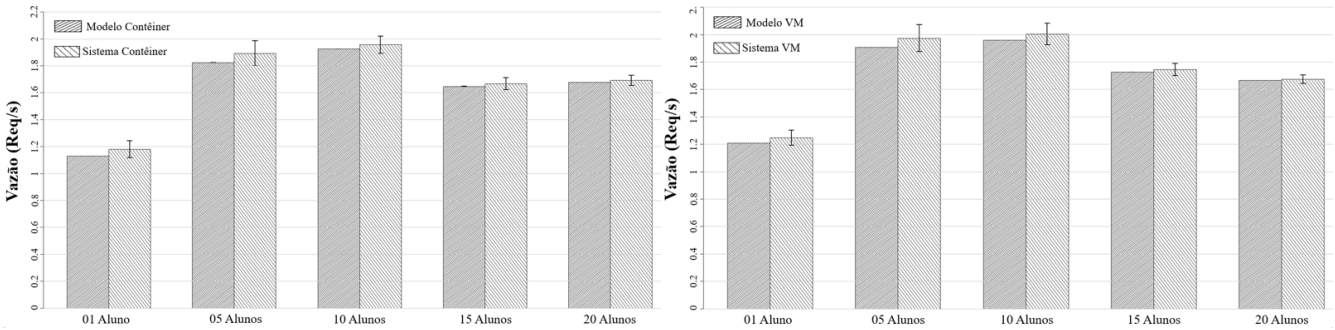

Figura 9. Comparação dos resultados para a vazão do sistema obtidos através dos experimentos e dos resultados obtidos através do modelo SPN utilizando contêineres (a) e VMs (b).

vazões médias correspondentes a 1.17850 req/s, 1.89220 req/s, 1.95690 req/s, 1.66720 req/s, 1.69150 req/s, respectivamente. Consequentemente, os ambientes baseados em VMs consideraram os mesmos tamanhos de turmas, as quais resultaram nos intervalos de confiança correspondentes a [1.1917;1.304], [1.8754;2.0717], [1.9247;2.0836], [1.6994;1.7902], [1.6438; 1.7048], respectivamente. Enquanto, o modelo da VM obteve as vazões médias correspondentes a 1.20936 req/s, 1.90691 req/s, 1.96029 req/s, 1.72798 req/s, 1.66626 req/s, respectivamente. De acordo com a Figura 9, os resultados dos modelos conseguiram representar o comportamento do sistema real, uma vez que os resultados estão dentro do intervalo de confiança (IC) de 95% obtidos através dos experimentos. Dessa forma, pode-se afirmar que o comportamento do modelo apresentou resultados precisos e consistentes em comparação o sistema real. Assim, diferentes cenários podem ser analisados a partir dos modelos SPN proposto, sem a necessidade da execução de novos experimentos. É importante destacar que o desempenho do sistema com contêineres e VMs entregaram os serviços com tempos computacionais aproximados.

7.2 Cenário I

Este cenário observou o comportamento da vazão em relação aos tempos de chegadas que variaram de 100 milissegundos (ms) até 10 segundos (seg) com intervalos de 100 ms para cada análise realizada. A ideia deste cenário foi analisar o comportamento da vazão submetida a diferentes tempos de chegadas. Considerando que a capacidade do *link* de comunicação estivesse adequada e não apresentasse falhas, acentuados intervalos entre uma requisição e outra podem sinalizar tendências de altas ociosidades e baixas demandas de processamento. Essas periódicas quedas de gargalos podem sugerir reduções na capacidade da operação em determinados horários do dia. Este cenário realizou uma demonstração desses aspectos de desempenho para que os resultados sejam úteis aos cenários posteriores.

Este estudo pressupôs uma turma com 20 alunos produzindo cargas de trabalhos, considerando apenas uma instância para o contêiner e outra para a VM. O tempo de serviço correspondeu a 0.59119 segundos para o contêiner e 0.59726 segundos para a VM. Cada avaliação forneceu um conjunto composto por 60 amostras das vazões médias influenciadas pelos efeitos dos tempos de chegadas entre as requisições. A primeira simulação adotou um tempo de chegada de 100 ms, a segunda 200 ms, a terceira 300 ms, e assim sucessivamente até a última simulação com tempo de chegada de 10 seg. Todas as simulações totalizaram 19 grupos de amostras relativas às médias das vazões de requisições por segundos. Os tempos de chegadas foram atribuídos igualmente às constantes `Tempo_de_Chegada` e `VM-Tempo_de_Chegada` em referência ao *delay* da transição `T_Chegada_Fila`, respectivamente. Os 20 alunos das turmas foram representados na constante `Número_de_Requisição` em referência às marcações do lugar `CT-Cliente`, o qual armazenou 260 *tokens* relativos às cargas de trabalhos com 260 requisições. As marcações do lugar `Docker/KVM` forneceu apenas uma instância para a constante `Número_de_Instâncias`. Os tempos de serviços dos contêineres e VMs foi atribuído à variável `Tempo_do_Serviço` em referência ao atraso da transição `T_Serviço`. É importante destacar que os experimentos realizados no sistema real forneceram as taxas de requisições por segundos que permitiram calcular os tempos de serviços de cada turma.

Os resultados para análise deste cenário são apresentados na Figura 10. Eles mostram que a vazão do sistema se man-

teve estável, em torno de 1.67 req/s, considerando tempos de chegadas variando entre 100 ms a 600 ms. No entanto, a partir do intervalo de 700 ms até 10 segundos, a vazão da turma caiu gradativamente até atingir uma taxa mínima de 0.1 req/s. Esse comportamento pode indicar que, em determinados períodos das 24 horas diárias, a capacidade da operação pode estar superdimensionada em relação à demanda e, por consequência, ser um indicativo de subutilização dos recursos alocados. Outro aspecto deste cenário é que a vazão máxima do serviço correspondeu à 1.67 req/s para o menor tempo de chegada. Considerando que os parâmetros do modelo não sejam alterados, uma eventual inclusão de novas turmas implicaria no aumento na demanda, porém a vazão do serviço permaneceria inalterada em razão da capacidade de processamento da instância está saturado. Além disso, a aplicação teria uma queda abrupta no seu desempenho devido ao aumento no tempo de resposta em decorrência do gargalo na operação.

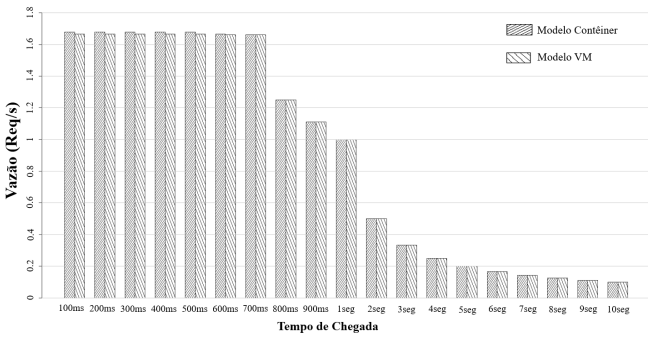

Figura 10. Vazão por tempo de chegada nas operações baseadas em contêineres e VMs

Uma segunda análise verificou o comportamento da vazão utilizando duas instâncias, ou seja, dois *tokens* no lugar `Docker/KVM`. Neste caso, os resultados indicaram uma estabilidade na taxa da vazão inerentes aos períodos de 100 ms e 200 ms. As taxas máximas atingidas corresponderam à 3.36 req/s para os dois contêineres e 3.33 req/s para as duas VMs. Considerando o período de 300 ms, as taxas da vazão de ambas abordagens reduziram em aproximadamente 3.32 req/s. Todavia, a queda de desempenho foi percebida a partir do período 400ms, onde a taxa da vazão caiu para 2.50 req/s até atingir a taxa mínima de 0.10 req/s relativa ao último período de 10 segundos. Nos períodos de 100 ms e 200 ms, observou-se que os tempos de respostas corresponderam a 77.9 segundos para os dois contêineres e 78.5 segundos para as duas VMs. Já no período de 300 ms, os tempos de respostas reduziram para 25.9 e 38.6 segundos em relação aos dois contêineres e às duas VMs, respectivamente. Nesse caso, notou-se um gargalo menor na operação a favor dos contêineres. Por outro lado, a partir do período 400 ms até 10 segundos, os tempos de respostas mantiveram-se estáveis em média de 0.60 segundos. Esse comportamento pode indicar prováveis casos de subutilização dos recursos em virtude da baixa de demanda, sugerindo redução na capacidade da operação, em caso necessidade de otimização ou redução de custo.

7.3 Cenário II

Este cenário avaliou o tempo de resposta em relação às cargas de trabalhos oriundas de variadas turmas. Nesta análise foram observados os gargalos gerados na operação em decorrência de aumento sucessivos de turmas com 20 alunos.

Em cada simulação iniciada, uma nova turma era incrementada até atingir um limite de 20 turmas. Nesse caso, a primeira análise foi realizada com uma turma, a segunda com duas turmas, a terceira com três turmas, e assim sucessivamente até atingir o último cenário com 20 turmas. Da mesma forma, os quantitativos de alunos cresceram de 20 em 20 unidades em detrimento dos incrementos das turmas. Sendo assim, todos esses cenários geraram gradativas cargas de trabalhos em cada simulação, totalizando a maior carga de trabalho com 400 alunos que demandaram um total de 5200 requisições no processamento da operação. No modelo, as variações das turmas foram definidas na constante `Número_de_Requisição` com o seu respectivo lugar `Clientes`. Um tempo de 100 ms foi definido na constante `Tempo_de_Chegada`, seguindo uma distribuição exponencialmente entre as requisições. O valor atribuído à `Tempo_de_Chegada` fez referência ao *delay* da transição `T_Chegada_Fila`. A capacidade da operação considerou apenas uma instância atribuída à constante `Número_de_Instâncias` referenciada pelo lugar `Docker/KVM`. Em virtude de todos os cenários adotarem turmas com 20 alunos para apenas uma instância, os tempos de serviços atribuídos ao contêiner e a VM corresponderam a 0.59119 e 0.59726 segundos, respectivamente. Dessa forma, esses valores foram atribuídos à constante `Tempo_do_Serviço` em referência ao atraso da transição `T_Serviço`.

Os resultados de cada escopos de turmas forneceram 60 amostras referentes aos seus respectivos tempos médios de respostas. Dada a análise de uma turma, os resultados revelaram que os tempos médios de respostas corresponderam a 155.38 segundos para o contêiner e 156.51 segundos para a VM, uma diferença de 1.13 segundos a favor do contêiner. Considerando os tempos de serviços vinculados à operação do contêiner e VM, o tempo médio de resposta aumentou gradativamente à medida que uma turma foi introduzida para criar um cenário de maior escopo. De acordo com a Figura 11, observou-se que duas turmas em atividades simultâneas corresponderam a 310.28 e 312.54 segundos para o contêiner e VM, respectivamente. Nesse caso, a diferença nos tempos de respostas dos dois ambientes correspondeu a 2.27 segundos a favor do contêiner. No caso da última análise, ou seja, as 20 turmas com atividades paralelas, os tempos médios de respostas resultaram em 3098.48 segundos para o contêiner e 3121.16 segundos para a VM. Uma diferença de 22.68 segundos a favor do contêiner. Em efeitos práticos, essa última análise representou um cenário cuja demanda saturou consideravelmente a aplicação, a ponto de tornar inviável a entrega do serviço em virtude da capacidade da operação está subdimensionada. Essa situação sugere que sejam aplicados ajustes equilibrados nos escopos das instâncias da operação a fim de

atender variações de demandas mínimas e máximas. Todavia, os cumprimentos de SLAs no menor tempo de resposta possível não pode significar alocação de recursos em excessos. Tais decisões podem caracterizar equívocos no planejamento da capacidade, resultando em gastos desnecessários no custo operacional.

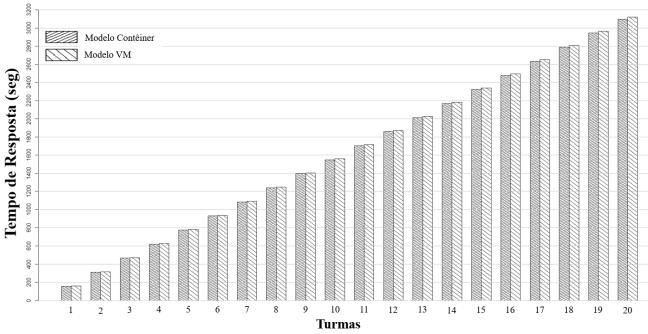

Figura 11. Tempo de Resposta por Turmas

7.4 Cenário III

O auto-escalonamento é um mecanismo aplicado frequentemente na automação do aumento ou redução da capacidade da operação conforme limiares definidos nos parâmetros de desempenho das instâncias. Dependendo do comportamento da carga de trabalho, os números de instâncias do serviço são ajustados automaticamente, a fim de cumprir o SLA esperado. Essa característica permite que o processamento da carga de trabalho seja balanceado para outras instâncias, sem causar prejuízo ao tempo de resposta e a vazão da operação. Todavia, é importante destacar que o aumento na capacidade da operação é uma tarefa que exige cautela na definição dos limites mínimos e máximos das instâncias, visto que as demandas das turmas podem crescer ou diminuir em períodos sazonais. Caso os limiares das instâncias sejam corretamente definidos no auto-escalonamento, o tempo de resposta não causará gargalos acentuados na operação em virtude da capacidade autoajustar conforme a demanda. Outro aspecto é que o tempo de serviço geral do *cluster* tende a diminuir à medida que cresce o agrupamento de instâncias. Por consequência, a vazão poderá alcançar taxas suficientes para atender a demanda com mínimos gargalos. Dessa forma, este cenário consegue estimar agrupamentos de instâncias suficientes para equilibrar a vazão das turmas, bem como, reduzir os gargalos causados pelo aumento no tempo de resposta.

Para este cenário, foram coletadas 60 amostras relativas a 10 agrupamentos de instâncias baseadas em contêineres e VMs, isto é, cada instância provisionada no *cluster* representou um aumento na sua capacidade. O maior agrupamento foi limitado a 10 instâncias. Nesse caso, a primeira análise foi realizada com uma instância, a segunda com duas instâncias, a terceira com três instâncias, e assim sucessivamente até atingir o último cenário com 10 instâncias. A carga de trabalho foi fixada em 10 turmas com 20 alunos. Sendo assim, um total de 2600 requisições representaram as demandas das 10 turmas. Esse montante foi atribuído à variável `Número_de_Requisição`, que representa o número de *tokens* no lugar `Clientes`. O tempo de chegada entre as requisições considerou atrasos próximos a 100 ms conforme uma distribuição exponencial. Esse valor foi atribuído à constante `Tempo_de_Chegada`, representando o `atrasos` da transição `T_Chegada_Fila`. Na constante `Tempo_do_Serviço`, que representa o *atraso*, foi definido com 0.59119 e 0.59726 segundos para o tempo médio de processamento de uma requisição por uma instância de contêiner e de VM, respectivamente. As análises deste cenário consideraram 10 agrupamentos de instâncias que variaram entre 1 a 10 contêineres ou VMs. Esses quantitativos de instâncias foram incrementados na constante `Número_de_Instâncias`, a qual foi referenciada no atributo *Marking* do lugar `Docker/KVM`.

A Figura 12 apresenta os resultados obtidos nas simulações realizadas nos 10 escopos de grupamentos de instâncias. A vazão do primeiro caso resultou em 1.68 req/s para um contêiner e 1.67 req/s para uma VM. Já o tempo de resposta foi de 1549.48 e 1560.81 segundos, para um contêiner e uma VM, respectivamente. Esse resultado mostra uma degradação do desempenho da aplicação em decorrência de apenas uma instância realizar o processamento de todas as 2600 requisições demandadas. Dessa forma, uma eventual demanda oriunda de 10 turmas concorrentes tornaria praticamente inviável o funcionamento da aplicação. Na análise do segundo caso, a vazões das instâncias agrupadas atingiram as taxas 3.36 req/s para dois contêineres e 3.33 req/s para duas VMs. Já o tempo de resposta foi de a 774.95 e 780.62 segundos, considerando dois contêineres e duas VMs, respectivamente. É possível notar que o aumento no número de instâncias reduziu o gargalo da operação em aproximadamente 50%. No quarto caso, as vazões das instâncias agrupadas atingiram as taxas 6.71 req/s para quatro contêineres e 6.66 req/s para quatro VMs. O tempo de resposta obtido foi de 387.54 e 390.38 segundos para quatro contêineres e quatro VMs, respectivamente. No sexto caso, os agrupamentos com 6 instâncias atingiram vazões cujas taxas foram 10.00 req/s para os contêineres e 9.99 req/s para as VMs. A operação com 6 instâncias reduziu o gargalo em 99.07% em comparação ao primeiro caso, fazendo o tempo de resposta diminuir para 14.41 e 15.69 segundos com a utilização de contêineres e VMs, respectivamente. É importante destacar que a inclusão da instância número 7 não surtiu efeito na vazão, visto que a sua taxa permaneceu em 10.00 req/s. No entanto, ocorreu uma queda no gargalo da operação, reduzindo os tempos de respostas dos contêineres e das VMs para 0.94 e 0.86 segundos, respectivamente. Sendo assim, um agrupamento com 10 instâncias representaria um superdimensionamento na capacidade, gerando desperdícios na alocação dos recursos.

Os resultados do Cenário I mostraram que o tempo de chegada foi um atributo que impactou diretamente na taxa da vazão. Foi observado que uma instância atingiu a uma vazão máxima assumindo os tempos de chegadas adotados. Todavia, o Cenário II mostrou que o tempo de resposta foi influenciado

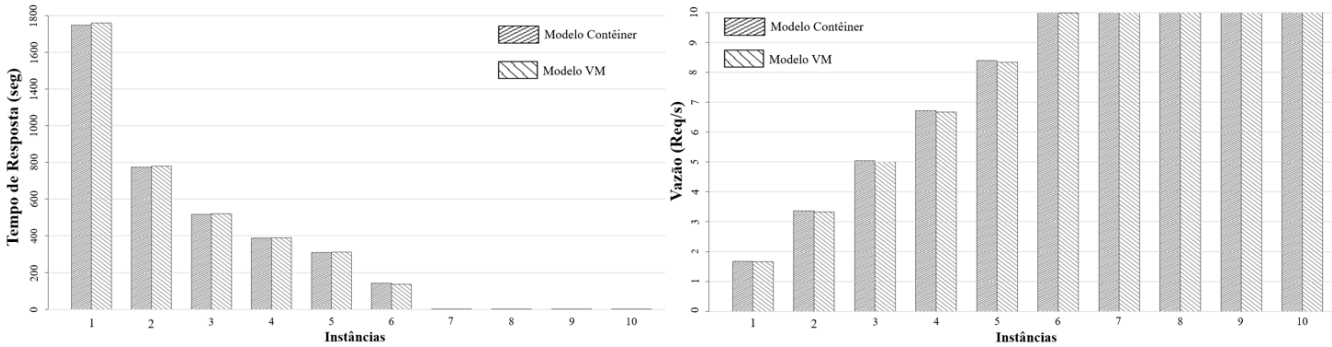

Figura 12. a) Tempo de Resposta por instâncias - b) Vazão por instâncias

com gargalos em virtude do limite de processamento suportado pela instância. Neste cenário em questão, foi observado que um agrupamento com 10 instâncias pode alcançar uma vazão com uma taxa maior quando o tempo de chegada é reduzido de 100 ms para 10 ms. Verificou-se que as 10 instâncias atingiram vazões cujas taxas corresponderam a 16.8 req/s e 16.7 req/ para os contêineres e VMs, respectivamente. Adicionalmente, os tempos de respostas de ambas tecnologias corresponderam a 155.49 e 156.62 segundos, respectivamente. Para um agrupamento com 20 instâncias, a vazão representou uma taxa equivalente a 33.76 req/ para os contêineres e 33.33 req/ para as VMs. Ao passo que, os tempos de respostas de ambas tecnologias corresponderam a 78.03 e 78.60 segundos, respectivamente.

Por fim, foi analisada uma operação cujo tempo de serviço para processar uma requisição representou 0.29560 segundos para o contêiner e 0.29863 segundos para a VM. Esses valores pressupõem um *cluster* cujo poder de processamento dos nós é superior em relação aos cenários anteriores. Dessa forma, a quantidade de instâncias suportadas por 10 turmas com 20 alunos reduziu o tamanho de 6 para 3 instâncias, mantendo as mesmas taxas da vazão e os mesmos tempos de respostas. Conclui-se que a redução do número de instâncias não causaria prejuízo ao desempenho da operação. Pelo contrário, poderia trazer uma contribuição positiva na redução do consumo de energia. Além do mais, a aquisição de *clusters* cujos nós possuem mais capacidade de processamento e memória, pode ser uma opção mais vantajosa financeiramente a longo prazo, dada a economia de espaço físico e a redução na temperatura do ambiente, em virtude da diminuição na quantidade de servidores. A seção a seguir mostra um cenário que avaliar o consumo de energia e os custos dos agrupamentos de instâncias.

7.5 Cenário IV

A análise de consumo de energia possibilita um maior controle e gestão nos gastos com energia elétrica dos *data centers*. Além do mais, uma boa prática de economia de energia promove uma cultura de preservação do meio ambiente e pode proporcionar um melhor custo-benefício de médio e a longo prazo na utilização de recursos computacionais. Os resultados obtidos acerca do consumo de energia também permitem realizar uma análise de custo com base no valor médio da tarifa cobrada por kWh. Para tanto, neste cenário é analisado o consumo de energia e o custo relacionado à vazão demandada pelas requisições das turmas. Nesta análise, a potência média demandada foi observada em decorrência da vazão suportada pelos agrupamentos de instâncias.

As simulações levaram em consideração um período de ausência de cargas de trabalhos destinadas aos contêineres e VMs. Para isso, o computador hospedeiro foi observado pelo medidor de consumo de energia durante um período de inatividade. Esse período de observação teve uma duração de 30 minutos e cada coleta de amostras considerou intervalos de 30 segundos. As coletas realizadas no *Watts Up Power Meter* forneceram 60 amostras referentes à potência média demandada. Em razão da inatividade do computador hospedeiro demandar um valor fixo de potência elétrica, foi avaliada separadamente as potências elétricas atribuídas às cargas de trabalhos e as instâncias agrupadas. Sendo assim, os casos avaliados demandaram uma potência média desvinculada da potência média relacionada à inatividade do computador hospedeiro. O consumo de energia atribuído apenas à inatividade representou uma potência base média de 25,04 *Watts*. Esse valor representa um custo fixo operacional apenas para manter a infraestrutura de ligado sem receber cargas de trabalhos dos clientes. Logo, o consumo de energia gerado a partir do valor 25,04 *Watts* representa de fato a potência demandada pelo processamento das requisições das turmas. Sendo assim. as análises posteriores utilizaram a potência base média (25,04 W) como valor de referência para derivar novos cálculos com base na Equação 1.

Os experimentos consideram um *cluster* contendo até 5 instâncias que variaram os números de contêineres e VMs de 1 a 5. Uma turma com 20 alunos gerou a carga de trabalho pelo cliente *Jmeter*. A primeira medição considerou uma instância, a segunda medição duas instâncias, a terceira medição três instâncias, assim sucessivamente até atingir cinco instâncias. Considerando os resultados obtidos, as amostras indicaram que cada instância agregada representa uma potência média correspondente a 2.06 W para os contêineres e 4.5 W para as VMs. A energia consumida foi calculada de acordo com a Equação 2, onde a variável *PB* representou a potência (W) base consumida pelo computador hospedeiro em inatividade,

ao passo que a variável *PD* referenciou a potência demandada pelas instâncias em atividades. A constante 1000 é usada para a conversão da unidade de potência *Watt* para a unidade *kilowatt* (kW). A variável $\triangle t$ corresponde ao período de um ano relativo à energia demandada no processamento realizado pelos escopos de instâncias definidos nas simulações. A unidade de tempo foi representada por horas, ou seja, um ano em horas correspondeu ao produto de 24 horas, 30 dias e 12 meses, totalizando 8.640 horas. No modelo, a métrica `kiloWatt_por_Tempo` (ver Tabela 3) adotou a Equação 1, multiplicando pelos números de instâncias em operação. A variável Tarifa representa o valor relativo à tarifa média nacional por KWh, praticado pela agência nacional de energia elétrica [11], com um valor de R$ 0,576 por kWh.

$$Custo(kWh) = \frac{PB + PD}{1000} \times \triangle t \times Tarifa \quad (2)$$

As medições demonstraram que tanto os contêineres quanto as VMs sem processar as cargas de trabalhos não indicaram impacto na utilização CPU, independentemente da quantidade de instâncias utilizadas nos cinco cenários experimentais. A ausência de demanda destinadas aos agrupamentos de instâncias manteve o percentual de ociosidade da CPU próximo a 99,8%. Assim, a ausência de demanda no computador hospedeiro representou um cenário cujo consumo de energia anual seria de 216.35 kW/ano. Dessa forma, o custo fixo anual para mantê-lo em ligado sem demanda seria em torno de R$ 124,62. É importante destacar que o consumo de energia anual e o custo anual foram calculados pela 2 com base na potência base (25,04 W). Portanto, ambos são valores que não possuem cargas de trabalhos vinculadas. Por outro lado, a potência demandada corresponde apenas aos valores gerados a partir das cargas de trabalhos processadas em casos de demandas, ou seja, desconsidera a potência base consumida pela inatividade do computador hospedeiro. Consequentemente, a potência total (PT) refere-se ao montante derivado da soma entre as variáveis PB e PD. Essa segunda variável teve a potência variada conforme as quantidades de contêineres (2.06 W) e de VMs (4.5 W) em processamento.

O tempo de chegada entre as requisições foi um fator determinante no valor da taxa da vazão, pois quando o tempo de chegada foi reduzido de 100 ms para 10 ms, os agrupamentos de instâncias conseguiram atingir suas maiores taxas da vazão. Essas elevadas taxas atribuídas às capacidades dos 10 agrupamentos de instâncias permitiram a métrica `kiloWatt_por_Tempo` calcular a potência média demandada oriundas dos fluxos de requisições das turmas. No modelo, as instâncias em atividades foram atribuídas ao lugar `Req_Processamento` conforme os tempos de chegadas atrelados aos fluxos de requisições da turma. O cenário III mostrou que 100 ms entre as chegadas das requisições não foi suficiente para exaurir a capacidade do agrupamento com 10 instâncias. No entanto, a Tabela 7 mostra que 10 milissegundos conseguiram produzir uma vazão equivalente a 16.79 req/s para os 10 contêineres e 16.66 req/s para as 10 VMs. Nesse caso, todas as instâncias foram utilizadas no processamento e os tempos de respostas apresentaram gargalos relativos a 155.48 segundos para os contêineres e 156.62 segundos para as VMS.

De acordo com as projeções anuais apresentadas na Tabela 7, a energia demandada por um contêiner e uma VM correspondeu a 17.80 kW/ano e a 38.88 kW/ano, respectivamente. Em decorrência da soma entre a potência demandada com a potência base referente a 216.35 kW/ano, os valores totais das duas operações corresponderam a 234,14 kW/ano e a 255.2 kW/ano, respectivamente. Os custos do caso 01 corresponderam a R$ 134.87 para os contêineres e R$ 147.01 para as VMs. Em relação à potência base, o contêiner e a VM do caso 01 representaram um aumento percentual de 8.23% e 17.97%, respectivamente. Nesse caso, a VM apresentou uma diferença de 21.08 kW/ano no consumo de energia total e de R$ 12.14 no custo total, resultando no aumento percentual de 9.00% em relação ao contêiner. Considerando as três instâncias agrupadas no caso 03, a potência elétrica demandada pelos contêineres e VMs corresponderam a 53.40 kW/ano e a 116.64 kW/ano, respectivamente. Os valores totais das duas operações corresponderam a 269.74 kW/ano e a 332.99 kW/ano, respectivamente. Na perspectiva financeira do caso 03, ambas tecnologias geraram custos totais referentes a R$ 155.37 e a R$ 191.81, respectivamente. Em relação à PB, as instâncias alocadas no caso 03 representaram aumentos percentuais de 24.68% e de 53.91%, respectivamente. Nesse caso, as VMs apresentaram diferenças de 63.24 kW/ano no consumo de energia total e de R$ 36.43 no custo total, resultando no acréscimo percentual de 23.45 % em relação aos contêineres. Considerando as 06 instâncias agrupadas no caso 06, a potência elétrica demandada pelos contêineres e VMs corresponderam a 106.79 kW/ano e a 233.28 kW/ano, respectivamente. Os valores totais das duas operações corresponderam a 323.14 kW/ano e a 449.93 kW/ano, respectivamente. Na perspectiva financeira do caso 06, ambas tecnologias geraram custos totais referentes a R$ 186.13 e a R$ 258.98, respectivamente. Em relação à PB, as instâncias alocadas no caso 06 representaram aumentos percentuais de 49.36% e de 107.83%, respectivamente. Nesse caso, as VMs apresentaram diferenças de 126.49 kW/ano no consumo de energia e de R$ 72.86 no custo total, resultando no acréscimo percentual de 39,14% em relação aos contêineres. As demais análises estão expostas na Tabela 7, as quais apresentaram projeções mais favoráveis aos contêineres acerca do consumo de energia e custo.

Todos dos resultados indicaram que as operações baseadas em VMs apresentaram um consumo de energia mais acentuados do que contêineres. Sobretudo quando os escopos aumentaram de capacidade à medida que as instâncias foram agregadas. Por exemplo, tendo como base o valor de referência, a última análise do caso 10 mostrou que o agrupamento com 10 instâncias gerou um aumento percentual de 82.27% para os contêineres e 179.71% para as VMs. Quando a comparação ocorreu entre contêineres e VMs cujos casos tiveram os mesmos tamanhos, observou-se que a operação da

Tabela 7. Consumo de energia e custo anual em relação aos agrupamentos de instâncias

Quantidade de Instâncias	Projeção Anual dos Ambientes baseados em Contêineres					Projeção Anual dos Ambientes baseados em VMs					Aumento % da VM
	Vazão	kW/ano	Total-kW/ano	Custo/ano	Total-Custo/ano)	Vazão	kW/ano	Total-kW/ano	Custo/ano	Total-Custo/ano	
01-Instância	1.68	17.80	234.14	R$ 10.25	R$ 134.87	1.67	38.88	255.23	R$ 22.39	R$ 147.01	9.00%
02-Instâncias	3.36	35.60	251.94	R$ 20.50	R$ 145.12	3.33	77.76	294.11	R$ 44.79	R$ 169.40	16.74%
03-Instâncias	5.04	53.40	269.74	R$ 30.76	R$ 155.37	5.00	116.64	332.99	R$ 67.18	R$ 191.80	23.45%
04-Instâncias	6.71	71.19	287.54	R$ 41.01	R$ 165.62	6.67	155.52	371.87	R$ 89.58	R$ 214.19	29.33%
05-Instâncias	8.39	88.99	305.34	R$ 51.26	R$ 175.87	8.33	194.40	410.75	R$ 111.97	R$ 236.59	34.52%
06-Instâncias	10.07	106.79	323.14	R$ 61.51	R$ 186.13	10.00	233.28	449.63	R$ 134.37	R$ 258.98	39.14%
07-Instâncias	11.75	124.59	340.93	R$ 71.76	R$ 196.38	11.66	272.16	488.51	R$ 156.76	R$ 281.38	43.28%
08-Instâncias	13.43	142.39	358.73	R$ 82.02	R$ 206.63	13.33	311.04	527.39	R$ 179.16	R$ 303.77	47.01%
09-Instâncias	15.11	160.19	376.53	R$ 92.27	R$ 216.88	15.00	349.92	566.27	R$ 201.55	R$ 326.17	50.39%
10-Instâncias	16.79	177.98	394.33	R$ 102.52	R$ 227.13	16.66	388.80	605.15	R$ 223.95	R$ 348.56	53.46%

VM do caso 01 demandou 9.00% a mais de energia elétrica do que o contêiner do caso 01. Adicionalmente, a operações das VMs caso 10 demandaram 53,46% a mais de energia elétrica do que os contêineres do caso 10. Na perspectiva do melhor aproveitamento dos recursos computacionais, um computador hospedeiro suportaria uma maior quantidade de instâncias baseadas em contêineres, visto que as VMs tendem a ocupar maiores parcelas dos níveis de utilização da CPU. Do ponto de vista da redução do custo, uma elevada quantidade de operação baseadas em VMs migradas para contêineres poderia representar uma redução considerável no custo do consumo de energia atribuído à operação do *data center*. Por isso, independentemente do valor da tarifa cobrada pela concessionária de energia elétrica, a diferença na energia demandada pelas VMs refletiria de médio a logo prazo no custo operacional da infraestrutura. Dessa forma, a adoção de ambientes baseados em contêineres pode indicar uma alternativa viável na minimização do custo operacional do *data center*. Muito embora, a proposta deste artigo não é sugerir preferencialmente uma tecnologia em detrimento de outra, uma vez que ambas podem ser complementares na composição de arquiteturas híbridas para diversas aplicações.

8. Conclusão e Trabalhos Futuros

Este trabalho propôs uma abordagem integrada de experimentos e modelo para o planejamento da capacidade de operações provisionadas por abordagens denominadas de conteinerização e virtualização. As análises do modelo abordaram cenários experimentais acerca da avaliação de desempenho, consumo de energia e custo de operações baseadas em contêineres e VMs. Para esse propósito, uma infraestrutura computacional compartilhou parcelas equiparadas de hardware com intuito de observar o comportamento da vazão das requisições por segundos, do tempo de resposta, da potência elétrica e dos custos em virtude de operações baseadas nos contêineres distribuídos pelo *Docker Engine* e máquinas virtuais distribuídas pelo *Hypervisor KVM*. O ambiente utilizou instâncias da plataforma *Moodle* para oferecer turmas de experimentação e, posteriormente, demandá-las com níveis de cargas de trabalhos que abrangeram conjuntos de requisições frequentemente utilizadas pelos alunos.

As amostras foram coletadas nos experimentos realizados em cinco turmas do *Moodle* com quantidades de alunos variadas. A vazão média de cada turma apresentou um intervalo de confiança com 95% que forneceu limites inferiores e superiores para verificar as conformidades entre as médias obtidas nas análises do modelo. As equivalências entre as médias do sistema real e do modelo foram validadas dentro da margem de erro predefinida. Dessa forma, o modelo apresentou resultados precisos e consistentes em comparação ao desempenho do sistema real. Uma vez validada a corretude do modelo, novos cenários foram analisados com perspectivas de escopos distintos. O primeiro cenário observou o comportamento da vazão de requisições por segundos em relação aos tempos de chegadas com variações de 100 milissegundos até 10 segundos, totalizando 19 escopos experimentais. Os resultados indicaram que o desempenho da operação sofreu quedas 14,89% e 40,42% na vazão quando os tempos de chegadas aumentaram para 700 ms e 1 segundo, respectivamente. O cenário II avaliou os gargalos na operação provenientes dos aumentos nas demandas das turmas. Os resultados mostraram a degradação no tempo de resposta à medida que os escopos das turmas acrescentavam 20 alunos, representando um ponto crítico na entrega do serviço.

O cenário III forneceu estimativas de escopos com instâncias suficientes para dar vazão às demandas das turmas, bem como, reduzir os gargalos causados pelos atrasos no tempo de resposta. Nesse cenário, uma análise considerou 10 turmas com 20 alunos e 100ms para os tempos de chegadas entre requisições. Os resultados revelaram que um agrupamento com 10 instâncias teria um desperdício de 40% da capacidade em decorrência da carga de trabalho das 10 turmas, visto que apenas 6 instâncias foram suficientes para suportar a demanda. Por fim, o cenário IV observou o consumo de energia e o custo relacionado à vazão demandada pelas requisições das turmas. Nessa análise, a potência elétrica demandada foi observada em decorrência da vazão suportada nas operações dos contêineres e VMs. Um agrupamento com 10 instâncias, ocupado na sua vazão máxima, gerou uma redução de 46,54% na demanda de energia favor dos contêineres. O custo da energia demandada levou em consideração a tarifa média nacional para realizar projeções custos anuais. Apesar das duas tecnologias coexistirem na maioria das arquiteturas de sistemas, os resultados indicaram que uma provável migração gradual de um ambi-

ente baseado em VMs para um de contêineres pode preservar a infraestrutura existente e ainda teria uma redução considerável no custo do consumo de energia atribuído à operação do *data center*. Além disso, a iniciativa poderia contribuir diretamente com a redução da emissão de gases de efeitos estufa. Todavia, é importante destacar que este trabalho não descarta a adoção de ambientes *Mooble* provisionados exclusivamente por VMs, porém sugere a adoção desta metodologia proposta para que a capacidade da operação seja otimizada na perspectiva de desempenho, do consumo de energia e do custo.

Em trabalhos futuros, almejamos avaliar outros ambientes AVAs, como o Claroline ou o TelEduc. Almejamos também avaliar a persistência da aplicação através de sistema de gerenciamento de banco de dados não relacionais, os quais tendem a ser mais aderentes às arquiteturas de microsserviços, na perspectiva de desempenho e escalabilidade. Esses bancos são conhecidos pelo termo *(Not only Structure Query Language - NoSQL)*. Outro possível trabalho futuro é criar SPNs para representar o comportamento de falha e reparo dos ambientes baseados em contêineres e VMs.

Agradecimentos

Esta pesquisa foi parcialmente financiada pelo CNPq - Brasil, processos 406263/2018-3 e 308266/2020-0.

Contribuições dos autores

Os autores Cleyton Gonçalves e Ermeson Andrade foram responsáveis pela escrita e pela organização do artigo, incluindo a coordenação da pesquisa. Os autores Júlio Mendonça e Gustavo Callou contribuíram na revisão do trabalho e na discussão dos resultados.

Referências

[1] JONES, N. How to stop data centres from gobbling up the world's electricity. *Springer Nature Limited*, v. 561, p. 163–166, September 2018.

[2] AMDOCS. *New Streamer 2021 Report*. 2021. ⟨https://www.amdocs.com/sites/default/files/New-Streamer-2021-Report-08Feb21.pdf⟩. [Online; acessado em: 23 de outubro de 2021].

[3] ANATEL. *Com maior uso da internet durante pandemia, número de reclamações aumenta*. 2020. ⟨https://g1.globo.com/economia/tecnologia/noticia/2020/06/11/com-maior-uso-da-internet-durante-pandemia-numero-de-/reclamacoes-aumenta-especialistas-apontam-problemas-/mais-comuns.ghtml⟩. [Online; acessado em: 23 de Outubro de 2021].

[4] SALEKHOVA, L. L.; GRIGORIEVA, K. S.; ZINNUROV, T. A. Using lms moodle in teaching clil: A case study. In: *2019 12th International Conference on Developments in eSystems Engineering (DeSE)*. [S.l.: s.n.], 2019. p. 393–395.

[5] LIMA, C. J. d. et al. Performance and power consumption evaluation of the moodle environment. *Revista Eletrônica de Gestão Organizacionalt*, v. 17, n. 5, p. 120–133, May 2019.

[6] Bhimani, J. et al. Accelerating big data applications using lightweight virtualization framework on enterprise cloud. In: *2017 IEEE High Performance Extreme Computing Conference (HPEC)*. [S.l.: s.n.], 2017. p. 1–7.

[7] Salah, T. et al. Performance comparison between container-based and vm-based services. In: *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*. [S.l.: s.n.], 2017. p. 185–190.

[8] BALBO, G. Introduction to generalized stochastic petri nets. In: . Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 83–131.

[9] MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, v. 77, n. 4, p. 541–580, April 1989.

[10] MARSAN, M. A. et al. *Modelling with Generalized Stochastic Petri Nets*. 1st. ed. USA: John Wiley & Sons, Inc., 1994.

[11] ANEEL. *Agência Nacional de Energia Elétrica*. 2020. ⟨http://www.aneel.gov.br/ranking-das-tarifas⟩. [Online; acessado em: 11 de Dezembro de 2020].

[12] DEHON, P. et al. Cvchatbot: Um chatbot para o aplicativo facebook messenger integrado ao ava moodle. In: . [S.l.: s.n.], 2018. p. 1623–1632.

[13] JR, J. C. C. Utilização do ava moodle e suas contribuições no processo de ensino-aprendizagem: um relato de experiência da plataforma em uma disciplina de ciências humanas voltada à saúde. In: . [S.l.: s.n.], 2019. v. 13, p. 6–26.

[14] MOODLE. *Moodle statistics*. 2021. ⟨https://moodle.net/stats/⟩. [Online; acessado em: 31 de Março de 2021].

[15] BOS, H.; TANENBAUM, A. *Sistemas Operacionais Modernos*. [S.l.]: PEARSON BRASIL, 2015. 325-341 p.

[16] REDHAT. *O que é virtualização?* 2020. ⟨https://www.redhat.com/pt-br/topics/virtualization/what-is-virtualization⟩. [Online; acessado em: 23 de Dezembro de 2020].

[17] VMWARE. *Virtualização*. 2020. ⟨https://www.vmware.com/br/solutions/virtualization.html⟩. [Online; acessado em: 24 de Dezembro de 2020].

[18] Mukhedkar, P.; Vettathu, A. *Mastering KVM Virtualization*. 1st. ed. [S.l.]: Packt Publishing, 2016. 01–10 p.

[19] Ivanov, K. *KVM Virtualization Cookbook*. 1st. ed. [S.l.]: Packt Publishing, 2017. 40 p.

[20] COSTA, L. H. M. K. *O que é virtualização?* 2021. ⟨https://www.gta.ufrj.br/ensino/eel879/trabalhos_v1_2017_2/kvm/⟩. [Online; acessado em: 09 de Novembro de 2021].

[21] Nickoloff, S. K. J.; Fisher, B. *Docker in Action*. 1st. ed. [S.l.]: Manning books, 2019. 30–40 p.

[22] Turnbull, J. *The Docker Boook - Containerization is the new virtualization*. 2016. ⟨https://github.com/eduleboss/the-best-docker-books/blob/master/books/The%20Docker%20Book%20-%20James%20Turnbull%20-%20v17.03.0.pdf⟩. [Online; acessado em: 07-Maio-2019].

[23] Cuadrado-Cordero, I.; Orgerie, A.; Menaud, J. Comparative experimental analysis of the quality-of-service and energy-efficiency of vms and containers' consolidation for cloud applications. In: *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. [S.l.: s.n.], 2017. p. 1–6.

[24] SILVA, B.; MACIEL, P.; ZIMMERMANN, A. Performability models for designing disaster tolerant infrastructure-as-a-service cloud computing systems. In: *8th International Conference for Internet Technology and Secured Transactions (ICITST-2013)*. [S.l.: s.n.], 2013. p. 647–652.

[25] PETRI, C. *Kommunikation mit Automaten*. [S.l.]: Rheinisch-Westfälisches Institut f. instrumentelle Mathematik an d. Univ., 1962. (Schriften des Rheinisch-Westfälischen Institutes für Instrumentelle Mathematik an der Universität Bonn).

[26] FIENI, G.; ROUVOY, R.; SEINTURIER, L. Smartwatts: Self-calibrating software-defined power meter for containers. In: *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. [S.l.: s.n.], 2020. p. 479–488.

[27] PHUNG, J.; LEE, Y. C.; ZOMAYA, A. Y. Lightweight power monitoring framework for virtualized computing environments. *IEEE Transactions on Computers*, v. 69, n. 1, p. 14–25, 2020.

[28] ZHENG, R. et al. Energy saving strategy of power system cluster based on container virtualization. In: *2020 Asia Energy and Electrical Engineering Symposium (AEEES)*. [S.l.: s.n.], 2020. p. 351–355.

[29] Chen, F.; Zhou, X.; Shi, C. The container deployment strategy based on stable matching. In: *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*. [S.l.: s.n.], 2019. p. 215–221.

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

[31] Yadav, R. R.; Sousa, E. T. G.; Callou, G. R. A. Performance comparison between virtual machines and docker containers. *IEEE Latin America Transactions*, v. 16, n. 8, p. 2282–2288, Aug 2018.

[32] Salah, T. et al. Performance comparison between container-based and vm-based services. In: *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*. [S.l.: s.n.], 2017. p. 185–190.

[33] Bhimani, J. et al. Accelerating big data applications using lightweight virtualization framework on enterprise cloud. In: *2017 IEEE High Performance Extreme Computing Conference (HPEC)*. [S.l.: s.n.], 2017. p. 1–7.

[34] Barik, R. K. et al. Performance analysis of virtual machines and containers in cloud computing. In: *2016 International Conference on Computing, Communication and Automation (ICCCA)*. [S.l.: s.n.], 2016. p. 1204–1210.

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

[36] Tadesse, S. S.; Malandrino, F.; Chiasserini, C. Energy consumption measurements in docker. In: *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*. [S.l.: s.n.], 2017. v. 2, p. 272–273.

[37] LITTLE, J. D. C. A Proof for the Queuing Formula: L = (lambda) W. *Operations Research*, v. 9, n. 3, p. 383–387, June 1961.

[38] IFSTAT. *Ifstat Network Monitor*. 2020. ⟨https://linux.die.net/man/1/ifstat⟩. [Online; acessado em: 02 de Novembro de 2020].

[39] NMON. *NMON for Linux*. 2020. ⟨http://nmon.sourceforge.net/pmwiki.php⟩. [Online; acessado em: 02 de Novembro de 2020].

[40] JMETER. *Apache JMeter*. 2020. ⟨http://jmeter.apache.org⟩. [Online; acessado em: 02 de Novembro de 2020].

[41] WATTSUP. *Watts Up Power Meter*. 2020. ⟨https://arcb.csc.ncsu.edu/~mueller/cluster/arc/wattsup/usb/watts-up-meters-manual.pdf⟩. [Online; acessado em: 02 de Novembro de 2020].

[42] SILVA, B. et al. Mercury: An integrated environment for performance and dependability evaluation of general systems. In: . [S.l.: s.n.], 2015.

[43] JAIN, R. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* [S.l.]: Wiley, 1991. I-XXVII, 1-685 p. (Wiley professional computing).

[44] KENETT, R. S.; ZACKS, S.; AMBERTI, D. Modern industrial statistics: with applications in r, minitab and jmp. In: . [S.l.: s.n.], 2014.

[45] R. *An Introduction to R*. 2021. ⟨https://cran.r-project.org/doc/manuals/r-release/R-intro.html⟩. [Online; acessado em: 14 de Novembro de 2021].

[46] GRIMA, P.; ALMARGO, L.; LLABRES, X. *Industrial Statistics with Minitab*. [S.l.: s.n.], 2012.

[47] APACHE. *Apache HTTP Server Documentation*. 2021. ⟨https://httpd.apache.org/docs-project/⟩. [Online; acessado em: 25 de Dezembro de 2021].

[48] MYSQL. *MySQL Documentation*. 2021. ⟨https://dev.mysql.com/doc/⟩. [Online; acessado em: 25 de Dezembro de 2021].

[49] PHP. *PHP Documentation*. 2021. ⟨https://www.php.net/docs.php⟩. [Online; acessado em: 25 de Dezembro de 2021].