INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DO RIO GRANDE DO NORTE

UIATAMARA JHULIENE SILVA FERREIRA

ANÁLISE DE TECNOLOGIAS DE VIRTUALIZAÇÃO E HARDWARE DE BAIXO CUSTO PARA INFRAESTRUTURA DE NUVEM DE PEQUENO PORTE

NATAL–RN
2017

UIATAMARA JHULIENE SILVA FERREIRA

ANÁLISE DE TECNOLOGIAS DE VIRTUALIZAÇÃO E HARDWARE DE BAIXO CUSTO PARA INFRAESTRUTURA DE NUVEM DE PEQUENO PORTE

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Redes de Computadores do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial à obtenção do título de Tecnólogo em Redes de Computadores.

Orientador: Prof. M.Sc. Francisco Sales de Lima Filho.
Coorientador: Prof. M.Sc. Marcelo Rômulo Fernandes.

NATAL-RN

2017

F383a Ferreira, Uiatamara Jhuliene Silva.
Análise de tecnologias de virtualização e hardware de baixo custo para infraestrutura de nuvem de pequeno porte. / Uiatamara Jhuliene Silva Ferreira – 2017.
43 f. : il.

Orientador: Me. Francisco Sales de Lima Filho.
Coorientador: Me. Marcelo Rômulo Fernandes.
Trabalho de Conclusão de Curso (Tecnologia em Redes de Computadores) – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, 2017.

1. Virtualização 2. Conteinerização. 3. Hardware de baixo custo. 4. Raspberry Pi. 5.Infraestrutura de nuvem. I. Lima Filho, Francisco Sales. II. Fernandes, Marcelo Rômulo. III. Título.

CDU 004.7

Ficha elaborada pela Seção de Informação e Referência da Biblioteca Sebastião Fernandes do IFRN.

UIATAMARA JHULIENE SILVA FERREIRA

ANÁLISE DE TECNOLOGIAS DE VIRTUALIZAÇÃO E HARDWARE DE BAIXO CUSTO PARA INFRAESTRUTURA DE NUVEM DE PEQUENO PORTE

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Redes de Computadores do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, em cumprimento às exigências legais como requisito parcial à obtenção do título de Tecnólogo em Redes de Computadores.

Trabalho de Conclusão de Curso apresentado em ___/___/____, pela seguinte Banca Examinadora:

BANCA EXAMINADORA

Prof. M.Sc. Francisco Sales de Lima Filho – Orientador
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

---

Prof. M.Sc. Marcelo Rômulo Fernandes – Coorientador
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

---

Prof. Dr. Jorgiano Márcio Bruno Vidal – Examinador
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

---

Prof. Esp. Moisés Cirilo de Brito Souto – Examinador
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

Dedico ao meu amado namorado, que esteve ao meu lado me auxiliando e incentivando em todas as situações ao decorrer deste trabalho e do curso. Sem a sua dedicação e o seu apoio seria impossível seguir a diante. Esta vitória é apenas a primeira de muitas que iremos juntos conquistar.

AGRADECIMENTOS

Ao Senhor Jesus Cristo, que com sua graça me concedeu a vida, esteve comigo em cada um dos meus passos me sustendo e guardando e é o principal responsável por esta conquista.

Aos meu pais, que me apoiaram em cada etapa de minha vida, investindo em minha educação e na formação do meu caráter.

Aos professores do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN), que colaboraram e construíram bases sólidas ao meu desenvolvimento e aprendizagem para o crescimento profissional. Seus nomes são inesquecíveis e por isso, dedico-lhes minha profunda admiração e respeito.

A todos aqueles que acreditaram na realização deste trabalho e deram-me força e estímulo para dar prosseguimento a esta pesquisa e obter sucesso. Em especial, ao meu orientador, Francisco Sales, que com excelência, paciência e dedicação orientou a mim e ao meu amado namorado, enchendo nossos corações de gratidão.

RESUMO

A crescente demanda por serviços ofertados em nuvem computacional, provocou nos últimos anos investimentos na ordem de milhões de reais na construção e operação de *data centers*. Visando reduzir os custos de aquisição e operação dessas infraestruturas, pesquisadores buscam desenvolver alternativas que entreguem serviços de computação em cenários mais específicos de nuvem a preços mais baixos. Nesse contexto, este trabalho faz uma análise comparativa de execução de uma aplicação típica da Internet em ambientes de virtualização tradicional (Máquina Virtual), conteinerização (*container* LXD) e em *hardware* de baixo custo (Raspberry Pi), com intuito de viabilizar a utilização de arranjos de *hardware* mais baratos para infraestruturas de nuvem de pequeno porte. Assim, com o apoio das ferramentas Jmeter e Collectl, foram elaborados experimentos de exaustão de recursos para operações básicas de banco de dados como gravação (INSERT), consulta (SELECT) e exclusão (DELETE) na aplicação WebTool. Os resultados revelaram que o Raspberry Pi equivale a 12% do *Container* LXD com relação quantidade máxima de usuários simultâneos no sistema na operação DELETE. Em contrapartida, na operação SELECT, o Raspberry Pi apresentou um ganho de 6% quando comparado a Máquina Virtual e representou 94% do *Container* LXD. Dessa forma, foi possível concluir que é viável, em infraestruturas de pequeno porte, implementar serviços de nuvem para aplicações típicas da Internet utilizando Raspberry Pi.

Palavras-chave: Virtualização. Conteinerização. *Hardware* de baixo custo. Raspberry Pi. Infraestrutura de nuvem.

ABSTRACT

The increasing demand for services offered in the cloud computing, has caused in recent years investments in the order of millions of reais in the construction and operation of data centers. Aiming to reduce the acquisition and operation costs of these infrastructures, researchers seek to develop alternatives that deliver computing services, in more specific cloud scenarios at lower prices. In this context, this work makes a comparative analysis of the execution of a typical Internet application in traditional virtualisation (Virtual Machine), containerization (Container LXD) and low cost hardware (Raspberry Pi) environments, with the aim of making feasible the use of cheaper hardware arrays for small cloud infrastructures. Thus, with the support of the tools Jmeter and Collectl, resource exhaustion experiments were developed for basic database operations such as write (INSERT), query (SELECT) and delete (DELETE) in the WebTool application. The results revealed that Raspberry Pi is equivalent to 12% of Container LXD with maximum number of concurrent users on the system in DELETE operation. In contrast, in the SELECT operation, Raspberry Pi presented a gain of 6% when compared to Virtual Machine and accounted for 94% of Container LXD. In this way, it was possible to conclude that it is feasible, in small infrastructures, to implement cloud services for typical Internet applications using Raspberry Pi.
Keywords: Virtualization. Conteinerization. Low cost hardware. Raspberry Pi. Cloud Infrastructure.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A crescente demanda por serviços ofertados em nuvem computacional, em suas tradicionais modalidades incluindo Infraestrutura como Serviços (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS), provocou nos últimos anos fortes investimentos em *data centers* para garantir a entrega desses serviços. De modo geral, isso resulta em investimentos na ordem de milhões de reais na construção e operação desses centros de dados.

Visando reduzir os custos de aquisição e operação dessas infraestruturas, pesquisadores industriais e acadêmicos buscam desenvolver alternativas mais baratas e que entreguem serviços de computação em cenários mais específicos de nuvem a preços mais baixos.

Nesse contexto, este trabalho faz uma análise comparativa de execução de uma aplicação típica da Internet em ambientes de virtualização tradicional (Máquina Virtual), conteinerização (*container* LXD) e em *hardware* de baixo custo (Raspberry Pi), com intuito de inferir a viabilidade de utilizar arranjos de *hardware* mais baratos para entrega de serviços de nuvem de pequeno porte.

Assim, com o apoio das ferramentas Jmeter e Collectl foram elaborados cenários de teste de exaustão de recursos com o objetivo de determinar o limite máximo de respostas com sucesso de cada ambiente para operações básicas de gravação (INSERT), consulta (SELECT) e exclusão (DELETE) em banco de dados utilizando a aplicação WebTool, desenvolvida na linguagem Python.

Os resultados extraídos do Jmeter em conjunto com a aplicação WebTool revelam que o Raspberry Pi equivale a 12% do *Container* LXD para a métrica adotada de quantidade máxima de usuários simultâneos no sistema na operação DELETE. Ainda nesse contexto, na operação SELECT, o Raspberry Pi apresentou um ganho de 6% quando comparado a Máquina Virtual. Além disso, mesmo com menor taxa de transmissão, o Raspberry Pi suportou 94% do total de usuários do *Container* LXD com menor utilização de CPU para a operação SELECT.

Por fim, foi possível também inferir que é viável, em infraestruturas de pequeno porte, implementar serviços de nuvem para aplicações típicas da Internet utilizando Raspberry Pi. A principal vantagem dessa utilização é que esses dispositivos são escaláveis, o que facilita a expansão dessas infraestruturas com um baixo custo.

1.1 OBJETIVOS GERAIS

Para a realização deste trabalho científico foram estabelecidos como objetivos gerais: **a)** produzir uma revisão bibliográfica da literatura relativa a computação em nuvem, virtualização e conteinerização; **b)** realizar um estudo a respeito de dispositivos com *hardware* de baixo custo, **c)** estudar o funcionamento de ferramentas de testes de exaustão de aplicações web e monitoramento de sistemas; **e)** analisar o desempenho de tecnologias de virtualização e *hardware* de baixo custo em diferentes cenários.

1.2 OBJETIVOS ESPÉCIFICOS

Como objetivos específicos deste trabalho foram definidos: **a)** implementar cenário de testes usando Máquina Virtual, *Container* LXD e Raspberry Pi; **b**) dominar os fundamentos e a utilização das ferramentas a serem implantadas; **c**) definir e executar um método de teste para esses cenários; **d**) realizar análise quantitativa dos testes realizados.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está organizado em 5 capítulos, incluindo o atual. O capítulo 2 apresenta um estudo teórico a respeito dos conceitos fundamentais de infraestruturas de nuvem, incluindo computação em nuvem e seus serviços, os tipos de *data centers*, virtualização e conteinerização, além dos conceitos de dispositivos de *hardware* de baixo custo.

Em seguida, o capítulo 3 aborda os materiais e os métodos utilizados nos experimentos, bem como os resultados obtidos em cada cenário de testes. Logo depois, no capítulo 4 estarão descritos os resultados e as discussões a respeito do que foi apresentado no capítulo anterior. Finalmente, no quinto e último capítulo são relatadas as considerações finais sobre toda a obra, incluindo as conclusões gerais do trabalho e propostas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo, são abordados os principais conceitos a respeito de infraestruturas de computação em nuvem, quais serviços podem ser oferecidos e as características de *data centers*. São apresentados aspectos de virtualização e conteinerização, como esses conceitos podem ser aplicados em *data centers*, bem como o uso de *hardware* de baixo em infraestrutura de nuvem.

2.1 COMPUTAÇÃO EM NUVEM

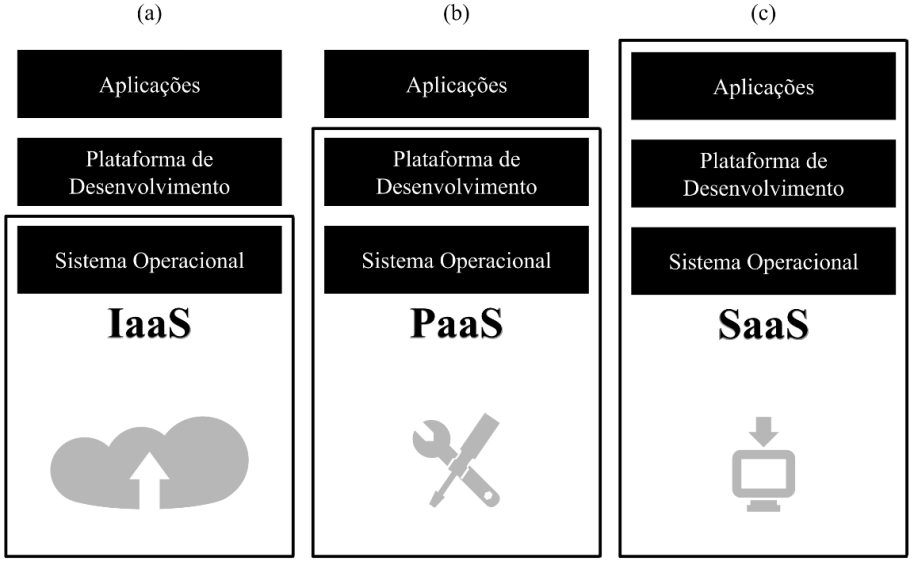
O *National Institute of Standards and Technology* (NIST) define computação em nuvem como um conjunto de recursos compartilhados (como redes, servidores, armazenamento, aplicações de *software* e serviços de gerenciamento de dados) que podem ser rapidamente fornecidos e liberados com o esforço de gerenciamento mínimo ou serviço de interação com o provedor (NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, 2011), além de não ter a necessidade dos usuários da nuvem conhecerem a localização e outros detalhes a respeito da infraestrutura de computação, conforme (GOPULARAM, B; PERIASAMY, 2012). A (SUN MICROSYSTEMS, INC., 2009) afirma que a computação em nuvem se baseia em tendências estabelecidas para dirigir o custo de entrega dos serviços enquanto aumenta a velocidade e a agilidade com que os serviços são implementados. De uma perspectiva, a nuvem não é nada novo por usar abordagens, conceitos e melhores práticas já definidas. De outra perspectiva, tudo é novo por mudar a forma de inventar, desenvolver, implementar, atualizar, manter e pagar por aplicações e infraestruturas na qual ela atua.

Há três distintos modos de introduzir os recursos da computação em nuvem em organizações que escolhem esse modelo para implementar aplicações, segundo o Microsoft Azure (MICROSOFT CORPORATION, 2016): nuvem pública, nuvem privada e nuvem híbrida. A medida que a nuvem é de propriedade de um provedor de serviços de nuvem de terceiros e operada por ele, que por sua vez fornece recursos de computação, chama-se nuvem pública. O próprio Microsoft Azure é um exemplo de nuvem pública. O termo nuvem privada é empregado quando os recursos de computação em nuvem são usados exclusivamente por uma única empresa ou organização. Uma nuvem privada pode estar localizada fisicamente no *data center* local da empresa. Por fim, a nuvem híbrida diz respeito a combinação entre

nuvens públicas e privadas, permitindo que dados e aplicativos sejam compartilhados entre elas.

A (SUN MICROSYSTEMS, INC., 2009) ainda assegura que provedores de serviços de nuvem tendem a oferecer serviços que podem ser agrupados em três categorias: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) e *Software as a Service* (SaaS). A IaaS oferece sistemas de armazenamento e recursos computacionais em forma de serviços padronizados através da rede, como mostra a Figura 1 (a). De acordo com (VERDI, ROTHENBERG, *et al.*, 2010), usuários que recebem esse tipo de serviço não possuem o controle da estrutura física, mas possuem controle sobre os sistemas operacionais, armazenamento, aplicações instaladas e, possivelmente, um controle limitado dos recursos de rede através de mecanismos de virtualização. Entre os exemplos comerciais de IaaS está incluso o Joyent (JOYENT, INC., 2016), cujo principal produto é uma linha de servidores virtualizados que fornecem uma infraestrutura sob demanda altamente disponível. Em contrapartida, a PaaS encapsula uma camada de *software* e fornece como serviço que pode ser usado para construir outros serviços de alto nível, conforme a Figura 1 (b). Um exemplo é a produção de uma plataforma de desenvolvimento, que é então fornecida a um cliente como um serviço. Entre os exemplos comerciais de PaaS, está incluso o Google App Engine (GOOGLE INC., 2016). Ainda existe o SaaS, que dispõe de uma completa aplicação oferecida como um serviço sob demanda, ilustrada na Figura 1 (c). Um dos exemplos largamente conhecidos de SaaS é o salesforce.com (SALESFORCE.COM, INC., 2016), entretanto muitos outros exemplos vieram ao mercado, incluindo a oferta do Google Apps for Work (GOOGLE INC., 2016) de serviços empresariais básicos.

Figura 1- Serviços oferecidos pela computação em nuvem.



Fonte: Baseado em (BROADSOFT JAPAN K.K, 2015).

A computação em nuvem obteve um alto crescimento durante o ano de 2016 em relação ao ano anterior. Um levantamento realizado pela RightScale (RIGHTSCALE, INC., 2016), aponta que nas grandes empresas o papel das equipes de TI com foco em nuvem tem evoluído ao longo dos últimos anos. Além das vantagens empresariais, os usuários de nuvem entrevistados ainda relataram a respeito da velocidade no acesso à infraestrutura (aumento de 57% para 62%), maior escalabilidade (aumento de 30% para 36%) e alta disponibilidade (aumento de 26% para 31%) nos últimos dois anos.

2.2 INFRAESTRUTURA DE DATACENTER

Segundo (VERDI, ROTHENBERG, *et al.*, 2010), as infraestruturas de nuvem consistem em *data centers* os quais hospedam servidores que, mediante diferentes níveis de organização e técnicas de virtualização, oferecem os serviços em nuvem. Dessa maneira, os *data centers* são a manifestação física da computação em nuvem, sendo a infraestrutura de rede a base da comunicação, interligando servidores físicos em grande escala.

Verdi (VERDI, ROTHENBERG, *et al.*, 2010) ainda afirma que, dependendo do tamanho da própria infraestrutura física e sua localização, os *data centers* podem ser classificados como: (1) mega, onde possuem aplicações que requerem alto uso de memória RAM, vários ciclos de CPU e/ou uma grande capacidade de armazenamento em disco rígido; (2) micro, o qual as aplicações não exigem tanta capacidade computacional, contudo são fortemente interativas (por exemplo, e-mail e aplicações de escritório como Google Docs); (3) nano, que surge da adoção do paradigma *peer-to-peer* (P2P) e da possibilidade de considerar os equipamentos dos usuários finais como uma extensão natural do *data center*, usando os recursos do usuário para migrar a execução de tarefas, armazenar dados e prover serviços através da instanciação de máquinas virtuais; por fim, *data centers* baseados em *containers* (4), onde são modularizados com até 2.000 servidores, formando um bloco computacional que necessita apenas de energia, água para refrigeração e conectividade de rede (fibra óptica) para ser colocado em funcionamento.

Cada categoria de *data center* abordada, possui seus critérios de construção incluindo a localização, a disponibilidade de recursos de energia, conectividade à Internet, além do tipo de provedor da infraestrutura e fornecedor dos equipamentos. Apesar disso, Verdi (VERDI, ROTHENBERG, *et al.*, 2010) diz que os *data centers* baseados em *containers* têm se tornado mais atrativos devido a economia de energia, pois os sistemas de energia são altamente eficientes, podendo economizar de 40% a 50% das despesas operacionais de *data centers*

tradicionais. Entre os exemplos comerciais desse tipo de *data center* estão inclusos o Portable Modular Data Center (PMDC) da IBM (IBM, 2016) e o Blackbox da Sun Microsystems (ORACLE, 2016).

2.3 COMPUTAÇÃO EM NUVEM EM HARDWARE DE DATACENTER

Com o crescimento de volumes de dados e a variedade de aplicações de Internet, os *data centers* se tornaram uma infraestrutura eficiente e promissora para apoiar o armazenamento de dados e fornecer a plataforma para a implantação de serviços de rede e aplicações (BARI, BOUTABA, *et al.*, 2012). Contudo, segundo (ERICSSON AB, 2016), um dos principais desafios relativos aos *data centers* de hoje é o baixo nível de utilização de recursos. Isso é causado pela fragmentação criada pela variedade de perfis de aplicação (tais como aplicações com rede, memória e computação intensiva). Para enfrentar esse desafio, os operadores de *data centers* utilizam tecnologias de virtualização e conteinerização, a fim de permitir e facilitar o compartilhamento e escalonamento de recursos. Para tanto, será tratado com mais detalhes a respeito de virtualização e conteinerização, a seguir nas subseções 2.3.1 e 2.3.2, respectivamente.

2.3.1 Virtualização

A computação em nuvem, conforme visto na subseção 2.1, tem ganhado popularidade principalmente em ambientes de *data centers*. Como resultado, existe a necessidade de organizações de TI entregar mais rapidamente serviços como IaaS, PaaS e SaaS. Além disso, otimizar e tornar eficiente as soluções para implementar e gerenciar os diversos tipos de *data centers*. Para (ORACLE, 2010), a virtualização é a tecnologia chave para otimizar esses recursos.

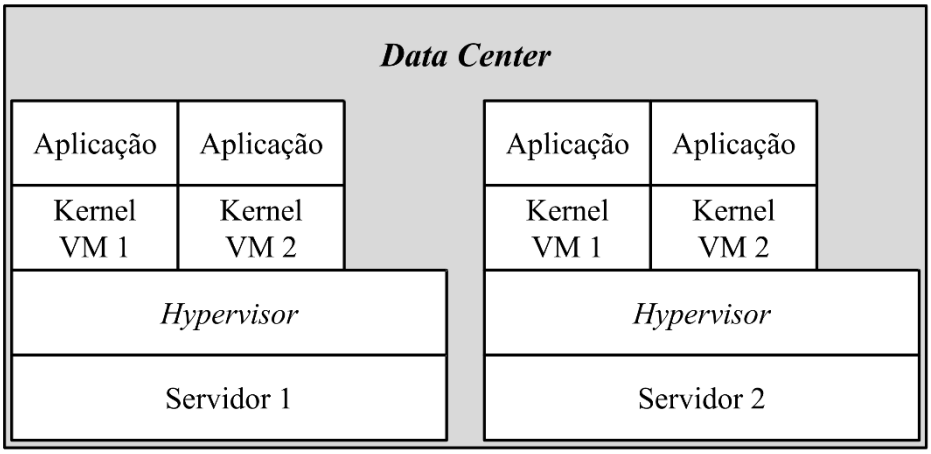
A virtualização foi primeiro introduzida na década de 1960, segundo (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2008), para permitir o particionamento de ambientes *mainframe*. Hoje, a virtualização está também em *data centers*, onde alguns ou todos os *hardwares* são virtualizados. As técnicas de virtualização também podem ser aplicadas em distintas camadas da infraestrutura de TI, incluindo rede, armazenamento, sistemas operacionais e aplicações.

De acordo com (VMWARE, INC., 2006), o principal benefício da virtualização é a habilidade de executar múltiplos sistemas operacionais em um único sistema físico e

compartilhar recursos de *hardware* (CPU, memória, dispositivos de entrada e saída) – conhecido como particionamento. A virtualização ainda facilita o gerenciamento de recursos entre partições.

Além disso, para (VMWARE, INC., 2006), a virtualização pode ser aplicada a nível de *hardware*, de *software* e de aplicação. No particionamento baseado em *software*, por exemplo, há duas propostas: *hosted* e *hypervisor*. A proposta *hosted* provê particionamento de serviços acima do sistema operacional e suporta uma ampla variedade de configurações de *hardware*. Contudo, a arquitetura *hypervisor* é mais eficiente já que se trata da primeira camada de *software* instalada acima do *hardware*, por isso é muitas vezes conhecida como "*bare metal*". Um *hypervisor*, segundo (SCHEEPERS, 2014), é um pedaço de *software* que cria e executa máquinas virtuais. É possível ver esse tipo de configuração em *data centers*, conforme ilustrado na Figura 2. Com os *hypervisors* e as máquinas virtuais que rodam sobre eles, os administradores de sistemas são capazes de otimizar o uso dos recursos físicos disponíveis e isolar partes individuais da infraestrutura de aplicação.

Figura 2 - Visão esquemática de máquinas virtuais em *data centers*.



Fonte: Baseado em (SCHEEPERS, 2014).

As tecnologias de virtualização que surgiram têm principal foco no kernel do Linux e, segundo (SCHEEPERS, 2014), são categorizadas como virtualização tradicional (*full-virtualization*), para-virtualização ou virtualização baseada em *containers*. Na virtualização tradicional, o sistema operacional ou seus componentes podem executar dentro da máquina virtual sem modificações, diz (VMWARE, INC., 2005). Ao contrário, a para-virtualização requer que o sistema operacional seja modificado antes que ele possa ser executado na máquina virtual. A virtualização baseada em *containers* será discutida com mais detalhes na subseção 2.3.2.

A International Business Machines Corporation (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2008) assegura que a virtualização de recursos aumenta a

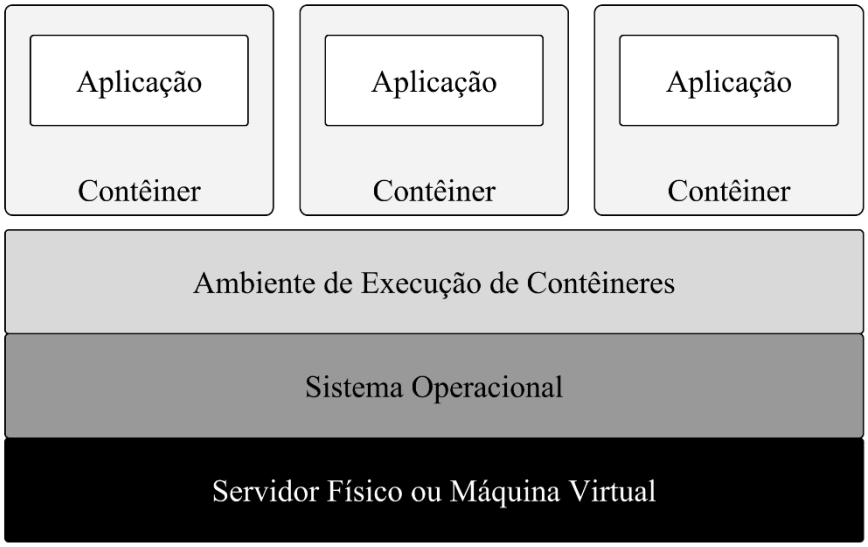
disponibilidade do sistema e reduz a complexidade das soluções de recuperação de desastres, assim minimizando o tempo de inatividade. Além disso, a virtualização pode ajudar no controle de expansão da infraestrutura por meio de servidores virtuais que atuam de forma segura em um ambiente de *hardware* compartilhado. Uma infraestrutura virtual, ainda, pode permitir o aumento da taxa de utilização do servidor de 15% para mais de 70%. Dessa forma, a infraestrutura ajuda a reduzir os custos de gerenciamento com uma plataforma de gestão comum e ferramental.

2.3.2 Conteinerização

A virtualização baseada em *container* usa o *kernel* do sistema operacional hospedeiro (pode estar em um servidor físico ou em uma máquina virtual) para executar várias instâncias isoladas, conforme esquematizado na Figura 3. Ao fazer uso do *kernel* do sistema operacional hospedeiro e não depender de um *hypervisor* para gerir os recursos, os *containers* tornam-se mais leves do que outras soluções, além de permitir instâncias completamente isoladas. Cada *container* tem seu próprio sistema de arquivos *root*, processos, memória, dispositivos e interface de rede (CODESHIP, 2015).

Em (HEWLETT PACKARD ENTERPRISE, 2016), diz que o uso de *containers* promete acelerar os ciclos de implantação de aplicações e aumentar a agilidade da computação em nuvem de forma fundamental. Entre os promovedores dessa tecnologia estão o LXC (WHAT'S LXC?, 2016), o LXD (WHAT'S LXD?, 2016) e o Docker (DOCKER INC., 2017). O LXC é um projeto *open source* que foi integrado com o *kernel* Linux afim de suportar múltiplos *containers* Linux no mesmo servidor físico. Já o LXD é um *container hypervisor* composto por um sistema *daemon*, um cliente de linha de comando (lxc) e um *plugin* OpenStack Nova. De outro lado está o Docker, que incorpora os principais componentes de *software* de um *container*, juntamente com suas dependências como binários, bibliotecas e arquivos de configuração.

Figura 3 - Visão esquemática de virtualização baseada em containers.



Fonte: Baseado em (CODESHIP, 2015).

Containers possibilitam a entrega de aplicações de diversas formas, uma delas está nos *data centers* com aplicações Web, armazenamento de computação, sistemas PaaS e gestão de recursos, por exemplo. Além da vasta utilidade, os *containers* apresentam ainda diversas vantagens, conforme (CISCO | RED HAT, 2014): redução de custos; aceleração no desenvolvimento de aplicações; menos sistemas operacionais para gerenciar; melhor utilização de recursos, entre outros.

2.4 INFRAESTRUTURA DE HARDWARE DE BAIXO CUSTO

Em 1980, quando a era dos computadores pessoais teve início, o custo de um computador estava acima de 3.000 dólares. Ao decorrer do aprimoramento da tecnologia de computação, os notebooks tornaram-se mais funcionais comparados aos computadores e com preços inferiores aos 400 dólares (OU, 2013). Seguindo essa evolução, uma nova geração de minicomputadores com baixo custo tem surgido. Essas plataformas, também conhecidas como *Single Board Computers* (SBCs), segundo (LENCSE e RÉPÁS, 2016), oferecem capacidades relativamente altas de computação em comparação com o preço ou o consumo de energia e, especialmente, ao espaço que ocupam. Para (CUSICK, MILLER, *et al.*, 2014), esses computadores trabalham com energia extremamente baixa, normalmente 5V, a velocidade do processador se aproxima de 1GHz e a memória RAM pode variar de 512 MB e 1GB.

Em 2006, segundo (NEWARK, A TRADEMARK OF PREMIER FARNELL CORP, 2014), um grupo da *University of Cambridge's Computer Laboratory*, decidiu abordar a

necessidade de uma plataforma de baixo custo que permitiria que crianças aprendessem a programar sem a necessidade de um computador tradicional em casa. O resultado foi um SBC de 35 dólares nomeado Raspberry Pi (RASPBERRY PI FOUNDATION, 201-?). Em 28 de julho de 2008, segundo (NEWARK, A TRADEMARK OF PREMIER FARNELL CORP, 2014), nasceu um outro desenvolvedor de SBCs, a organização BeagleBoard.org (BEAGLEBOARD.ORG, 2016). A BeagleBoard.org foi formada para desenvolver modernos minicomputadores que tenham baixo custo, o resultado foi uma plataforma de desenvolvimento conhecida com BeagleBoard.

Há uma variedade de SBCs, incluindo o CHIP (NEXT THING CO, 2016), e o Banana Pi (BANANA PI, 2014). Entretanto, o Raspberry Pi destaca-se pelo seu baixo preço e popularidade. A primeira versão foi lançada em 2012 e atualmente está na sua terceira versão (Raspberry Pi 3). O Raspberry Pi 3, por exemplo, custa aproximadamente 37 dólares e contém 1 GB RAM, suporta Wireless LAN 802.11n, Bluetooth 4.1, 4 portas USB, porta HDMI, entre outras características (RASPBERRY PI FOUNDATION, 201-?). Muitas dos SBCs de hoje estão se tornando tão poderosos a ponto de começar a ter a mesma capacidade de computadores e *tablets* modernos. Ainda de acordo com (NEWARK, A TRADEMARK OF PREMIER FARNELL CORP, 2014), a utilização de SBCs em aplicações sem fio e *Internet of Things* (IoT) irá certamente continuar. RENNER (2015) diz que o número estimado de dispositivos em 2020 pode variar de 26 a 200 bilhões.

2.5 COMPUTAÇÃO EM NUVEM EM HARDWARE DE BAIXO CUSTO

Data centers que suportam serviços em nuvem, como visto na subseção 2.2, frequentemente consistem de dezenas de milhares de máquinas em uma única rede (GREENBERG, HAMILTON, *et al.*, 2009). O significativo investimento financeiro necessário para replicar tais infraestruturas representa um dos principais obstáculos na implementação prática. Com a introdução dos SBCs, abordado na subseção 2.4, conhecidos pelo seu baixo custo e baixo consumo de energia, tornou mais acessível a construção de *data centers* baseados em nuvem com menor infraestrutura.

Como alternativa, surgiu a prática conhecida como *testbed*, a qual refere-se a uma plataforma de experimentação controlada, onde soluções podem ser implementadas e testadas em um ambiente que se assemelha as condições do ambiente real, segundo (CONSORTIUM, 2015). Baseado nisso, alguns pesquisadores têm focado na implementação de simuladores para ambientes de computação em nuvem. O CloudSim (CALHEIROS, RANJAN, *et al.*,

2010), por exemplo, simula toda a pilha de computação em nuvem, incluindo camadas de rede e virtualização de um *data center*. Há também o GreenCloud (KLIAZOVICH, BOUVRY; KHAN, 2012), um simulador à nível de *data centers* baseados em nuvem com foco na avaliação da consciência energética. Por outro lado, há o MDCSim (LIM, SHARMA, *et al.*, 2009), que simula características de *hardware* específicas de diferentes *data centers* tal como servidores e *switches*.

Apesar disso, alguns simuladores *testbed* de nuvem ainda consistem de um número razoável de servidores, como afirma AVETISYAN, CAMPBELL, *et al.* (2010), dificultando sua realização. Como alternativa, há modelos de *data centers* composto por *clusters* de SBC's, um exemplo é o trabalho proposto por TSO, WHITE, *et al.* (2013), intitulado "*The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures"*.

Assim como em ambientes de *data centers* tradicionais, a virtualização e a conteinerização também estão presentes em cenários que utilização *Single Board Computers*. Há uma camada de *software* entre o sistema operacional e as aplicações nos dispositivos de IoT, conhecida como *middleware*. Tanto a virtualização tradicional mostrada na subseção 2.3.1, quanto a tecnologia de *containers* apresentada na subseção 2.3.2 são tipos de *middleware*, como afirma RENNER (2015).

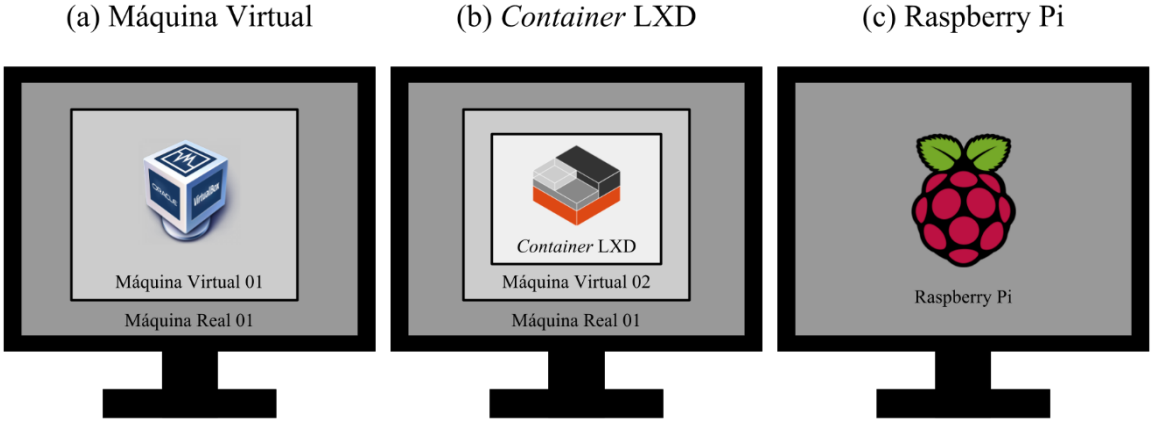
3 ANÁLISE DE TECNOLOGIAS DE VIRTUALIZAÇÃO E HARDWARE DE BAIXO CUSTO PARA INFRAESTRUTURA DE NUVEM DE PEQUENO PORTE

Após a abordagem dos conceitos fundamentais para a compreensão deste trabalho, o atual capítulo irá tratar da análise de desempenho realizada em tecnologias de virtualização – **máquina virtual** e ***container* LXD** – além do **Raspberry Pi** (um *hardware* de baixo custo), incluindo ainda os materiais e métodos utilizados e os experimentos efetuados.

3.1 MATERIAIS E MÉTODOS

Este trabalho apresenta a característica de um estudo quantitativo com métodos empíricos, uma vez que prioriza apontar numericamente o comportamento de três tecnologias, além de compreender e interpretar esses comportamentos. Para isso, foram selecionados três tipos de servidores apresentadas na Figura 4: **máquina virtual**, ***container* LXD** e **Raspberry Pi**.

Figura 4 – Tecnologias utilizadas.



Fonte: Elaborado pelo autor (2017).

A **Máquina Virtual 01** está instalada no Virtual Box (VIRTUAL BOX, 2016), uma ferramenta de virtualização desenvolvida pela Oracle Corporation, em um *host* físico chamado de Máquina Real 01 na Figura 4 (a). O ***container* LXD** está hospedado na Máquina Virtual 02 com as mesmas características da anterior, com exceção de ter o Ubuntu Server 16.04 como sistema operacional, conforme a Figura 4 (b). O Ubuntu Server foi selecionado em razão de atender as características necessárias para a instalação e configuração de *containers* LXD. Já o **Raspberry Pi** não necessita de um *host* hospedeiro, pois trata-se de um *hardware* completo, como mostra a Figura 4 (c). O Quadro 1 contém as especificações de

cada servidor citado, incluindo o tipo de sistema operacional, CPU (núcleos), memória RAM, armazenamento e dados de rede.

Quadro 1 - Especificação técnicas.

	Máquina Real 01	**Máquina Virtual 01**	***Container* LXD**	**Raspberry Pi**
Sistema operacional	Ubuntu Desktop 14.04	Debian Jessie	Debian Jessie	Raspbian
CPU (Núcleos)	4	4	4	4
Memória RAM (GiB)	8	1	0,97	0,90
Armazenamento (GiB)	465,67	14,90	14,90	14,90
Placa de rede	Gigabit Ethernet	Gigabit Ethernet	Gigabit Ethernet	Fast Ethernet
Tipo de disco	Disco rígido	Disco rígido	Disco rígido	Cartão MicroSD

Fonte: Elaborado pelo autor (2017).

Para analisar o desempenho individual desses servidores, foi utilizado a aplicação WebTool (FILHO, 2017) categorizada pela literatura como CRUD, onde realiza funções básicas de armazenamento persistente em banco de dados, são elas: criar (*Create*), ler (*Read*), atualizar (*Update*) e deletar (*Delete*). O Quadro 2 apresenta a descrição das operações contidas na aplicação e a Figura 5 exibe a página Web da aplicação.

Quadro 2 - Operações disponíveis na aplicação.

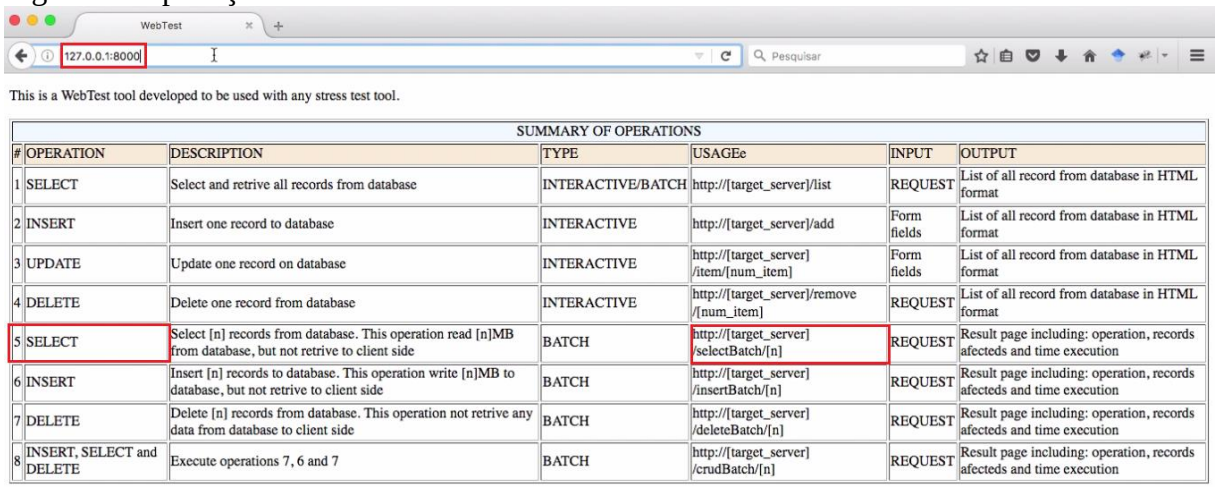
Função	**Tipo**	**Descrição**
INSERT	INTERACTIVE/ BATCH	Recebe a requisição em uma página contendo o número de inserções, processa as inserções no banco de dados e devolve a resposta em uma página HTML. Em inserções do tipo INTERACTIVE adiciona apenas um registro ao banco de dados. Em inserções do tipo BATCH é possível adicionar mais de um registro.
SELECT	INTERACTIVE/ BATCH	Recebe a requisição de uma página contendo o número de registros a serem retornados, processa a consulta e devolve a resposta em uma página HTML. O tipo INTERACTIVE apenas lista os registros presentes no banco de dados. O tipo BATCH é possível selecionar mais de um registro.

DELETE	INTERACTIVE/ BATCH	Recebe a requisição em uma página contendo o número de remoções, processa no banco de dados e devolve a resposta em uma página HTML. O tipo INTERACTIVE remove apenas um registro do banco de dados. No tipo BATCH é possível remover mais de um registro.
UPDATE	INTERACTIVE	Recebe a requisição de uma página contendo o número de registros a serem atualizados, processa as atualizações no banco de dados e devolve a resposta em uma página HTML. Possui apenas o tipo INTERACTIVE, o qual atualiza um registro no banco de dados.
INSERT, SELECT and DELETE	BATCH	Recebe a requisição em uma página contendo o número de registros, processa os testes de INSERT, SELECT e DELETE e devolve a resposta em uma página HTML. Por ser do tipo BATCH, é possível receber mais de um registro.

Fonte: Elaborado pelo autor (2017).

Entre as operações disponíveis na WebTool, neste trabalho será utilizada apenas as operações de INSERT, SELECT e DELETE, todas do tipo BATCH. Essas operações foram selecionadas em razão da natureza do trabalho, assim é possível inserir, selecionar e deletar de acordo com a capacidade de cada servidor. Para o funcionamento da aplicação WebTool foi instalado em cada cenário o Python (PYTHON SOFTWARE FOUNDATION, 2017) versão 2.7.9, o *framework* Django (DJANGO SOFTWARE FOUNDATION, 2017) versão 1.10.5 e o banco de dados SQLite (SQLITE CONSORTIUM, 2017) versão 3.8.7.1.

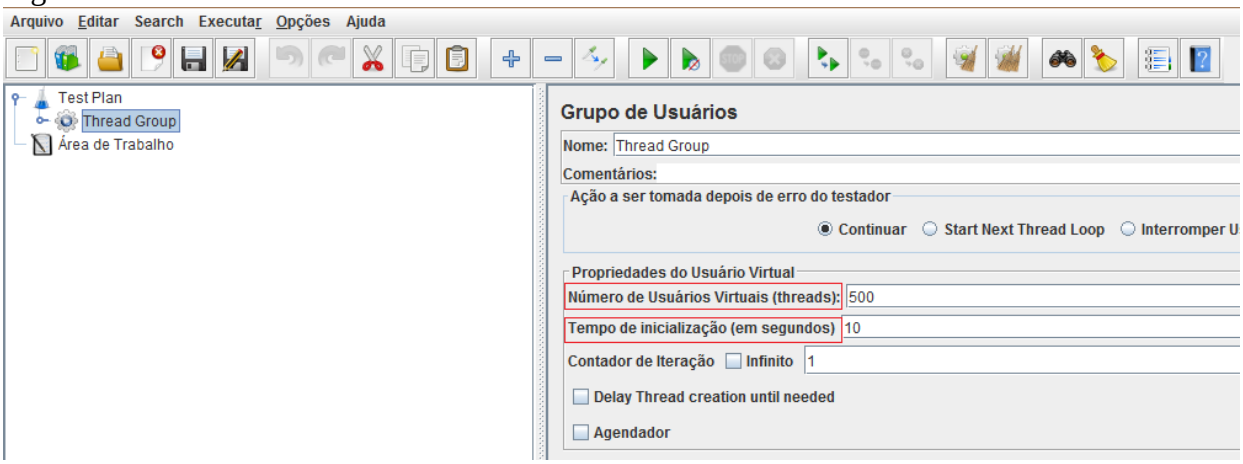
Figura 5 - Aplicação WebTool.



Fonte: Elaborado pelo autor (2017).

Para testar e monitorar o comportamento dos servidores foram utilizadas as ferramentas Jmeter (APACHE SOFTWARE FOUNDATION, 2016) e Collectl (COLLECTL, 2016). O Collectl é um pacote presente no Debian/Raspbian que permite o monitoramento de subsistemas incluindo CPU, disco, memória, rede, processos, entre outros. Neste trabalho, foram monitorados o comportamento de CPU, de memória RAM e utilização de rede. O Jmeter permite elaborar planos de teste de modo a simular acessos à serviços hospedados em um servidor ou grupo de servidores. Esses recursos permitem, como foi feito nos experimentos, avaliar o comportamento geral do serviço e sua infraestrutura sob diferentes situações de carga. Afim de analisar o desempenho dos servidores ilustrados na Figura 4, foi elaborado e executado o plano de teste do Jmeter conforme capturas de tela mostrada na Figura 6.

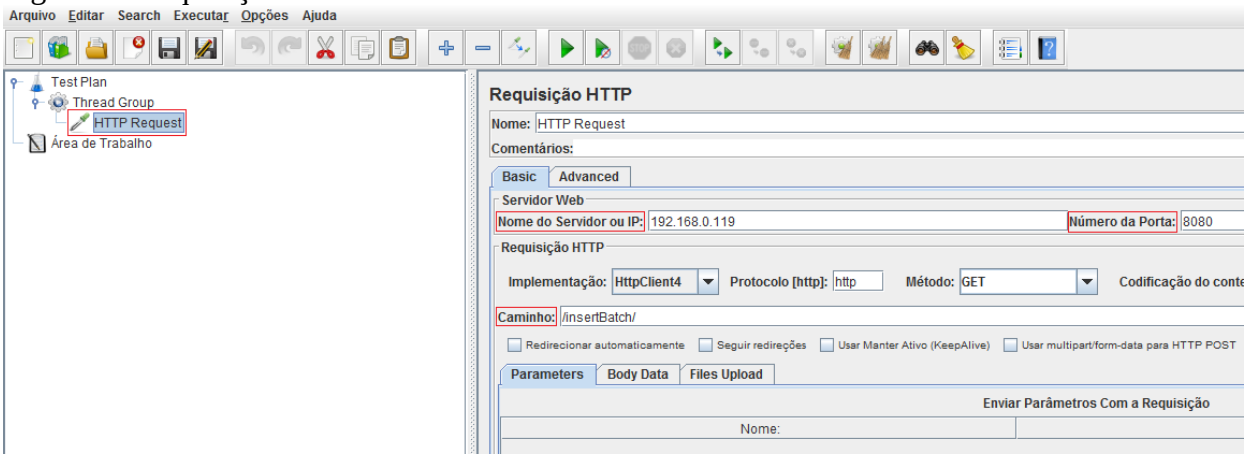
Figura 6 - Plano de teste no Jmeter.



Fonte: Elaborado pelo autor (2017).

Além disso, foi possível adicionar um testador de requisições HTTP informando o IP do servidor onde está em funcionamento a WebTool, a porta onde o servidor está executando e o caminho que será acessado pelos usuários virtuais, essas informações estão visíveis na Figura 7.

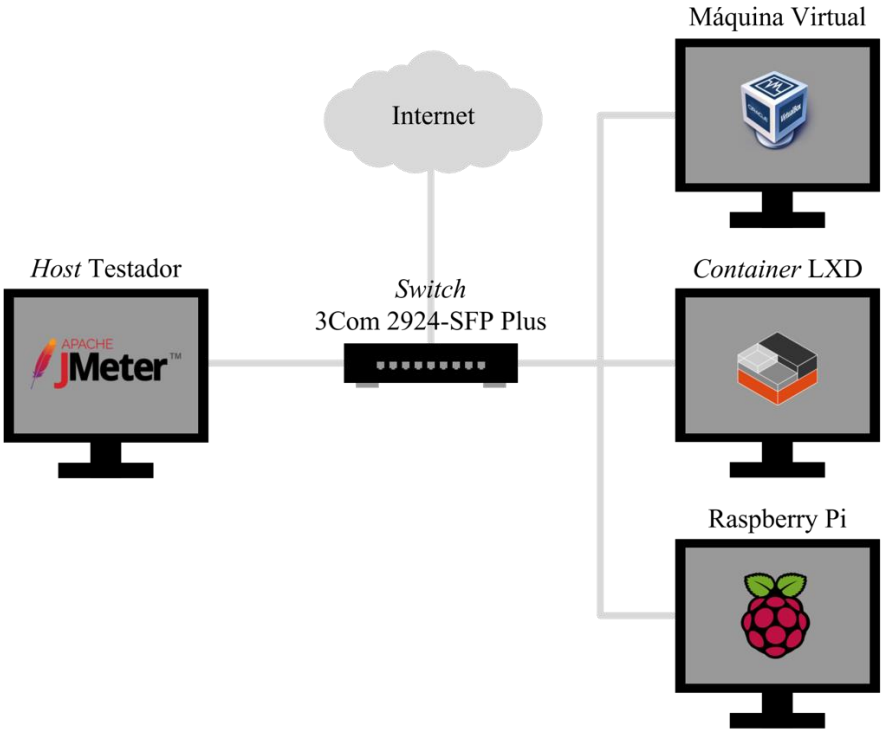
Figura 7 - Requisição HTTP no Jmeter.



Fonte: Elaborado pelo autor (2017).

Os experimentos para medir o desempenho dos servidores foram realizados no cenário de rede demostrado na Figura 8, que abrange um *switch* 3Com 2924-SFP Plus com acesso à Internet, uma estrutura interna contendo um ***Host* Testador** (mesmas especificações da Máquina Real 01 na **Erro! Fonte de referência não encontrada.**) com o Jmeter versão 3.1, a **áquina Virtual**, o ***Container* LXD** e o **Raspberry Pi**.

Figura 8 - Cenário de rede.



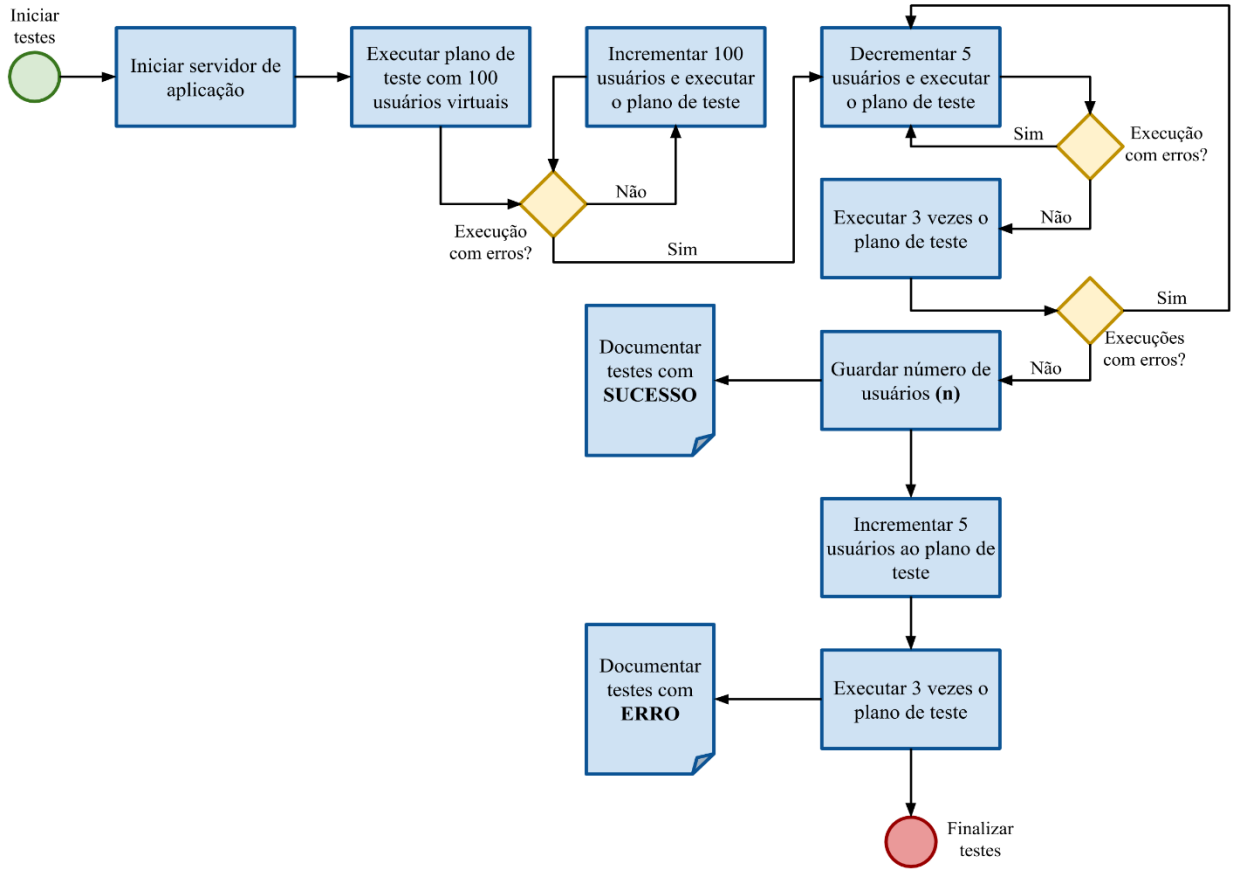
Fonte: Elaborado pelo autor (2017).

Neste cenário foram realizados três experimentos os quais serão descritos a seguir na subseção 3.2. Em cada um deles foram efetuados testes das operações INSERT, SELECT e DELETE. Para realizar esses testes, foi utilizado no campo “Caminho” na requisição HTTP

do Jmeter os parâmetros */insertBatch*, */selectBatch* ou */deleteBatch* (formato de requisição estabelecido pela aplicação WebTool), conforme a Figura 7 vista acima. Ainda, foi adicionado o endereço de IP dos servidores no campo "Nome do Servidor ou IP".

Para encontrar os limiares de usuários simultâneos com sucesso e o ponto de falha suportado pelos servidores foi adotado para cada operação um algoritmo que está na Figura 9, a seguir. Tal algoritmo é executado com a finalidade de estabelecer um padrão de teste e reduzir a probabilidade de haver diferenças entre eles.

Figura 9 - Algoritmo aplicado nas operações.



Fonte: Elaborado pelo autor (2017).

No algoritmo há duas etapas: testes com sucesso e com erro. Durante os testes de sucesso é inicialmente executado o plano do Jmeter com 100 usuários virtuais (número descoberto empiricamente) e incrementado outros 100 até encontrar o primeiro erro. Quando encontrado, são decrementados 5 usuários até descobrir o limite de usuários simultâneos sem falhas. A fim assegurar que esse limite é verdadeiro ainda são executados mais três testes com a mesma quantidade (*n*). Caso haja erros em alguma das três execuções, o algoritmo volta para decrementar mais 5 usuários e realizar novamente todas as atividades seguintes. Se não houver erros, o número (*n*) de usuários é guardado para realizar a segunda etapa de testes. Em

seguida, são incrementados 5 usuários ao plano de teste e então executado três vezes para garantir o ponto onde o sistema começa a ter falhas nas respostas das requisições e então os testes são finalizados.

3.2 EXPERIMENTOS

Conforme visto na subseção 3.1, neste trabalho foi avaliado o desempenho da **Máquina Virtual**, do ***Container* LXD** e do **Raspberry Pi** através dos resultados de três experimentos. Para isso, foi utilizado as ferramentas Colectl e Jmeter conforme detalhado a seguir:

a) O Collectl e o servidor de desenvolvimento da aplicação executando na **Máquina Virtual**. Além da interface de linha de comando do Jmeter executando no ***Host* Testador**;
b) O Collectl e o servidor de desenvolvimento da aplicação executando no ***Container* LXD**. Além da interface de linha de comando do Jmeter executando no ***Host* Testador**;
c) Collectl e o servidor de desenvolvimento da aplicação executando no **Raspberry Pi**. Além da interface de linha de comando do Jmeter executando no ***Host* Testador**.

Em todos os experimentos foi aplicado o algoritmo mostrado anteriormente na subseção 3.1. Portanto, com base nos resultados computados pelo Jmeter, é possível perceber as diferenças entre cada cenário, especialmente em relação ao limite de usuários virtuais simultâneos, média de duração dos testes e média de requisição por segundo (*throughput*) de cada operação. A Tabela 1, apresentará os resultados dos testes de sucesso aplicados aos experimentos.

Tabela 1 - Resultados do Jmeter dos experimentos em testes com sucesso.

Experimento	**Operação**	**Usuários Virtuais**	**Duração dos testes** (média em segundos)	**Throughput** (requisição por segundo)
a Máquina Virtual	INSERT	575	12	53
	SELECT	920	22	44
	DELETE	515	12	50
b *Container* LXD	INSERT	465	12	44
	SELECT	1045	22	49
	DELETE	520	12	49
c Raspberry Pi	INSERT	90	12	9
	SELECT	980	22	46
	DELETE	65	12	7

Fonte: Elaborado pelo autor (2017).

Além disso, foi possível coletar o instante de falha em cada experimento. A Tabela 2, contém a quantidade de usuários dos testes com erros, suas respectivas durações (média), além da porcentagem média de erros.

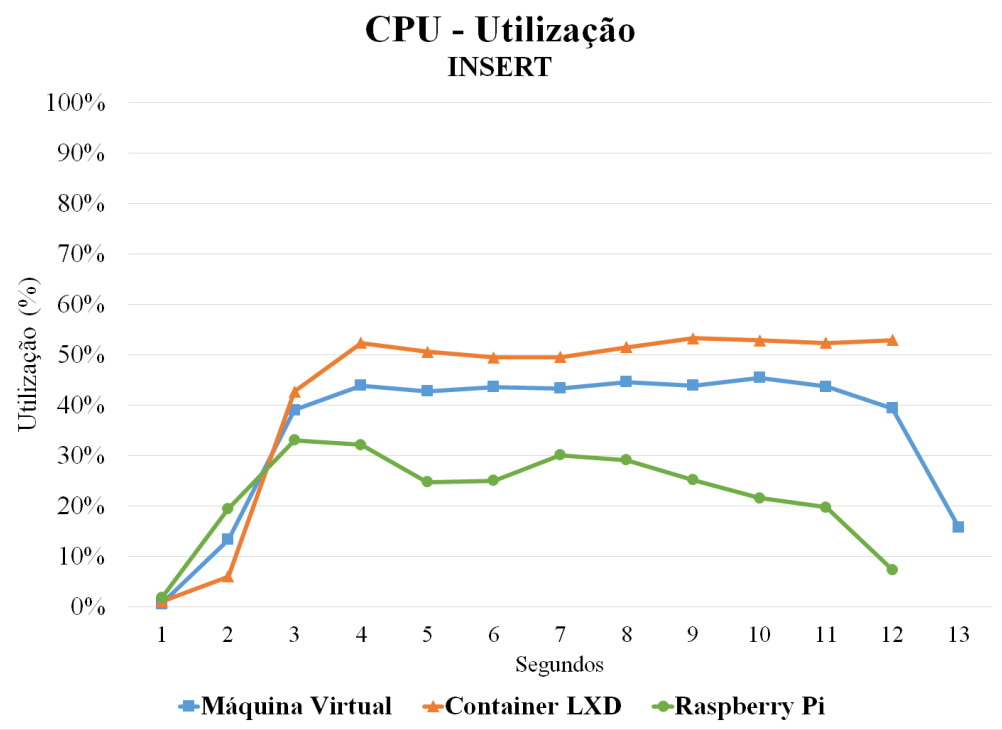
Tabela 2 - Resultados do Jmeter dos exeprimentos em testes com erro.

Experimento	**Operação**	**Usuários Virtuais**	**Duração dos testes** (média em segundos)	**Taxa de erros** (média em %)
a Máquina Virtual	INSERT	580	12	0,06
	SELECT	925	22	0,11
	DELETE	520	12	6,22
b *Container* LXD	INSERT	470	12	0,21
	SELECT	1050	27	0,06
	DELETE	525	12	0,06
c Raspberry Pi	INSERT	95	13	2,11
	SELECT	985	27	0,10
	DELETE	70	13	9,52

Fonte: Elaborado pelo autor (2017).

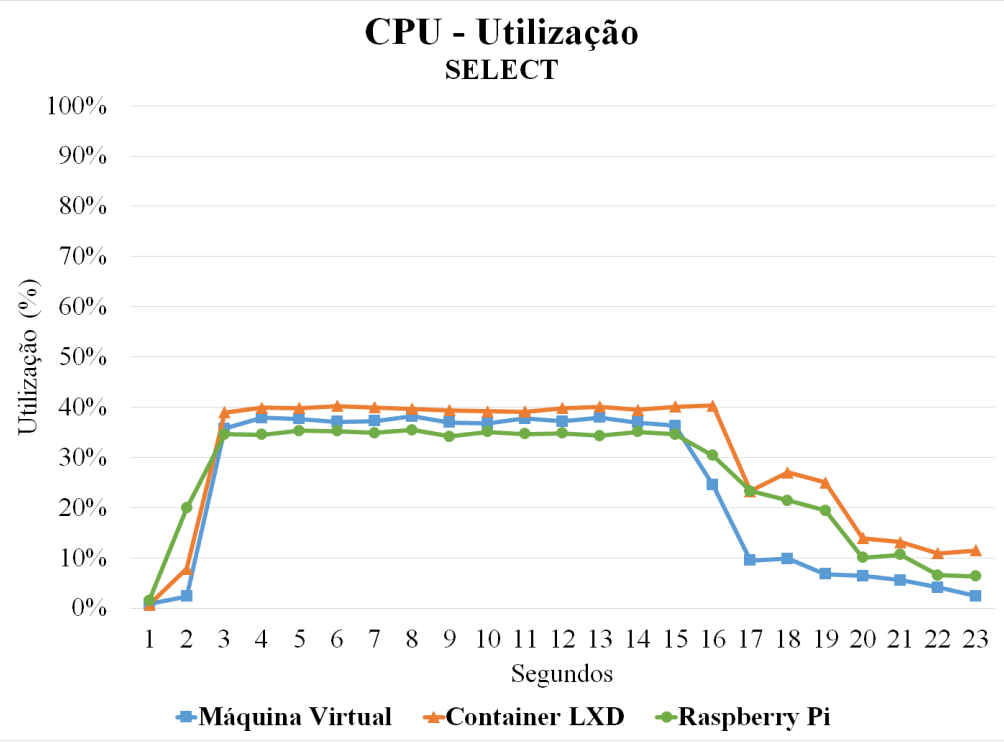
Fundamentado na duração de cada experimento da Tabela 1, foi coletado o comportamento de CPU, memória RAM e rede do seu respectivo servidor por meio do Collectl. Na Figura 10, Figura 11 e Figura 12 a seguir, mostram graficamente os resultados da utilização de CPU por segundo na **Máquina Virtual**, no ***Container* LXD** e no **Raspberry Pi** durante as operações de INSERT, SELECT e DELETE.

Figura 10 - Utilização de CPU na operação INSERT.



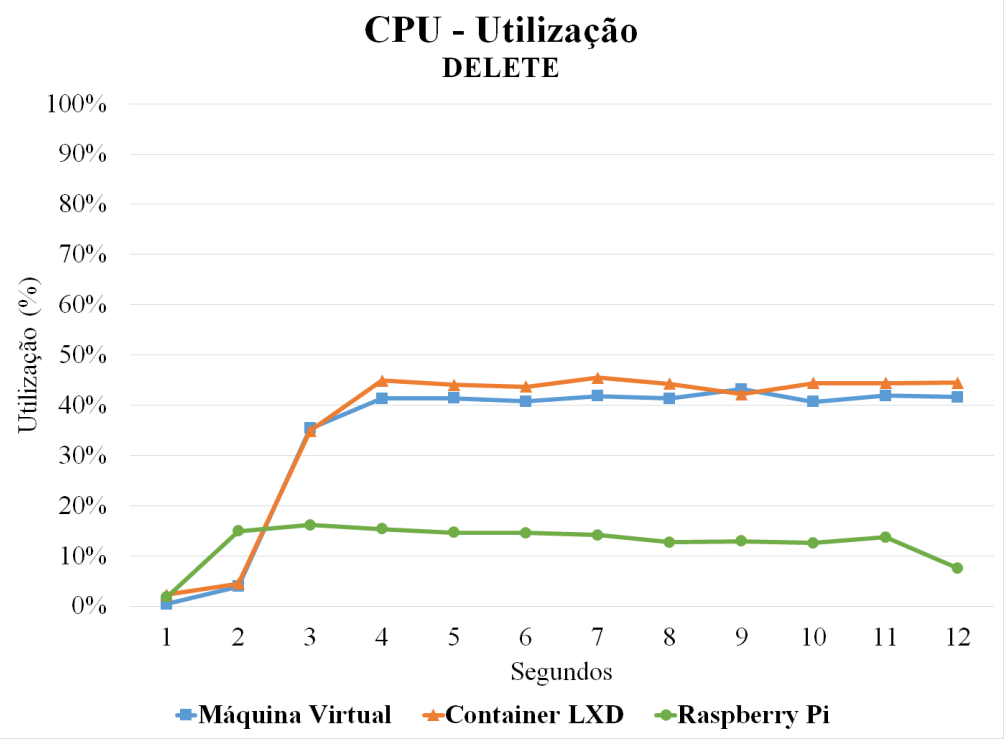
Fonte: Elaborado pelo autor (2017).

Figura 11 - Utilização de CPU na operação SELECT.



Fonte: Elaborado pelo autor (2017).

Figura 12 - Utilização de CPU na operação DELETE.



Fonte: Elaborado pelo autor (2017).

É possível observar, que a **Máquina Virtual** e o ***Container* LXD** possuem resultados semelhantes em todas operações. Em contrapartida, o **Raspberry Pi** apresenta resultados inferiores, com exceção da operação SELECT. Diferentemente da utilização da CPU, a memória RAM dos servidores não teve largas alterações durante os testes realizados pelo Jmeter. Por esse motivo, os resultados extraídos do Collectl serão apresentados na Tabela 3.

Tabela 3 - Resultados da memória RAM dos servidores.

Servidor	**Total** (MiB)	**INSERT** (Média em MiB/s)	**SELECT** (Média em MiB/s)	**DELETE** (Média em MiB/s)
Máquina Virtual	1.000	521,53	406,12	514,94
Container LXD	991	324,06	171,40	294,82
Raspberry Pi	925,52	224,40	214,78	232,71

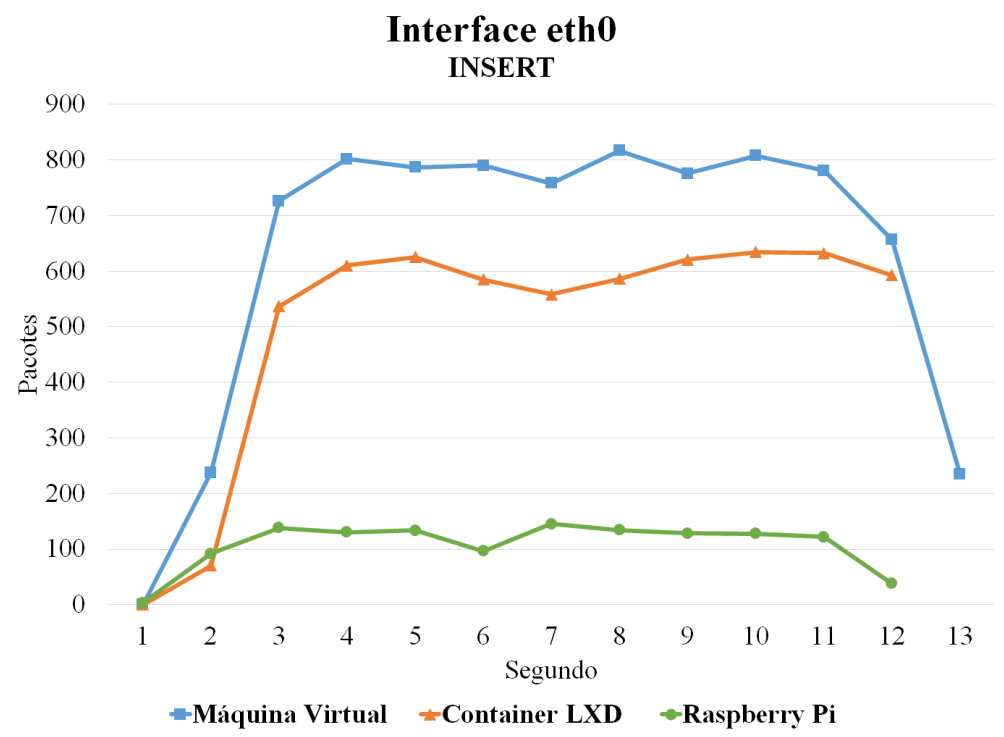
Fonte: Elaborado pelo autor (2017).

De acordo com esses resultados percebe-se que, em cada servidor, foi usado da memória aproximadamente a mesma quantidade de MiB/s (média) durante as operações de INSERT e DELETE.

Além disso, também foi coletado o desempenho da interface de rede eth0[1] dos servidores. Na Figura 13, Figura 14 e Figura 15 exibem graficamente a quantidade de pacotes de entrada e saída durante as três operações realizadas neste trabalho.

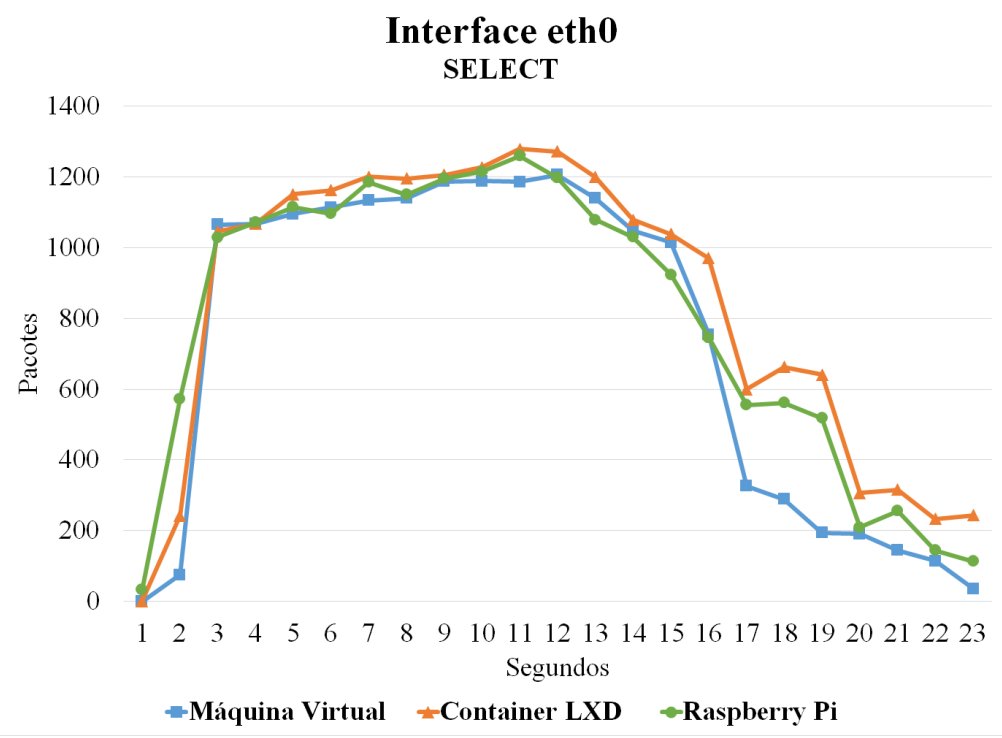
[1] Primeira interface de rede dos servidores.

Figura 13 - Interface eth0 da operação INSERT.



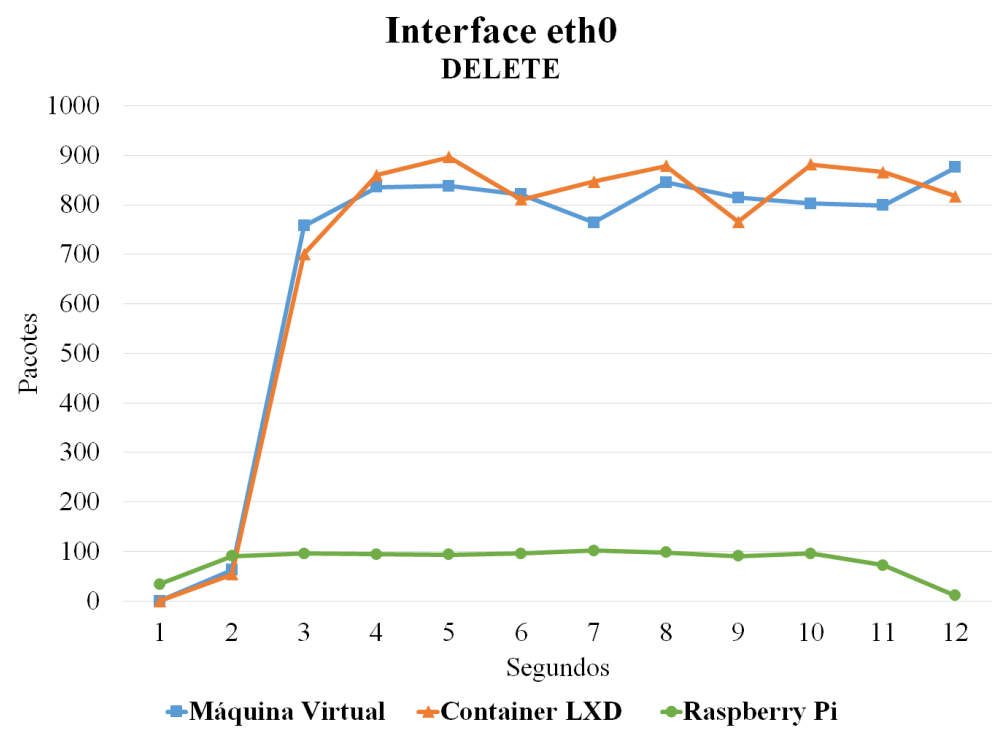
Fonte: Elaborado pelo autor (2017).

Figura 14 - Interface eth0 da operação SELECT.



Fonte: Elaborado pelo autor (2017).

Figura 15 - Interface eth0 da operação DELETE.



Fonte: Elaborado pelo autor (2017).

Nota-se que apenas durante a operação de SELECT há um volume semelhante de pacotes (tanto de entrada quanto de saída) que passam pela interface eth0 em todos os servidores.

4 **RESULTADOS E DISCUSSÕES**

Embasado nos resultados dos experimentos realizados, os quais foram descritos na subseção 3.2, neste capítulo será realizado uma discussão geral dos resultados apresentando conclusões e generalizações a respeito da atual pesquisa. Primeiramente, a Máquina Virtual, o *Container* LXD e o Raspberry Pi foram selecionados para realizar esta pesquisa por se tratarem de tecnologias que estão sendo altamente utilizadas em diferentes infraestruturas, como visto no capítulo 2.

Percebe-se que o experimento do Raspberry Pi demonstrou resultados inferiores comparado aos experimentos da Máquina Virtual e do *Container* LXD, como o limite de usuários simultâneos com sucesso que resultou em menor utilização de CPU, memória e rede. É importante destacar que não foi apresentado neste trabalho o monitoramento de disco com o Collectl, pois a ferramenta não coletou os dados do *Container* LXD e do Raspberry Pi. Uma suposta razão para isso é que o Raspberry Pi possui memória *flash* e o *Container* LXD, apesar de estar sobre uma máquina virtual com memória magnética, seu disco é apenas um formato de arquivo. Contudo, há a possibilidade de ter sido por outras razões que estão inclusas na documentação do Collectl. Com a finalidade de esclarecer os diferentes resultados, o Quadro 3 exibe uma comparação individual do Raspberry Pi com cada um dos servidores a respeito do limite de usuários simultâneos.

Quadro 3 - Comparação do percentual de usuários simultâneos.

Comparativo entre	**Operação**	**Percentual**
Raspberry Pi e Máquina Virtual	INSERT	O Raspberry Pi suporta aproximadamente 16% do total de usuários da Máquina Virtual nesta operação.
	SELECT	O Raspberry Pi suporta aproximadamente 106% do total de usuários da Máquina Virtual nesta operação.
	DELETE	O Raspberry Pi suporta aproximadamente 13% do total de usuários da Máquina Virtual nesta operação.
Raspberry Pi e *Container* LXD	INSERT	O Raspberry Pi suporta aproximadamente 19% do total de usuários do Container LXD nesta operação.
	SELECT	O Raspberry Pi suporta aproximadamente 94% do total de usuários do Container LXD nesta operação.
	DELETE	O Raspberry Pi suporta aproximadamente 12% do total de usuários do Container LXD nesta operação.

Fonte: Elaborado pelo autor (2017).

Existem diferentes motivos para esse comportamento, uma possibilidade é que o Raspberry Pi possui uma placa de rede Fast Ethernet ao contrário dos outros dois servidores que contêm placa Gigabit Ethernet. Ou seja, teoricamente o Raspberry Pi representa 10% da

capacidade de transmissão da Máquina Virtual e do *Container* LXD. Então, conclui-se que em uma infraestrutura de nuvem a qual utiliza-se de virtualização, por exemplo, com um *cluster* de 10 dispositivos Raspberry Pi é possível ter uma taxa de transmissão semelhante à dos outros dois servidores. Apesar disso, uma das vantagens de se utilizar 10 desses dispositivos para abrigar uma aplicação CRUD em um *data center* ao invés de uma Máquina Virtual ou um *Container* LXD é que, em casos de falhas ou danos no *hardware* do servidor, todas as operações estarão distribuídas entre esses dispositivos e não afetará com tal intensidade nas respostas aos usuários.

Para que uma operação interrompa as respostas das requisições é natural que hajam razões específicas. Neste trabalho não foi diferente, o motivo pelo qual os servidores suportaram até um determinado número de usuários simultâneos nas operações de INSERT e DELETE ocorreu devido a limitação do banco de dados. A aplicação WebTool utiliza o SQLite como SGBD[2], o qual não suporta um alto nível de simultaneidade por ser propositalmente feito para leves aplicações. Posteriormente, a razão que limitou as operações de SELECT nos servidores estava no *Host* Testador. Durante essa operação, o Jmeter apresentou um erro chamado de *Connect Exception* (ORACLE AND/OR ITS AFFILIATES, 2016), que acontece quando há vários *threads*[3] ocupados servindo as requisições existentes e as novas requisições são mantidas em espera, então, se o tempo de resposta estipulado atinge o limite estabelecido, as novas requisições não serão atendidas. Portanto, o motivo desta falha está na limitação do *Host* Testador.

Portanto, conclui-se que para operações do tipo INSERT é mais viável usar uma máquina virtual do que um *container* LXD, pois apesar de consumir mais memória a Máquina Virtual usada neste trabalho permitiu uma quantidade maior de usuários simultâneos e utilizou menos CPU do que o *Container* LXD. Já na operação SELECT, o *Container* LXD e o Raspberry Pi apresentaram melhores resultados. O *Container* LXD usou menos de sua memória RAM. Em contrapartida, mesmo com menor taxa de transmissão, o Raspberry Pi suportou 94% do total de usuários do *Container* LXD e menor utilização de CPU. Tais conclusões se baseiam no intervalo de tempo em que o *Host* Testador suportou as requisições, mas provavelmente em cenários onde a limitação não está no testador os resultados podem ser diferentes. Finalmente, em operações de DELETE a Máquina Virtual e o *Container* LXD mostrou melhor performance em todos os subsistemas.

[2] Sistema Gerenciador de Banco de Dados
[3] Sequência de instruções programadas.

De modo geral, com base nos resultados obtidos, foi possível inferir que é viável, em infraestruturas de pequeno porte, implementar serviços de nuvem para aplicações típicas da Internet utilizando Raspberry Pi. A principal vantagem dessa utilização é que esses dispositivos são escaláveis, o que facilita a expansão dessas infraestruturas com um baixo custo.

5 **CONSIDERAÇÕES FINAIS**

Este capítulo irá retratar as considerações finais deste trabalho, tal como as relações entre as tecnologias comparadas e como podem ser aplicadas em infraestruturas de nuvem, bem como apresentar perspectivas de trabalhos futuros como desdobramentos desta obra.

5.1 CONCLUSÕES

No decorrer deste trabalho, procurou-se realizar uma pesquisa teórica, elaborando assim um referencial contendo os conceitos fundamentais a respeito de infraestruturas de nuvem, que tipo de serviços elas oferecem e como tais infraestruturas podem ser implementadas. Além disso, buscou-se elencar diferentes possibilidades de abrigar os serviços oferecidos por essas infraestruturas.

Entre essas possibilidades, neste trabalho foram selecionados máquina virtual, *container* LXD e Raspberry Pi como servidores que abrigam aplicações do tipo CRUD. Para a realização dos experimentos foram construídos ambientes para abrigar a aplicação WebTool, além de isolar essas infraestruturas em uma mesma localização de rede.

Com as ferramentas Colleclt e Jmeter foi possível avaliar o desempenho de subsistemas como CPU, memória e rede dos servidores selecionados. Baseado nos resultados, foram efetuadas comparações do comportamento dessas tecnologias e como elas podem serem utilizadas e aplicadas em infraestruturas de nuvem.

Apesar das tecnologias possuírem sistemas operacionais, memórias e armazenamentos semelhantes e ainda em um mesmo ambiente físico, a natureza delas levaram a resultados diferentes. Os resultados do Raspberry Pi (em sua maioria), por exemplo, foram inferiores comparados a Máquina Virtual e o *Container* LXD utilizados neste trabalho.

Por fim, podemos concluir que os objetivos desta obra foram alcançados, mostrando que, embora o Raspberry Pi possua limitações em *hardware*, é possível aumentar a quantidade de dispositivos para substituir a taxa de transmissão de uma máquina virtual ou um *container* LXD quando o intuito é distribuir um determinado serviço para reduzir a probabilidade de perder todo o sistema em casos de falhas. Além disso, os resultados obtidos de forma geral abriram espaço para trabalhos futuros que possam explorar arranjos mais elaborados apontando a viabilidade das tecnologias selecionadas em ambientes de produção.

5.2 TRABALHOS FUTUROS

Como sugestão, para dá continuidade à pesquisa realizada neste trabalho visualizamos como desafiadoras as seguintes propostas:

a) Realizar comparações entre as tecnologias abordadas neste trabalho adicionando um ou mais *hosts* testadores e alterar banco de dados da aplicação, como MySQL ou PostgreSQL;
b) Construir um arranjo em *cluster* de dispositivos Raspberry Pi com capacidade equivalente a um servidor de rede típico de datacenter e comparar o desempenho em aplicações típicas da web, bem como a eficiência energética entre eles;
c) Implementar um mini datacenter com dispositivos Raspberry Pi para uso distribuído em aplicações de Internet das Coisas (IoT);
d) Implementar um mini datacenter didático utilizando dispositivos Raspberry Pi e Containers LXD;
e) Implementar uma ferramenta de gerenciamento Web de Containers LXD sob dispositivos Raspberry Pi.

Vislumbramos a possibilidade de aplicar ferramentas matemáticas e estatísticas mais robustas como Teoria de Filas e estimativas para melhorar o rigor científico nesses trabalhos futuros.

REFERÊNCIAS

APACHE JMETER. In: **APACHE software foundation**. [S.l.], c2016. Disponível em: <http://jmeter.apache.org/>. Acesso em: 8 fev. 2017.

AVETISYAN, A. I. et al. Open cirrus: A global cloud computing testbed. **IEEE Computer Society**, Gainesville, v.43, n. 4, p. 42-50, abr. 2010.

BANANA PI. In: **BANANA pi**. [S.l.], c2014. Disponível em: <http://www.bananapi.org/>. Acesso em: 08 ago. 2016.

BARI, M. F. et al. Data Center Network Virtualization: A Survey. **IEEE Communications Surveys & Tutorials**, Hsinchu, v.15, n. 2, p. 909-928, set. 2012.

BEAGLEBOARD.ORG. In: **BeagleBoard.org**. Oakland, 2016. Disponível em: <http://beagleboard.org/>. Acesso em: 30 out. 2016.

BROADSOFT JAPAN K.K. **SaaS, PaaS, IaaS?**. [S.l.], c2015. Disponível em: <http://pbxl.co.jp/en/saas-paas-iaas/>. Acesso em: 9 mar. 2017.

CALHEIROS, R. N. et al. **CloudSim: a toolkit for modeling and simulation of cloud**. [S.l.], c2010. Disponível em: < http://www.buyya.com/papers/CloudSim2010.pdf >. Acesso em: 8 mar. 2017.

CHIP. In: **NEXT thing co**. [S.l.], 2016. Disponível em: <https://getchip.com/pages/chip>. Acesso em: 08 ago. 2016.

CISCO | RED HAT. **Linux Containers: Why They're in Your Future and What Has to Happen First**. [S.l.], c2014.

CODESHIP. **Why Containers and Docker are the Future**. [S.l.], 2015.

COLLECTL. [S.l.], 2016. Disponível em: <http://collectl.sourceforge.net/>. Acesso em: 8 fev. 2017.

CONSORTIUM, I. I. **IIC Quarterly Reports**. [S.l.], 2015.

CUSICK, J. J. et al. **Design, Construction, and Use of a Single Board Computer Beowulf Cluster: Application of the Small-Footprint, Low-Cost, InSignal 5420 Octa Board**. [S.l], 2014. Disponível em: <https://arxiv.org/ftp/arxiv/papers/1501/1501.00039.pdf>. Acesso em: 8 mar. 2017.

DJANGO. In: **DJANGO software foundation**. [S.l.], 2017. Disponível em: <https://www.djangoproject.com/>. Acesso em: 8 fev. 2017.

DOCKER. In: **DOCKER inc**. [S.l.], 2017. Disponível em: <https://www.docker.com/>. Acesso em: 8 mar. 2017.

ERICSSON AB. **Hyperscale Cloud**. [S.l.], 2016.

FILHO, F. S. D. L. **WebTest-Tool**. [S.l.], 2017. Disponível em: <https://gitlab.devops.ifrn.edu.br/2504795/WebTest-Tool>. Acesso em: 8 mar. 2017.

GOOGLE APP ENGINE. In: **GOOGLE inc**.[S.l.], 2016. Disponível em: <https://appengine.google.com/>. Acesso em: 1 ago. 2016.

GOOGLE APPS FOR WORK. In: **GOOGLE inc**. [S.l.], 2016. Disponível em: <https://apps.google.com.br/intx/pt-BR/>. Acesso em: 1 ago. 2016.

GOPULARAM, B. P.; B, Y. C.; PERIASAMY, P. Highly Scalable Model for Tests Execution in Cloud Environments. In: 2012 18th Annual International Conference on Advanced Computing and Communications, 2012, Bangalore. p. 14-16.

GREENBERG, A. et al. **The cost of a cloud: research problems in data center networks**. New York, 2009. Disponível em:<http://mvdirona.com/jrh/TalksAndPapers/CostOfClouds_CCR.pdf>. Acesso em: 8 mar 2017.

HEWLETT PACKARD ENTERPRISE. **Server virtualization—How containers are changing the cloud and application landscape**. [S.l.], 2016.

IBM. In: **IBM**. São Paulo, 2016. Disponível em: <http://www.ibm.com/br-pt/>. Acesso em: 30 out. 2016.

INTERNATIONAL BUSINESS MACHINES CORPORATION. **IBM Systems Virtualization: Servers, Storage, and Software**. New York, 2008.

JOYENT. In: **JOYENT, inc**. c2016. Disponível em: <https://www.joyent.com/>. Acesso em: 30 jul 2016.

KLIAZOVICH, D.; BOUVRY, P.; KHAN, S. U. Greencloud: A packet-level simulator of energy-aware cloud computing data centers. **Springer Science+Business Media**, [S.l.], v. 62, n. 3, p. 1263–1283, dez. 2012.

LENCSE, G.; RÉPÁS, S. **Benchmarking Further Single Board Computers for Building a Mini Supercomputer for Simulation of Telecommunication Systems**. [S.l.], 2016. Disponível em: <http://www.hit.bme.hu/~lencse/publications/IJATES2-2016-SBC-published.pdf>. Acesso em: 8 mar. 2017.

LIM, S.-H. et al. MDCSim: A multi-tier data center simulation, platform. In: 2009 IEEE International Conference on Cluster Computing and Workshops, 2009, [S.l.], p. 1-9.

MICROSOFT CORPORATION. **O que é computação em nuvem?** Seattle, 2016. Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-cloud-computing/>. Acesso em: 18 jul. 2016.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **The NIST Definition of Cloud Computing**. Gaithersburg, 2011.

NEWARK, A TRADEMARK OF PREMIER FARNELL CORP. **A Brief History of Single Board Computers**. [S.l.], 2014.

ORACLE. In: **ORACLE**. [S.l.], c2016. Disponível em: <https://www.oracle.com/sun/index.html>. Acesso em: 30 out. 2016.

_____. **The Most Complete and Integrated Virtualization: From Desktop to Datacenter**. Redwood Shores, c2010.

ORACLE AND/OR ITS AFFILIATES. **Class ConnectException**. [S.l.], 2016. Disponível em: <https://docs.oracle.com/javase/7/docs/api/java/net/ConnectException.html>. Acesso em: 9 fev. 2017.

OU, J. **Pi and The Sky : Using Smart Offloading to Improve Performance on Low-cost Computers**. Oslo, 2013. Disponível em: <https://www.duo.uio.no/bitstream/handle/10852/37445/Ou-Jun.pdf?sequence=4&isAllowed=y>. Acesso em: 9 mar. 2017.

PYTHON. In: **PYTHON software foundation**. [S.l.], 2017. Disponível em: <https://www.python.org/>. Acesso em: 9 mar. 2017.

RASPBERRY PI. In: **RASPBERRY pi foundation**. [S.l.: 201-?]. Disponível em: <https://www.raspberrypi.org/>. Acesso em: 08 ago. 2016.

RENNER, M. **A Performance Evaluation of Container Technology as a Model for the Infrastructure of the Internet of Things**. Finland, 2015. Disponível em: <https://arxiv.org/ftp/arxiv/papers/1603/1603.02955.pdf>. Acesso em: 9 mar. 2017.

RIGHTSCALE, INC. **RightScale 2016 State of the Cloud Report**. [S.l.], 2016.

SALESFORCE. In: **SALESFORCE.COM, inc**. San Francisco, c2016. Disponível em: <http://www.salesforce.com/>. Acesso em: 30 jul. 2016.

SCHEEPERS, M. J. **Virtualization and Containerization of Application Infrastructure: A Comparison**. Enschede, 2014. Disponível em: <http://referaat.cs.utwente.nl/conference/21/paper/7449/virtualization-and-containerization-of-application-infrastructure-a-comparison.pdf>. Acesso em: 9 mar. 2017.

SQLITE. In: **SQLITE consortium**. [S.l.], 2017. Disponível em: <https://www.sqlite.org/>. Acesso em: 8 fev. 2017.

SUN MICROSYSTEMS, INC. **Introduction to Cloud Computing Architecture**. Santa Clara, c2009.

TSO, F. P. et al. The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures. In: 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops, 2013, [S.l.], p. 108-112.

VERDI, F. L. et al. Novas Arquiteturas de Data Center para Cloud Computing. **Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**, Gramado, p. 103-148, maio 2010.

VIRTUAL BOX. In: **ORACLE**. [S.l.], 2016. Disponível em: <https://www.virtualbox.org/>. Acesso em: 9 fev. 2016.

VMWARE, INC. **Virtualization: Architectural Considerations And Other Evaluation Criteria**. Palo Alto, 2005.

_____. **Virtualization Overview**. Palo Alto, 2006.

WHAT'S LXC?. [S.l.], 2016. Disponível em: <https://linuxcontainers.org/lxc/>. Acesso em: 5 ago. 2016.

WHAT'S LXD?. [S.l.], 2016. Disponível em: <https://linuxcontainers.org/lxd/>. Acesso em: 19 jan. 2016.