Avaliação de Desempenho de Estratégias de Virtualização

Alternative Title: Performance Evaluation of Virtualization Strategies

Haltielles Silva
Universidade Federal de Itajubá
Itajubá, Minas Gerais
haltiellestil@unifei.edu.br

Bruno T. Kuehne
Universidade Federal de Itajubá
Itajubá, Minas Gerais
brunokuehne@unifei.edu.br

Dionisio M. L. Filho
Universidade Federal do Mato Grosso do Sul
Pioneiros, Mato Grosso do Sul
dionisio.leite@ufms.br

Maycon L. M. Peixoto
Universidade Federal da Bahia
Salvador, Bahia
maycon.leone@ufba.br

Rafael M. D. Frinhani
Universidade Federal de Itajubá
Itajubá, Minas Gerais
frinhani@unifei.edu.br

Bruno G. Batista
Universidade Federal de Itajubá
Itajubá, Minas Gerais
brunoguazzelli@unifei.edu.br

RESUMO

Cloud computing is a computational model in which providers offer on-demand services to clients in a transparent way. In this context, the efficient provisioning of resources is extremely important, since it is directly related to Quality of Service (QoS), with the Service Level Agreement (SLA) compliance and the cost of the service. In this way, virtualization becomes a crucial factor in resource provisioning. In this paper, performance evaluations were performed considering environments configured with different virtualization strategies: by virtual machines and by containers. In the first set of experiments, we evaluated the boot time of the system with the KVM, Xen and Docker tools and with variations in the number of instances and virtual cores. In the second, the same mechanisms were analyzed during the execution of different benchmarks (Apache and Smallpt) and with variations in the number of virtual resources provisioned. In the results, Docker proved to be more efficient when compared to other tools.

KEYWORDS

Avaliação de Desempenho; Estratégias de Virtualização; Computação em Nuvem; Provisionamento de Recursos; Qualidade de Serviço.

1 INTRODUÇÃO

Nos últimos anos um dos tópicos mais discutidos na área de Tecnologia de Informação (TI) tem sido computação em nuvem. Segundo o NIST (*National Institute of Standards and Technology*), "computação em nuvem é um modelo que permite ubiquidade, conveniência e acesso sob demanda para um conjunto de recursos compartilhados que são configuráveis e que podem ser rapidamente entregues com um esforço mínimo de gestão por parte dos usuários [15].

A nuvem é um ambiente altamente escalável, no qual a demanda de serviços pode mudar instantaneamente. Dessa forma, a alocação automática de recursos para atender essa demanda torna-se um tópico interessante tanto no âmbito acadêmico quanto no industrial. O correto provisionamento permite um melhor uso dos recursos computacionais disponíveis e, consequentemente, de toda a infraestrutura que compõe a nuvem, pois o mapeamento entre a carga e os recursos é mais eficiente. Além disso, ele ajuda no cumprimento das exigências feitas pelos clientes, sejam elas de desempenho e/ou segurança, e provê um grande dinamismo ao sistema.

A tarefa de prover mais recursos computacionais a um cliente, por exemplo, é altamente viável e de fácil disponibilização, uma vez que os recursos computacionais são virtualizados e podem ser considerados ilimitados na visão dos clientes. Dessa forma, a virtualização é a tecnologia chave na computação em nuvem. Nesse contexto, ela refere-se principalmente à abstração dos recursos físicos de TI aos clientes e aplicativos que os usam. Além disso, permite que os servidores, dispositivos de armazenamento, hardware e outros recursos sejam tratados em conjunto ao invés de sistemas distintos, de modo que esse conjunto de recursos possa ser alocado por demanda [28]. Sendo assim, a virtualização é adaptada a uma infraestrutura de nuvem dinâmica, pois oferece vantagens importantes no compartilhamento, gerenciamento e isolamento dos recursos [23].

A virtualização é uma técnica que possibilita a emulação de uma ou mais estações de trabalho/servidores em um único computador físico, que assume o papel de vários computadores lógicos (instâncias). Essas instâncias são conhecidas como máquinas virtuais (*Virtual Machines* - VMs) ou contêineres, dependendo da estratégia de virtualização aplicada. Cada instância oferece um ambiente completo similar a uma máquina física. Com isso, cada instância pode ter seu próprio sistema operacional, aplicativos e serviços de rede [3] [1].

A virtualização surgiu como uma forma de isolar uma determinada aplicação através de uma camada intermediária de *software* criada por cima do sistema, que abstrai os recursos da máquina. O método mais comum de virtualização consiste na criação de máquinas virtuais, as quais representam sistemas operacionais inteiros por cima do sistema hospedeiro. Atualmente, existem diversos métodos de virtualização disponíveis, sendo um dos mais tradicionais a virtualização por hipervisor.

A virtualização por hipervisor requer um “supervisor” de máquinas virtuais rodando por cima do Sistema Operacional (SO), provendo uma abstração completa da máquina virtual. Cada máquina virtual tem seu próprio sistema operacional isolado dos demais, o que possibilita que um único hospedeiro tenha diversos SOs diferentes. Além disso, cada máquina virtual inicializa o seu próprio kernel e espaço de usuário, o que oferece um bom isolamento [4].

Uma outra metodologia que vem ganhando destaque nos últimos anos é a virtualização baseada em contêineres, também conhecida

como virtualização a nível de sistema operacional. Ela é uma alternativa menos custosa em diversos fatores quando comparada à virtualização por hipervisor. Essa técnica consiste no compartilhamento dos recursos físicos da máquina através da criação de instâncias de espaços de usuário que compartilham o kernel do sistema hospedeiro. Um contêiner pode conter uma única aplicação ou um sistema inteiro, de forma que, no primeiro, a aplicação é executada isoladamente no contêiner, e, no segundo, uma instância completa é executada, de forma que diversas aplicações podem ser executadas simultaneamente em um único contêiner [4].

No entanto, a utilização de hipervisores e contêineres acarreta em uma sobrecarga sobre o sistema hospedeiro [18]. Tal sobrecarga deve ser considerada, pois impacta sobre a Qualidade do Serviço (QoS - *Quality of Service*) prestado.

Em face ao exposto, o objetivo deste artigo é avaliar o impacto da utilização de máquinas virtuais e contêineres em um ambiente de prestação de serviços como o de computação em nuvem. Para isso, inicialmente será realizada uma análise do tempo de inicialização de cada abordagem de virtualização, considerando os virtualizadores Xen, KVM e Docker. Em seguida, uma outra análise será realizada considerando os mesmos mecanismos durante a prestação de dois serviços distintos: um de requisições Web (Apache Benchmark) e outro de renderização de imagens (Smallpt Benchmark). Em ambas a variação no provisionamento de recursos foi considerada. Dessa forma, a quantidade de máquinas virtuais e de núcleos virtuais (vCPUs) foi modificada de acordo com o planejamento de experimentos, a fim de analisar o desempenho do sistema mediante a saturação do ambiente.

Este artigo está estruturado da seguinte forma: a Seção 2 discorre sobre a importância da virtualização e das estratégias disponíveis; a Seção 3 apresenta os trabalhos relacionados; na Seção 4 é detalhado o planejamento dos experimentos; na Seção 5 são discutidos os resultados; por fim, na Seção 6 são apresentadas as conclusões e premissas para trabalhos futuros.

2 ESTRATÉGIAS DE VIRTUALIZAÇÃO

A virtualização refere-se principalmente à abstração dos recursos físicos de TI aos clientes e aplicativos que os usam[27]. Ela permite que os servidores, dispositivos de armazenamento, hardware e outros recursos sejam tratados em conjunto ao invés de sistemas distintos, de modo que esse conjunto de recursos possa ser alocado por demanda [28]. Sendo assim, a virtualização é adaptada a uma infraestrutura de nuvem dinâmica, pois oferece vantagens importantes no compartilhamento, gerenciamento e isolamento dos recursos [23].

Entre os principais objetivos e benefícios obtidos com o uso da virtualização estão o uso eficiente dos recursos; redução dos custos com gerenciamento de recursos; maior flexibilidade e portabilidade; isolamento; e eliminação de problemas com compatibilidade [16].

Um ambiente virtualizado consiste de três partes básicas. A primeira consiste do sistema real, nativo ou hospedeiro (*host system*), que contém os recursos reais de *hardware* e *software* do sistema. A segunda, o sistema virtual, também denominado sistema convidado (*guest system*), que opera sobre o sistema real. Em alguns casos, vários sistemas virtuais podem coexistir, executando simultaneamente sobre o mesmo sistema real. Por fim, a camada de virtualização, conhecida como hipervisor ou monitor VMM (*Virtual Machine Monitor*), que constrói as interfaces virtuais a partir da interface real.

Diferentes abordagens consideradas na construção de hipervisores implicam na definição de estratégias para a virtualização, sendo que as mais utilizadas são a Virtualização Total (*Full Virtualization*), a Paravirtualização (*Paravirtualization*), e mais recentemente, o Contêiner. A utilização de uma técnica ou outra normalmente fica a cargo do domínio das instâncias e do sistema que será implementado [12] [22]. A Figura 1 apresenta uma comparação entre as estratégias de virtualização por máquinas virtuais e por contêiner.

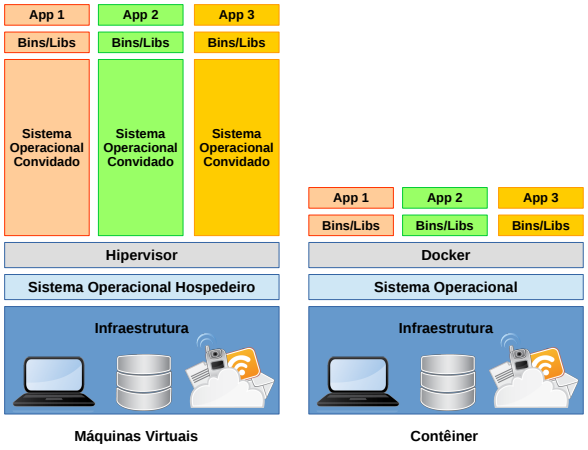


Figura 1: Estratégias de virtualização, adaptado de [26].

A virtualização total provê uma réplica do *hardware* subjacente, de forma que o sistema operacional e as aplicações possam ser executadas como se fossem executadas diretamente sobre o *hardware* [3]. Dessa forma, o sistema operacional hóspede é instalado, sem modificações, sobre o hipervisor, o que é a grande vantagem dessa estratégia. Essa é a abordagem utilizada na maioria dos hipervisores de sistemas clássicos, como QEMU, VMWare e KVM (*Kernel-based Virtual Machine*) [32].

A desvantagem da virtualização total é que o sistema convidado executa mais lentamente, uma vez que todos os acessos ao *hardware* são intermediados pelo hipervisor [12]. Dessa forma, o hipervisor terá que interceptar e emular todas as instruções sensíveis (instrução que altera o estado do sistema) executadas pelos sistemas convidados, o que gera um custo elevado em plataformas de *hardware* sem suporte adequado à virtualização.

Além disso, alguns problemas técnicos gerados pela forma que os sistemas operacionais são implementados devem ser contornados, já que esses foram implementados para serem executados como uma instância única em uma máquina física, não disputando recursos com outros sistemas operacionais. Por exemplo, um sistema operacional convencional implementa memória virtual por meio de paginação. Todo o procedimento de gerência de alocação, liberação e controle de acesso às páginas deve ser respeitado. Para isso, o espaço de endereçamento do sistema hóspede deve ser convertido para um real gerando uma disputa de recursos, o que acarreta em queda de desempenho [14].

Por outro lado, na paravirtualização o sistema operacional é modificado, de forma que a chamada de uma instrução sensível é substituída pela chamada a um tratador de interrupção de *software* (*Virtual Monitor Calls*), o que permite um melhor acoplamento entre os sistemas convidados e o hipervisor [3]. Com isso o hipervisor não precisa testar instrução por instrução, o que representa um ganho significativo de desempenho das máquinas virtuais. Outro ponto positivo da paravirtualização é que os dispositivos de *hardware* são acessados por *drivers* da própria máquina virtual, não sendo necessário o uso de *drivers* genéricos que inibem o uso da capacidade total do dispositivo [12]. Os primeiros hipervisores a adotar a paravirtualização foram o Denali [29] e o Xen [2].

Embora exija que o sistema convidado seja adaptado ao hipervisor, o que é a sua grande desvantagem pois diminui a sua portabilidade, a paravirtualização permite que o sistema convidado acesse alguns recursos do *hardware* diretamente, sem a intermediação ativa do hipervisor. Nesses casos, o acesso ao *hardware* é apenas monitorado pelo hipervisor, que informa ao sistema convidado seus limites, como as áreas de memória e de disco disponíveis. Para o acesso aos demais dispositivos como *mouse* e teclado, o hipervisor apenas gerencia a ordem de acesso no caso de múltiplos sistemas convidados em execução simultânea.

Por fim, uma estratégia de virtualização conhecida como contêiner tem se destacado. Ela se diferencia da virtualização total e da paravirtualização pelo fato de, ao invés de carregar um sistema operacional completo, ela agrega apenas as bibliotecas e configurações necessárias para a aplicação funcionar (bins/libs), o que torna os sistemas eficientes, leves e auto-suficientes [24]. Em outras palavras, o contêiner é um pacote leve, autônomo e executável de um software que inclui tudo o que é necessário para executá-lo (código, tempo de execução, ferramentas do sistema, bibliotecas do sistema, configurações, etc), eliminando softwares irrelevantes para o sistema. Disponível para aplicações Windows e Linux, o *software* em um contêiner sempre será o mesmo independente do ambiente. Atualmente, o Docker é a ferramenta de contêiner mais utilizada no âmbito industrial e acadêmico [6].

Os contêineres e máquinas virtuais possuem sistemas similares de isolamento e alocação, mas seu funcionamento se difere pelo fato de os contêineres virtualizarem o sistema operacional e não o hardware. A próxima seção apresenta alguns trabalhos disponíveis na literatura.

3 TRABALHOS RELACIONADOS

Há na literatura diversos trabalhos sobre avaliação de desempenho de virtualizadores, tais como os estudos apresentados em [31], [19], [13], [9] e [20].

Rajasekaran *et al.* [21] apresentam um estudo sobre a degradação em uma rede que contém várias máquinas virtuais utilizando o virtualizador VMware. *Web services* foram utilizados como aplicação nas máquinas virtuais e foi observado que a sobrecarga gerada nas máquinas virtuais é maior do que a apresentada pelas máquinas físicas. Com cargas pesadas, as taxas de perda de pacotes TCP (*Transmission Control Protocol*) e o tempo médio de resposta de transações HTTP (*HyperText Transfer Protocol*) aumentaram com o número de VMs. O *host* sem VMs e sujeito à mesma carga de usuários sofreu quase nenhuma perda e alcançou uma maior utilização da rede. Os autores mostraram que o fluxo destinado a uma máquina virtual é comprimido no tempo, devido ao custo de virtualização e, ainda, que a compressão pode ser extrema devido ao aumento do número de máquinas virtuais ativas.

Em Menon *et al.* [17] é apresentado um conjunto de ferramentas para o virtualizador Xen denominado Xenoprof. O Xenoprof permite coordenar o perfil de várias máquinas virtuais em um sistema para obter a distribuição de eventos de *hardware*, tais como ciclos de *clock*, *cache* e erros de TLB (*Translation Lookaside Buffer*). Os autores avaliaram as aplicações que fazem uso intensivo de rede, por considerar que essa é uma atividade muito onerosa ao sistema. Eles ainda enfatizam a observação da rede uma vez que a implementação de entrada e saída (I/O) do Xen pode prejudicar o desempenho de aplicações que fazem uso intensivo de recursos de I/O. O uso de recursos de *hardware* de rede como a segmentação do TCP e o *checksum* podem se tornar gargalos para o sistema.

Sotomayor *et al.* [25] utilizam a virtualização para contornar as limitações na manipulação de recursos físicos em uma grade computacional. Os autores argumentam que, apesar da sobrecarga gerada, a virtualização é o melhor caminho para a configuração e disponibilização de recursos na grade. O desempenho é melhorado com o uso de *cache* das imagens das máquinas virtuais e o reuso das mesmas, criando assim *templates* de máquinas virtuais. Foi mostrado para um conjunto específico de *jobs*, que a utilização de *cache* melhorou o desempenho em operações de entrada e saída das máquinas virtuais.

O estudo apresentado por Koh *et al.* [11] aponta as desvantagens da virtualização em relação ao isolamento de desempenho, ou seja, a interferência que uma máquina virtual gera em outra máquina virtual. Segundo os autores, a caracterização da carga de trabalho é um fator importante, pois, o desempenho é afetado de forma diferente em configurações diferentes, causando variações significativas na taxa de transferência do sistema observado nas máquinas virtuais. A caracterização da carga de trabalho que gera interferência no desempenho é importante, podendo maximizar a utilização do ambiente. Nesse estudo o virtualizador Xen foi modificado para coletar os índices de desempenho das máquinas virtuais. Por meio da análise dos dados coletados, foi possível identificar grupos de aplicações que geram interferências no comportamento geral do sistema. Além disso, os autores utilizaram os resultados para desenvolver modelos matemáticos para prever o desempenho de outras aplicações baseando-se em características das cargas de trabalho.

Na visão de White e Pilbeam [30], é importante observar as limitações e exigências do *host* físico que será utilizado para a virtualização. Os autores apresentam uma revisão dos métodos de virtualização e dos *softwares* de virtualização, e discutem uma análise dos problemas de desempenho inerentes a cada uma dessas abordagens. Os autores mencionam o fato de haver divergências de resultados em outros trabalhos sobre virtualização devido a não maturidade das técnicas de virtualização utilizadas. Além disso, são discutidas as formas de virtualização de recursos como: emulação, virtualização nativa, paravirtualização, virtualização em nível de sistema operacional, virtualização de recursos (como rede e vídeo) e virtualização de aplicações. Segundo os autores, a emulação é a técnica que possui o pior desempenho, enquanto que as técnicas

que garantem o melhor desempenho são a paravirtualização (Xen) e a virtualização no nível do sistema operacional (KVM).

No trabalho apresentado por [5] foi desenvolvido a realocação de recursos no Xen, visando garantir o SLA (*Service Level Agreement*), por meio do algoritmo *Credit Scheduler*. Os autores executaram um *benchmarck* sobre a aplicação hospedada no ambiente virtualizado, coletando os resultados dos testes e utilizando-os para mapear as métricas de baixo nível (processador e memória), sob diferentes cargas de trabalho, provendo dados de configuração e de desempenho desse ambiente. Estes dados foram utilizados em conjunto com os SLAs das VMs para que, dada uma configuração vigente de uma máquina Xen e as políticas do SLA, o subsistema fosse capaz de identificar qual a melhor maneira de reconfigurar os seus recursos computacionais providos, buscando uma melhora na utilização de recursos das VMs.

Com a grande necessidade de virtualização em servidores, a otimização do desempenho dos mesmos na execução das tarefas é algo crucial para o cumprimento do SLA. No estudo conduzido por [8] foram realizados testes com as VMMs: Hyper-V, ESXI, OVM, VirtualBox, Xen, onde cada virtualizador foi submetido a testes e análises de CPU com Nbench, NIC Benchmark com netperf, tempo de compilação do kernel linux, teste de armazenamento com filebench e testes de memória com ramspeed. Com base nos testes o ESXi apresentou o melhor desempenho em relação aos demais.

Para executar aplicações cientificas em HPC (*High Performance Computing*), geralmente problemas são encontrados como a compatibilidade, por exemplo. Uma solução para este problema é a virtualização. Ermakov e Vasyukov [7] desenvolveram um trabalho que busca comparar a implementação de um HPC utilizando *Docker* e KVM através de testes de performance utilizando MPICH, Intel Linpack benchmark, High-performance linpack e OpenFOAM. Os autores concluíram que o desempenho em HPC depende de muitos fatores, como a rede envolvida na comunicação e o tipo do processador. Além disso, eles afirmam que os testes devem ser realizados no ambiente onde será feito o HPC, pois dependendo da sua infraestrutura os resultados podem oscilar, chegando a casos onde o *Docker* é melhor, e casos onde a Virtualização com KVM é melhor.

Dos trabalhos analisados, não foram encontrados na literatura trabalhos que consideram o tempo de inicialização das estratégias de virtualização por máquina virtual e por contêiner, utilizando as ferramentas Xen, KVM e Docker. Por essa razão, análises de desempenho foram realizadas considerando a instanciação de diferentes quantidades e configurações de VMs e contêineres, permitindo uma comparação entre as três ferramentas. Em seguida, novas análises foram realizadas considerando a execução de benchmarks (Apache e Smallpt) pelas VMs e contêineres com diferentes configurações de vCPUs e quantidades de VMs sobre o sistema hospedeiro. O objetivo foi analisar o desempenho do ambiente com as estratégias de virtualização sob diferentes tipos de serviços. A próxima seção discorre sobre a metodologia adotada.

4 PLANEJAMENTO DOS EXPERIMENTOS

O estudo apresentado neste artigo contempla um volume razoável de experimentos e foi desenvolvido em um ambiente prototipado. Dessa forma, em todas as fases foi elaborado um planejamento dos experimentos. Depois de realizar a coleta de dados, os resultados obtidos foram analisados utilizando-se métodos estatísticos seguindo a metodologia de planejamento fatorial completo proposta por Jain [10], na qual os fatores correspondem às características do ambiente e os níveis aos possíveis valores que podem ser adotados.

Dessa forma, considerando a possibilidade de diferentes fatores interferirem no ambiente, cada experimento foi executado dez vezes e, a partir dos resultados obtidos de cada experimento, foi calculada a média, o desvio padrão e o intervalo de confiança dos dados amostrais obtidos com 95% de confiança.

A utilização de técnicas de virtualização e contêiner dependem das características do sistema hospedeiro. Para o presente estudo, o hardware utilizado para os testes é descrito na Tabela 1.

Tabela 1: Especificações do sistema hospedeiro.

Componente:	**Descrição**
Processador:	Intel Core I7-2600 de 3.4GHz
Memória:	8192MB DDR3 de 1333MHz
Fonte:	Corssair CX550 de 550w
Motherboard:	IPMH61R1
SSD:	Kingston 120G

Dois conjuntos de experimentos foram definidos. O primeiro consiste em avaliar o tempo gasto para a inicialização das ferramentas KVM, Xen e Docker, variando a quantidade de vCPUs e de VMs, como mostra a Tabela 2. Dessa forma, foi possível verificar o tempo total gasto desde a inicialização de todas as instâncias até o ponto em que elas estão prontas para utilização (ociosas).

Tabela 2: Fatores e níveis para o teste de inicialização.

Fatores	**Níveis**
Ferramenta	KVM, Xen, Docker
Quantidade de instâncias	1, 2, 4, 8, 16
Quantidade de vCPUs	1, 2, 4, 8

O segundo conjunto de experimentações consiste em analisar o desempenho de cada ferramenta com variações na quantidade de recursos computacionais e com diferentes cargas. Para isso, foram realizados experimentos considerando diversas configurações de máquinas virtuais e contêineres, executando cargas do tipo *System* e *CPU-Bound*, por meio dos seguintes benchmarks:

- **Apache**[1]: consiste em um benchmark *System-Bound* que mensura a quantidade de requisições por segundo que um sistema consegue atender, com 100 requisições realizadas simultaneamente em um total de 1 milhão. Nos experimentos foi considerada como variável de resposta a Quantidade Média de Requisições Atendidas por segundo por todas as instâncias, em todas as replicações.
- **Smallpt**[2]: consiste em um benchmark *CPU-Bound* que realiza a renderização de imagens utilizando o algoritmo Monte Carlo. Nos experimentos foi considerado como variável de

[1] https://openbenchmarking.org/test/pts/apache
[2] https://openbenchmarking.org/test/pts/smallpt

resposta o Tempo Médio de Execução do benchmark, considerando todos os recursos virtuais provisionados de acordo com o planejamento dos experimentos.

A Tabela 3 apresenta os fatores e níveis definidos no planejamento do segundo conjunto de experimentos, enquanto que a próxima seção discute os resultados das experimentações.

Tabela 3: Fatores e níveis para os testes com benchmarks.

Fatores	Níveis
Ferramenta	KVM, Xen, Docker
Quantidade de instâncias	1, 2, 4, 8, 16
Quantidade de vCPUs	1, 2, 4, 8
Benchmark	Apache, Smallpt

5 AVALIAÇÃO DE DESEMPENHO

Nesta seção, avaliações de desempenho de dois conjuntos de experimentos são apresentadas. Na primeira, uma análise dos tempos de inicialização das ferramentas de virtualização por máquinas virtuais e por contêiner Xen, KVM e Docker foi realizada, considerando o comportamento do sistema com variações na quantidade de recursos virtuais (quantidade de instâncias e vCPUs). Na segunda, cada técnica foi analisada com variações na quantidade de recursos virtuais, executando dois tipos de benchmarks, Apache (*System-Bound*) e Smallpt (*CPU-Bound*).

5.1 Análise dos Tempos de Inicialização

Nos resultados apresentados na Figura 2 e na Tabela 4 foi possível verificar que em todos os casos o Docker obteve os menores tempos de inicialização do ambiente, em segundos, com as configurações definidas em cada experimento. Isso ocorreu devido às características dessa ferramenta em carregar apenas pacotes essenciais, ao contrário do Xen e KVM. Como exemplo, pode-se citar o experimento com maior sobrecarga e concorrência pelos recursos físicos, 16 instâncias com 8 vCPUs cada, no qual o Docker obteve um tempo de inicialização 87,57% menor que o Xen e 96,38% que o KVM.

Tabela 4: Tempos médios de inicialização em segundos.

Qtd Instâncias	Qtd vCPUs	KVM	Xen	Docker
1 instância	1	27,18	4,48	2,43
	2	27,18	4,47	2,44
	4	27,20	5,69	2,45
	8	27,21	4,62	2,44
2 instâncias	1	30,07	8,39	2,84
	2	32,08	8,30	2,85
	4	32,11	8,44	2,84
	8	33,19	8,51	2,84
4 instâncias	1	40,95	17,11	3,65
	2	40,02	17,11	3,65
	4	37,98	17,29	3,65
	8	44,16	16,74	3,64
8 instâncias	1	62,29	33,13	5,23
	2	61,46	32,15	5,23
	4	68,14	36,23	5,27
	8	77,51	34,97	5,24
16 instâncias	1	185,92	67,56	8,49
	2	196,32	72,44	8,49
	4	224,77	75,28	8,51
	8	223,83	81,00	8,44

O virtualizador Xen obteve o segundo menor tempo de inicialização em todos os experimentos. Ainda no experimento com maior sobrecarga sobre o sistema hospedeiro, verificou-se que ele apresentou uma redução no tempo de aproximadamente 65,35% em relação ao KVM.

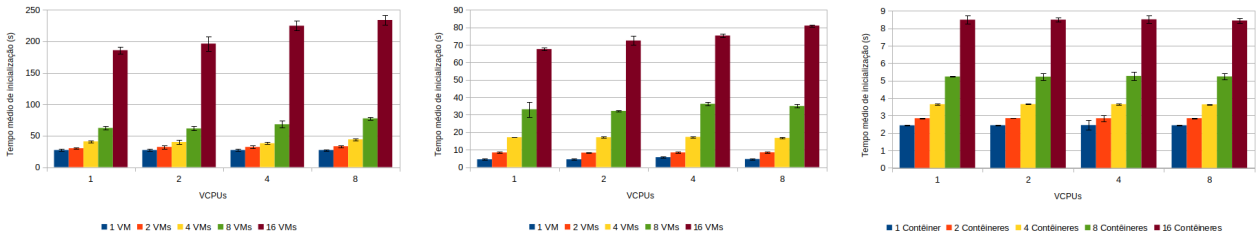
Por fim, considerando a saturação do sistema hospedeiro, foi possível verificar que, com exceção do Docker, o comportamento dos ambientes configurados com o Xen e KVM foi similar perante o aumento da quantidade de VMs e vCPUs. Em outras palavras, à medida que a quantidade de recursos computacionais aumentou, a competição por recursos físicos tornou-se maior, impactando diretamente no tempo de inicialização do sistema. Por outro lado, mesmo com o aumento dos recursos computacionais, o ambiente manteve o mesmo comportamento com a utilização do Docker, sendo necessário mais experimentos para quantificar o ponto de saturação do sistema com esse virtualizador.

5.2 Análise da Execução dos Benchmarks

A Figura 3 e a Tabela 5 apresentam a quantidade média de requisições atendidas por segundo por todas as instâncias na execução do benchmark Apache, com variações na quantidade de recursos computacionais. Em quase todos os experimentos o Docker mostrou-se superior aos virtualizadores Xen e KVM, uma vez que ele é mais leve e, dessa forma, executa uma quantidade menor de tarefas concorrentes no processador virtual. As exceções foram os experimentos com 1 vCPU e variação na quantidade de instâncias, 1, 2 e 4, nos quais o KVM mostrou-se superior aos demais. Em uma comparação com o Docker, o KVM foi superior em aproximadamente 28% com 1 instância, 17% com 2 e 23% com 4. Isso ocorreu pois, para esses casos, o mecanismo do Docker responsável pelo mapeamento entre os recursos virtuais e físicos atribui apenas 50% de cada núcleo físico para cada núcleo virtual. Dessa forma, para as configurações de recursos utilizada nos experimentos em questão, parte dos recursos físicos ficaram ociosos.

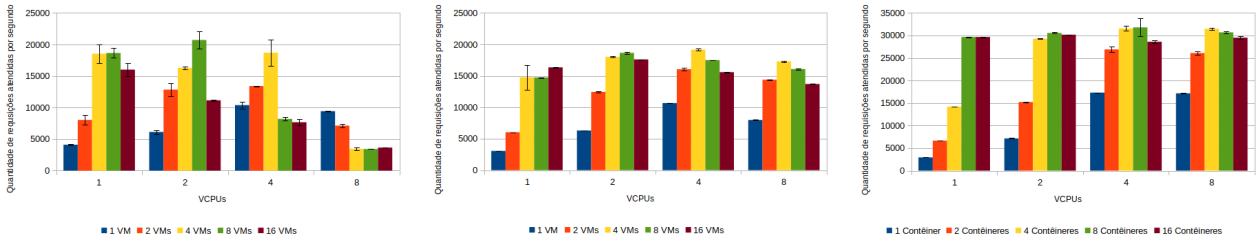
Tabela 5: Quantidade média de requisições atendidas (RA) por segundo.

Qtd Instâncias	Qtd vCPUs	KVM	Xen	Docker
1 instância	1	4096,80	3061,24	2936,26
	2	6122,39	6302,04	7144,33
	4	10380,46	10661,95	17300,18
	8	9418,19	7975,06	17150,81
2 instâncias	1	8027,32	6027,61	6650,38
	2	12839,67	12434,14	15218,88
	4	13405,75	16044,14	26899,58
	8	7133,32	14368,89	26089,25
4 instâncias	1	18532,28	14758,87	14202,42
	2	16294,03	18010,69	29283,09
	4	18688,03	19152,55	31542,95
	8	3443,65	17292,38	31456,74
8 instâncias	1	18649,65	14729,98	29661,83
	2	20714,21	18648,14	30547,87
	4	8189,84	17497,56	31788,76
	8	3417,40	16071,97	30679,87
16 instâncias	1	16009,98	16335,68	29661,83
	2	11143,32	17599,28	30156,67
	4	7665,23	15569,71	28592,28
	8	3643,61	13687,17	20503,89



(a) KVM (b) Xen (c) Docker

Figura 2: Tempos médios de inicialização das ferramentas.



(a) KVM (b) Xen (c) Docker

Figura 3: Quantidade média de requisições atendidas por segundo pelo ambiente.

Além disso, em todos os virtualizadores a quantidade de requisições atendidas aumentou à medida que mais recursos computacionais (instâncias e vCPUs) foram alocados. No entanto, esse comportamento mudou nos experimentos com 8 e 16 VMs e variações de 1, 2, 4 e 8 vCPUs. Para esses casos, verificou-se uma saturação do sistema hospedeiro, causado pela concorrência entre os recursos virtuais pelos recursos físicos, o que proporcionou uma queda na quantidade de requisições atendidas pelo sistema. O mesmo comportamento foi verificado no Xen para os experimentos com 8 VMs e variações de 4 e 8 vCPUs, bem como para os experimentos com 16 VMs e variações de 2, 4 e 8 vCPUs.

No KVM, verificou-se que o sistema teve o seu desempenho prejudicado quando o aumento de recursos virtuais atingiu a saturação dos recursos físicos. Esse comportamento pode ser observado nos experimentos com 8 vCPUs e variação na quantidade de instâncias (1, 2, 4, 8 e 16).

A Figura 4 e a Tabela 6 apresentam o tempo gasto pelos recursos virtuais alocados para a execução do benchmark Smallpt, o qual aplica a renderização de imagens por meio do algoritmo Monte Carlo.

Verificou-se que os resultados foram similares para os ambientes com Xen, KVM e Docker com instanciações de 8 e 16 VMs e variações na quantidade de vCPUs. Nestes casos, a quantidade de recursos alocados prejudicou o tempo de execução do benchmark devido à disputa por recursos físicos entre os recursos virtuais.

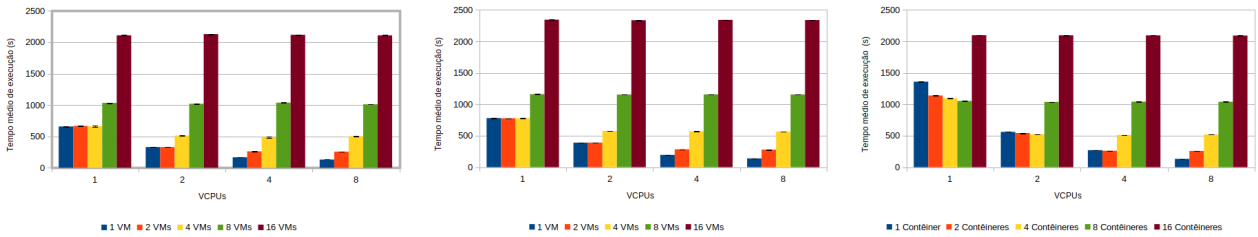
Com relação ao Docker, verificou-se que tempo de execução reduziu nos experimentos com 1 vCPU e variação na quantidade de VMs (1, 2 e 4 VMs). Além disso, verificou-se que o tempo de execução foi similar nas instanciações de 1, 2 e 4 VMs com 2 vCPUs

Tabela 6: Tempos médios de execução (TME) em segundos.

Qtd Instâncias	Qtd vCPUs	KVM	Xen	Docker
1 instância	1	657,63	779,72	1359,00
	2	329,97	389,33	560,56
	4	167,26	194,56	270,69
	8	130,75	138,71	131,24
2 instâncias	1	666,69	776,76	1139,25
	2	330,44	389,61	542,31
	4	261,27	283,63	261,63
	8	257,20	278,20	258,05
4 instâncias	1	662,18	777,72	1095,81
	2	512,58	575,17	524,77
	4	488,65	570,23	512,95
	8	502,92	564,18	521,04
8 instâncias	1	1033,47	1160,22	1052,83
	2	1019,54	1154,69	1037,46
	4	1037,99	1156,60	1041,79
	8	1011,33	1154,98	1039,81
16 instâncias	1	2109,68	2343,77	2096,90
	2	2128,67	2335,03	2096,06
	4	2118,46	2341,23	2094,73
	8	2109,21	2337,35	2091,60

cada, e nos experimentos com 1 e 2 VMs e 4 vCPUs. Esses comportamentos permitem estabelecer um limiar válido para as três técnicas, no qual é possível verificar o ponto onde o aumento de recursos virtuais começa a prejudicar o desempenho do ambiente devido à concorrência por recursos físicos.

De maneira geral, tem-se então um comportamento inicial estabelecido com variações na quantidade de VMs e apenas 1 vCPU por VM, o qual demonstrou a necessidade de mais recursos computacionais para que o tempo de execução fosse reduzido. Essa redução no tempo permaneceu até que os recursos físicos se tornaram limitantes devido à concorrência entre os recursos virtuais.



(a) KVM (b) Xen (c) Docker

Figura 4: Tempos médios de execução do benchmark.

O comportamento descrito pode ser observado nos experimentos com 8 vCPUs, nos quais verificou-se os menores tempos de execução com 1 VM e 8 vCPUs. Dessa forma, partindo desta configuração, pode-se verificar que o aumento na quantidade de VMs proporcionou o aumento dos tempos nos outros experimentos.

Por fim, um outra análise importante diz respeito à escalabilidade dos recursos, a qual visa responder a seguinte hipótese: em termos de escalabilidade de recursos para atender um carga *CPU-Bound*, é melhor aumentar a quantidade de instâncias ou de vCPUs dentro de uma mesma instância?

De acordo com os experimentos é melhor ter mais vCPUs em uma instância ao invés de mais instâncias com uma quantidade menor de vCPUs. Isso ocorre, devido a troca de contexto entre as VMs que estão concorrendo para acessar o recurso físico.

O comportamento descrito pode ser observado nos experimentos com 8 instâncias e 1 vCPU, 4 instâncias e 2 vCPUs, 2 instâncias e 4 vCPUs e 1 instância e 8 vCPUs, nos quais, embora a quantidade total de vCPUs presente no sistema seja a mesma (8 vCPUs em todos os experimentos), a redução da quantidade de instâncias junto com o aumento de vCPUs permitiu uma redução nos tempos médios de execução.

6 CONCLUSÕES

A utilização de computação em nuvem permite aumentar dinamicamente a capacidade de prestação de serviços de uma empresa ou de atender às solicitações de serviços dos clientes, sem que estes precisem investir em infraestrutura como compra de hardware, de licenças de software ou treinamento de funcionários. Neste contexto, a virtualização desempenha um papel fundamental para o provisionamento e gerenciamento de recursos, e consequentemente, na qualidade do serviço prestado.

Neste trabalho foram realizadas avaliações de desempenho de ambientes configurados com diferentes estratégias de virtualização: por máquinas virtuais e por contêiner, utilizando as ferramentas KVM, Xen e Docker. No primeiro conjunto de experimentações foi analisado o tempo de inicialização das três ferramentas com diferentes quantidades de recursos (instâncias e vCPUs).

Verificou-se nos resultados uma superioridade do Docker com relação aos outros dois hipervisores, seguido pelo Xen e KVM. Isso ocorreu pelo fato do contêiner armazenar apenas o que é necessário para executar a aplicação, sem necessidade de virtualizar hardware como os hipervisores.

Em seguida, outro conjunto de experimentos foi realizado, no qual foi analisado o desempenho das três ferramentas durante a execução dos benchmarks Apache (*System-Bound*) e Smallpt (*CPU-Bound*), e com variações na quantidade de recursos virtualizados.

Com relação à quantidade de requisições atendidas fornecida pelo benchmark Apache, o Docker apresentou novamente os melhores resultados. O KVM mostrou-se mais eficiente que o Xen quando uma quantidade menor de recursos virtuais foi alocada. No entanto, este comportamento mudou à medida que a quantidade de recursos aumentou. Neste ponto, o Xen foi mais eficiente em relação ao KVM durante o atendimento das requisições.

Por outro lado, para o benchmark Smallpt, o qual fornece o tempo de execução da renderização de imagens como variável de resposta, as três técnicas apresentaram comportamentos similares. Destaca-se o conjunto de experimentos no Docker com 1 vCPU e variações na quantidade de instâncias. Neste caso, verificou-se que a quantidade inicial de recursos alocados prejudicou o desempenho do ambiente. No entanto, com o aumento da quantidade de recursos, o tempo de execução foi reduzido até que a quantidade de recursos físicos tornou-se um fator limitante. Isso ocorreu devido à competição por recursos físicos entre os recursos virtuais.

O presente trabalho não finaliza as possibilidades de estudos com relação a avaliação de ambientes virtualizados com máquinas virtuais e com contêineres, podendo ainda abordar outros aspectos e pontos que complementariam o trabalho desenvolvido. Entre eles destacam-se a utilização de uma infraestrutura em nuvem para os testes, com variações mais expressivas na quantidade de recursos alocados, bem como nos tipos de virtualizadores e benchmarks.

REFERÊNCIAS

[1] Khalid Alhamazani, Rajiv Ranjan, Karan Mitra, Fethi Rabhi, Prem Prakash Jayaraman, Samee Ullah Khan, Adnene Guabtni, and Vasudha Bhatnagar. 2014. An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art. *Computing* 97, 4 (2014), 357–377.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. 2003. Xen and the art of virtualization. In *ACM SIGOPS Operating Systems Review*, Vol. 37. ACM, 164–177.

[3] A. Carissimi. 2008. Virtualização: da teoria a soluções. *Minicursos do Simpósio Brasileiro de Redes de Computadores–SBRC* 2008 (2008), 173–207.

[4] Antonio Celesti, Davide Mulfari, Maria Fazio, Massimo Villari, and Antonio Puliafito. 2016. Exploring container virtualization in IoT clouds. In *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*. IEEE, 1–6.

[5] Elder de Macedo Rodrigues. 2009. *Realocaçao de recursos em ambientes virtualizados.* Ph.D. Dissertation. Pontifícia Universidade Católica do Rio Grande do Sul.

[6] Docker. 2017. What is Docker. https://www.docker.com/what-docker

[7] Alexey Ermakov and Alexey Vasyukov. 2017. Testing Docker Performance for HPC Applications. *arXiv preprint arXiv:1704.05592* (2017).

[8] Waldemar Graniszewski and Adam Arciszewski. 2016. Performance analysis of selected hypervisors (Virtual Machine Monitors-VMMs). *International Journal of Electronics and Telecommunications* 62, 3 (2016), 231–236.
[9] S. Ibrahim, B. He, and H. Jin. 2011. Towards pay-as-you-consume cloud computing. In *Services Computing (SCC), 2011 IEEE International Conference on*. IEEE, 370–377.
[10] R. Jain. 1991. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. New York, NY, USA, Wiley.
[11] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu. 2007. An analysis of performance interference effects in virtual environments. In *Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on*. IEEE, 200–209.
[12] Y. Li, W. Li, and C. Jiang. 2010. A survey of virtual machine system: Current technology and future trends. In *Electronic Commerce and Security (ISECS), 2010 Third International Symposium on*. IEEE, 332–336.
[13] Q. Liu, C. Weng, M. Li, and Y. Luo. 2010. An In-VM measuring framework for increasing virtual machine security in clouds. *Security & Privacy, IEEE* 8, 6 (2010), 56–62.
[14] D.M.F. Mattos. 2008. Virtualização: VMWare e Xen. *Universidade Federal do Rio de Janeiro* (2008).
[15] P. Mell and T. Grance. 2011. The NIST definition of cloud computing (draft). *NIST special publication* 800 (2011), 145.
[16] I. Menken and G. Blokdijk. 2010. *Cloud Computing Foundation Complete Certification Kit-Study Guide Book and Online Course*. Emereo Pty Ltd.
[17] A. Menon, J.R. Santos, Y. Turner, G.J. Janakiraman, and W. Zwaenepoel. 2005. Diagnosing performance overheads in the xen virtual machine environment. In *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*. ACM, 13–23.
[18] Alfonso Pérez, Germán Moltó, Miguel Caballer, and Amanda Calatrava. 2018. Serverless computing for container-based architectures. *Future Generation Computer Systems* 83 (2018), 50–59.
[19] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu. 2010. Understanding performance interference of i/o workload in virtualized cloud environments. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 51–58.
[20] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, C. Pu, and Y. Cao. 2012. Who is your neighbor: net I/O performance interference in virtualized clouds. (2012).
[21] V. Rajasekaran, J. Martin, and J. Westall. 2004. On the Impact of Virtual Machine Overhead on TCP Performance. In *Communication and Computer Networks*. ACTA Press.
[22] P Vijaya Vardhan Reddy and Lakshmi Rajamani. 2014. Performance Evaluation of Hypervisors in the Private Cloud based on System Information using SIGAR Framework and for System Workloads using Passmark. *International Journal of Advanced Science and Technology* 70 (2014), 17–32.
[23] B.P. Rimal, E. Choi, and I. Lumb. 2009. A taxonomy and survey of cloud computing systems. In *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*. IEEE, 44–51.
[24] Piet Smet, Bart Dhoedt, and Pieter Simoens. 2018. Docker layer placement for on-demand provisioning of services on edge clouds. *IEEE Transactions on Network and Service Management* (2018).
[25] B. Sotomayor, K. Keahey, and I. Foster. 2006. Overhead matters: A model for virtual resource management. In *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*. IEEE Computer Society, 5.
[26] Xili Wan, Xinjie Guan, Tianjing Wang, Guangwei Bai, and Baek-Yong Choi. 2018. Application deployment using Microservice and Docker containers: Framework and optimization. *Journal of Network and Computer Applications* 119 (2018), 97–109.
[27] VM Ware. 2017. Virtualization Essentials. https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/ebook/gated-vmw-ebook-virtualization-essentials.pdf
[28] Yi Wei and M Brian Blake. 2014. Proactive virtualized resource management for service workflows in the cloud. *Computing* (2014), 1–16.
[29] A. Whitaker, M. Shaw, and S.D. Gribble. 2002. Denali: A scalable isolation kernel. In *Proceedings of the 10th workshop on ACM SIGOPS European workshop*. ACM, 10–15.
[30] J. White and A. Pilbeam. 2010. A survey of virtualization technologies with performance testing. *arXiv preprint arXiv:1010.3233* (2010).
[31] X. You, X. Xu, J. Wan, and C. Jiang. 2009. Analysis and evaluation of the scheduling algorithms in virtual environment. In *Embedded Software and Systems, 2009. ICESS'09. International Conference on*. IEEE, 291–296.
[32] Xiaobo Zhou and Chang-Jun Jiang. 2014. Autonomic Performance and Power Control on Virtualized Servers: Survey, Practices, and Trends. *Journal of Computer Science and Technology* 29, 4 (2014), 631–645.