UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

CRISTIANO CARLOS MATTE

Viabilidade do emprego de processadores ARM para infraestrutura de computação em nuvem (IaaS)

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Alexandre da Silva Carissimi

Porto Alegre
2016

AGRADECIMENTOS

Agradeço, inicialmente, a todos os amigos e colegas que estiveram presentes nos meus anos de graduação, sempre dispostos a ouvir, debater e aconselhar.

Ao meu orientador, Alexandre Carissimi, por todo o apoio, paciência e dedicação dispendidas para que este trabalho se tornasse possível.

À Bruna, minha namorada, por todo carinho e incentivo recebido nessa difícil etapa final de graduação, e pela compreensão dos momentos de pouca atenção em que dedicava meu tempo ao desenvolvimento deste trabalho. Te amo!

Aos meus pais, por todo o apoio e oportunidades dadas para que eu chegasse até aqui e, como diz meu pai, "realizasse os sonhos de sua juventude". Admiro demais o esforço de vocês para que eu e meu irmão tivessem as melhores condições de vida possíveis. Agradeço imensamente o apoio nas minhas tomadas de decisões e a preocupação e presença nos momentos mais difíceis dessa caminhada. Amo vocês!

Em especial, ao meu irmão, figura mais presente em minha vida durante os anos de graduação. Obrigado por toda a companhia, pelas ideias trocadas e momentos que passamos juntos. Obrigado por ter acompanhado e auxiliado minha transformação pessoal, e por ter sido meu grande ombro amigo nos momentos mais difíceis, dando conselhos sempre valiosos. Este trabalho é dedicado a você.

RESUMO

A computação em nuvem, desde a última década, vem apresentando uma larga expansão e mostrando ser uma alternativa que oferece maior flexibilidade e menores custos aos usuários. Além disso, a computação em nuvem é benéfica ao meio-ambiente ao utilizar os recursos de forma mais eficiente e ao utilizar amplamente tecnologias de virtualização, técnicas consideradas sustentáveis no âmbito da computação verde. Outra maneira de se obter uma computação sustentável é através da utilização de processadores de baixo consumo de energia elétrica. Nesse sentido, o presente trabalho apresenta uma análise da viabilidade do emprego de um minicomputador de placa única, com processador ARM, como hardware para execução de máquinas virtuais, com o objetivo de oferecer uma nuvem computacional que segue o modelo de infraestrutura como serviço (IaaS). Para isso, o sistema desenvolvido utiliza o minicomputador Cubietruck em sua infraestrutura, além de soluções de software livre como Xen, para virtualização, e OpenNebula, para gerência de nuvem.

Palavras-chave: computação em nuvem, virtualização, computação verde, ARM, Cubietruck, Xen, OpenNebula.

Viability study of the use of ARM processors for cloud computing infrastructure (IaaS)

ABSTRACT

Since the last decade, cloud computing has been expanding and proving to be an alternative that offers greater flexibility and lower costs to users. Furthermore, cloud computing is beneficial to the environment by utilizing resources more efficiently and by making extensive use of virtualization technologies, techniques that are considered sustainable in the field of green computing. Another way to achieve a sustainable computing is through the use of low-power processors. In this sense, this work presents an viability analysis of the use of a single board computer that uses an ARM processor as the hardware for virtual machines execution, aiming to offer a cloud that implements the infrastructure as service (IaaS) model. The developed system uses the Cubietruck minicomputer in its infrastructure, as well as free software solutions such as Xen for virtualization and OpenNebula for cloud management.

Keywords: cloud computing, virtualization, green computing, ARM, Cubietruck, Xen, OpenNebula.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AMD	*Advanced Micro Devices*
API	*Application Programming Interface*
ARM	*Advanced RISC Machines*
AWS	Amazon *Web Services*
BIOS	*Basic Input/Output System*
DTB	*Device Tree Binary*
EC2	*Elastic Compute Cloud*
GCC	*GNU C Compiler*
GNU	*GNU's Not Unix!*
GPL	*General Public Licence*
GPPD	Grupo de Processamento Paralelo e Distribuído
HDMI	*High-Definition Multimedia Interface*
HP	*Hewllet-Packard*
IaaS	*Infrastructure as a Service*
IP	*Internet Protocol*
IBM	*International Bussiness Machines*
JVM	Java *Virtual Machine*
KVM	*Kernel-based Virtual Machine*
LAN	*Local Area Network*
LVM	*Logical Volume Manager*
NAND	*Not AND*
NFS	*Network File System*
NIST	*National Institute of Standards and Technology*
PaaS	*Plataform as a Service*

QEMU	*Quick Emulator*
RAM	*Random Access Memory*
REST	*Representation State Transfer*
RISC	*Reduced Instruction Set Architecture*
S3	*Simple Storage Service*
SaaS	*Software as a Service*
SDN	*Software Defined Network*
SOAP	*Simple Object Access Protocol*
SSH	*Secure Shell*
QEMU	*Quick Emulator*
UART	*Universal asynchronous receiver/transmiter*
USB	*Universal Serial Bus*
VGA	*Video Graphics Array*
VPN	*Virtual Private Network*

SUMÁRIO

1 INTRODUÇÃO

A utilização de soluções de computação em nuvem vem passando por uma grande ascenção nos últimos anos. É cada vez mais comum que usuários de computador utilizem email, redes sociais, serviços de *streaming* e armazenamento e diversas outras aplicações web, mesmo muitas vezes não sabendo que por trás desses serviços há uma infraestrutura de nuvem. Da mesma maneira, desenvolvedores e empresas da área de tecnologia da informação passaram a investir menos em infraestrutura própria e a utilizar recursos fornecidos pela nuvem para hospedar seus produtos e serviços.

Este crescente interesse é justificado, primeiramente, pelo modelo de negócio que a computação em nuvem introduz. Com ela, a computação deixa de ser comercializada como um produto e passa a ser comercializada como um serviço. Dessa maneira, os usuários pagam apenas pelos recursos que eles efetivamente consomem e não precisam investir na aquisição de recursos próprios, muitas vezes não utilizados em sua total capacidade. Além disso, a responsabilidade pela gerência e manutenção da infraestrutura contratada passa a ser do provedor de computação em nuvem, e o usuário pode se preocupar apenas com aquilo que lhe é de interesse ou de seu negócio.

Dado o grande interesse por parte dos usuários, esse mercado passou a ser interessante também para os provedores de computação em nuvem. Gigantes da tecnologia como Amazon, IBM, Google e Microsoft, entre outros, investem cada vez mais em sua infraestrutura e em suas soluções de nuvem, e esses investimentos tem gerado bons resultados financeiros. O lucro da Amazon com suas soluções de nuvem superou o lucro com o varejo no último quarto de 2015 (MCBRIDE; MEDHORA, 2016), e a Microsoft, embora tenha apresentado queda nos lucros e na receita no mesmo período, notou um aumento das receitas em seus negócios de nuvem (WINGFIELD, 2016).

Além de ser benéfica para os usuários e os provedores, a computação em nuvem também é benéfica para o meio-ambiente ao lidar de forma sustentável com grandes desafios da computação verde. A computação verde (*green computing*) trata do estudo e da prática de uma computação ambientalmente sustentável, incluindo todo o ciclo de vida dos computadores e seus periféricos: projeto, fabricação, utilização e descarte (MURUGESAN, 2008). Deseja-se que esses processos sejam feitos de maneira eficiente com um efeito mínimo no meio-ambiente. Nesse sentido, a computação em nuvem provê uma utilização eficiente do hardware através da utilização de recursos sob demanda, onde novos recursos são alocados apenas em momentos

de alta demanda e liberados em seguida, evitando o desperdício. Além disso, a utilização da mesma infraestrutura para atender diversos usuários e empresas faz com que a taxa de utilização dos servidores seja mais alta. Essas características, além de sustentáveis do ponto de vista da redução no número de recursos adquiridos, são também benéficas por gerarem economia no consumo de energia elétrica.

O impacto da computação em nuvem para uma computação mais sustentável pode ser grande. Segundo (MICROSOFT, 2016), a migração para soluções de nuvem pode representar uma redução no consumo de energia de 30% a 90%, quando comparado à utilização de uma infraestrutura de servidores própria. Nesse mesmo sentido, (BARR, 2015) afirma que migrar para os serviços da Amazon representa uma redução de 88% na emissão de carbono na atmosfera. Além disso, para fornecer soluções ainda mais sustentáveis, a Amazon busca alcançar um compromisso de utilizar apenas fontes de energia renováveis para abastecer toda sua infraestrutura, através da construção e ampliação de seus parques eólicos e solares (AMAZON, 2016).

Outra maneira de reduzir o consumo de energia elétrica na computação é utilizar processadores de baixo consumo. Nesse contexto, destacam-se os processadores ARM, projetados, principalmente, para sistemas embarcados onde há restrições de consumo de energia. Os processadores ARM passaram por uma grande ascenção nos últimos anos devido ao surgimento dos *smartphones* e *tablets*, segmento em que a empresa possui o domínio do mercado. Além da aplicação em dispositivos móveis, esses processadores também são utilizados em diversas soluções embarcadas e em minicomputadores de placa única.

A utilização de práticas ambientalmente sustentáveis ganha uma importância cada vez maior e atinge as mais diversas áreas, tornando-se necessário manter uma reflexão constante sobre novas formas de utilizarmos os recursos disponíveis. Nesse sentido, a computação em nuvem, além dos inúmeros benefícios para seus usuários, cumpre um papel importante para a redução do desperdício e utilização eficiente dos recursos. A combinação desse modelo com processadores de baixo consumo passa a ser um campo de estudo interessante para que consigamos atingir uma computação sustentável.

1.1 Objetivo

O presente trabalho possui dois objetivos principais. Primeiro, apresentar os conceitos e tecnologias de computação em nuvem e virtualização. Segundo, avaliar a viabilidade do emprego do *Cubietruck*, um minicomputador de placa única com processador ARM, como hardware para execução de máquinas virtuais, visando o oferecimento de uma nuvem computacional que segue o modelo de infraestrutura como serviço (IaaS). Para isso, será feito um estudo das capacidades de virtualização do *Cubietruck* e sua compatibilidade com aplicações de virtualização existentes, bem como a possibilidade de integrá-lo a uma infraestrutura de nuvem gerida pelo OpenNebula, um gerenciador de nuvem de código aberto.

1.2 Organização do trabalho

Este trabalho é organizado em 4 capítulos, além desta introdução. O capítulo 2 apresenta os principais conceitos de virtualização e computação em nuvem, além das principais ferramentas existentes na área. O capítulo 3 fornece a arquitetura e a implementação do sistema proposto neste trabalho, abordando as diferentes tecnologias e ferramentas de software e hardware utilizados. No capítulo 4, é feita uma avaliação do sistema desenvolvido para analisar a viabilidade de seu emprego. Por fim, o capítulo 5 apresenta as conclusões e resultados do trabalho, bem como sugestões de melhorias para trabalhos futuros.

2 COMPUTAÇÃO EM NUVEM E VIRTUALIZAÇÃO

Desde a segunda metade da década de 2000, a utilização de soluções que empregam computação em nuvem vem passando por uma grande ascensão. Embora existam registros que utilizam esse termo há mais tempo, a disponibilidade de redes de alta capacidade, o surgimento de computadores e meios de armazenamento de baixo custo e a ampla adoção de virtualização de hardware permitiram o largo crescimento da computação em nuvem a partir da última década. Desde então, o número de sistemas e o interesse dos usuários nessas soluções vem aumentando. A rápida expansão é justificada pelos inúmeros benefícios que podem ser obtidos, como economia de recursos e energia elétrica, eficiência no uso de recursos, redução de custos, escalabilidade, segurança e flexibilidade.

A computação em nuvem tem como grande alicerce tecnológico a virtualização. É através da virtualização de recursos como máquinas, redes e armazenamento que uma arquitetura de nuvem fornece seus serviços sem ter a necessidade de dedicar uma máquina física exclusiva para cada serviço contratado. Assim, um servidor físico pode estar executando diversos serviços, de diversos usuários, sobre recursos virtuais, aproveitando-se dos benefícios citados anteriormente. Além de ser benéfica para o provedor de computação em nuvem, a utilização da virtualização beneficia os clientes: pode-se alocar mais recursos virtuais em momentos que a demanda de processamento é alta, e diminuir o número de recursos contratados quando a demanda for menor. Dessa forma, paga-se só pelos recursos utilizados, e a computação passa a ser vista como um serviço ao invés de ser vista como um produto.

As seções deste capítulo apresentarão os principais conceitos de virtualização e de computação em nuvem, bem como exemplos de aplicações de máquinas virtuais, serviços de nuvens existentes e ferramentas que fazem a gerência de uma infraestrutura de nuvem.

2.1 Virtualização

A virtualização consiste na técnica de abstrair características físicas de recursos computacionais para outros sistemas, aplicações ou usuários finais que com eles interagem (IBM, 2007), através da criação de versões virtuais de recursos. Quando um recurso é virtualizado, ele é basicamente separado de uma peça de hardware físico. Assim, recursos virtuais podem ser temporariamente associados a recursos físicos, e essa associação pode mudar ao longo do tempo (SMITH; NAIR, 2003).

O conceito de virtualização surgiu no início da década de 1970, quando era comum que cada *mainframe*, mesmo de um único fabricante, tivesse seu próprio sistema operacional. Para permitir que aplicações legadas de outros sistemas operacionais executassem em um *mainframe*, surgiram as máquinas virtuais. Assim, podia-se executar aplicações de uma plataforma em outra desde que houvesse uma máquina virtual para a máquina alvo que desse suporte para a plataforma original das aplicações (CARISSIMI, 2008).

A utilização da virtualização, entretanto, não se restringe às máquinas virtuais. Ela é uma técnica muito ampla e pode ser aplicada a diversos recursos computacionais como processador, memória, rede, armazenamento e dispositivos de entrada e saída. As máquinas virtuais são apenas um caso de aplicação de virtualização, onde uma camada de software oferece um ambiente similar ao de uma máquina física (CARISSIMI, 2008). Dessa forma, uma única máquina física pode executar diversas instâncias de máquinas virtuais, cada uma com seu próprio sistema operacional, bibliotecas e aplicativos.

O emprego mais comum de máquinas virtuais, atualmente, está na virtualização de servidores. É comum que infraestruturas de rede utilizem a filosofia de "um servidor por serviço", visando atender diferentes arquiteturas clientes e garantir a segurança dos serviços ao isolar fisicamente os servidores. Essa prática, no entanto, não utiliza todo o potencial do processador, visto que a maior parte dos servidores operam apenas entre 5% e 15% de sua capacidade de carga total (VMWARE, 2016). Além disso, o custo de aquisição e manutenção da infraestrutura é alto, já que cada máquina executa exclusivamente um único servidor.

Ao empregar a virtualização de servidores, cada servidor físico é particionado e executa múltiplos servidores virtuais. Essa técnica é conhecida como consolidação de servidores. Assim, o poder de processamento da máquina física é utilizado de maneira mais eficiente. Além disso, há uma redução de custos tanto na aquisição de máquinas quanto em energia elétrica, refrigeração, cabeamento, espaço físico, suporte e manutenção. Outra vantagem é a facilidade que as máquinas virtuais possuem de serem replicadas e possuírem seus estados salvos, fornecendo aos servidores um alto grau de flexibilidade e portabilidade. De fato, de acordo com (BURGER, 2012), não é incomum alcançar uma consolidação de servidores onde dez servidores virtuais executam em um único servidor físico.

Os benefícios da utilização de máquinas virtuais também são observados quando elas são utilizadas em *desktops*. Através delas, um usuário pode executar aplicações que foram feitas para sistemas operacionais diferentes do nativo. Em ambientes de testes e desenvolvimento, as máquinas virtuais permitem uma instalação rápida ao isolar a aplicação em um ambiente

conhecido e controlado. Dessa forma, diferentes instalações com diferentes requerimentos e dependências podem ser isoladas em diferentes máquinas virtuais.

Na computação em nuvem, a virtualização exerce um papel central com a utilização de máquinas virtuais. Através da criação de um sistema escalável de múltiplos recursos computacionais independentes, os recursos ociosos podem ser alocados e utilizados de maneira mais eficiente. Dessa forma, a virtualização fornece agilidade para melhorar o desempenho de operações e reduz custos ao aumentar a utilização da infraestrutura. O cliente do provedor de computação em nuvem também obtém vantagens desse modelo. Além da redução de custos, a utilização de máquinas virtuais na infraestrutura de nuvem permite que mais máquinas virtuais sejam contratadas em momentos de alta demanda, e liberadas posteriormente quando a demanda for normalizada.

2.1.1 Máquinas virtuais

Uma máquina virtual é implementada através da adição de software a uma plataforma, dando a esta a aparência de outra plataforma, ou ainda de múltiplas plataformas. É possível que uma máquina virtual possua um sistema operacional, conjunto de instruções, ou ambos, que diferem daqueles implementados na máquina física em que ela executa. Nesse contexto, surgem os conceitos de sistema hospedeiro e sistema hóspede. O sistema hospedeiro é o sistema que executa diretamente sobre o hardware de uma máquina física. Já o sistema hóspede é o sistema que executa sobre uma máquina virtual.

De acordo com (SMITH; NAIR, 2003), o modo como as máquinas virtuais são implementadas pode ser classificado de duas maneiras: máquinas virtuais de processo e máquinas virtuais de sistema. Essa classificação é devida a diferente maneira como o conceito de “máquina” pode ser percebido. Do ponto de vista de um processo, uma máquina é o espaço de memória a ele destinado, os registradores do processador e as instruções que permitem que seu código seja executado. O sistema de entrada e saída e o acesso ao disco são percebidos de maneira abstrata, através de chamadas feitas ao sistema operacional. Além disso, processos são efêmeros: eles são criados, executam por um período de tempo e terminam.

Já de um ponto de vista de mais alto nível, uma máquina dá suporte a um sistema inteiro. Um sistema é um ambiente completo que executa inúmeros processos, aloca memória, gerencia dispositivos de entrada e saída e sistemas de arquivos e permite que os processos interajam com o sistema operacional que é parte do sistema. Além disso, ao contrário dos processos, o ambiente de um sistema persiste ao longo do tempo.

Desse modo, uma máquina virtual de processo é capaz de suportar um único processo, sendo criada e destruída juntamente com esse processo hóspede. As máquinas virtuais em nível de linguagem de programação são exemplos de máquinas virtuais de processo.

As máquinas virtuais em nível de linguagem de programação são utilizadas por algumas linguagens de alto nível e permitem aos desenvolvedores a criação de aplicações que executem em diversas plataformas diferentes sem que haja necessidade de códigos-fonte e compilações específicos para cada plataforma. Nesses casos, o código compilado não gera instruções de nenhuma arquitetura específica, mas sim instruções intermediárias que devem ser interpretadas por uma máquina virtual. Para cada plataforma que a aplicação for executar, deve existir uma máquina virtual que interprete o código intermediário gerado e converta para as instruções específicas daquela plataforma. A máquina virtual Java (Java *Virtual Machine* – JVM) é o maior exemplo desse tipo de máquina virtual. Para executar um programa escrito na linguagem Java, é necessário que a JVM esteja instalada no computador.

Já em nível de sistema operacional, múltiplas instâncias isoladas de espaços de usuário coexistem em um mesmo sistema operacional. Essas instâncias são normalmente chamadas de *containers* ou *jails*. O interesse no uso desses *containers* está na segurança existente ao isolar os múltiplos espaços e no melhor aproveitamento do hardware ao dividir uma única máquina em diversos espaços de usuário. Essas máquinas virtuais são mais eficientes que as máquinas virtuais em nível de hardware, visto que há um único sistema operacional em execução. Por outro lado, há uma menor flexibilidade, já que não é possível que um *container* execute um sistema operacional diferente do sistema hospedeiro. A aplicação *chroot*, do sistema Unix, é um exemplo desse tipo de máquina virtual. Outro exemplo é o serviço Docker (DOCKER, 2016), que permite a contratação de ambientes de desenvolvimento que executam sobre *containers*.

No caso das máquinas virtuais de sistema, um hardware virtual é criado para prover um ambiente de um sistema completo. Essas máquinas atuam em nível de hardware e suportam múltiplos processos de usuário, sistemas de arquivos, acesso a dispositivos de entrada e saída e, em alguns casos, interface gráfica. Nesse caso, o sistema operacional hóspede pode ser diferente do sistema operacional hospedeiro, bem como o conjunto de instruções utilizado pelas aplicações das máquinas virtuais pode ser diferente do conjunto de instruções da máquina física em que elas executam.

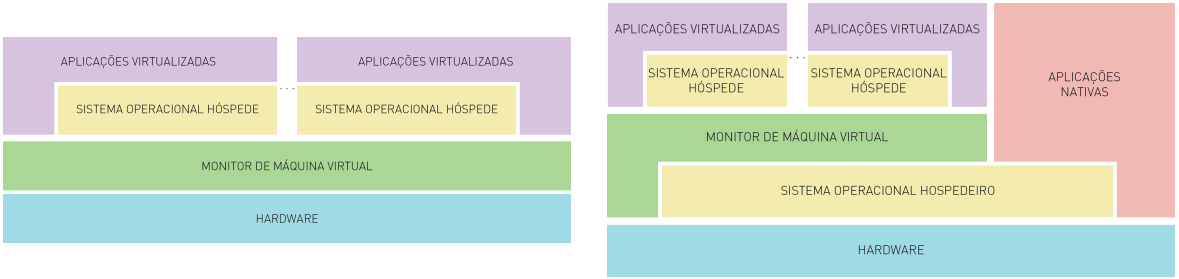
No contexto de máquinas virtuais de sistema, surge o conceito de monitor de máquina virtual, também chamado de hipervisor. O monitor de máquina virtual é responsável por dividir a utilização do hardware físico entre múltiplos sistemas hóspedes, através do mapeamento dos

recursos virtuais para recursos físicos. Além disso, o monitor de máquina virtual gerencia a execução das instruções dos sistemas hóspedes. Instruções sensíveis, que acessam recursos compartilhados, são interceptadas pelo hipervisor, verificadas e então executadas pelo próprio hipervisor, de maneira transparente ao sistema hóspede. Por fim, nos casos em que o conjunto de instruções utilizado pela máquina virtual é diferente do conjunto de instruções da plataforma física, é o hipervisor que executa a tradução dessas instruções.

Os monitores de máquinas virtuais são divididos em dois tipos que classificam a maneira que eles são implementados. Os monitores do tipo I, também chamados de *bare metal*, são instalados diretamente sobre o hardware, conforme ilustrado na Figura 2.1a. Dessa maneira, o hipervisor executa no mais alto nível de privilégio, podendo interceptar e implementar todas as ações dos sistemas hóspedes que interagem com recursos de hardware. Este tipo de implementação é mais eficiente, mas requer que os *drivers* dos dispositivos estejam disponíveis para o hipervisor e que o computador seja totalmente reconfigurado para que o monitor de máquina virtual seja a primeira camada de software acima do hardware.

Por outro lado, os monitores de máquinas virtuais do tipo II, também chamados de hospedados, executam aplicações virtualizadas sobre um sistema operacional hospedeiro, conforme mostra a Figura 2.1b. Assim, as máquinas virtuais executam de maneira similar a uma aplicação comum do sistema hospedeiro. Os hipervisores hospedados possuem a vantagem de poder utilizar *drivers* de dispositivos e outros serviços do sistema operacional hospedeiro, mas perdem eficiência por estarem em uma camada superior de software.

Figura 2.1 – Tipos de monitores de máquinas virtuais



(a) Monitor de máquina virtual de tipo I (b) Monitor de máquina virtual de tipo II

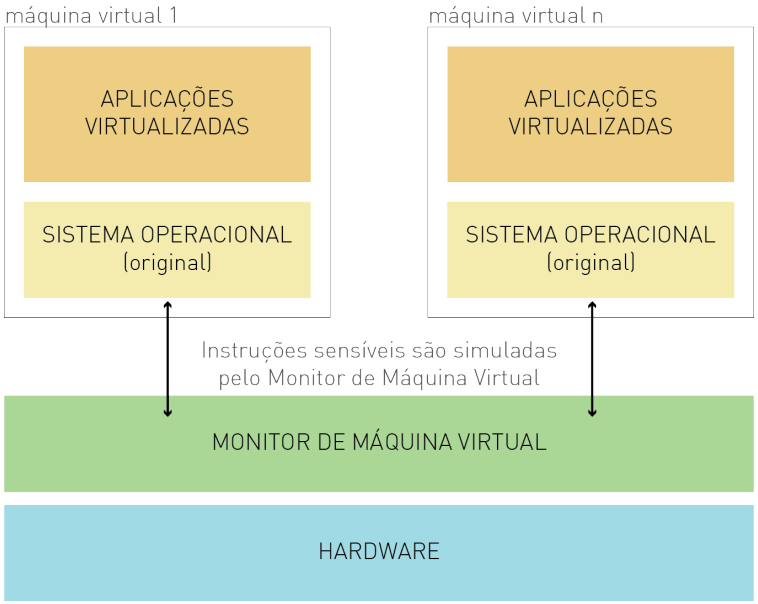
Fonte: Adaptado de (SMITH; NAIR, 2003)

2.1.2 Virtualização completa e paravirtualização

A implementação de um monitor de máquina virtual deve tratar de questões que estabelecem a maneira como seus serviços são oferecidos às suas máquinas virtuais e sistemas operacionais hóspedes. Nesse sentido, surgem os conceitos de virtualização total ou completa e paravirtualização.

Na virtualização total, ilustrada na Figura 2.2, o monitor de máquina virtual cria uma réplica virtual de uma máquina física, incluindo uma BIOS virtual, dispositivos virtuais e gerência de memória virtual. Dessa maneira, o sistema operacional hóspede e suas aplicações executam como se estivessem sobre um hardware físico. A grande vantagem dessa abordagem é que o sistema operacional não tem conhecimento de que está sendo virtualizado e, portanto, não requer modificação alguma. Isso, porém, também traz alguns inconvenientes.

Figura 2.2 – Virtualização total



Fonte: Adaptado de (CARISSIMI, 2008)

O primeiro problema é a complexidade existente em criar um ambiente virtual completo para o sistema operacional hóspede. Em relação aos dispositivos, existe uma variedade de placas e periféricos muito grande, tornando difícil o suporte do hipervisor para todos os dispositivos. Como solução, as implementações de virtualização total utilizam um conjunto de dispositivos genéricos que, apesar de funcionar bem na maior parte dos casos, não garante a utilização da capacidade total de um recurso de harware físico.

Ainda em relação a criação de um ambiente virtual, existem questões técnicas que devem ser contornadas para gerenciar a disputa de recursos entre diversos sistemas hóspedes, visto que os sistemas operacionais são implementados para serem executados como uma única instância em uma máquina física. Um exemplo disso é a gerência de memória virtual de sistemas que utilizam paginação. É necessário mapear o espaço de endereçamento de cada sistema hóspede para o espaço de endereçamento real, utilizado por todos os sistemas. Isso pode gerar uma queda no desempenho das máquinas virtuais.

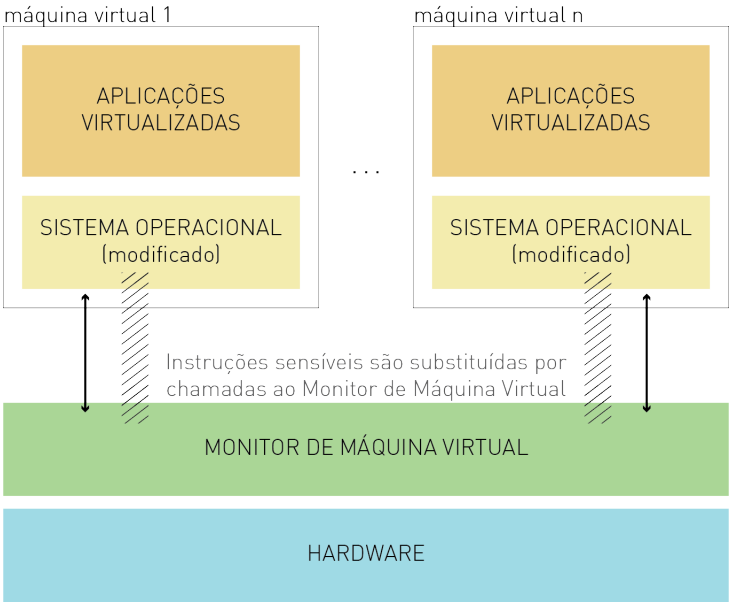
Outro problema existente na virtualização total é a execução de instruções sensíveis feitas pelo sistema hóspede. Certas instruções, como as instruções de entrada e saída, alteram ou dependem do estado de recursos do sistema. No caso das máquinas virtuais, os efeitos das instruções executadas por uma máquina virtual não podem alterar o estado de qualquer outra máquina virtual, aplicação ou hardware físico. Para que isto ocorra, a virtualização total utiliza uma técnica chamada de tradução binária, onde as instruções executadas pelo sistema operacional hóspede são testadas pelo monitor de máquina virtual para verificar se elas são sensíveis ou não. Caso a instrução seja sensível, ela é simulada pelo hipervisor para o sistema hóspede. Todo esse procedimento representa um custo de processamento.

A paravirtualização, ilustrada na Figura 2.3, surge como uma forma de resolver os problemas encontrados na virtualização total. Nessa abordagem, o sistema operacional hóspede é modificado para chamar o monitor de máquina virtual sempre que for executar uma instrução sensível. Assim, não é necessário que o hipervisor teste as instruções do sistema hóspede. Além disso, na paravirtualização os recursos de hardware são acessados por *drivers* de dispositivos da própria máquina virtual, eliminando a desvantagem da utilização de dispositivos virtuais genéricos.

Nota-se que, embora a paravirtualização explore melhor os recursos da máquina física e não adicione o custo de processamento da tradução binária, ela requer a modificação do sistema operacional hóspede para que ele passe a ter conhecimento sobre a existência da máquina virtual. Isso faz com que sua compatibilidade seja baixa, visto que não é possível modificar sistemas operacionais de código fechado, como o Microsoft Windows. Além disso, embora a maior vantagem da paravirtualização seja o ganho de desempenho, a presença de suporte à virtualização em processadores tem auxiliado na melhora de desempenho da virtualização total. O suporte de hardware para a virtualização é chamado de *Hardware Assisted Virtualization*. Em 2005 e 2006, os fabricantes de processadores Intel e AMD criaram extensões para a arquitetura x86 para darem suporte à virtualização. Estas soluções foram desenvolvidas de forma independente pelas duas fabricantes e são incompatíveis entre si, embora ambas

tenham o propósito de resolver os principais problemas da virtualização total, como a execução de instruções sensíveis pelo sistema operacional hóspede e a queda de desempenho causada pela gerência de memória virtual.

Figura 2.3 – Paravirtualização



Fonte: Adaptado de (CARISSIMI, 2008)

A solução da AMD, chamada de AMD *Virtualization* ou AMD-V, implementa funções no processador que são executadas pelo monitor de máquina virtual e permitem que ele controle se determinadas instruções de um sistema operacional hóspede são permitidas (CARISSIMI, 2008). A solução da Intel é chamada de *Intel Virtualization Technology*, e adiciona dois novos modos de operação do processador: *root* e não-*root*. O monitor de máquina virtual executa no modo *root*, enquanto as máquinas virtuais executam no modo não-*root*. Ao executar no modo não-*root*, instruções sensíveis executadas pelos sistemas operacionais hóspedes irão transferir o processador para o modo *root*, permitindo ao hipervisor executar essas instruções garantindo que não haverá risco para o sistema (UHLIG et al., 2005).

Por fim, ressalta-se que o suporte de hardware para virtualização deve ser explicitamente utilizado pelas ferramentas de virtualização. Como não são todos os processadores que possuem tal suporte, algumas ferramentas podem apresentar problemas de compatibilidade ou desempenho se utilizadas com certos processadores (CARISSIMI, 2008). Portanto, essas questões devem ser levadas em consideração no momento da escolha das ferramentas a serem utilizadas em um ambiente de virtualização.

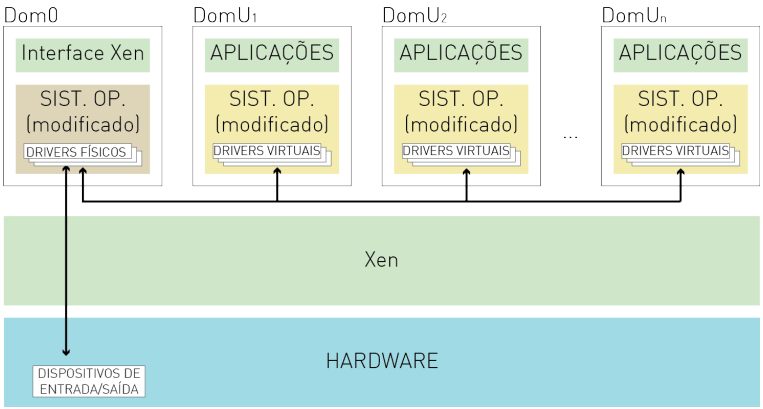
2.1.3 Ferramentas existentes

Devido a ampla utilização da virtualização nos mais diversos contextos, existem vários monitores de máquinas virtuais disponíveis no mercado. Esses monitores variam quanto ao tipo, emprego de virtualização total ou paravirtualização, arquiteturas alvo, licenciamento e custo. Esta seção apresentará alguns dos hipervisores mais utilizados atualmente.

Originado de um projeto de pesquisa da Universidade de Cambridge, o hipervisor Xen[1] surgiu em 2003 e suporta, atualmente, as arquiteturas x86, x86-64 e ARM. O Xen é um projeto de software livre, disponível sob a licença GNU *General Public Licence* (GPL) e atualmente mantido pela empresa Citrix. A Citrix suporta o desenvolvimento do projeto de código aberto e também vende soluções empresariais do software.

Além do conceito de monitor de máquina virtual, a arquitetura do Xen utiliza também a noção de domínios. O hipervisor executa diretamente sobre o hardware (tipo I) e é responsável por gerenciar processador, memória e interrupções. Ele não faz nenhuma gerência de entrada e saída e não possui *drivers* de dispositivos. Essa função é do domínio privilegiado, chamado de domínio 0. O domínio 0 é a primeira máquina virtual iniciada pelo sistema, executando um núcleo Linux modificado com os *drivers* da máquina física e *drivers* especiais para tratar os acessos à rede e ao disco feitos por todas as outras máquinas virtuais, de domínio não privilegiado (domínio U). O domínio 0 ainda possui uma aplicação que provê uma interface com o mundo externo, que permite a gerência das máquinas virtuais de domínio U. A Figura 2.4 ilustra a arquitetura de hipervisor e domínios utilizada pelo Xen.

Figura 2.4 – Arquitetura do hipervisor Xen



Fonte: Adaptado de (XEN, 2016)

[1] https://www.xenproject.org

Em suas primeiras versões, o Xen utilizava apenas a técnica de paravirtualização. Embora justificada por questões de desempenho, essa decisão, por requerer uma versão modificada do sistema, limitou o seu emprego a sistemas operacionais Unix, especialmente aqueles que possuem código aberto. A partir da terceira versão, o Xen passou a dar suporte à virtualização completa, permitindo o uso de sistemas operacionais não modificados, como o Microsoft Widows. Entretanto, o suporte à virtualização completa se restringe às máquinas com processadores que possuam suporte de hardware à virtualização.

Outro monitor de máquina virtual que possui código aberto, também distribuído sob a licença GNU GPL, é o KVM[2] (*Kernel-based Virtual Machine*). O KVM consiste em módulos Linux que convertem o núcleo do sistema operacional em um monitor de máquina virtual do tipo I, e está disponível para as arquiteturas x86 e x86-64. Seu desenvolvimento e manutenção são financiados pela *Open Virtualization Alliance*.

Normalmente, um processo pode executar em dois modos: núcleo e usuário. O modo usuário é o padrão para as aplicações, sendo que uma aplicação pode passar para o modo núcleo quando precisar de alguns serviços privilegiados, como acesso ao disco. O KVM adiciona um novo modo ao sistema, chamado de modo convidado. Os processos de modo convidado são aqueles que executam dentro de uma máquina virtual. O modo convidado possui seu próprio sistema operacional e ambiente de usuário. Entretanto, do ponto de vista do sistema hospedeiro, uma máquina virtual do KVM é vista como um processo normal.

O KVM implementa virtualização total e requer um processador com suporte de hardware à virtualização. A gerência de memória das máquinas virtuais é feita pelo próprio núcleo do sistema operacional hospedeiro e a gerência de entrada e saída é feita através do software de emulação QEMU, também de código livre.

Outro monitor de máquina virtual disponível para as arquiteturas x86 e x86-64 é o VirtualBox[3], da empresa Oracle. O núcleo do seu código é mantido sob a licença GNU GPL, mas certas funcionalidades utilizam uma licença proprietária. Ao contrário do Xen e do KVM, trata-se de um monitor do tipo II, e sua instalação está disponível para os sistemas Windows, macOS, Linux e Solaris. O uso pessoal é gratuito, enquanto licenças empresariais são pagas, bem como consultorias e assistências fornecidas pela Oracle.

O VirtualBox implementa virtualização total e não exige que o processador possua suporte de hardware para virtualização, embora essas funcionalidades são utilizadas quando

[2] http://www.linux-kvm.org
[3] https://www.virtualbox.org

disponíveis. Nos casos em que não há suporte de hardware, o hipervisor testa as instruções privilegiadas executadas pelas máquinas virtuais e substitui as que são sensíveis por versões seguras equivalentes. Além disso, um conjunto genérico de dispositivos é disponibilizado ao sistema operacional hóspede, e os discos rígidos são emulados por uma ferramenta própria do produto.

A VMware[4] é uma companhia subsidiária da Dell que oferece uma ampla gama de aplicações e serviços de virtualização para as arquiteturas x86-64, abrangendo desde *desktops* até *data centers*. Os produtos da VMWare utilizam virtualização total e não exigem suporte do processador à virtualização, embora essa funcionalidade seja utilizada quando disponível. Entre as aplicações oferecidas, destacam-se o VMWare *Workstation* e o VMWare ESXi, ambos de licenças proprietárias.

O VMWare *Workstation* é um hipervisor de tipo II voltado à criação de máquinas virtuais em *desktops*. A aplicação está disponível para Windows e Linux em duas versões: *Player* e *Pro*. A versão *Player* é gratuita para uso pessoal, enquanto a versão *Pro* é paga e oferece algumas funcionalidades a mais, como a possibilidade de salvar o estado de máquinas virtuais para posterior recuperação. O VMWare ESXi é um hipervisor de tipo I, pago e voltado à virtualização de *data centers*, permitindo que um servidor físico seja particionado em múltiplos servidores virtuais.

Por fim, a Microsoft também oferece sua solução de virtualização para a arquitetura x86-64, chamada de *Hyper-V*[5]. O *Hyper-V* é um monitor de máquina virtual de tipo I que implementa virtualização total e requer suporte do processador à virtualização. Sua licença é proprietária e ele é disponibilizado como parte do Windows Server e das versões mais avançadas do Windows 8 e Windows 10. Embora o *Hyper-V* possa ser utilizado em *desktops*, seu maior foco é a virtualização de *data centers*, sendo suportada a virtualização de versões mais recentes do Windows e algumas distribuições Linux.

A arquitetura do *Hyper-V* isola diferentes máquinas virtuais utilizando o conceito de partição raiz e partição filha. Uma partição é uma unidade lógica de isolamento, suportada pelo hipervisor, onde um sistema operacional executa. O hipervisor possui ao menos uma partição raiz executando, responsável por diversas funções de virtualização, como disponibilizar *drivers* de dispositivos para as máquinas virtuais das partições filhas e gerenciar o ciclo de vida dessas. As partições filhas não tem acesso direto aos recursos de hardware, sendo esses controlados

[4] https://www.vmware.com
[5] https://www.microsoft.com/en-us/cloud-platform/virtualization

pelo hipervisor ou então direcionados aos *drivers* da partição raiz. Nota-se que, conceitualmente, a arquitetura de partições do *Hyper-V* é semelhante à arquitetura de domínios utilizada pelo Xen.

2.2 Computação em nuvem

A computação em nuvem pode ser descrita como um modelo que fornece poder computacional como um serviço público, de maneira parecida aos serviços de energia elétrica e telefonia (CARISSIMI, 2015). Ela surgiu através da evolução e adoção de diversas tecnologias e paradigmas existentes, oferecendo, através da Internet, acesso a um conjunto compartilhado de recursos computacionais que podem ser configurados. Esses recursos incluem redes, servidores, armazenamento, aplicações e serviços.

Entre as tecnologias que permitiram o surgimento da computação em nuvem destaca-se a virtualização. De fato, (CARISSIMI, 2015) define a computação em nuvem como "a utilização massiva da virtualização para a criação de um modelo de negócio". O modelo de negócio mencionado é o "pague o quanto usa" (*pay-as-you-go*). Nesse modelo, o usuário paga apenas pelos recursos que forem consumidos. Dessa maneira, em um momento de alta demanda, o usuário pode contratar mais recursos para evitar a degradação do desempenho de seus serviços. Quando a demanda for normalizada, os recursos adicionais contratados podem ser liberados, e o usuário pagará apenas pelo tempo que eles foram utilizados. Essa capacidade de alocar e liberar recursos conforme a necessidade é chamada de elasticidade.

De fato, a elasticidade é possível justamente pela utilização da virtualização pelos provedores de serviços em nuvens. Em um ambiente virtualizado, torna-se fácil instanciar e migrar máquinas virtuais conforme a solicitação de clientes. Além disso, a portabilidade e flexibilidade proporcionadas pela virtualização permite que várias aplicações de diferentes sistemas operacionais executem sobre uma mesma máquina física. Dessa forma, o provedor compartilha e aluga seus recursos a diversos usuários, amortizando os custos de manutenção e aquisição de sua larga infraestrutura. Por fim, a redução dos custos com estruturas físicas e energia elétrica proporcionadas pela virtualização vão ao encontro do conceito de computação verde.

Outra característica importante da computação em nuvem é que o fornecimento dos serviços é feito pelo modelo *self-service*, permitindo que o usuário contrate e gerencie seus recursos através da Internet, via web, sem a necessidade de intervenção do provedor do serviço (CARISSIMI, 2015). Dessa maneira, o usuário apenas contrata e configura os recursos que

deseja, mas não precisa conhecer os detalhes de como eles são implementados internamente. Esses detalhes são abstraídos pelo provedor, permitindo ao cliente preocupar-se apenas com os detalhes de seu negócio. Essa simplicidade de uso descrita torna a computação em nuvem ainda mais acessível.

Por trás dessa simplicidade enxergada pelos clientes da nuvem, há uma complexa infraestrutura mantida pelo provedor dos serviços. O Instituto Nacional de Padrões e Tecnologia dos EUA (*National Institute of Standards and Technology* – NIST) define a infraestrutura de nuvem como uma coleção de hardware e software que contém uma camada física e uma camada de abstração. A camada física consiste nos recursos de hardware que são necessários para sustentar os serviços oferecidos pela nuvem. A camada de abstração consiste em aplicações que executam na camada física e que apresentam as características essenciais da nuvem. Conceitualmente, a camada de abstração situa-se acima da camada física.

2.2.1 Tipos de nuvem

O NIST classifica as nuvens em quatro modelos de implementação distintos: pública, privativa, comunitária e híbrida. Essa classificação define o propósito e a natureza da nuvem, e cada tipo distingue-se dos demais por características como posse, tamanho e disponibilidade de acesso à nuvem. A descrição de cada modelo de implementação a seguir é baseada em (MELL; GRANCE, 2011).

As nuvens públicas são aquelas em que os recursos oferecidos estão disponíveis para uso do público em geral. Esse modelo de implementação segue o modelo de negócio tradicional da computação em nuvem, onde o usuário contrata os recursos conforme necessário e paga apenas pelos recursos utilizados. Assim, os serviços e a infraestrutura do provedor de serviços atendem vários clientes. É o modelo adequado para negócios que desejam contratar recursos computacionais como serviços, evitando gastos com infraestrutura. Alguns exemplos de provedores de computação em nuvem que seguem esse modelo de implantação são a Amazon *Web Services*, Microsof *Azure*, Google e Dropbox.

Por outro lado, nas nuvens privativas a infraestrutura é utilizada para prover serviços a uma única organização. A posse e gerenciamento da infraestrutura podem ser terceirizados, mas não há mais o modelo de negócio *pay-as-you-go* na nuvem privativa, visto que a organização deve arcar com o custo total da infraestrutura, incluindo aquisição de máquinas, energia elétrica e manutenção. O acesso a uma nuvem privativa só pode ser feito por usuários autorizados. Esse modelo de implantação pode ser útil para organizações que possuam requisitos críticos de

privacidade e proteção de seus dados, quando verificado que os atributos de segurança das nuvens públicas não são suficientes para seus negócios. Soluções de nuvens privativas são oferecidas por empresas como Microsoft, IBM e HP.

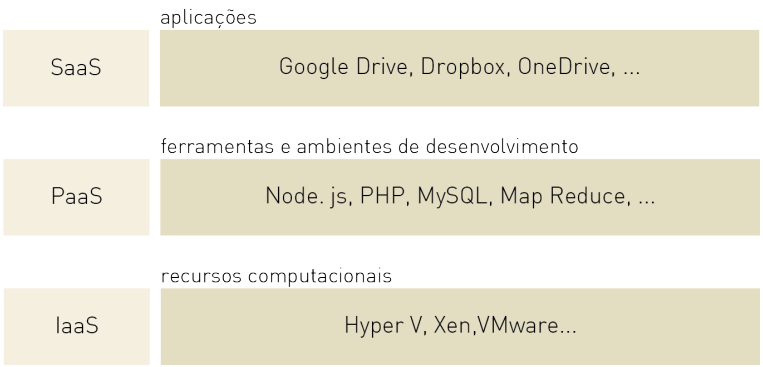
Já as nuvens comunitárias dividem uma infraestrutura física entre diversas organizações que geralmente possuem requisitos de privacidade, desempenho e segurança parecidos. A posse e gerenciamento da infraestrutura pode ser de uma ou mais organizações da comunidade ou terceirizados, sendo os custos divididos entre todos os membros. Esse tipo de nuvem pode, por exemplo, ser utilizado por governos de países para prover uma infraestrutura computacional aos seus diversos órgãos.

Por fim, as nuvens híbridas são a composição de duas ou mais infraestruturas de nuvens de modelos distintos que continuam existindo como entidades únicas, mas são interconectadas por tecnologias que permitem a portabilidade de aplicações e dados entre elas. Esse tipo de nuvem pode ser adequado para a redução de custos de certas organizações, mantendo-se os dados e aplicações mais críticos em uma nuvem privada que atende a requisitos de segurança mais rígidos, e as demais informações e serviços em uma nuvem pública.

2.2.2 Serviços de nuvem

O NIST define também três diferentes modelos de serviços de nuvem: software como serviço (*Software as a Service* – SaaS), plataforma como serviço (*Plataform as a Service* – PaaS) e infraestrutura como serviço (*Infrastructure as a Service* – IaaS). Esses modelos são classificados pelo nível de abstração dos serviços oferecidos pela nuvem e podem ser representados como camadas de uma pilha, como ilustrado na Figura 2.5. A seguir, cada modelo de serviço será descrito com base em (MELL; GRANCE, 2011).

Figura 2.5 – Arquitetura em camadas dos serviços de nuvem



Fonte: Adaptado de (CARISSIMI, 2015)

No modelo SaaS, os serviços oferecidos ao usuário são aplicações que executam sobre uma infraestrutura de nuvem. Essas aplicações estão prontas para serem utilizadas, e tipicamente são oferecidas mediante pagamento de mensalidades ou anuidades. Não há necessidade que o usuário adquira, configure e mantenha a aplicação contratada. Além disso, os serviços geralmente são disponibilizados através de navegadores web, não exigindo a instalação de qualquer aplicação extra no computador do usuário.

Nesse modelo, o usuário não controla ou gerencia os recursos da infraestrutura da nuvem como rede, servidores, sistema operacional e armazenamento. O controle é limitado a algumas configurações específicas do software contratado. Dessa forma, os serviços oferecidos pelo modelo SaaS destinam-se ao usuário final. São exemplos desses serviços o Google *Drive*, Dropbox, Microsoft *OneDrive* e *Gmail*.

No PaaS, a infraestrutura de nuvem fornece ao usuário um ambiente de desenvolvimento com compiladores de linguagens de programação, bibliotecas, depuradores, banco de dados e servidores. Esse modelo destina-se, portanto, aos desenvolvedores de software, que podem desenvolver e executar suas aplicações sem o custo e a complexidade de comprar e gerenciar uma infraestrutura de hardware e ambiente de software.

É importante ressaltar que as plataformas fornecidas podem oferecer ferramentas específicas para o desenvolvimento com certas tecnologias, não sendo um modelo genérico que possa ser configurado de acordo com as tecnologias utilizadas pelos usuários. Do mesmo modo que no software como serviço, o usuário também não controla ou gerencia a rede, os servidores, o sistema operacional e o armazenamento da infraestrutura de nuvem. Entretanto, é possível configurar as aplicações fornecidas para desenvolvimento e até mesmo utilizar a plataforma para hospedar as aplicações desenvolvidas. Alguns serviços do Microsoft *Azure* e o Google *App Engine* são exemplos de PaaS.

Por fim, o IaaS oferece ao consumidor um sistema computacional composto por processadores, memória, armazenamento e qualquer outro recurso computacional, permitindo ao usuário instalar e executar seu sistema operacional e quaisquer aplicações que sejam necessárias. O consumidor não controla a infraestrutura da nuvem, mas pode configurar livremente a infraestrutura contratada. Tipicamente, essa infraestrutura é alocada através de uma máquina virtual no ambiente do provedor de nuvem.

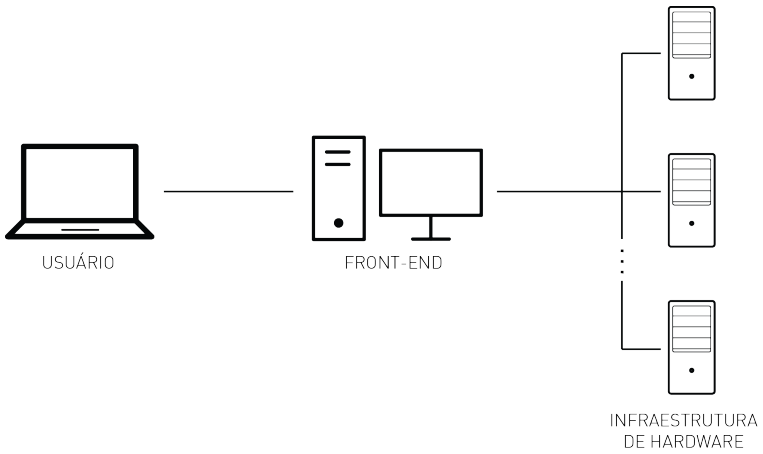
Por estar em um nível de abstração mais baixo, a infraestrutura como serviço destina-se a equipes de tecnologia da informação, que podem configurar ambientes de desenvolvimento e produção de acordo com as tecnologias utilizadas. A tarifação desse serviço é feita pela quantidade de recursos destinados ao cliente durante um certo período de tempo, não sendo

considerado se o recurso está sendo realmente utilizado. São exemplos de provedores IaaS a Amazon *Elastic Compute Cloud* (EC2), o Google *Compute Engine* e alguns serviços do Microsoft *Azure*.

2.2.3 Gerenciadores de nuvem

Tipicamente, uma solução em nuvem possui três atores envolvidos: o usuário, um portal de acesso (*front-end*) e a infraestrutura de hardware. A relação entre esses atores está representada na Figura 2.6.

Figura 2.6 – Arquitetura de uma solução em nuvem



O usuário utiliza a interface disponibilizada pelo portal de acesso para contratar e gerenciar recursos na nuvem. Essa interface geralmente é uma página web acessada via navegador, mas, em alguns casos, também pode ser uma interface de linha de comando acessada por um terminal.

O *front-end* é a parte do provedor de nuvem que faz o intermédio entre o usuário e a infraestrutura de hardware. Ele é um servidor que executa um gerenciador de nuvem. O gerenciador de nuvem é a ferramenta responsável pela gerência da infraestrutura existente, monitorando constantemente o estado dos recursos físicos. É ele quem aloca, destrói e distribui os recursos virtuais entre os diversos recursos físicos, executando também operações como migração de máquinas virtuais e balanceamento de carga. É também o gerenciador quem disponibiliza a interface de acesso aos recursos da nuvem ao usuário.

Por fim, a infraestrutura de hardware consiste em todos os recursos físicos disponibilizados para alocação dos recursos virtuais pelo gerenciador de nuvem. Sobre o

hardware, existe uma camada de software que é utilizada pelo gerenciador para controlar os recursos alocados. No caso da utilização de máquinas virtuais, essa camada de software é um monitor de máquina virtual.

2.2.4 Serviços de nuvem e gerenciadores existentes

O crescente interesse por soluções de nuvem provocou o surgimento de diversos serviços e ferramentas que proveem algum modelo de serviço de nuvem. Esta seção apresentará alguns dos serviços mais utilizados atualmente e os gerenciadores de nuvem de código aberto mais conhecidos.

Entre as soluções voltadas ao usuário final, oferecendo o modelo de software como serviço, destacam-se alguns produtos do Google, da Microsoft e o Dropbox. O Google oferece uma série de serviços que podem ser acessadas via navegador. Alguns exemplos são o *Drive*, que permite o armazenamento, compartilhamento e edição de arquivos na nuvem; *Docs*, que permite a edição de arquivos de texto, planilhas e apresentações diretamente no navegador; *YouTube*, que oferece um serviço de hospedagem e reprodução de vídeos por *streaming*; e *Maps*, que disponibiliza mapas do mundo inteiro com serviços como consultas de trajetos, trânsito e transporte público. A Microsoft, por sua vez, também oferece um serviço de armazenamento de arquivos na nuvem, chamado de *OneDrive*. Há ainda o *Office* 365, que oferece a linha de produtos do Microsoft *Office* como um serviço pago mensalmente ou anualmente, integrado ao *OneDrive*, e com a possibilidade de criar e editar arquivos do *Office* diretamente do navegador. Por fim, o Dropbox, que é outro serviço de armazenamento e compartilhamento de arquivos em nuvem.

Voltado aos desenvolvedores de softwares, o Google *App Engine* é um exemplo de plataforma como serviço que oferece um ambiente de desenvolvimento e hospedagem de aplicações web. Atualmente, sua versão estável oferece suporte às linguagens Java, Python, Go e PHP, juntamente com diversas bibliotecas utilizadas pelos desenvolvedores dessas linguagens. O serviço é gratuito até certo nível de consumo de recursos, sendo cobrado adicionalmente por mais armazenamento, largura de banda e horas de execução. O *App Engine* oferece ainda serviços de elasticidade, alocando automaticamente mais recursos para aplicação nos momentos em que a demanda for maior.

Seguindo o modelo de infraestrutura como serviço, destacam-se o Amazon *Web Services* (AWS) e o Microsoft *Azure*. A AWS oferece mais de 70 serviços de nuvem dentre os quais se destacam o Amazon *Elastic Compute Cloud* (EC2) e o Amazon *Simple Storage Service*

(S3). O EC2 é um serviço que oferece ao usuário a possibilidade de instanciar máquinas virtuais com imagens pré-definidas ou definir completamente sua configuração de sistema. Os sistemas operacionais disponíveis incluem diversas distribuições Linux, OpenSolaris, FreeBSD e versões do Windows Server, sendo ainda possível escolher diferentes processadores e quantidade de memória e de armazenamento. O serviço é elástico, permitindo que mais recursos sejam alocados em momentos de maior demanda. O EC2 utiliza o Xen como ferramenta de virtualização, e a cobrança é feita pelo tempo de utilização dos recursos alocados, não levando em consideração se eles foram utilizados ou não.

O S3 é um serviço de armazenamento destinado a grandes objetos de dados. Esses objetos podem possuir até 5 *terabytes* e são armazenados em *buckets*, conceito utilizado pela Amazon para identificar o espaço de armazenamento de cada usuário. Tanto *buckets* quanto objetos de dados podem ser criados, listados e baixados utilizando APIs REST e SOAP e o protocolo BitTorrent. A cobrança é feita de acordo com o espaço contratado pelo usuário.

O Microsoft Azure também oferece dezenas de serviços de nuvens, dentre os quais destacam-se o *Virtual Machines* e o *Blob Storage*. O *Virtual Machines* é um serviço semelhante ao Amazon EC2, e permite ao usuário configurar uma máquina virtual com diferentes configurações de hardware e alguma distribuição Linux ou versão do Windows. A cobrança é feita sobre os recursos utilizados por minuto. O *Blob Storage* é semelhante ao Amazon S3, oferecendo armazenamento para grandes arquivos na nuvem com desempenho, alta disponibilidade e segurança. Os arquivos do *Blob Storage* podem ser gerenciados através de uma API REST e de bibliotecas disponibilizadas pela Microsoft para diversas linguagens, facilitando a integração com os produtos de desenvolvedores clientes do serviço. A cobrança do *Blob Storage* é feita de acordo com o espaço utilizado pelo usuário.

Todos os serviços citados até então implementam o modelo de nuvem pública. Entretanto, certas organizações possuem o interesse em manter uma nuvem privativa, por motivos que podem envolver requisitos críticos de privacidade e proteção de seus dados ou até mesmo custos. A estrutura física de uma nuvem privativa é semelhante ao de uma nuvem pública, e para gerenciar os recursos físicos e virtuais e oferecer os serviços em nuvem é necessário que exista um gerenciador de nuvem. Nesse contexto, podem ser empregados diversos gerenciadores de código aberto que oferecem um modelo de serviço IaaS, como o Eucalyptus, OpenStack, CloudStack e OpenNebula.

O Eucalyptus[6] é uma plataforma que permite a criação de nuvens híbridas e privadas compatíveis com o AWS e pertence, desde 2008, à Hewlett-Packard (HP). Os usuários podem mover instâncias de máquinas virtuais entre uma nuvem privada do Eucalyptus e a Amazon EC2 para criar uma nuvem híbrida.

A arquitetura do Eucalyptus é dividida em componentes, e cada máquina física é um nó que executa máquinas virtuais utilizando os hipervisores Xen, KVM ou VMware. Cada nó executa o *node controller*, que monitora e envia informações sobre seu nó a um *cluster controller*. Cada *cluster controller* é responsável por gerenciar a execução das máquinas virtuais pertencentes aos seus nós. Por fim, o *cloud controller* gerencia todos os *cluster controllers* e oferece acesso aos recursos da nuvem aos usuários, através de uma interface de linha de comando e de uma interface web. Existem ainda dois componentes para armazenamento de dados, chamados *storage controller* e *Walrus*. Enquanto o *Walrus* oferece armazenamento persistente para as imagens de máquinas virtuais do Eucalyptus e qualquer outro tipo de dado, o *storage controller* atua no nível do *cluster controller* e do *node controller* para gerenciar os discos virtuais das máquinas virtuais em execução.

O OpenStack[7] é uma plataforma de computação em nuvem distribuída sob a licença Apache 2.0. Ele surgiu de uma iniciativa da NASA com a empresa Rackspace e é atualmente mantido pela OpenStack *Foundation*, uma organização sem fins lucrativos apoiada por diversas companhias. A plataforma possibilita a implantação tanto de nuvens públicas como de nuvens privadas.

A arquitetura do OpenStack é modular e composta de diversos serviços que podem ser utilizados conforme a necessidade da nuvem a ser criada. O *Nova* é o módulo responsável pelo ciclo de vida das máquinas virtuais e suporta diversos hipervisores como Xen, KVM e VMware. O módulo *Neutron* é responsável pela conectividade de rede da plataforma, permitindo a definição de SDNs (*Software Defined Networks*) e VPNs (*Virtual Private Networks*), além de outros serviços de rede. Para armazenamento utiliza-se o *Swift*, que oferece a persistência de objetos dados não estruturados que podem ser acessados por uma API REST; o *Cinder*, que armazena os discos virtuais das instâncias de máquinas virtuais em execução; e o *Glance*, que persiste imagens de máquinas virtuais utilizadas para instanciar novas máquinas. Completa os serviços considerados essênciais o módulo *Keystone*, que oferece autenticação e autorização para todos os demais serviços do OpenStack. Por fim, a administração de uma nuvem

[6] http://www.eucalyptus.com
[7] http://www.openstack.org

OpenStack é feita através de uma aplicação web chamada *Dashboard*, ou através de linhas de comando.

Outra plataforma distribuída sob a licença Apache 2.0 é a CloudStack[8], mantida pela Apache *Software Foundation*, uma organização sem fins lucrativos que mantém diversos softwares de código aberto. O CloudStack suporta diversos hipervisores comerciais da Citrix e do VMWare, além de KVM e Microsoft *Hyper-V*. Pode-se utilizar a plataforma para criar nuvens públicas, privativas e híbridas, sendo que nesse último caso é possível integrar com os serviços EC2 e S3 da AWS.

A arquitetura do CloudStack permite desde a configuração de uma nuvem com infraestrutura extremamente básica até a gerência de recursos em uma infraestrutura geograficamente distribuída. Em ambos os casos, a nuvem é gerenciada por um servidor chamado de *Management Server*, que controla a alocação de recursos e os ciclos de vida das máquinas virtuais, a distribuição de endereços de IP, a gerência de armazenamento das imagens de máquinas e discos virtuais e fornece uma interface web para os administradores e usuários da plataforma.

Por fim, o OpenNebula[9] é outra plataforma de computação em nuvem que oferece a possibilidade de implementação de nuvens públicas, privativas e híbridas, também distribuída sob a licença Apache 2.0. A plataforma suporta os hipervisores KVM, Xen, VMware e vCenter, permitindo ainda a integração com o serviço EC2 da AWS.

O OpenNebula foi projetado para ser de simples instalação, configuração e manutenção, não sendo necessária a integração de diversos módulos para prover os serviços. A arquitetura da plataforma é dividida em *hosts*, servidores de armazenamento e um servidor *front-end*. Um *host* é uma máquina física que executa máquinas virtuais e é conectada ao *front-end*. Uma única nuvem do OpenNebula pode trabalhar com *hosts* que utilizam diferentes hipervisores. Os servidores de armazenamento são divididos em três classes: de sistema, que armazena as imagens de máquinas virtuais em execução; de imagens, que persiste repositórios de imagens de disco para instanciar novas máquinas virtuais; e de arquivos, que armazena qualquer outro tipo de arquivo. Nas infraestruturas mais simples, os servidores de armazenamento podem executar na mesma máquina do *front-end*. Por fim, o *front-end* é o servidor responsável por gerenciar e monitorar todos os *hosts* e suas máquinas virtuais, os *clusters* em que os *hosts* podem estar organizados, as configurações de rede e as configurações de autenticação. O *front-*

[8] http://cloudstack.apache.org
[9] http://opennebula.org

end ainda fornece uma interface de linha de comando e uma interface web, chamada de *Sunstone*, para os administradores e usuários da plataforma acessarem os serviços oferecidos.

2.3 Considerações finais

A computação em nuvem trouxe um modelo de computação cada vez mais presente em nosso cotidiano, utilizando diversas tecnologias já existentes para fornecer a computação como um serviço que é cobrado de maneira proporcional aos recursos utilizados. Entre as diversas tecnologias utilizadas, o grande alicerce da computação em nuvem é a virtualização, que além de prover alta flexibilidade e portabilidade, permitindo a alocação e liberação de recursos conforme necessário, ainda promove uma utilização eficiente dos recursos de hardware.

A utilização do hardware de forma eficiente faz parte de uma preocupação central das infraestruturas de nuvem: o consumo de energia elétrica. De fato, o uso de servidores consolidados e a utilização de recursos sob demanda contribuem com a redução do consumo e, por consequência, com a computação verde, que visa trazer as práticas de sustentabilidade para a computação. Ainda assim, os grandes *data centers* utilizados pelos provedores de computação em nuvem consomem grandes quantidades de energia elétrica, não apenas com a infraestrutura de hardware, mas também com toda a estrutura de refrigeração necessária. Nesse sentido, este trabalho propõe, no próximo capítulo, uma solução para uma infraestrutura de nuvem de baixo custo e de baixo consumo de energia elétrica.

3 PROPOSTA E IMPLEMENTAÇÃO

Dado o crescente interesse pelas soluções da computação em nuvem e a necessidade de alcançarmos uma computação sustentável, este capítulo apresenta a proposta e a implementação deste trabalho, que trata da implementação de uma infraestrutura de nuvem que provê o modelo IaaS.

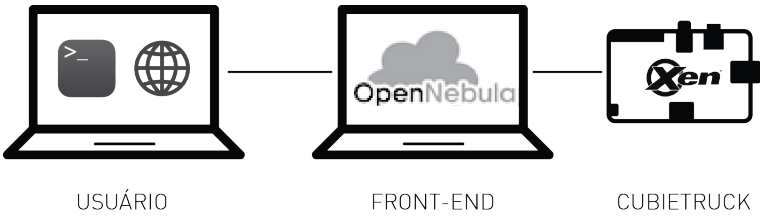
Para atingir o objetivo, foram estabelecidos dois requisitos de projeto. O primeiro requisito determina que deve ser utilizado hardware com processador de baixo consumo de energia elétrica na infraestrutura de nuvem implementada. O segundo requisito estabelece que o sistema implementado deve utilizar apenas soluções de software livre.

As seções deste capítulo apresentarão a arquitetura do sistema implementado, com as tecnologias empregadas em cada parte da solução, e irão descrever as instalações e configurações necessárias para o correto funcionamento do sistema. Os detalhes com todos os passos necessários para a configuração do sistema estão listados no Apêndice deste trabalho.

3.1 Arquitetura do sistema

A arquitetura do sistema implementado segue a arquitetura apresentada na Figura 2.6, com a seleção de ferramentas de software e hardware para executar as tarefas de cada ator. A Figura 3.1 ilustra a arquitetura final da solução desenvolvida com as tecnologias utilizadas.

Figura 3.1 – Arquitetura da solução implementada



Do lado do usuário, é necessário que haja um navegador web e uma interface de linha de comando com um programa de acesso remoto via SSH (*Secure Shell*), independente do sistema operacional e hardware utilizados. O navegador é utilizado para acessar a interface web fornecida pelo *front-end* que permite instanciar e gerenciar máquinas virtuais, enquanto a interface de linha de comando é utilizada para acessar remotamente as máquinas virtuais instanciadas.

O *front-end* executa o gerenciador de nuvem OpenNebula, que monitora e gerencia a infraestrutura de hardware, os serviços de armazenamento de arquivos e imagens de máquinas virtuais, os modelos de máquinas virtuais que podem ser instanciadas, as configurações de rede das máquinas virtuais e as permissões de usuários e grupos para acesso aos serviços da nuvem. Além disso, o OpenNebula fornece o *Sunstone*, um sistema web onde administradores e usuários da nuvem podem ter acesso aos serviços fornecidos. A máquina que executa os serviços do *front-end* é um computador pessoal de arquitetura Intel x86, executando o sistema operacional GNU/Linux de distribuição Ubuntu 14.04.

Por fim, a infraestrutura de hardware é composta pela Cubietruck, um mini computador de placa única, de baixo consumo, e de arquitetura ARM Cortex-A7. Para a execução e gerência de máquinas virtuais, utiliza-se o Xen como monitor de máquina virtual. O objetivo final é a utilização de um *cluster* com vários computadores Cubietruck, mas este trabalho utiliza apenas um como prova de conceito.

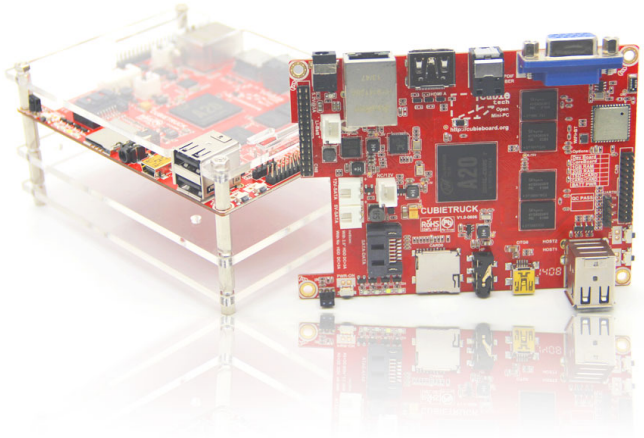
3.2 Cubietruck

Atualmente, o mercado dispõe de uma variedade de minicomputadores de placas únicas que utilizam processadores ARM. O público alvo desses dispositivos é principalmente desenvolvedores, estudantes e entusiastas da Internet das Coisas, que utilizam esses computadores integrados a sensores e atuadores para automatizar certas tarefas do cotidiano. A escolha sobre qual dispositivo específico utilizar pode considerar vários fatores, como a facilidade de uso, qualidade da documentação disponível, capacidade de hardware e compatibilidade com software existente. Neste trabalho, foi utilizado o Cubietruck por estar disponível no Grupo de Processamento Paralelo e Distribuído (GPPD) do Instituto de Informática da UFRGS.

O Cubietruck é a terceira geração da série de mini computadores Cubieboard, um projeto de hardware livre desenvolvido pela companhia chinesa Cubietech. O processamento é feito pelo chip Allwinner A20, que inclui um processador ARM Cortex-A7 de 1GHz, dois núcleos, e uma unidade de processamento gráfico ARM Mali400 MP2. A memória RAM disponível é de 2GB DDR3, a uma frequência de 480MHz. Em termos de armazenamento, existe uma memória NAND de 8GB, além de entradas para cartão microSD e SATA 2.0. Para entrada e saída, o Cubietruck dispõe de duas portas USB e uma Mini USB, saídas de vídeo VGA e HDMI, uma porta RJ45 para Ethernet Gigabit, Wi-Fi, Bluetooth e dois barramentos de pinos com diversas interfaces de comunicação, como UART e PS2. Por fim, a alimentação é

feita por uma fonte de 5 volts e 2 amperes, e o sistema operacional disponível de fábrica é o Android 4.2.2. A Figura 3.2 ilustra um computador Cubietruck.

Figura 3.2 – Um mini computador Cubietruck



Fonte: (CUBIETECH, 2016)

A primeira tentativa de inicialização feita pelo Cubietruck é através da verificação de existência de um *bootloader* na entrada de cartão microSD. Nesse sentido, utilizou-se o U-Boot, um *bootloader* de código aberto distribuído sob a licença GNU GPL e destinado a dispositivos embarcados. A instalação do U-Boot se deu através de um processo de compilação cruzada, onde o compilador gera código executável para uma plataforma diferente daquela em que o compilador está sendo executado. Para essa tarefa, foi utilizado um pacote adicional ao GNU C *Compiler* (GCC), um compilador de código aberto. O mesmo processo de compilação cruzada foi utilizado para gerar os executáveis do monitor de máquina virtual e do núcleo do sistema operacional utilizados.

Após o processo de compilação, o binário do U-Boot é transferido para o setor de inicialização do cartão microSD, já formatado no formato *ext4*. Além disso, o *bootloader* consome ainda dois arquivos. O primeiro é um *device tree binary* (DTB), um arquivo com uma estrutura de dados que descreve o hardware do dispositivo sendo utilizado. Esse arquivo é gerado juntamente com a compilação do núcleo do sistema operacional. O segundo é uma imagem com os comandos que o U-Boot deve executar após ser inicializado. Isto inclui a configuração de certas variáveis, como a raiz do sistema de arquivos, além dos arquivos que

devem ser carregados para a memória e seus endereços e o núcleo do sistema operacional a ser inicializado. Os arquivos carregados para o sistema desenvolvido são os binários do monitor de máquina virtual e do núcleo do sistema operacional.

3.3 Xen

A primeira camada de software executada após o carregamento pelo U-Boot é o monitor de máquina virtual Xen, em sua versão 4.4. A opção por esse hipervisor justifica-se pelo amplo esforço e comprometimento de sua comunidade em manter uma versão específica para a arquitetura ARM, além atender ao requisito de utilização de soluções de software livre.

A instalação do Xen também foi feita pelo processo de compilação cruzada. Não são necessárias alterações e configurações adicionais, visto que existe compatibilidade nativa para a arquitetura ARM. Após a compilação, o arquivo binário gerado é transferido para o cartão microSD, e seu carregamento e inicialização fica sob a responsabilidade do *bootloader*.

3.3.1 Domínio 0

Conforme mencionado na seção 2.1.3, a arquitetura do Xen possui dois tipos de domínios: o domínio 0, que executa uma única máquina virtual em modo privilegiado e provê *drivers* e uma interface para gerência das demais máquinas virtuais; e os domínios U, onde executam todas as máquinas virtuais instanciadas via domínio 0 e não possuem acesso privilegiado.

Em ambos os domínios, o sistema operacional utilizado foi o GNU/Linux, pois além de ser uma opção de software livre, ele oferece compatibilidade com o Xen, segurança e confiabilidade. A compilação do núcleo Linux requer um arquivo de configurações, que lista todas as funcionalidades a serem incluídas no binário a ser gerado. Para facilitar esse processo, uma série de arquivos com configurações pré-definidas para diversos ambientes são fornecidos. Neste trabalho, foi utilizado como base um arquivo de configuração padrão para a arquitetura ARM do Cubietruck, e foram adicionadas as configurações que habilitam a compatibilidade do sistema operacional com o Xen. Novamente, o binário gerado é transferido para o cartão microSD, e é o *bootloader* quem carrega e inicializa o sistema operacional.

A distribuição do sistema operacional utilizada no domínio 0 foi a Ubuntu 14.04, por ser compatível com a versão do hipervisor e do gerenciador de nuvem utilizados. A instalação do Ubuntu foi feita utilizando a ferramenta *debootstrap*, que permite instalar a base de um

sistema Debian ou Ubuntu em um subdiretório de outro sistema já instalado. Nesse caso, a instalação foi feita no diretório que apontava para o cartão microSD. Após a instalação, é necessário ainda configurar a tabela de sistemas de arquivos e o arquivo de interfaces de rede do sistema operacional.

Além das ferramentas e programas disponibilizados pela distribuição utilizada, é necessário instalar no sistema do domínio 0 uma ferramenta que fornece acesso ao núcleo do hipervisor, permitindo gerenciar e instanciar máquinas virtuais. Essa ferramenta é chamada *xen-utils-4.4* e é disponibilizada pelo próprio Xen, podendo ser obtida utilizando-se o programa *apt-get*, o gerenciador de pacotes para Linux padrão da distribuição Ubuntu.

3.3.2 Domínios U

Para instanciar máquinas virtuais a partir do domínio 0, o Xen consome um arquivo com os parâmetros da máquina virtual a ser criada. Esse arquivo contém configurações como o núcleo do sistema operacional e o sistema de arquivos a serem utilizados, a quantidade de memória RAM e de processadores virtuais a serem dedicados, as interfaces de rede e o terminal a serem atribuídos à máquina virtual.

Neste trabalho, utilizou-se o mesmo núcleo de sistema operacional utilizado no domínio 0 nas máquinas virtuais de domínio U. A interface de rede foi criada utilizando-se *bridges*, que permite que uma interface física de rede seja compartilhada entre múltiplas interfaces virtuais. Esse procedimento é feito através da adição de uma interface de *bridge* ao arquivo de interfaces de rede do sistema operacional do domínio 0. No arquivo de configuração do domínio U, é estabelecido que essa interface de *bridge* deve ser utilizada.

Como distribuição do GNU/Linux das máquinas virtuais, optou-se por utilizar uma das imagens disponibilizadas pelo Linaro. O Linaro é uma organização formada por diversas companhias que visa prover software livre e gratuito para os processadores da família ARM (LINARO, 2016). Entre as ferramentas disponibilizadas, incluem-se uma série de imagens de disco baseadas na distribuição Ubuntu para a arquitetura ARM. Foi utilizada a versão Linaro *Developer* para as máquinas virtuais, por ser uma imagem mais enxuta e que inclui diversas ferramentas para desenvolvedores. Essa versão não inclui, por exemplo, suporte à interface gráfica, sendo possível apenas utilizar o modo texto. Nesse sentido, conforme ressaltado na seção 2.2.2, destaca-se que os modelos de nuvem de IaaS são destinados a equipes de tecnologias da informação, composta por usuários avançados e familiarizados com a interface de linha de comando.

A forma como os discos das máquinas virtuais são disponibilizados ao Xen também deve ser especificada. É possível utilizar tanto o modo físico, quando o disco da máquina virtual está disponível em algum volume do sistema; quanto o modo emulado, quando a imagem de disco está nos formatos *qcow*, *qcow2* ou *vhd*, e o Xen recorre ao software QEMU (*Quick Emulator*) para emular o disco da máquina virtual. Enquanto o modo emulado oferece mais flexibilidade para cópias e restaurações de estado, o modo físico possui melhor desempenho por executar diretamente sobre o disco físico.

O modo utilizado no sistema desenvolvido foi o físico, através da utilização da ferramenta LVM (*Logical Volume Manager*), um mapeador de dispositivos que oferece gerenciamento de volumes lógicos para o núcleo Linux. Desta maneira, utilizou-se uma partição do cartão microSD para a criação de um grupo de volumes do LVM e, nesse grupo, são inseridos os volumes lógicos com a imagem do sistema de arquivos da máquina virtual.

Por fim, os parâmetros de memória e de processador foram utilizados de maneira variável entre diferentes instâncias de máquinas virtuais, a fim de avaliar o desempenho com diferentes configurações de memória e processador.

3.4 OpenNebula

Dentre as opções de gerenciadores de nuvem de código aberto disponíveis, o sistema implementado utilizou o OpenNebula, na versão 4.14, por oferecer compatibilidade com o hipervisor Xen e com o hardware ARM utilizado.

A arquitetura do OpenNebula, conforme explicitado na seção 2.2.4, é dividida em *hosts*, servidores de armazenamento e um servidor *front-end.* Na solução implementada, o *host* é o Cubietruck, e os servidores de armazenamento e *front-end* foram configurados em uma única máquina por tratar-se de uma infraestrutura pequena.

Para executar os servidores de armazenamento e *front-end*, a máquina utilizada foi um computador pessoal de arquitetura Intel x86, com 4GB de RAM e processador Core i5 de 2,4 GHz com 4 núcleos. O sistema operacional utilizado foi o GNU/Linux, em sua distribuição Ubuntu 14.04.

No *front-end*, a instalação do OpenNebula foi feita utilizando-se o gerenciador de pacotes *apt-get*. Dois pacotes foram instalados, um chamado *opennebula* e outro chamado *opennebula-sunstone*. O primeiro é responsável pelas funcionalidades do gerenciador de nuvem, enquanto o segundo instala o Sunstone, um sistema web que fornece uma interface gráfica para acesso aos serviços da nuvem. Uma vez instalados, é necessário habilitar o suporte

ao Xen. Embora esta opção esteja desabilitada por padrão, habilitá-la requer uma simples edição em um arquivo de configuração do OpenNebula.

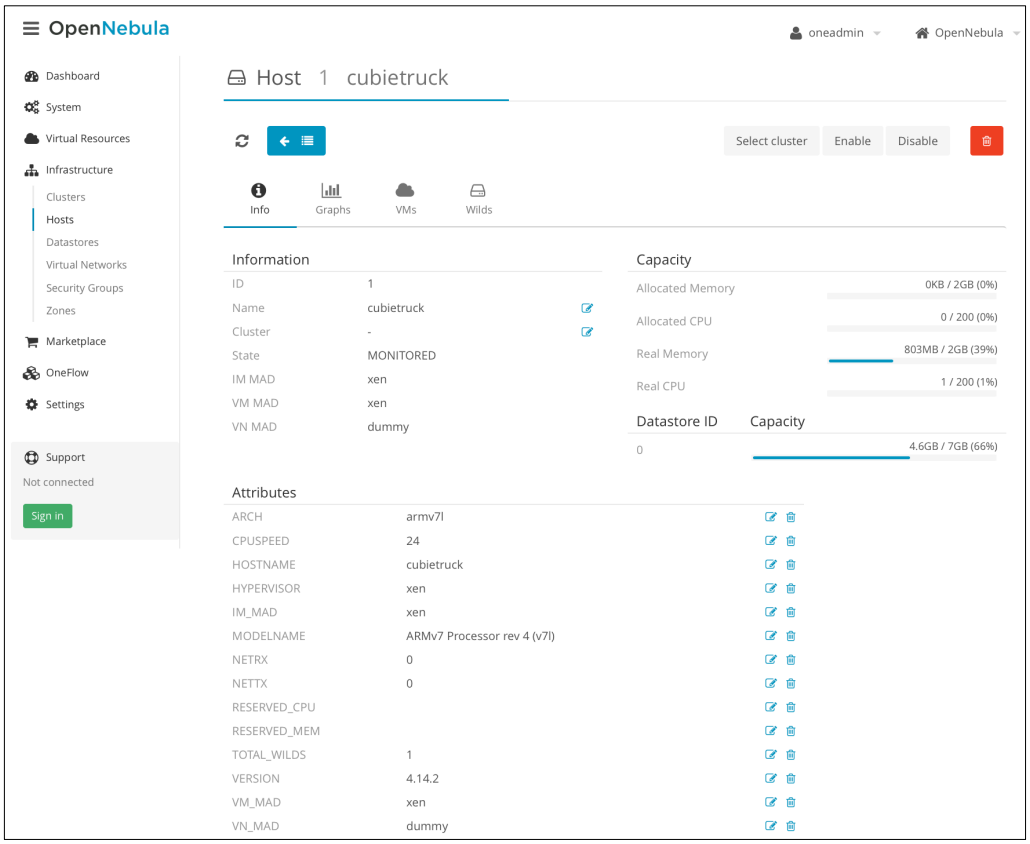
A instalação do OpenNebula cria um usuário no sistema operacional chamado *oneadmin*. Esse usuário provê acesso e é responsável por todas as tarefas do gerenciador de nuvem, como monitoramento e gerência dos *hosts* e dos servidores de armazenamento. Nesse sentido, é necessário que todos os *hosts* possuam um usuário com o mesmo nome e que permita acesso remoto via SSH sem exigência de senha ao *front-end*, permitindo o monitoramento e envio de tarefas.

Para permitir acesso do *front-end* ao Cubietruck, um usuário chamado *oneadmin* foi adicionado ao sistema operacional do domínio 0, com os mesmos identificadores de usuário e grupo do *oneadmin* do *front-end*. Além disso, é necessário adicionar as chaves de acesso SSH do usuário do *front-end* à lista de chaves autorizadas a acessar o usuário no Cubietruck sem necessidade de inserir uma senha. Por fim, deve-se permitir que o *oneadmin* no Cubietruck execute comandos privilegiados sem a necessidade de senhas, permitindo acesso aos comandos do Xen e do LVM.

Uma vez que ambos os lados já estejam configurados para a utilização do OpenNebula e estejam conectados na mesma rede, é possível adicionar o Cubietruck como *host* do gerenciador de nuvem. Para isso, basta informar ao *front-end* o endereço IP e o monitor de máquina virtual utilizado pelo *host*. A Figura 3.3 exibe a página do Sunstone com as informações do *host* Cubietruck após sua adição à nuvem.

A instalação do OpenNebula cria, por padrão, dois servidores de armazenamento: um de sistema, que armazena as imagens de máquinas virtuais em execução; e outro de imagens, responsável por armazenar imagens de discos para instanciar novas máquinas virtuais. Como este trabalho utiliza o LVM para prover discos para as máquinas virtuais, é necessário criar um novo servidor de armazenamento de imagens do tipo LVM. Esses servidores de armazenamento LVM, por questões de desempenho, transferem as imagens para os *hosts* no momento em que elas são adicionadas à nuvem, mantendo apenas um ponteiro para sua localização em cada *host*. Quando uma máquina virtual é instanciada, a imagem utilizada é replicada no *host* para uso exclusivo daquela máquina virtual, e o servidor de armazenamento de sistema manterá um ponteiro para essa versão replicada.

Figura 3.3 – *Host* Cubietruck adicionado à nuvem do OpenNebula



Após a configuração dos servidores de armazenamento, é possível adicionar imagens de discos a serem utilizadas pela máquina virtual. Nesse caso, foi adicionada a imagem da Linaro mencionada na seção anterior.

O último passo para permitir que sejam instanciadas máquinas virtuais é criar um modelo de máquina virtual. Esse modelo é responsável por definir uma máquina, configurando a quantidade de processadores e de memória, a imagem de disco e o núcleo do sistema operacional a serem utilizados e configurações de rede a serem aplicadas. No sistema implementado, foi criado um modelo com 100 cotas de processador e 512MB em cotas de memória RAM, utilizando a imagem de disco adicionada ao servidor de armazenamento LVM. Ressalta-se que as configurações de cotas de processador e memória RAM são definidas apenas com valores sugeridos, e podem ser facilmente alteradas no momento de instanciar uma máquina virtual a partir de um modelo.

É importante destacar o modelo de cotas utilizado pelo OpenNebula para alocar processadores e memória RAM para as máquinas virtuais. Cada *host* possui cem cotas para cada núcleo de processador disponível, e uma cota para cada *megabyte* de memória RAM. No caso do Cubietruck, estão disponíveis duzentas cotas de processador e 2GB de cotas de

memória RAM. O somatório das cotas utilizadas pelas máquinas virtuais em execução em um *host* não pode ser maior que o total de cotas disponíveis naquele *host*, tanto no caso de processador quanto no caso de memória RAM.

A partir de um modelo, é possível instanciar máquinas virtuais através do Sunstone de maneira bastante fácil. Para isso, basta selecionar o modelo a ser utilizado, nomear a máquina virtual e, se desejado, alterar alguns parâmetros como número de processadores e quantidade de memória RAM. Uma vez criada, deve-se aguardar que o Sunstone informe que a máquina virtual foi instanciada com sucesso. A partir disso, basta conectar-se remotamente utilizando o protocolo SSH para utilizar a máquina virtual. As Figuras 3.4 e 3.5 mostram, respetivamente, as informações de uma máquina virtual instanciada no Sunstone e um terminal conectado remotamente a esta máquina virtual.

Figura 3.4 – Máquina virtual instanciada no OpenNebula

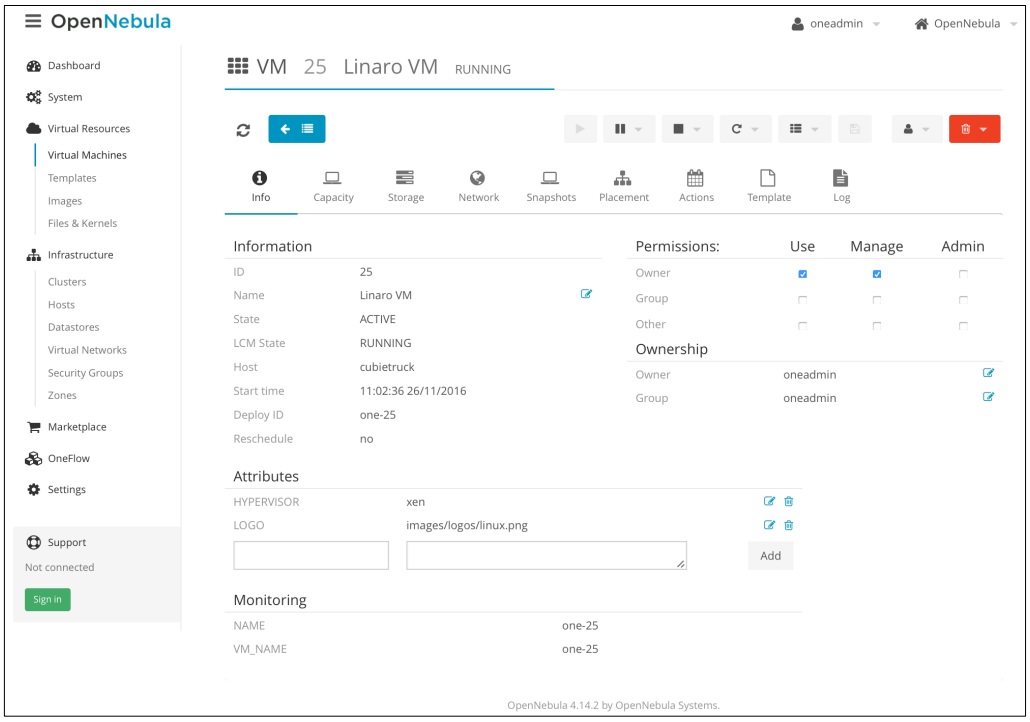
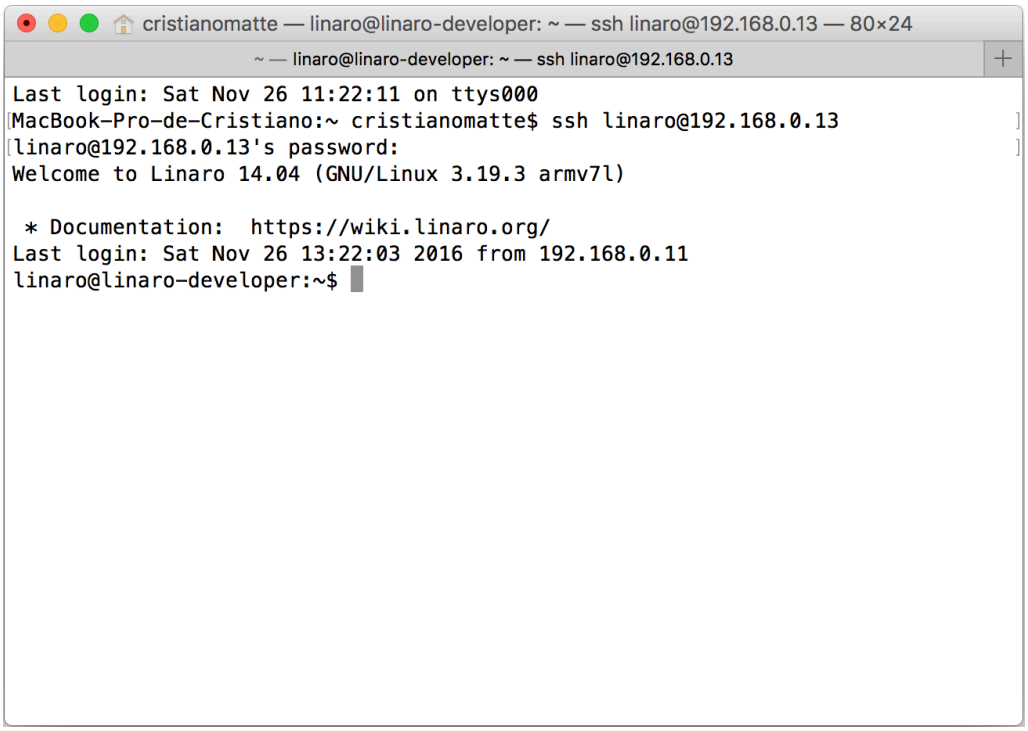


Figura 3.5 – Acesso remoto a uma máquina virtual instanciada



3.5 Dificuldades encontradas

Ao longo da implementação do sistema descrito neste capítulo, algumas dificuldades foram encontradas.

O primeiro problema encontrado foi em relação a falta de sinal de vídeo HDMI da Cubietruck após a instalação do Linux. De fato, segundo a comunidade linux-sunxi[10], o suporte ao HDMI só é possível se for habilitada a utilização de dispositivos *framebuffer* no arquivo de configurações utilizado para compilar o sistema operacional. Após adicionar essa configuração e recompilar o núcleo Linux, a conexão de vídeo HDMI passou a funcionar.

Outra dificuldade encontrada foi em relação a utilização de imagens de disco emuladas para instanciar máquinas virtuais. Conforme mencionado anteriormente, o Xen recorre ao emulador QEMU para emular imagens de disco. A execução do QEMU no Cubietruck, entretanto, apresentou problemas relacionados a falta de um dispositivo de vídeo presente. Mesmo considerando o fato de que o Xen não depende da emulação de vídeo feita pelo QEMU e adicionando parâmetros ao emulador para não utilizar dispositivo de vídeo, não houve sucesso

[10] A linux-sunxi é uma comunidade de software livre dedicada a prover suporte a sistemas operacionais de código aberto para dispositivos com os *chips* da Allwinner (LINUX-SUNXI, 2016).

nas tentativas de utilizar imagens de disco emuladas. Este problema foi contornado através da utilização de imagens de disco físicas com o LVM.

Por fim, a adição do Cubietruck como *host* do OpenNebula também apresentou problemas. A configuração automática do *host*, feita através da instalação do pacote *opennebula-node,* cria o diretório do usuário *oneadmin* utilizando NFS (*Network File System*), um sistema de arquivos remoto. Desse modo, o diretório do *oneadmin* de todos os *hosts* aponta para o diretório do *oneadmin* do *front-end.* Embora o Cubietruck pudesse ser adicionado como *host* com sucesso dessa maneira, a transferência de arquivos como imagens de discos do *front-end* para os *hosts* apresentou diversas falhas, sem entretanto exibir mensagens de erro significativas. A resolução desse problema se deu através da configuração manual do OpenNebula do *host*, criando o usuário *oneadmin* sem a utilização do NFS.

3.6 Considerações finais

Com base na proposta deste trabalho, este capítulo apresentou o projeto e a implementação de uma solução de infraestrutura de nuvem utilizando uma arquitetura ARM de baixo consumo de energia elétrica, ressaltando tanto o hardware empregado, como o Cubietruck, quanto as ferramentas utilizadas, como o monitor de máquinas virtuais Xen e o gerenciador de nuvem OpenNebula.

A partir da implementação, o próximo capítulo apresenta a avaliação do sistema desenvolvido em termos de desempenho, permitindo verificar a viabilidade de sua utilização para a solução de problemas reais.

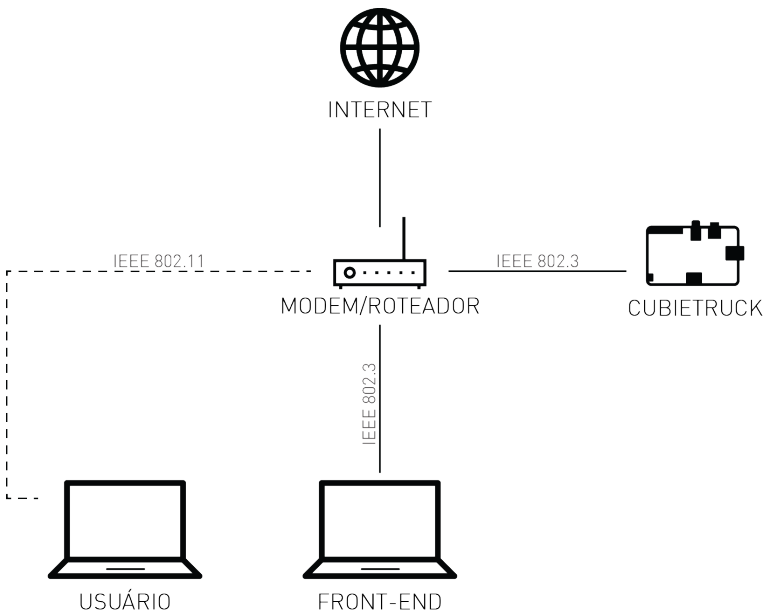
4 AVALIAÇÃO EXPERIMENTAL

Após a implementação da solução proposta, é necessário avaliar o desempenho da solução em nuvem e das máquinas virtuais instanciadas. Nesse sentido, este capítulo apresenta o ambiente experimental e a metodologia utilizados nos experimentos, seguido dos resultados de cada experimento realizado e uma discussão sobre os valores obtidos.

4.1 Ambiente experimental

Os experimentos foram realizados em uma rede LAN (*Local Area Network*) residencial, com todos os componentes do sistema conectados a mesma rede através das interfaces locais RJ45 e Wi-Fi de um roteador. Utilizou-se um equipamento diferente para cada ator de uma solução em nuvem: Um Cubietruck como *host*; um notebook como *front-end*; e um segundo notebook como usuário, utilizado para acessar o gerenciador de nuvem e as máquinas virtuais instanciadas através dele. A Figura 4.1 ilustra o ambiente experimental utilizado na validação.

Figura 4.1 – Ambiente experimental de avaliação



O Cubietruck, conforme descrito na seção 3.3, executa o hipervisor Xen e possui, no domínio privilegiado, o sistema operacional GNU/Linux de distribuição Ubuntu 14.04 e núcleo modificado para prover compatibilidade com o Xen. O processador possui dois núcleos de 1GHz e a memória RAM disponível é de 2GB, sendo 512MB consumidos pelo domínio

privilegiado. As máquinas virtuais executam o sistema operacional de distribuição Linaro *Developer* com o mesmo núcleo do domínio privilegiado, configuradas com um disco de 1GB. As configurações de processador e memória variaram para cada avaliação.

O *front-end* é um computador de arquitetura x86 de 64 bits, com processador Intel Core i5-2430M de 4 núcleos de 2,40GHz e memória de 4GB, executando a distribuição Ubuntu 14.04 do sistema operacional GNU/Linux.

Por fim, no lado do usuário, foi utilizado o navegador Safari 10.0 para acessar o gerenciador de nuvem através da interface Sunstone e o terminal padrão do sistema operacional macOS 10.12 para acesso remoto às máquinas virtuais acessadas.

4.2 Metodologia

Foram conduzidos três diferentes tipos de experimentos para validar a solução implementada, a fim de avaliar o desempenho em diferentes momentos do ciclo de vida das máquinas virtuais e em diferentes aplicações em que elas podem ser empregadas. Esses experimentos são:

- Tempo de instanciação das máquinas virtuais, através da medição do tempo entre a criação de uma máquina virtual no gerenciador de nuvem e a indicação de que essa máquina está pronta para ser utilizada.
- Execução de dois *benchmarks* clássicos, chamados *drhystone* e *whetstone*, que avaliam, respectivamente, o desempenho das máquinas virtuais na execução de operações com inteiros e ponto flutuante.
- Execução de *benchmark* de servidores web executando nas máquinas virtuais, avaliando o desempenho de entrada e saída de dados.

Em cada experimento, as medições foram repetidas dez vezes e foi obtido a média dos resultados da amostra. Os experimentos executados em cenários com mais de uma máquina virtual, com exceção da máquina responsável por executar a avaliação, mantiveram todas as máquinas virtuais ociosas, exceto onde for relatado o contrário.

Para efeitos de comparação, os experimentos de *benchmarks* também foram realizados no computador utilizado no *front-end* e em um Cubietruck sem hipervisor, utilizando apenas a distribuição Ubuntu 14.04 do sistema operacional GNU/Linux.

4.3 Tempo de instanciação de máquinas virtuais

O tempo para instanciar máquinas virtuais foi avaliado em cinco diferentes contextos: instanciação da primeira, segunda, terceira, quarta e quinta máquina virtual. Isso permite verificar se há alguma variação no tempo total quando a carga de processamento do gerenciador de nuvem e do Cubietruck é maior

À cada máquina virtual, foram atribuídas 40 cotas de processador e 256MB de cotas de memória RAM. Quando a soma de cotas de processador ou memória atribuídas às máquinas virtuais instanciadas atingirem o valor máximo suportado pelo Cubietruck (200 cotas de processador e 2GB de cotas de memória), o OpenNebula bloqueia a instanciação de novas máquinas virtuais.

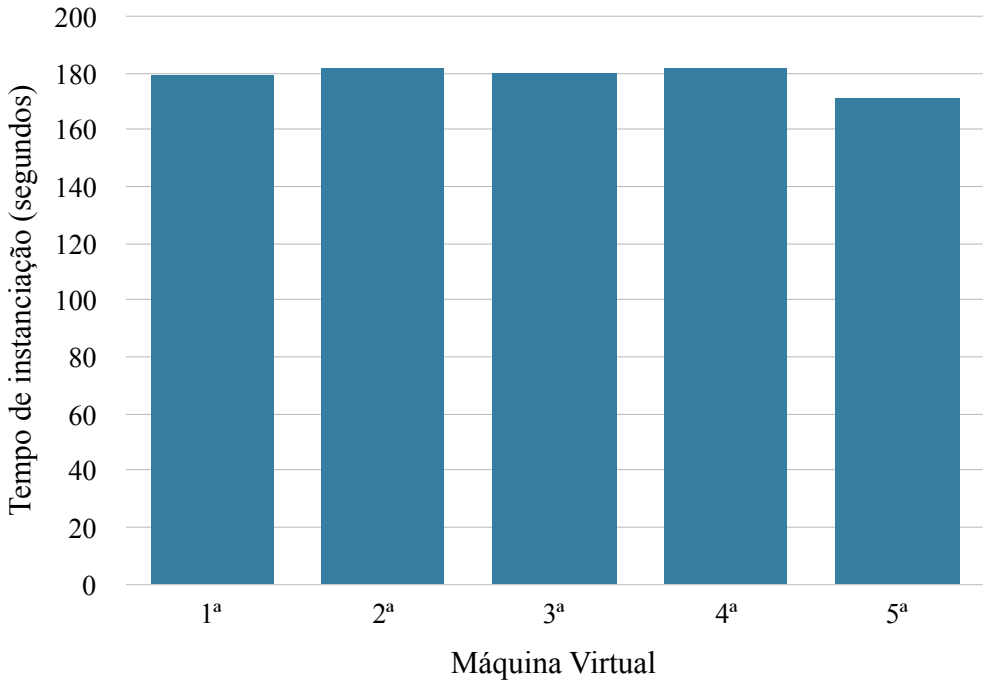
O cálculo considerou o tempo entre o comando para o OpenNebula para que a máquina virtual fosse instanciada e a resposta do gerenciador de nuvem informando que a máquina estava pronta para uso. A Tabela 4.1 exibe os resultados obtidos neste experimento.

Tabela 4.1 – Tempo de instanciação de máquinas virtuais

Máquina virtual	*Tempo de instanciação (segundos)*	*Desvio padrão*	*Intervalo de confiança (95% de confiança)*
1ª	179	7,20	[174,54 , 183,46]
2ª	182	6,70	[177,85 , 186,15]
3ª	180	8,46	[174,76 , 185,24]
4ª	182	8,76	[176,57 , 187,43]
5ª	171	7,80	[166,17 , 175,83]

O tempo necessário para instanciar uma máquina virtual é relativamente alto, justificado pela necessidade de se fazer uma cópia da imagem do disco exclusiva para uso da instância sendo criada. Percebe-se, entretanto, que não há interferência da existência de máquinas virtuais já em execução no tempo necessário para instanciar novas máquinas virtuais. A Figura 4.2 ilustra, através de um gráfico, a manutenção do tempo praticamente constante ao instanciar novas máquinas virtuais.

Figura 4.2 - Tempo de instanciação de máquinas virtuais



Além de instanciar novas máquinas virtuais, é possível desligar e ligar as existentes. O desligamento mantém a máquina no OpenNebula e a imagem de disco criada, mas libera memória e processador do Cubietruck para que ele possa executar novas máquinas. Neste caso, conforme demonstrado pela Tabela 4.2, o tempo necessário para religar uma máquina é muito pequeno, e também permanece praticamente constante independente do número de máquinas instanciadas. Os valores demonstram que, de fato, é a criação da imagem de disco feita pelo OpenNebula no Cubietruck que ocupa a maior parcela do tempo de instanciação de novas máquinas.

Tabela 4.2 – Tempo de religamento de máquinas virtuais

Máquinas virtuais instanciadas	*Tempo para religar uma máquina virtual (segundos)*	*Desvio padrão*	*Intervalo de confiança (95% de confiança)*
Uma	3	0,48	[2,71 , 3,29]
Duas	2	0,51	[1,68 , 2,32]
Três	2	0,42	[1,74 , 2,26]
Quatro	2	0	[2, 2]
Cinco	2	0,42	[1,74 , 2,26]

4.4 Desempenho em *benchmarks* clássicos

Foram executados dois *benchmarks* clássicos que avaliam o desempenho de processamento das máquinas virtuais: *dhrystone* e *whetstone*. Para fins de comparação, essas avaliações foram também executadas no Cubietruck sem o Xen e no computador utilizado como *front-end*. Nos cenários com máquinas virtuais, 50 cotas de processador e 512MB de cotas de memória RAM foram atribuídas a cada máquina, e apenas uma executou o *benchmark*, enquanto as outras permaneceram ociosas.

O *dhrystone* é um programa que executa diversas instruções aritméticas de números inteiros, como chamadas de funções, indireções de ponteiros, atribuições e laços de execução. Esse tipo de instrução é bastante comum em aplicações como compiladores, navegadores e editores de texto. Sua versão original é de 1984, mas este trabalho executou a segunda versão, que corrige problemas causados por otimizações excessivas aplicadas pelos compiladores. A Tabela 4.3 exibe os resultados expressos em VAX MIPS, um padrão de desempenho bastante utilizado em *benchmarks*.

Tabela 4.3 – Resultados do *benchmark dhrystone*

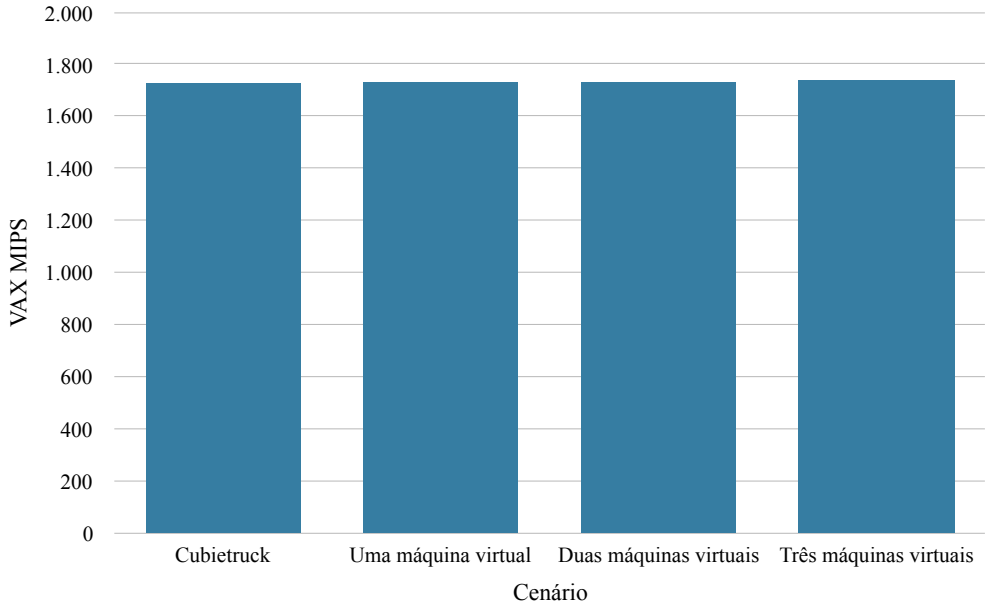
Cenário	*VAX MIPS*	*Desvio padrão*	*Intervalo de confiança (95% de confiança)*
Front-end	18.130,47	315,72	[17.934,79 , 18.326,16]
Cubietruck	1.727,33	2,94	[1.725,50 , 1.729,15]
Uma máquina virtual	1.730,30	7,70	[1.725,52 , 1.735,08]
Duas máquinas virtuais	1.731,55	7,54	[1.726,87 , 1.736,22]
Três máquinas virtuais	1.737,68	3,43	[1.735,55 , 1.739,81]

Percebe-se que o desempenho do computador utilizado no *front-end* é bastante superior aos demais cenários. Este resultado é esperado, visto que o processador do Cubietruck é destinado a sistemas embarcados e voltado ao baixo consumo de energia elétrica, enquanto o *front-end* possui um processador de arquitetura x86, destinado a computadores pessoais e bastante voltado a prover um bom desempenho.

Entretanto, a comparação entre os *benchmarks* executados no Cubietruck mostra que os resultados permanecem praticamente constantes entre diferentes cenários. O fato da avaliação estar sendo executada em uma máquina virtual, mesmo quando há duas máquinas virtuais instanciadas, não alterou o desempenho de processamento.

Os resultados da execução do *dhrystone* estão representados no gráfico da Figura 4.3.

Figura 4.3 – Comparação entre diferentes cenários de execução do *dhrystone*



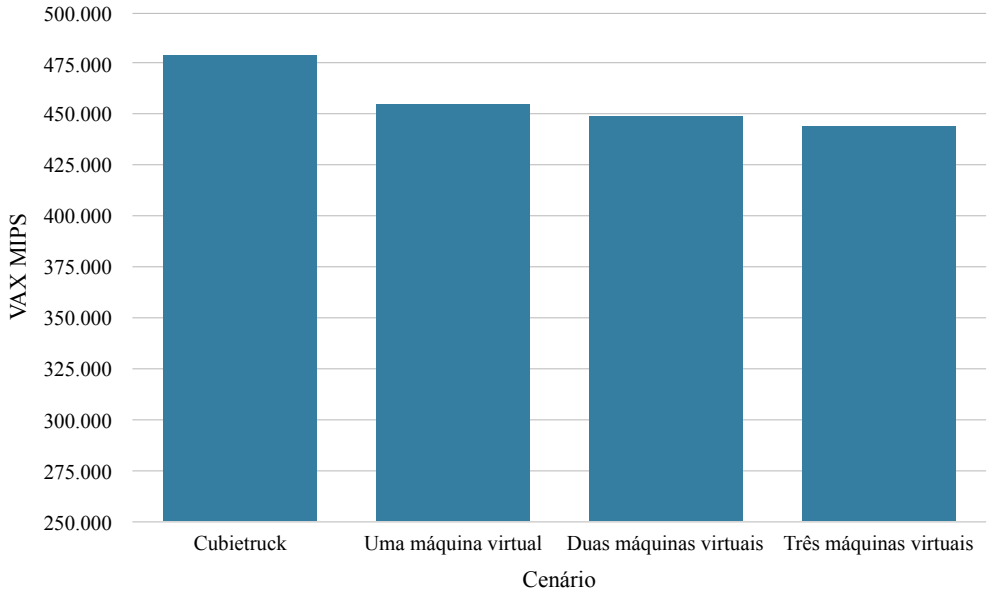
O *whetstone*, diferentemente do *dhrystone*, é um programa que executa instruções aritméticas de números de ponto flutuante. Essas instruções são comuns em aplicações científicas, como cálculos matemáticos e simulações físicas, além de amplamente utilizadas em processamento gráfico. Os resultados obtidos estão na Tabela 4.4 e estão expressos em MWIPS (*Million Whetstone Instructions Per Second* – Milhões de Instruções Whetstone Por Segundo).

Tabela 4.4 – Resultados do *benchmark whetstone*

Cenário	*MWIPS*	*Desvio padrão*	*Intervalo de confiança (95% de confiança)*
Front-end	3.677.224	60.594,24	[3.639.668 , 3.714.780]
Cubietruck	479.515	1.195,13	[478.774 , 480.256]
Uma máquina virtual	455.285	5.315,11	[451.991 , 458.579]
Duas máquinas virtuais	448.793	8.974,28	[443.230 , 454.355]
Três máquinas virtuais	444.028	2.545,05	[442.451 , 445.606]

Novamente, o desempenho do computador utilizado para *front-end* foi bastante superior aos cenários que utilizaram o Cubietruck, pelos mesmos motivos apresentados na avaliação do *dhrystone*. Nesse *benchmark*, entretanto, o desempenho obtido nas máquinas virtuais foi inferior a execução sobre um sistema nativo no Cubietruck. A Figura 4.4 exibe um gráfico demonstrando essa avaliação.

Figura 4.4 – Comparação entre diferentes cenários de execução do *whetstone*



4.5 Desempenho de servidor web

Para avaliar o desempenho de entrada e saída, foi utilizado um servidor web simples, fornecido como um módulo da linguagem Python, que consiste apenas na disponibilidade de um arquivo de 2MB. Foi utilizado o software *Siege*, dedicado a testes e *benchmarks* de servidores web, para disparar dez requisições de dez usuários simultâneos para obtenção do arquivo, totalizando cem requisições por execução da avaliação. Ressalta-se que, no contexto do *Siege*, o termo disponibilidade é utilizado para expressar a porcentagem de requisições que foram respondidas com sucesso pelo servidor.

Essa avaliação foi executada em cinco diferentes cenários: no computador utilizado no *front-end* e em um Cubietruck sem hipervisor, para permitir comparação com as avaliações em máquinas virtuais; na solução de nuvem implementada, com uma e com duas máquinas virtuais instanciadas, onde apenas uma máquina executou o *benchmark* e a outra permaneceu ociosa; e na solução de nuvem implementada, com duas máquinas virtuais instanciadas e executando, simultaneamente, o *benchmark Siege*. Nos cenários que envolvem máquinas virtuais, foram atribuídas 100 cotas de processador e 512MB de cotas de memória RAM a cada máquina. A Tabela 4.5 mostra os resultados obtidos nessa avaliação.

Tabela 4.5 – Resultados dos *benchmarks* executados pelo *Siege*

Cenário	*Benchmark*	*Resultado*	*Desvio padrão*	*Intervalo de confiança (95% de confiança)*
Front-end	Disponibilidade	98,6%	0,54	[98,26 , 98,93]
	Tempo de resposta	11,27 segundos	0,94	[10,68 , 11,85]
	Taxa de transações	0,68 transações/segundo	0,03	[0,66 , 0,70]
	Taxa de transferência	1,37 MB/segundo	0,05	[1,33 , 1,40]
Cubietruck	Disponibilidade	99,4%	0,89	[98,84 , 99,95]
	Tempo de resposta	12,86 segundos	1,02	[12,23 , 13,49]
	Taxa de transações	0,67 transações/segundo	0,06	[0,63 , 0,70]
	Taxa de transferência	1,34 MB/segundo	0,11	[1,26 , 1,41]
Uma máquina virtual	Disponibilidade	99%	0	[99 , 99]
	Tempo de resposta	9,71 segundos	1,99	[8,47 , 10,94]
	Taxa de transações	0,82 transações/segundo	0,14	[0,73 , 0,91]
	Taxa de transferência	1,65 MB/segundo	0,28	[1,47 , 1,82]
Duas máquinas virtuais, apenas uma executando o servidor web	Disponibilidade	99%	0,70	[98,56 , 99,43]
	Tempo de resposta	11,28 segundos	1,58	[10,29 , 12,26]
	Taxa de transações	0,73 transações/segundo	0,07	[0,68 , 0,77]
	Taxa de transferência	1,46 MB/segundo	0,15	[1,37 , 1,56]
Duas máquinas virtuais, ambas executando o servidor web	Disponibilidade	92,8%	2,88	[91,02 , 94,58]
	Tempo de resposta	34,96 segundos	4,15	[32,39 , 37,53]
	Taxa de transações	0,24 transações/segundo	0,03	[0,22 , 0,26]
	Taxa de transferência	0,48 MB/segundo	0,06	[0,44 , 0,52]

Nessa avaliação, mesmo a comparação dos cenários executados no Cubietruck com a execução no computador do *front-end*, onde houve grande disparidade na avaliação da seção 4.4, mostra resultados semelhantes para a carga de trabalho aplicada. Isso mostra um bom desempenho de entrada e saída de rede do Cubietruck e, consequentemente, das máquinas virtuais instanciadas.

Quando dois servidores web são executados em duas máquinas virtuais diferentes e avaliados simultaneamente, é visível a queda de desempenho. Esse resultado era esperado, visto que a mesma interface de rede física está sendo dividida e utilizada por diferentes interfaces virtuais. Como as duas interfaces virtuais estão sendo utilizadas ao mesmo tempo, a queda de desempenho é justificada pela utilização do protocolo TCP, que divide a taxa de transferência entre as diferentes conexões.

4.6 Considerações finais

As avaliações executadas mostraram um bom desempenho das máquinas virtuais que executam sobre o Cubietruck em comparação à execução nativa sobre o mesmo hardware, sobretudo na execução de programas comuns, que envolvem instruções de aritmética de inteiros. O tempo de inicialização das máquinas virtuais, embora possa parecer alto, é pouco significativo se o tempo de uso das máquinas for grande. Além disso, o tempo de instanciação mantém-se constante, independente do número de máquinas virtuais instanciadas, não frustrando as expectativas dos usuários.

Os *benchmarks* de desempenho do processador mostram uma diferença esperada entre o Cubietruck e um computador pessoal de arquitetura x86. Nesse caso, ressalta-se que a energia consumida por um trabalho depende da potência e do tempo. Essa relação entre as grandezas está expressa na Equação 4.1 abaixo.

$$E = P * t \tag{4.1}$$

Nesse sentido, a utilização de um computador de baixa potência e, consequentemente, de baixo consumo, deve levar em consideração que se o tempo para executar aplicações for muito alto, o consumo de energia para realizar os trabalhos pode ser maior que um computador de potência maior, mas que executa as aplicações em menor tempo.

5 CONCLUSÃO

A evolução das técnicas de virtualização e o aumento na dispibilidade de redes de alta capacidade permitiram, na última década, o surgimento de um novo modelo de negócios na computação: a computação em nuvem. Com uma vasta gama de produtos oferecidos, usuários e empresas passaram a utilizar capacidade computacional como um serviço pago proporcionalmente ao seu uso e ao seu tempo de utilização. Da mesma maneira, os provedores de computação em nuvem podem disponibilizar recursos não utilizados por um usuário para outros usuários.

Embora a concentração do poder computacional em grandes *data centers* é benéfica sob vários aspectos da computação verde, pois provê um melhor aproveitamento dos recursos e utiliza consolidação de servidores, esses centros de processamento consomem grandes quantidades de energia elétrica, demandada tanto pelo hardware quanto pelos sistemas de refrigeração necessários.

Nesse sentido, torna-se interessante combinar as vantagens e a flexibilidade da computação em nuvem com um hardware de baixo consumo de energia elétrica, a fim de prover uma computação ainda mais sustentável. Os processadores ARM surgem como uma excelente alternativa nesse caso, pois são projetados visando o baixo consumo de energia e possuem uma vasta presença no mercado de dispositivos embarcados.

O objetivo deste trabalho, além de apresentar os principais conceitos e ferramentas de virtualização e computação em nuvem, foi propor uma solução em nuvem de modelo IaaS (infraestrutura como serviço) que utilizasse, em sua infraestrutura de hardware, computadores de processador ARM. Para isso, utilizou-se o Cubietruck, um minicomputador de placa única e processador ARM disponível no GPPD. As soluções de software utilizadas foram todas de código aberto: Linux, como sistema operacional; Xen, como monitor de máquina virtual; e OpenNebula, como gerenciador de nuvem.

A avaliação da solução implementada mostrou bons resultados. O tempo para instanciar máquinas virtuais mantém-se constante conforme mais máquinas virtuais são instanciadas. O tempo de religamento é bastante rápido. Em termos de poder de processamento, os resultados podem ser considerados satisfatórios, levando em consideração que o projeto de um processador deve fazer uma escolha entre baixo consumo de energia e poder de processamento, onde tenta-se priorizar um sem negligenciar o outro. Além disso, foi possível perceber que, para programas que utilizam instruções que envolvem aritmética de inteiros, o desempenho manteve-se praticamente constante com o aumento do número de máquinas virtuais. Programas

que utilizam muita aritmética de ponto flutuante, entretanto, apresentam uma queda de desempenho se há muitas máquinas virtuais intanciadas. Por fim, a execução de um servidor web mostrou um bom desempenho do Cubietruck, apresentando resultados semelhantes a um computador de arquitetura x86 para as cargas de trabalho utilizadas. Seria necessário, entretanto, variar a carga de trabalho para saturar o processador ou a rede e verificar o desempenho do Cubietruck comparado ao computador de arquitetura x86.

Ressalta-se que, como este trabalho apresentou-se como prova de conceito, apenas um Cubietruck foi utilizado na solução de nuvem implementada. Para trabalhos futuros, pode-se utilizar um *cluster* com vários Cubietrucks para prover e avaliar uma solução mais próxima das existentes no mercado. Nesse sentido, é interessante também que seja avaliado o desempenho de aplicações que envolvam processamento paralelo e distribuído, através da utilização de bibliotecas como Open MPI e OpenMP, utilizando diversas máquinas virtuais para solucionar um problema.

Acima de tudo, este trabalho propôs uma alternativa ao hardware atualmente utilizado pelos provedores de nuvem que visa prover as vantagens da computação em nuvem combinado ao baixo consumo de energia elétrica. O estudo de soluções diversas e a constante preocupação com questões de sustentabilidade é fundamental para que consigamos cumprir os compromissos da computação verde. Destaca-se, nesse sentido, o esforço dos maiores provedores de nuvem em utilizar geração de energia limpa em seus *data centers*, e a necessidade da conscientização dos usuários sobre questões que envolvem computação e sustentabilidade. Essas discussões e medidas são cruciais para que tenhamos uma computação que continue resolvendo nossos problemas sem criar problemas para o meio ambiente.

REFERÊNCIAS

AMAZON. **AWS & Sustainability.** 2016. Disponível em < https://aws.amazon.com/pt/about-aws/sustainability/ >. Acesso em: out. 2016.

BARR, J. **Cloud Computing, Server Utilization, & the Environment.** 2015. Disponível em: < https://aws.amazon.com/pt/blogs/aws/cloud-computing-server-utilization-the-environment/ >. Acesso em: out. 2016.

BURGER, T. **The Advantages of Using Virtualization Technology in the Enterprise.** 2012. Disponível em < https://software.intel.com/en-us/articles/the-advantages-of-using-virtualization-technology-in-the-enterprise >. Acesso em: set. 2016.

CARISSIMI, A. da S. Virtualização: da teoria a soluções. In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 26., 2008, Rio de Janeiro, RJ. **Livro texto dos minicursos.** Rio de Janeiro: Sociedade Brasileira de Computação, 2008. p. 173-207.

CARISSIMI, A. da S. Desmistificando a Computação em Nuvem. In: Escola Regional de Alto Desempenho, 15., 2015, Gramado, RS. **Anais.** Porto Alegre: Sociedade Brasileira de Computação, 2015. p 3-24.

CLOUD COMPUTING WIKIPEDIA. **Cloud Computing.** 2016. Disponível em: < https://en.wikipedia.org/wiki/Cloud_computing >. Acesso em: set. 2016.

CUBIETECH. **Cubieboard 3.** 2016. Disponível em: < http://www.cubietech.com/product-detail/cubieboard3 >. Acesso em: nov. 2016.

DOCKER. **Docker – Build, Ship, and Run Any App, Anywhere.** 2016. Disponível em < https://www.docker.com >. Acesso em: set. 2016.

IBM. **Virtualization in Education.** 2007. Disponível em: < http://www-07.ibm.com/solutions/in/education/download/Virtualization%20in%20Education.pdf >. Acesso em: set. 2016.

JOE DOG SOFTWARE. **Siege Manual.** 2012. Disponível em < https://www.joedog.org/siege-manual/ >. Acesso em: dez. 2016.

LEONARD, THOMAS. **Running Xen on the Cubieboard 2.** 2016. Disponível em < https://mirage.io/wiki/xen-on-cubieboard2 >. Acesso em: ago. 2016.

LINARO. **About Linaro.** 2016. Disponível em < http://www.linaro.org/about/ >. Acesso em: nov. 2016.

LINUX-SUNXI. **linux-sunxi.org.** 2016. Disponível em < https://linux-sunxi.org/Main_Page >. Acesso em: nov. 2016.

LONGBOTTOM, ROY. Linux PC Benchmarks Ubuntu – Roy Longbottom's PC Benchmark Collection. 2016. Disponível em < http://www.roylongbottom.org.uk/linux%20benchmarks.htm >. Acesso em: nov. 2016.

MCBRIDE, S.; MEDHORA, N. **Amazon profit crushes estimates as cloud-service revenue soars.** 2016. Disponível em: < http://www.reuters.com/article/us-amazon-results-idUSKCN0XP2WD >. Acesso em: out. 2016.

MELL, P.; GRANCE, T. The NIST Definition of Cloud Computing. **NIST Special Publication 800-145.** Gaithersburg, MD: National Institute of Standards and Technology, 2011. Disponível em < http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf >. Acesso em: set. 2016.

MICROSOFT. **Sustainable Cloud Services | Microsoft Environment.** 2016. Disponível em: < https://www.microsoft.com/about/csr/environment/solutions/cloud/ >. Acesso em: out. 2016.

MURUGESAN, S. Harnessing Green IT: Principles and Practices. **IEEE IT Professional.** [S.l.], v. 10, n. 1, p. 24-33, jan.–fev. 2008.

OPENNEBULA. **OpenNebula 4.14 Documentation.** 2016. Disponível em < http://docs.opennebula.org/4.14/ >. Acesso em: ago. 2016.

SMITH, J. E.; NAIR, R. **An overview of virtual machines architectures.** [S.l.]: Elsevier Science, 2003.

UHLIG, R.; NEIGER, G.; RODGERS, D.; SANTONI, A. L.; MARTINS, F. C. M.; ANDERSON, A. V.; BENNETT, S. M.; KÄGI, A.; LEUNG, F. H.; SMITH, L. Intel Virtualization Technology. **Computer**, [S.l.], v. 38, n. 5, p. 48-56, mai. 2005.

VIRTUALIZATION WIKIPEDIA. **Virtualization.** 2016. Disponível em < https://en.wikipedia.org/wiki/Cloud_computing >. Acesso em: set. 2016.

VMWARE. **Server Virtualization & Consolidation.** 2016. Disponível em < http://www.vmware.com/solutions/consolidation.html >. Acesso em: set. 2016.

XEN. **Xen Project Software Overview.** 2016. Disponível em < https://wiki.xenproject.org/wiki/Xen_Project_Software_Overview >. Acesso em: nov. 2016.

XEN. **Xen ARM With Virtualization Extensions / Allwinner.** 2014. Disponível em < https://wiki.xenproject.org/wiki/Xen_ARM_with_Virtualization_Extensions/Allwinner >. Acesso em: ago. 2016.

WINGFIELD, N. **Microsoft Profit and Revenue Fall, but Cloud Computing Grows.** 2016. Disponível em: < http://www.nytimes.com/2016/01/29/technology/microsoft-earnings.html >. Acesso em: out. 2016.

APÊNDICE
INSTALAÇÃO E CONFIGURAÇÃO DO SISTEMA IMPLEMENTADO

O sistema de nuvem implementado requer a configuração do Cubietruck, que atua como *host* do OpenNebula, e do *front-end*. Esse apêndice lista os passos e comandos necessários para configurar ambos, assumindo a utilização do sistema operacional Ubuntu 14.04. As linhas linhas iniciadas com sustenido (#) são comentários sobre os comandos que devem ser utilizados.

Configuração do Cubietruck

A configuração do Cubietruck é feita utilizando um cartão microSD. Com o cartão conectado a um computador de arquitetura x86, os comandos a seguir devem ser executados.

```
# Instalação do compilador cruzado
apt-get install gcc-arm-linux-gnueabihf

# Compilação do u-boot
git clone git://git.denx.de/u-boot.git
cd u-boot
git checkout v2016.03
make CROSS_COMPILE=arm-linux-gnueabihf- Cubietruck_defconfig
make CROSS_COMPILE=arm-linux-gnueabihf-

# Compilação do Linux
git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
cd linux
git checkout v3.19
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- multi_v7_defconfig

# Deve-se alterar o arquivo de configurações para adicionar suporte
# ao Xen e ao LVM
vi .config
# As configurações a seguir foram adicionadas
CONFIG_XEN_DOM0=y
CONFIG_XEN=y
CONFIG_NETFILTER=y
CONFIG_NETFILTER_ADVANCED=y
CONFIG_BRIDGE_NETFILTER=y
CONFIG_STP=y
CONFIG_BRIDGE=y
CONFIG_SYS_HYPERVISOR=y
CONFIG_XEN_BLKDEV_FRONTEND=y
CONFIG_XEN_BLKDEV_BACKEND=y
CONFIG_AHCI_SUNXI=y
CONFIG_XEN_NETDEV_FRONTEND=y
CONFIG_XEN_NETDEV_BACKEND=y
CONFIG_INPUT_AXP20X_PEK=y
CONFIG_INPUT_XEN_KBDDEV_FRONTEND=y
CONFIG_HVC_DRIVER=y
```

```
CONFIG_HVC_IRQ=y
CONFIG_HVC_XEN=y
CONFIG_HVC_XEN_FRONTEND=y
CONFIG_MFD_AXP20X=y
CONFIG_REGULATOR_AXP20X=y
CONFIG_FB_SYS_FOPS=y
CONFIG_FB_DEFERRED_IO=y
CONFIG_XEN_FBDEV_FRONTEND=y
CONFIG_MMC_SUNXI=y
CONFIG_VIRT_DRIVERS=y
CONFIG_XEN_BALLOON=y
CONFIG_XEN_SCRUB_PAGES=y
CONFIG_XEN_DEV_EVTCHN=y
CONFIG_XEN_BACKEND=y
CONFIG_XENFS=y
CONFIG_XEN_COMPAT_XENFS=y
CONFIG_XEN_SYS_HYPERVISOR=y
CONFIG_XEN_XENBUS_FRONTEND=y
CONFIG_XEN_GNTDEV=y
CONFIG_XEN_GRANT_DEV_ALLOC=y
CONFIG_SWIOTLB_XEN=y
CONFIG_XEN_PRIVCMD=y
ONFIG_PHY_SUN4I_USB
CONFIG_HAS_IOPORT=y
CONFIG_MD=y
CONFIG_BLK_DEV_DM_BUILTIN=y
CONFIG_BLK_DEV_DM=y
CONFIG_DM_BUFIO=y
CONFIG_DM_SNAPSHOT=y

# Uma vez adicionada as configurações, compila-se o Linux
ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make zImage dtbs

# Compilação do Xen
git clone git://xenbits.xen.org/xen.git
cd xen
git checkout RELEASE-4.4.0
make dist-xen XEN_TARGET_ARCH=arm32 CROSS_COMPILE= arm-linux-gnueabihf-
CONFIG_EARLY_PRINTK=sun7i

# Formatação e particionamento do cartão microSD
export card=/dev/mmcblk0
export p=p

dd if=/dev/zero of=${card} bs=1M count=1
sfdisk -R ${card}
cat << EOT | sfdisk --in-order -L -uM ${card}
1,4096,L
,,8e
EOT
mkfs.ext4 /dev/mmcblk0p1

# Instalação da distribuição Ubuntu 14.04 no cartão microSD
export distro=trusty
mount ${card}${p}1 /mnt/
debootstrap --arch=armhf --foreign $distro /mnt/
cp /usr/bin/qemu-arm-static /mnt/usr/bin/
chroot /mnt /usr/bin/qemu-arm-static /bin/sh -i
/debootstrap/debootstrap --second-stage

cat << EOT > etc/apt/sources.list
```

```
deb http://ports.ubuntu.com/ $distro main universe
deb-src http://ports.ubuntu.com/ $distro main universe
deb http://ports.ubuntu.com/ $distro-security main universe
deb-src http://ports.ubuntu.com/ $distro-security main universe
deb http://ports.ubuntu.com/ $distro-updates main universe
deb-src http://ports.ubuntu.com/ $distro-updates main universe
EOT

# Configuração de senha do root
passwd

# Adição de usuário e configuração de sua senha
useradd admin
passwd admin

# Volta para a raiz do sistema original
exit

# Configuração do arquivo de partições
vi /mnt/etc/fstab
# O arquivo deve possuir o seguinte conteúdo:
/dev/mmcblk0p1  / ext4   rw,relatime,data=ordered       0 1

# Configuração do arquivo de resolução de endereços
vi /mnt/etc/resolv.conf
# O arquivo deve possuir o seguinte conteúdo:
nameserver 8.8.8.8

# Configuração do arquivo de interfaces de rede
vi /mnt/etc/network/interfaces
# O arquivo deve possuir o seguinte conteúdo:
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
  up ip link set eth0 up

auto br0
iface br0 inet dhcp
        bridge_ports eth0
        bridge_fd 9
        bridge_hello 2
        bridge_maxage 12
        bridge_stp off

# Instalação do kernel
cp {diretório do linux}/arch/arm/boot/zImage /mnt/boot/

# Instalação do Device Tree Binary
cp {diretório do linux}/arch/arm/boot/dts/sun7i-a20-cubitruck.dtb /mnt/boot/

# Instalação do Xen
cp {diretório do xen}/xen/xen /mnt/boot/

# Instalação do u-boot
dd if={diretório do u-boot}/u-boot-sunxi-with-spl.bin of=${card} bs=1024
seek=8

# Instalação do arquivo de configuração do u-boot
apt-get install u-boot-tools
```

```
vi boot.cmd

# O arquivo boot.cmd deve possuir o seguinte conteúdo:
setenv fdt_addr       0xaec00000
setenv fdt_high       0xffffffff
setenv kernel_addr_r  0xaf600000
setenv xen_addr_r     0xaea00000 # 2M
ext2load mmc 0 ${xen_addr_r} /boot/xen
setenv   bootargs   "console=dtuart   dtuart=/soc@01c00000/serial@01c28000
dom0_mem=512M,max:512M"
ext2load mmc 0 ${fdt_addr} /boot/sun7i-a20-cubietruck.dtb
fdt addr ${fdt_addr} 0x40000
fdt resize
fdt chosen
fdt set /chosen \#address-cells <1>
fdt set /chosen \#size-cells <1>
ext2load mmc 0 ${kernel_addr_r} /boot/zImage
fdt mknod /chosen module@0
fdt  set  /chosen/module@0  compatible  "xen,linux-zimage"  "xen,multiboot-
module"
fdt set /chosen/module@0 reg <${kernel_addr_r} 0x${filesize} >
fdt  set  /chosen/module@0  bootargs  "console=tty1  ro  root=/dev/mmcblk0p1
rootwait clk_ignore_unused"
bootz ${xen_addr_r} - ${fdt_addr}

# Geração e instalação da imagem do arquivo boot.cmd
mkimage -C none -A arm -T script -d boot.cmd boot.scr
cp boot.src /mnt/boot/

# Finalização do processo
umount /mnt
```

A partir dessa configuração inicial, o cartão microSD pode ser inserido no Cubietruck para inicialização do sistema. Com o sistema operacional executando no Cubietruck, deve-se prosseguir com a configuração do LVM, Xen e OpenNebula.

```
# Instalação dos pacotes necessários
apt-get install openssh-server
apt-get install xen-utils-4.4
apt-get install lvm2

# Criação do volume físico do LVM na segunda partição do cartão microSD
pvcreate /dev/mmcblk0p2

# Criação do grupo de volumos do LVM
# O nome desse grupo deve ser o mesmo do utilizado para criar o servidor
# de imagens LVM no OpenNebula
vgcreate vg0 /dev/mmcblk0p2

# Criação do usuário e grupo oneadmin para utilização pelo OpenNebula
# O usuário e o grupo devem possuir os mesmos IDs do usuário e grupo
# oneadmin do front-end. Neste exemplo, o ID utilizado é 9869.
groupadd -g 9869 oneadmin
useradd --uid 9869 -g oneadmin -d /var/lib/one oneadmin

# Adição do usuário oneadmin ao grupo sudo
sudo usermod -aG sudo oneadmin
```

```
# Para permitir a execução de comandos privilegiados sem exigência de
# senha, deve-se alterar a entrada do grupo sudo no arquivo /etc/sudoers
# para o seguinte:
%sudo ALL=(ALL:ALL) NOPASSWD:ALL
```

Configuração do *front-end*

No *front-end*, deve-se instalar e configurar o gerenciador de nuvem OpenNebula e o servidor web Sunstone. Os comandos necessários estão listados abaixo.

```
# Instalação dos pacotes
wget -q -O- http://downloads.opennebula.org/repo/Ubuntu/repo.key | apt-key
add -
echo "deb http://downloads.opennebula.org/repo/4.14/Ubuntu/14.04/ stable
opennebula" > /etc/apt/sources.list.d/opennebula.list
apt-get update
apt-get install opennebula opennebula-sunstone
```

A instalação dos pacotes do OpenNebula adiciona, automaticamente, o usuário `oneadmin` no *front-end*. Para que o Sunstone receba requisições de qualquer endereço IP, deve-se alterar a linha "`host: 127.0.0.1`" do arquivo `/etc/one/sunstone-server.conf` para "`:host: 0.0.0.0`".

O OpenNebula requer acesso via SSH do *front-end* aos *hosts* sem a exigência de senha. Para configurar esse acesso, os comandos necessários estão listados a seguir.

```
# Configuração do SSH
su - oneadmin
cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys

cat << EOT > ~/.ssh/config
Host *
    StrictHostKeyChecking no
    UserKnownHostsFile /dev/null
EOT

chmod 700 ~/.ssh/
chmod 600 ~/.ssh/id_dsa.pub
chmod 600 ~/.ssh/id_dsa
chmod 600 ~/.ssh/authorized_keys
```

Após a configuração do SSH, o diretório `/var/lib/one/.ssh` do *front-end* deve ser copiado para o mesmo caminho no Cubietruck.

O último passo antes de adicionar o Cubietruck como *host* do OpenNebula é habilitar os *drivers* do Xen no gerenciador de nuvem. Para isso, deve-se acessar o arquivo `/etc/one/oned.conf` e descomentar as linhas a seguir.

```
IM_MAD = [
    name       = "xen",
    executable = "one_im_ssh",
    arguments  = "xen" ]

VM_MAD = [
    name       = "xen",
    executable = "one_vmm_exec",
    arguments  = "xen4",
    default    = "vmm_exec/vmm_exec_xen4.conf",
    type       = "xen" ]
```

Por fim, deve-se adicionar o Cubietruck como um *host* do OpenNebula. Assumindo que o endereço IP do Cubietruck é 192.168.0.15, o comando a ser executado está listado abaixo.

```
onehost create 192.168.0.15 -i xen -v xen -n dummy
```

A partir desse momento, Cubietruck e *front-end* estão integrados, e o OpenNebula pode monitorar o estado de seu *host*. Para adicionar novos servidores de arquivos, imagens de disco e modelos de máquinas virtuais, pode-se utilizar o Sunstone, acessado, por padrão, na porta 9869.