UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Cloudlets Móveis: Deslocamento de serviços para redução de latência

Rodrigo Rezende

São Cristóvão

2016

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Rodrigo Rezende

Cloudlets Móveis: Deslocamento de serviços para redução de latência

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Ricardo José Paiva de Britto Salgueiro

Coorientador: Profa. Dra. Edilayne Salgueiro

São Cristóvão
2016

Rodrigo Rezende
Cloudlets Móveis:
Deslocamento de serviços para redução de latência/ Rodrigo Rezende. – São Cristóvão, 2016-
71 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Ricardo José Paiva de Britto Salgueiro

Dissertação (Mestrado) – UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO, 2016.

1. *Cloudlet*. 2. *Dockers*. I. Ricardo Salgueiro. II. Universidade Federal de Sergipe. III. Programa de Pós-Graduação em Ciência da Computação. IV. *Cloudlets* Móveis: Deslocamento de serviços para redução de latência

CDU 02:141:005.7

Rodrigo Rezende

Cloudlets Móveis: Deslocamento de serviços para redução de latência

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PROCC) da Universidade Federal de Sergipe (UFS) como parte de requisito para obtenção do título de Mestre em Ciência da Computação.

BANCA EXAMINADORA

Prof. Dr. Ricardo José Paiva de Britto Salgueiro, Presidente
Universidade Federal de Sergipe (UFS)

Profa. Dra. Edilayne Meneses Salgueiro, Co-orientadora
Universidade Federal de Sergipe (UFS)

Prof. Dr. Edward David Moreno, Membro
Universidade Federal de Sergipe (UFS)

Prof. Dr. Paulo Romero Martins Maciel, Membro
Universidade Federal de Pernambuco (UFPE)

Cloudlets Móveis: Deslocamento de serviços para redução de latência

Este exemplar corresponde à redação da Dissertação de Mestrado, sendo a defesa do mestrando **Rodrigo Rezende** para ser aprovada pela banca examinadora.

Trabalho aprovado. São Cristóvão, 29 de agosto de 2016:

Prof. Dr. Ricardo José Paiva de Britto Salgueiro
Orientador

Profa. Dra. Edilayne Meneses Salgueiro
Co-orientadora

Prof. Dr. Edward David Moreno
Membro

Prof. Dr. Paulo Romero Martins Maciel
Membro

É preferível ser um pato que deseja crescer dentre cisnes
que um cisne sem a capacidade de crescer

Agradecimentos

Muitos me foram importantes para a conclusão desta dissertação que me seria insuficiente esta página para citar todos e demonstrar meu agradecimento. Dentre os grupos posso agradecer ao PoP-SE na pessoa de Dilton, ao TJSE pela concessão de espaço e tempo, à UNIT pela compreensão e à UFS pela oportunidade. Individualmente devo agradecer à minha família, em especial à minha esposa Susana pelo companheirismo, incentivo e paciência entre todos os dias de impaciência que ela absorvia de mim, e aos meus pais pelo suporte e incentivo me oferendo tempo cuidando de alguns problemas cotidianos. A Itauan, Saulo e Wesley que me ajudaram com estudos e discussões, sem eles não teria conseguido. Agradeço aos meus orientadores Ricardo e Edilayne pela paciência, insistência e acreditarem mais em mim do que eu mesmo, sei que me trazer de volta aos estudos disputando com 3 empregos não era função de vocês mas obrigado por acreditarem. Por não ser injusto, credito os agradecimentos aos que estiveram mais próximos de mim nos dias finais e mais crítico mas todos amigos, companheiros, colegas e inócuos me foram úteis em algum momento de alguma forma, os agradeço também. Um beijo pra Rita, sempre presente quando escrevia.

Resumo

Serviços em nuvem tem se tornado uma constante para solução dos problemas computacionais convencionais, algumas empresas a exemplo da Amazon ou Microsoft investiram em infraestrutura para vender recursos como PaaS ou IaaS. Subsequentemente novos dispositivos, agora portáteis, tem se disseminado e precisaram adotar o modelo da nuvem de provimento de recursos criando dessa maneira a *Mobile Cloud Computing*. Mesmo com os vários modelos de arquitetura para essa nova nuvem para dispositivos móveis, a evolução aponta em total transparência para as aplicações e usuários quanto à localização dos recursos, demandando que os equipamentos móveis necessitem uma conexão permanente e uma abordagem de espelhamento de processos conhecidas como *Cloudlets*. As *Cloudlets* devem prover recursos exigidos que garantam uma experiência de percepção de uso confortável ao usuário, contornando problemas como alta latência, processamento local e indisponibilidade de conexão.

Para a obtenção dessa qualidade de percepção, a *Cloudlet* deve estar o mais próximo possível do usuário e acompanhá-lo durante seu deslocamento. Este trabalho apresenta uma extensão ao simulador *CloudSim* que possibilita a análise do efeito do deslocamento das *Cloudlets* através da nuvem.

Como alternativa, este trabalho discute a adoção de um modelo baseado em *containeres* e *dockers* que permita um tempo de subida menor e menos dados referentes ao ambiente para serem deslocados entre os centro de processamento das *cloudlets*. Para avaliação da alternativa foi desenvolvida uma extensão do *CloudSim* que permite a mobilidade dos serviços na nuvem. Como resultado a estensão demonstrou que, para os cenários propostos, é possível reduzir o tempo e o overhead necessários para os deslocamento, beneficiando a percepção do usuário e abrindo novas premissas para avaliação, implementação e otimização.

Palavras-chaves: *Cloudlet*, mobilidade, *docker*.

Abstract

Cloud Service have been usual to solve several computer issues, some companies like Amazon or Microsoft decide invest a big amount of money to sell computer resources as PaaA or IaaS. Subsequently, new devices, now portables, emerge as common and needs cloud computing to provide resources, creating a new structure known as mobile cloud computing. Even showing several models to its architecture in mobile cloud computing, natural behavior converge to total transparency to applications and users concerning to resource location, requiring that mobile devices maintains a permanent connection and an approach that needs a procces mirroring, knows as Cloudlets. These Cloudlets should provides required resources to guarantee a cofortable perception experience to users, giving an alternate option to latency problems, local processing and unavailable connections. To obtain this quality of perception, the Cloudlet should be closest to user as possible and follow him all the time during his travel. The usual model about Cloudlets consists on services running at virtual machine, that are costly what concerns moving and startup time. As alternative this work proposes a docker container based model, which will permit a smaller startup time less tranfer data about system between datacenters evolved. This alternative shows, at proposed scenarios, is possible reduce startup overhead and moving time, benefeting users perception and opening new perspectives to implement and optimizing.

Key-words: *Cloudlet, handoff, docker.*

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

CDN	*Content Delivery Network*
GB	*GigaByte*
Gbps	*Gigabits por segundo*
Ghz	*Gigahertz*
IaaS	*Infrastructure as a Service*
ICMP	*Internet Control Message Protocol*
ID	*Identification*
IPC	*Inter-process communication*
LXC	*Linux Container*
MB	*MegaByte*
Mbps	*Megabits por segundo*
MIPS	*Milhões de Instruções Por Segundo*
ms	*milissegundo*
PaaS	*Platform as a Service*
PID	*Process Identification*
QoS	*Quality of Service*
RTT	*Round Trip Time*
SaaS	*Software as a Service*
SDN	*Software Defined Network*
SLA	*Service Level Agreement*
TB	*TeraByte*
VM	*Virtual Machine*

Sumário

1 Introdução

Há alguns poucos anos, os cálculos e informações humanas eram armazenados em centenas de folhas de papel, impressos em prensas mecânicas e equações desenhadas manualmente por cientistas. Com o advento da computação e criação de computadores esses documentos passaram a ser dados armazenados e gerados instantaneamente por poderosos processadores.

Inicialmente essas máquinas de processar eram compostas por válvulas e ocupavam centenas de metros quadrados dispostos em salas específicas ou mesmo prédios inteiros. A necessidade de processamento aumentou e o tamanho desses cérebros eletrônicos diminuiu em proporções equivalentes, o que anteriormente estava distribuído em salas com a miniaturização passou a ocupar a área equivalente a uma geladeira convencional. Ainda assim, segundo Stallings (2010), mesmo menores continuavam a ocupar salas específicas e centralizava todo o processamento em apenas um nó onde os usuários se conectavam, no formato de *mainframes*.

A maneira para otimizar o uso dos *mainframes* foi a virtualização, técnica descrita em (BAYS et al., 2015) que permite a criação de múltiplas plataformas virtuais em uma única plataforma física, permitindo arquiteturas heterogêneas executarem o mesmo *hardware*. Essa técnica possibilitou o atendimento de clientes em larga escala otimizando o uso dos grandes computadores e acesso remoto. Seguindo uma evolução natural das necessidades, a demanda por cálculos aumentou, conseguindo ser suprida pela engenharia de novos materiais e métodos de fabricação dos processadores, garantindo até mesmo a miniaturização dos componentes a ponto do equipamento completo caber numa escrivaninha. Chegaram os computadores pessoais, mais conhecidos como PCs, com preços acessíveis e prometendo que mais cientistas conseguissem grandes cálculos com orçamento mais modesto.

Esse foi o início da popularização do invento que permitiu a mais rápida evolução cultural e científica já registrada na história da humanidade, outrora limitada à capacidade humana de processar informações e restrita a pequenos grupos privilegiados com acesso a dados importantes não difundidos tão amplamente. Da grande quantidade de computadores pessoais mundialmente dispersa surge uma outra necessidade implícita, embora não menos importante que é a de como difundir com os pares científicos distantes fisicamente as informações e os dados gerados.

Estimulada pela guerra e pelo interesse comercial, as redes de computadores locais começam a ter suas fronteiras expandidas, tendendo para uma rede de redes locais, científicas, comerciais, militares de escala global; a intercomunicação entre essas diversas redes serve como embrião para Internet.

Com a possibilidade de atender a todos os habitantes do globo, e fora dele, a Internet começa a se popularizar no campo cultural, exigindo uma capacidade de processamento ainda maior do que era esperado. Em grande parte, essa demanda se deve a aspectos comerciais,

onde empresas tiveram êxito em monetizar ao suprir pequenas demandas individuais em larga escala, oferecendo serviços especializados de alta qualidade a todos financeiramente capazes, independente da relevância apresentada. Ao oferecer esses serviços as empresas voltaram a centralizar os equipamentos em *datacenters*, locais especializados destinados a proteger os equipamentos que proveriam os dados e processamento aos clientes. Espaço físico, equipamentos, custos e enormes prédios especializados voltaram a ser o desafio da computação. Um novo serviço para um cliente implicaria em um novo hardware no *datacenter* da empresa; incluir o serviço da maneira mais econômica possível se tornou uma questão latente.

A volta da tecnologia de Virtualização foi a solução encontrada para otimizar processamento e custos. Com ela seria possível utilizar o processamento ocioso dos equipamentos, criar serviços e servidores sob demanda sem a necessidade de mais espaço físico ou um consumo energético proporcional. A Virtualização proporcionou também um incremento na qualidade de experiência do usuário, garantindo balanceamento, criação de servidores sob necessidade e alocação de recursos de acordo com a necessidade de maneira instantânea.

Com as novas possibilidades para criação de máquinas virtualizadas para atender a demanda crescente criou-se o hábito e necessidade de grande desempenho com alta disponibilidade nos serviços. Esse fato gerou outro desafio, pois embora houvesse a possibilidade de criar servidores sob necessidade de um serviço específico, com a globalização da Internet, onde se criaria o servidor e como ele atenderia às necessidades passou a ser o questionamento. O local de sua criação determinaria o desempenho de rede, conectividade, rotas, priorizações etc.

Cloud Computing, ou computação nas nuvens, criou parâmetros para reger essa criação de servidores virtuais, abstraindo a camada de infraestrutura e tornando a experiência do usuário mais confortável. Ao se demandar mais capacidade da nuvem (não importa se armazenamento, processamento, vazão de rede, atraso etc) onde esse recurso seria alocado não importaria mais, ele pode ser obtido de um *datacenter* na Europa ou na América do Norte. Para o usuário essa informação pode ser desprezada caso a desempenho ou custo seja satisfatório.

Essas nuvens computacionais para alocação de recursos podem ser públicas com propósito geral, privadas para empresas que tenham filiais ou mistas se a necessidade julgar apropriado. Segundo (PEREZ, 2014), o mercado esperado para *Cloud Computing* em 2014 deveria ter movimentado cerca de 9,5 bilhões de dólares, valor proporcional a atual popularidade da tecnologia.

1.1 Justificativa

Nos últimos anos aplicações baseadas em dispositivos móveis tornaram-se mais comuns em vários campos de uso tais como entretenimento, saúde, jogos, negócios, redes sociais e notícias. A razão dessa popularidade deve-se que a computação móvel é capaz de oferecer ao usuário exatamente a ferramenta que o usuário precisa independente do momento e da localidade em que ocorre a demanda. Mobilidade é uma característica da computação pervasiva, tão em

evidência atualmente, permitindo a continuidade da atividade independente do deslocamento ou local onde se encontra a atividade.

Tais aplicações exigem um modelo de serviço diferenciado, até então inexistente e impossível de ser suprido. As características diferenciadas utilizando a nuvem abriram espaço para uma classe, chamada de *mobile cloud computing*. Essa nova espécie de nuvem para atender os dispositivos móveis deve garantir a ubiquidade dos dispositivos móveis e o desempenho confortável ao usuário, que não tem a menor preocupação onde o seu serviço deverá ser executado.

Para apromirar a qualidade de percepção de serviço para o usuário, tal serviço deve estar o mais próximo possível do cliente. A mobilidade do usuário exige também a migração do serviço para que ele não se distancie muito, essa migração não pode ser avaliada pelos simuladores atuais, eles não a implementavam. Criar uma função que seja possível avaliar essa mobilidade dos serviços ou *Cloudlets* se mantinha como um problema em aberto.

No aspecto das *Cloudlets*, ou serviços, um dos principais problemas envolve o atraso de sua comunicação com o cliente em um dispositivo móvel, a diminuição do atraso na inicialização do serviço favorece sua utilização e migração entre os *datacenters*, pois, ao reduzir o tempo necessário para inicialização, o custo envolvido na transferência e subida é igualmente reduzido.

Otimizar o tempo necessário para iniciar um serviço e sua migração entre *datacenters* distintos que compõem uma ou mais nuvem computacional e avaliar se apresenta como uma questão pertinente e em aberto.

1.2 Objetivos

O objetivo principal desta dissertação é apresentar uma alternativa de implementação das *Cloudlets* que resulte em uma redução da latência para movimentação do serviço entre *datacenters* ou nuvens locais e possa ser comprovada através de simulação. A análise dessa alternativa precisa ser validade através de simuladores, que não possuem métodos de avaliar a mobilidade das *Cloudlets* ou mesmo serviços, para prover parte do objetivo principal é estender o *CloudSim* para que ele seja capaz de avaliar mobilidade de serviços e máquinas virtuais.

Objetivos Específicos

Para atingir o objetivo principal desta dissertação deve-se atender igualmente aos seguintes objetivos específicos:

- Propor um modelo alternativo da implementação das *Cloudlets* utilizando *dockers containers*;
- Implementar novas funcionalidades no simulador *CloudSim* para permitir o deslocamento dos serviços;

- Demonstrar o ganho de desempenho que se obtém com o uso de *dockers containers* para *Cloudlets* móveis;

1.3 Organização da Dissertação

Para facilitar a compreensão, esta dissertação distribuiu-se em capítulos que agregassem conhecimento e permitissem a assimilação do conteúdo de maneira lógica-incremental. São esses capítulos:

- No Capítulo 2 apresenta-se a fundamentação teórica dos conceitos de *Mobile Cloud Computing*, das *Cloudlets* e das ferramentas utilizadas para conclusão desta dissertação;
- No Capítulo 3 são expostos trabalhos relacionados que motivaram este documento e serviram como base comparativa de resultados;
- No Capítulo 4 é descrito o funcionamento da mobilidade das *Cloudlets* e como essa questão foi incluída no *Cloudsim*;
- No Capítulo 5 são descritos os ambientes para simulação bem como a análise e comparação de resultados;
- A conclusão das simulações, as dificuldades encontradas e os trabalhos futuros se apresentam no Capítulo 6.

2 Fundamentação Teórica

Neste capítulo são apresentados os principais conceitos de *Cloud Computing*, *Mobile Cloud Computing* e das *Cloudlets* (base desta dissertação). As ferramentas utilizadas para implementação das *Cloudlets* em máquinas virtuais, tais como *OpenStack* e *OpenStack++* também são apresentadas, além de uma descrição do simulador *CloudSim* que foi utilizado neste trabalho. Na seção 2.1 é apresentado um histórico e descrição sobre *Cloud Computing* com uma subseção 2.1.1 descrevendo as nuvens federadas. Na seção 2.2 é definido *Mobile Cloud Computing* e na subseção 2.2.1 o principal assunto vinculado: *Cloudlet*. As tecnologias utilizadas como *CloudSim*, *Docker container*, *OpenStack* e *OpenStack++* são tratadas na seção 2.3, seção 2.4, seção 2.5 e subseção 2.5.1 respectivamente.

2.1 *Cloud Computing*

O termo *Cloud Computing*, ou computação na nuvem em tradução livre para o português, foi citado academicamente pela primeira vez por Chellappa em 1997(CHELLAPPA, 1997) e em 1999 uma empresa chamada Net Centric tentou registrar o termo para garantir o direito autoral, foi rejeitado. A história da *Cloud Computing* pode ser vista com as datas mais importantes entre 1959 e 2001 na Tabela 2.1, segundo Rajaraman (2014).

Existem muitas definições para *Cloud Computing*, uma das mais utilizadas advém do NIST[1], que descreve em (MELL; GRANCE, 2011) como um modelo que permite acesso a uma rede sob demanda ubíqua e conveniente a um conjunto compartilhado de recursos computacionais configuráveis (e.g. redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com o mínimo de esforço para gerenciamento ou interação com o serviço do provedor. Neste modelo, a nuvem é composta de cinco características essenciais:

- *Self-service* sob demanda - O usuário tem disponível para contratação todo recurso computacional existente a exemplo poder de processamento, espaço de armazenamento ou execução de aplicativos diretamente do provedor de serviços sem interação humana alguma;
- Amplo acesso à rede - Todo recurso computacional disponível pode ser acessado de qualquer local, a qualquer momento com qualquer dispositivo capaz que possa acessar a rede mundial de computadores;
- Conjunto de recursos - O conjunto de recursos disponíveis devem ser reunidos de tal maneira que garanta o serviço contratado. O *pool* desses recursos podem ser geograficamente

[1] *National Institute of Standards and Technology*

Tabela 2.1 – História da *Cloud Computing*.

1959	John McCarthy nota a necessidade de computadores de tempo compartilhado.
1961	Em palestra, John McCarthy sugere que os computadores terão utilidade similar ao serviço de telefonia.
1966	Douglas Parkhill publica um livro intitulado "Desafios da computação utilitária"†.
1995	Amazon inicia a venda de livros através da *World Wide Web*.
1999	Salesforce.com inicia a venda de *Software* como serviço utilizando a *World Wide Web*.
1999	Ian Foster e Carl Kesselman publicam o livro *The Grid: Blueprint for a new computing infrastructure* e desenvolvem a ferramenta Globus para criação de um computador em *grid*.
2004	Google inicia a oferta do serviço de *email*.
2006	Amazon apresenta o conceito de computação paga pelo uso (*Amazon Web Services*) e *Elastic Cloud Computing* (EC2).
2006	Google apresenta *Google Apps* com 2 GB de espaço de armazenamento em sua infraestrutura.
2010	Microsoft lança seu serviço de provimento de nuvem, Azure.
2011	IBM oferece nuvem inteligente.

Fonte – (RAJARAMAN, 2014)

† – *Challenges of the Computer Utility*.

distribuídos em vários *datacenters*, seu poder computacional pode ser compartilhado entre vários usuários. Eles devem ser alocados dinamicamente aos consumidores de acordo com a demanda. O usuário pode ter conhecimento ou não de que *datacenter* o recurso provém, o que em algumas situações tem causado desconfortos diplomáticos;

- Rápida Elasticidade - O consumidor pode solicitar mais recursos quando necessário e pode liberá-los quando não forem precisos. Do ponto de vista do consumidor, os recursos são ilimitados e ele pagará apenas o quê e quando utilizar;

- Serviços mensuráveis - O usuário da nuvem é capaz de monitorar e controlar o uso dos recursos para acompanhar suas contas relativas ao uso. Esse controle permite também que a nuvem seja adaptativa, balanceando automaticamente a carga de acordo com o comportamento de uso.

As características da *Cloud Computing* favoreceram empresas e usuários que não queriam ou não podiam alocar grandes recursos financeiros para construção e manutenção de *datacenters*. Grandes empresas, a exemplo da Amazon e Microsoft, que já possuíam um vasto

parque computacional observaram a oportunidade de otimizar os seus recursos computacionais e monetizarem vendendo o serviço de nuvem a clientes menores. Nesse modelo as empresas que vendem o serviço investem na sua infraestrutura e gerência com a finalidade de vender parte da sua capacidade computacional, a qualidade do serviço é regido por um contrato com o provedor e o custo final do *datacenter* é rateado entre os diversos clientes nos casos de nuvens públicas, diminuindo o custo final de manutenção do ambiente e dos recursos conforme demonstramos em (REZENDE et al., 2016).

Neste modelo a infraestrutura deve ser dinâmica, virtualizada e de fácil alocação por parte dos clientes, o serviço final normalmente não exige grandes requisitos de conhecimento do equipamento ou ambiente. Apesar da facilidade de uso da estrutura, sistemas internos e aplicações particulares não são geridas pelos provedores da nuvem, exigindo a mesma necessidade de outras alternativas com relação ao conhecimento das aplicações ou a contratação de consultorias específicas para *Cloud Computing*.

A segurança desses ambientes, apesar de ser de inteira responsabilidade do provedor, adentra numa seara um tanto nebulosa onde cada *datacenter* pode possuir suas leis específicas. Por ser geograficamente distribuída, a nuvem pode estar presente em vários países e com isso serem regidas por leis locais diferentes das dos clientes. Alguns provedores declaram explicitamente onde irão executar as aplicações, para que o usuário decida onde irá alocar o recurso. Também é preciso reconhecer que a proteção do ambiente difere da segurança da aplicação, se uma aplicação particular sua corretude comportamental depende exclusivamente do cliente.

Quanto a implantação e uso, a nuvem pode ser classificada de quatro maneiras diferentes. Essa classificação pode levar em consideração o custo, a responsabilidade ou a distribuição, conforme os tipos listados a seguir:

- Nuvem Pública - A infraestrutura é disponível a qualquer cliente que tenha condições de pagar, seu foco é oferecer serviços em nuvem. Ela é de responsabilidade e controle exclusivamente do provedor de serviço. Pode possuir uma esparsa distribuição geográfica para atender satisfatoriamente clientes distantes, sua infraestrutura é compartilhada entre os diversos clientes;
- Nuvem Privada - A infraestrutura computacional criada serve para atender um única organização ou empresa em particular. A responsabilidade é apenas da instituição que a criou, podendo contratar terceiros para gerenciar e dar manutenção. A distribuição geográfica normalmente esta em apenas um local, porém grandes empresas podem distribuí-la entre seus principais polos.
- Nuvem Comunitária - Formada por um grupo de interesse em comum com organizações colaborativas. A responsabilidade é delegada a uma das empresas que compõem o consórcio de interesse ou uma empresa especializada e sua distribuição pode ser esparsa para

atender os diversos interesses. Essa modalidade é utilizada normalmente com a exigência de grande poder computacional aliado a um custo reduzido por ser rateado.

- Nuvem Híbrida - A estrutura é combinada entre duas ou mais das classificações anteriores. Uma empresa pode possuir uma nuvem com dois ou mais modelos citados, isso a classificaria como nuvem híbrida.

Com relação aos serviços de *Cloud Computing* existem 3 modelos de oferta: *Software* como serviço (SaaS - *Software as a Service*), Plataforma como Serviço (PaaS - *Platform as a Service*) e Infraestrutura como Serviço (IaaS - *Infrastructure as a Service*). Normalmente essas classificações ajudam a determinar o serviço ofertado pelas nuvem públicas. Abaixo uma breve descrição de cada modelo de serviço com a Figura 2.1 delimitando cada camada e exemplo de empresas que os ofertam.

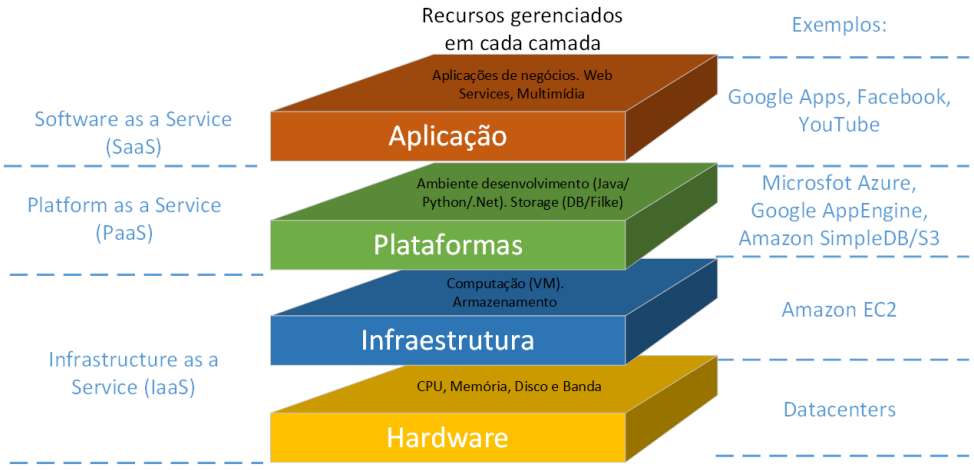

Figura 2.1 – Arquitetura de *Cloud Computing*.

Fonte: Produzido pelo autor

- IaaS – *Infraestructure as a Service*: A oferta ao usuário final é o provimento de processamento, armazenamento, redes, e outros recursos fundamentais na computação onde o cliente é capaz de implementar um software inespecífico, o qual pode incluir um sistema operacional e aplicações. O cliente não pode manipular a infraestrutura da nuvem mas pode controlar o sistema operacional, armazenamento e aplicações no recurso oferecido pela nuvem;

- SaaS – *Software as a Service*: A oferta ao usuário final é a aplicação do provedor executando numa estrutura de nuvem. As aplicações são acessíveis de vários dispositivos através de uma interface leve e adequada, como um navegador. O cliente não controla estruturas da nuvem tal como sistema operacional, armazenamento ou mesmo configurações específicas das aplicações individualmente;

- PaaS – *Plataform as a Service*: A oferta ao usuário final é oferecer na infraestrutura de nuvem criada pelo cliente aplicações com uso específicos de biblioteca, linguagens de programação e serviços oferecidos pelo provedor da nuvem. O usuário não gerencia ou controla as camada inferiores da infraestrutura da nuvem, mas tem controle sobre o sistema operacional, armazenamento e desenvolvimento das aplicações; além de possivelmente o controle limitado de determinados componentes de rede;

2.1.1 *Federação de Nuvens*

Apesar de não constituir formalmente um modelo, grandes provedores de serviços que precisam atender clientes geograficamente distribuídos utilizam-se de uma arquitetura de redes federadas para conseguir oferecer o serviço mais adequadamente aos clientes. Nesta distribuição as nuvens dos grandes provedores são distribuídas administrativamente dentre os diversos *datacenters* que possuem, sob a gerência de um Coordenador de Nuvem em cada *datacenter* distribuído que decidirá em qual região a aplicação do cliente será executada.

Assim, nessa distribuição pode-se considerar que as nuvens formam uma federação de confiança interconectadas pela rede de dados do provedor de serviços, assim é possível obter alguns benefícios financeiros e de desempenho, (WANG; WU, 2009):

- Aumentar a capacidade dos provedores de SaaS em alcançar níveis mais altos de QoS (*Quality of Service* - Qualidade de Serviço) aos clientes e oferecerem serviços mais aprimorados otimizando a região de execução e a escala em que são ofertados;
- Incrementar os limites de ofertas dinâmicas e automáticas aos clientes fazendo uso dos recursos entre nuvens. Uma nuvem da federação pode-se comportar como cliente de outra nuvem da mesma federação de acordo com a demanda;
- Permitir a auto-adaptação perante falhas como desastres naturais e durante manutenções programadas ou não. Ao ocorrer esses eventos a carga pode ser distribuída dentre os integrantes da federação e garantir o SLA[2] contratado pelos clientes, reduzindo o custo por penalidades.

Além dos benefícios supra-citados, a nuvem pode oferecer informações para que as aplicações dos clientes em execução possam fazer uso delas e favorecer o próprio desempenho diante a demanda; exemplos de aplicações que possam fazer uso dessa função são redes sociais e redes orientadas a conteúdo (CDN).

[2] *Service Level Agreement* - Acordo de Nível de Serviço: Contrato entre duas partes: entre a entidade que pretende fornecer o serviço e o requisitante ou interessado. Esse contrato descreve o serviço, suas metas de nível, papéis e responsabilidades.

2.2 *Mobile Cloud Computing*

Suprida a necessidade e otimização que a nuvem ofereceu com sua flexibilidade, agilidade e escalabilidade, em 2010 a Gartner (PETTEY; TUDOR, 2010) previu acertadamente que *Cloud Computing* teria o maior investimento proporciomal a outras tecnologias durante os 5 anos seguintes. Após 4 anos, outra pesquisa do Gartner (RIVERA, 2014) indicava que a tendência para os próximos dois anos seria a integração do real e virtual utilizando computação móvel como cliente/nuvem para suas aplicações: *Mobile Cloud Computing*.

Porém, ao contrário da *Cloud Computing*, dispositivos móveis são intrinsecamente limitados por seu poder de processamento, armazenamento e limite energético. Essas são características que estavam disponíveis nas gerações anteriores e que, por fazerem parte das premissas de mobilidade, não estão disponíveis em dispositivos portáteis.

Apesar dos limitantes, algumas soluções móveis se mostraram bastante eficientes, como o caso do *Apple iCloud* ao permitir que seus usuários obtivessem suas músicas, fotos, calendários, documentos etc em seus dispositivos móveis de forma instantânea e com espaço de armazenamento virtualmente ilimitado. Seus dados estavam em nuvens computacionais disponíveis convencionais e transparente aos seus usuários. A *Cloud Computing* permitiu a superação dos limites inerentes dos dispositivos móveis, integrando a solução existente para os dispositivos convencionais aos dispositivos móveis e gerando o início da *Mobile Cloud Computing*.

Bahl et al. (2012) denominam como *mCloud* uma nuvem, um ambiente que oferece seus recursos através de dispositivos móveis. Para o utilizador a nuvem não seria transparente, porém uma extensão do seu dispositivo. Neste caso inicial de integração, a fronteira entre o dispositivo e a nuvem seria bastante nítida para o uso dos recursos devido à própria natureza: ela foi produzida para uso corporativo, a arquitetura existente na primeira onda de *Cloud Computing* não era focada em dispositivos móveis. O modelo de programação utilizado em computação móvel precisaria ter uma forte sincronização que a nuvem exigia.

A geração seguinte de aplicativos iniciaram um processo de desvanecimento dos limites entre o que era nuvem e o que era local, mesclando o real e o virtualizado. Por exemplo tornou-se possível deslocar-se em uma cidade com a posição de movimento e direção atualizada a todo momento e descobrir locais ou ruas próximas sem necessariamente possuir o arquivo de mapas no celular, as informações viriam da nuvem e o posicionamento e renderização seria fornecido pelo equipamento local.

Naturalmente novas aplicações como sofisticados jogos multijogadores que respondiam a gestos e que possuíam gráficos elaborados surgiram e exigiram que o *hardware* portátil acompanhasse a demanda. Mesmo outros aplicativos sem grande apelo gráfico necessitavam de outras capacidades tais como reconhecimento de fala, processamento de fala, aprendizado de máquina, reconhecimento de imagem, planejamento e tomada de decisões etc. Aplicativos com necessidades que não era suprida por limitações dos equipamentos portáteis existentes.

Com o crescimento da quantidade das aplicações móveis permitindo a popularização da computação ubíqua, alguns obstáculos tornaram-se mais evidentes. Para mobilidade os parâmetros a serem considerados para os novos desafio são diferentes. Em (HA et al., 2013b) os dispositivos móveis não seguem a tendência regida pela Lei de Moore[3]; velocidade do clock do processador, tamanho da memória e capacidade dos discos se tornaram necessidades secundárias. O primordial para os dispositivos móveis são peso do equipamento, tamanho reduzido, tempo de vida da bateria, conforto ergonômico e uma eficiente dissipação de calor; características que não se apresentam como limitação temporária, mas intrínseca à própria mobilidade, sem perspectiva de solução definitiva, ao menos na maioria dos itens.

Por outro lado, a nuvem estava suprindo as demandas clássicas dos dispositivos convencionais de uma maneira acessível e eficiente. Obviamente a situação não passou desapercebida e a solução direta foi utilizar a nuvem para suprir todas as demandas (à exceção de potência energética) imediatas dos dispositivos móveis, incorporando desvanecimento das limites ocorrida na geração anterior de aplicativos e utilizando poder de processamento, poder gráfico, espaço etc. da *Cloud Computing*.

Em (KOVACHEV; CAO; KLAMMA, 2011), os aplicativos para dispositivos podem ser classificados em *Online* e *Offline*:

- *Offline*: Nessa categoria estão a maioria das aplicações para dispositivos móveis. Eles atuam como aplicativos completos locais, utilizando os próprios recursos com periódicas sincronizações com os servidores. Normalmente possuem uma grande integração com o equipamento e acesso facilitado, funcionando desconectado e desempenho limitado pelo hardware.

- *Online*: Aplicações *Online* presumem que a conexão entre o cliente e o servidor está presente na maior parte do tempo, devido a isso elas estão presentes quase em sua maioria nos *Smartphones*. Se caracterizam por serem multiplataforma, acessíveis de qualquer local ou dispositivo, podem utilizar recursos remotos em conjunto com os locais desfazendo os limites do dispositivos locais. Apesar desse aparente benefício esse modelo impõe a existência de latências relativamente grandes para respostas em tempo real que algumas aplicações deveriam exigir e algum desconforto na usabilidade devido a alguns momentos de desconexão ou de espera de resposta do servidor.

Infelizmente, por questões de hardware, atualmente não há possibilidade de uma opção perfeitamente adaptativa entre os dois modelos, onde os recursos locais seriam explorados caso não estivessem disponíveis na nuvem ou vice-versa.

[3] A complexidade para componentes com custos mínimos tem aumentado em uma taxa de aproximadamente um fator de dois por ano ... Certamente em um curto prazo pode-se esperar que esta taxa se mantenha, se não aumentar. A longo prazo, a taxa de aumento é um pouco mais incerta, embora não haja razões para se acreditar que ela não se manterá quase constante por pelo menos 10 anos.

Independente do modelo utilizado, *Online* ou *Offline*, uma *Mobile Cloud Computing* ou *mCloud* pode ter arquiteturas de implementação das aplicações diferenciadas, a exemplo:

- Aplicações Modularizadas: Nesta arquitetura as aplicações são implementadas divididas em camadas de função, cada camada pode ser executada no dispositivo móvel ou na nuvem que irá suprir sua demanda. Ao desenvolver um *middleware* para aplicação ele pode ser executado remotamente ou localmente para otimizar os parâmetros importantes para sua execução, sua comunicação pode ser direta como aplicações *Online* ou utilizando sincronização como no modelo *Offline*. Como exemplo dessa arquitetura é possível citar o *framework* AlfredO (RELLERMEYER; RIVA; ALONSO, 2008) que permite a distribuição de módulos de função entre o dispositivo móvel e o servidor, o módulo mínimo estaria no cliente como interface gráfica para execução do serviço no servidor.

- Aplicações Móveis: Neste modelo as aplicações devem ser acessadas a partir de diferentes dispositivos durante sua execução, mantendo a mesma qualidade de resposta ao usuário. Para isso o processo em questão deve ser capaz de migrar entre os dispositivos em que são exigidos, apresentando um ciclo de execução similar ao de uma máquina virtual suspendendo e resumindo sistema em outro *Host* de execução. A diminuição da latência é sua vantagem em relação ao modelo convencional de virtualização, ao evitar migrar todo o sistema operacional as operações de transferência e restauração apresentam otimizações.

- *Ad-hoc Mobile Clouds*: Apesar do oximoro que essa classificação aparente, sua definição abrange perfeitamente seus conceitos. Uma nuvem *Ad-hoc Mobile Cloud* se apresenta como uma nuvem entre os diversos dispositivos móveis próximos pertencentes a uma mesma rede, normalmente de uma mesma operadora. Os usuários envolvidos neste caso se comprometem a compartilhar os próprios recursos, permitindo que execuções de processos de outros usuários possam colaborativamente serem executadas localmente em seus equipamentos. Normalmente referem-se a um mesmo processo ou aplicação executada comunitariamente.

- Execução Aumentada: Em uma analogia à realidade aumentada, essa arquitetura visa transcender os limites impostos pelo *hardware* dos equipamentos portáteis. Para garantir a transparência para o usuário (KOVACHEV; CAO; KLAMMA, 2011) cita dois métodos de execução das aplicações; um onde o processo é replicado de forma idêntica na nuvem com recursos ilimitados e o outro onde uma aplicação possuiria um sistema operacional completo para alimentar a execução portátil. Em ambos os casos o resultado entrega ao usuário e aplicação um processo com desempenho de execução praticamente sem restrições.

Apesar dos diversos aspectos a serem considerados em *Mobile Cloud Computing*, presume-se que os fatores que limitam a total adoção das aplicações móveis, referem-se a

latência e problemas de redes. Sendo assim, essa dissertação concentra o foco na avaliação do desempenho de um modelo baseado na Execução Aumentada, e apresenta uma contraproposta fundamentado por Aplicações Móveis.

2.2.1 *Cloudlet*

Conforme explicado na seção anterior, fica claro que os recursos ofertados pela nuvem são uma excelente opção para exceder os limites dos dispositivos portáteis com relação a processamento e armazenamento. Porém, por estarem em *datacenters* distantes, às vezes internacionalmente distribuídos, essa opção apresenta impeditivos críticos, a exemplo da latência gerada ou limitação da conectividade.

A viagem entre diversos roteadores e vários saltos entre os enlaces ou tecnologias heterogêneas causam um atraso damasiadamente crítico para algumas aplicações em tempo real ou que fazem uso de realidade aumentada. A ausência de conectividade temporária com nuvens públicas como Amazon EC2 ou Azure também criam experiências desconfortáveis para o usuário, eventualmente apesar da rede de telefonia celular estar ativa, enlaces WAN podem apresentar instabilidades. Em (HA et al., 2013a), altas latências podem ser críticas o suficiente para afetar as novas aplicações móveis que exigem uma resposta em tempo real. Conforme apresentado em (SATYANARAYANAN et al., 2013) mesmo contornado o problema dos atrasos, a distância gera um acréscimo na vulnerabilidade da rede, que apresenta uma maior vulnerabilidade com relação ocorrência de desastres que poderiam causar interrupção na rede.

Como solução para contornar os problemas citados, garantindo maior confiança nas aplicações móveis, Satyanarayanan et al. (2009), propuseram um modelo arquitetural de três camadas para fornecimento de recursos computacionais aos dispositivos portáteis. Esse modelo cria um elemento intermediário - a *Cloudlet* - responsável por garantir os recursos com baixa latência e maior confiabilidade.

A estratégia em questão visa aproximar a nuvem dos dispositivos móveis, levando os recursos a redes que estão a um salto de distância dos clientes. O modelo de *Cloudlet* distribui a topologia em três camadas distintas onde há a nuvem como provedora de recursos ofertados aos clientes na camada mais baixa e intermediariamente existe a *Cloudlet* servindo como *proxy* ou substituto para execução dos processos conforme descrito na Figura 2.2. Essa estrutura deve ser completamente transparente para o utilizador, oferecendo uma visão de fartura dos recursos e apresenta quatro principais atributos segundo Satyanarayanan et al. (2009):

1. Exclusivamente contexto de *software*: Não há contexto de *hardware* do equipamento, mas possui cache de estado da nuvem. Pode possuir também *buffer* de dados originados de um dispositivo móvel isolados de forma segura do restante da nuvem. Ao evitar contexto de *hardware* permite-se a cada *cloudlet* se isolar do ambiente de execução, garantindo a auto-gerência do processo.

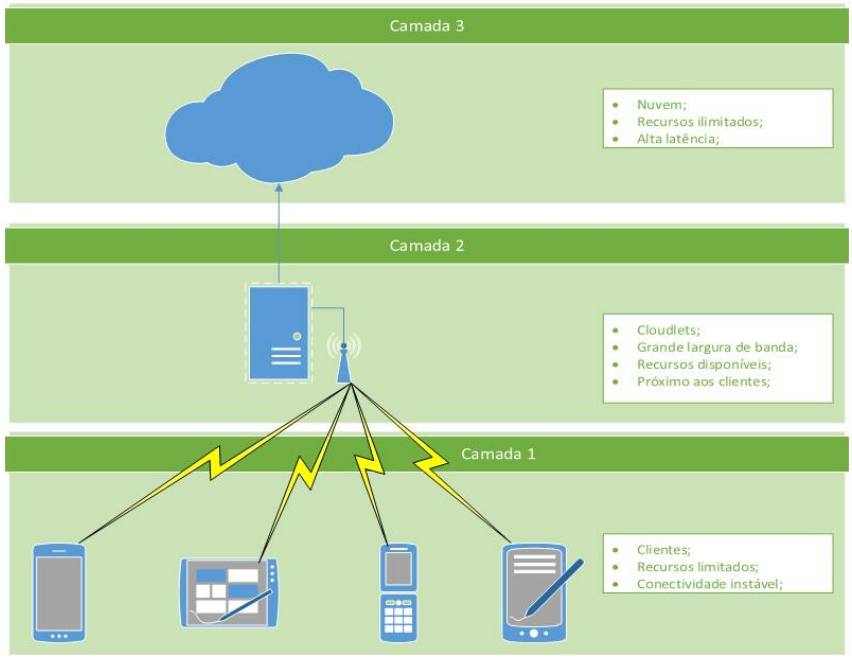

Figura 2.2 – Modelo de camadas adotado por *Cloudlets*.

Fonte: Produzido pelo autor

2. Poderoso poder computacional, bem conectado e seguro: Possui poder computacional suficiente (CPU, memória etc) para garantir a execução satisfatória em intensas demandas de um ou mais equipamento portátil. Apresenta excelente conectividade com a nuvem pública sem as restrições energéticas do cliente.

3. Próximo ao utilizador: É logicamente próxima aos equipamentos móveis associados. "Proximidade lógica" é definido como: possuir baixa latência na comunicação ponto-a-ponto e alta velocidade para transmissão dos dados. Frequentemente proximidade lógica implica em proximidade física, porém devido ao efeito "*last mile*[4]", o inverso pode não se apresentar verdadeiro: proximidade física pode não implicar em proximidade lógica.

4. Construído através da nuvem: Incorpora as necessidades ou instruções de carga dos dispositivos móveis para máquinas virtuais na nuvem, utilizando estruturas bem conhecidas como Amazon EC2 ou OpenStack com cada *Cloudlet* apresentando sua funcionalidade bem definida no sistema.

Por trazerem a nuvem próxima aos seus clientes utilizando máquinas virtuais através de equipamentos intermediários e funções específicas, a *Cloudlet* é frequentemente descrita como *"'Cloud in a box"'*, fazendo uma comparação como a criação de uma pequena nuvem dentro de uma caixa dedicada. Porém não deve-se fazer confusão entre Nuvem e *Cloudlet*, segundo Ha et al. (2013a) existem três diferenças notórias entre elas:

[4] Limitações impostas pela tecnologia que devem ter custo acessível e atender a uma larga escala de clientes.

- Provisionamento Rápido - A velocidade de provisionamento importa para a *Cloudlet*. Atualmente as Nuvens são otimizadas para iniciar imagens de máquinas virtuais que já existam em sua base de dados, não é oferecido uma opção rápida para instanciar uma imagem personalizada. A imagem em questão precisa ser personalizada individualmente e ser transferida para o *datacenter*, o que pode levar muito tempo de acordo com a largura de banda disponível. As *Cloudlets* seriam uma cópia do serviço em questão, tratada como uma máquina virtual (por falta de melhor analogia) que deveria ser transferida e iniciada;

- *Hand-off* - Uma vez instanciada a *Cloudlet*, ela deve se manter o mais próximo possível do seu cliente, normalmente a um salto de distância. Ao se deslocar, o ideal seria que a *Cloudlet* acompanhasse o dispositivo móvel e garantir a mínima distância possível mantendo a conexão, o desempenho e a transparência.

- Reconhecimento de Vizinhança - Ao se deslocar, o cliente deve requisitar a movimentação da *Cloudlet* para que seu desempenho se mantenha adequado, porquanto as *Cloudlets* devem constantemente descobrir ao menos um pequeno *datacenter* próximo ao seu cliente que possa abrigá-la quando necessário.

2.3 *CloudSim*

O *CloudSim* é um *framework* para modelagem e simulação de serviços e infraestrutura de computação nas nuvens desenvolvido inicialmente pelo *Cloud Computing and Distributed Systems Laboratory* (CLOUDS), na Universidade Melbourne, Austrália. Adotado frequentemente pela academia, o *CloudSim* tem se tornado o simulador de código aberto mais popular

A *Cloud Computing* ofereceu uma solução viável em larga escala, capaz de ser adaptada para as necessidade dos dispositivos móveis através da *Mobile Cloud Computing*, porém a implantação dessas estruturas nos *datacenters* dos provedores de serviço se mostrou uma atividade complexa devido à sua variedade de necessidade. Analisar o desempenho das políticas de provisionamento das máquinas virtuais em uma nuvem real apresentava alguns desafios conforme descrito em (WANG; WU, 2009):

- As Nuvens apresentavam necessidades diversificadas, perfis de escalonamento e demandas, tamanhos variados de sistemas e recursos;

- Os clientes tinham requisitos diferentes de QoS e SLA, eventualmente conflitantes ou incompatíveis e que não permitiam uma análise detalhada da situação;

- As aplicações, especialmente as novas, não possuíam um padrão de requisitos para seu uso e carga de execução/comunicação dificultando a predição de um dimensionamento.

Ademais, a constante reavaliação dos parâmetros das análises de desempenho de execução da nuvem demandariam grandes esforços dos analistas. Essa necessidade aumentaria o custo de gerenciamento e provisionamento da nuvem que deveria focar no menor custo final ao cliente, sendo assim torna-se inviável as constantes análises de desempenho em Nuvens reais que executavam ambientes de serviços em larga escala. A solução: simulação. Ferramentas que permitissem a simulação desses ambiente críticos de Nuvens seriam capazes reproduzir resultados condizentes a um custo baixo para os provedores de serviço, recriando situações pré-determinadas, em um ambiente controlado, que fosse possível repetir diversos testes e traçar opções para implantações na Nuvem real.

Para este trabalho, seria necessário um simulador que além das premissas de redes, possuísse avaliações específicas para as necessidade de Nuvens, flexível o suficiente para sua personalização, possibilitando a simulação da proposta através de métodos reconhecidos no âmbito acadêmico. A solução escolhida foi o *Cloudsim*, uma ferramenta descrita em (CALHEIROS et al., 2011) e utilizada por diversos pares que esta focada em aplicações na Nuvem analisando o provisionamento e testes de recursos, além de permitir uma grande flexibilidade através da personalização do seu código bem documentado. Dentre as funções que o *CloudSim* oferece, as destacadamente importantes para esse trabalho foram:

- Suporte para modelagem e simulação de ambientes *Cloud Computing* em larga escala, incluindo *datacenters* com suas características de recursos;
- Uma plataforma personalizável para modelagem de Nuvens, provedores de serviços. provisionamento e política de alocação de recursos;
- Suporte à simulação de conexões de rede personalizável entre os elementos do sistema simulado;
- Facilidade de simulação ambientes de Nuvem Federada, utilizando domínios públicos ou privados através de parâmetros já existentes ou a inclusão de novos de acordo com a necessidade;
- Avaliação dos recursos de virtualização mais comuns existentes e possibilidade de alterações para novos ou planejados;
- Utilização da conhecida topologia BRITE (MEDINA et al., 2001) para modelar largura de banda de enlaces de comunicação e latências associadas.

Escrito em Java, o *CloudSim* permitiu uma aprendizagem menos intrincada, possibilitando que o foco desta dissertação fosse menos deslocado para a questão da codificação. As modificações no simulador visaram incluir o conceito de *Cloudlet* em seu código e funções não existentes em seu código original tais como movimentação das mesmas *Cloudlets*, reutilizando

alguns dos recursos de avaliação de máquinas virtuais já existentes e os adaptando para *dockers*. As adaptações seguiram as recomendações descritas em (PAGARE; KOLI, 2015) e manteve o padrão de classes demonstrado na Figura 2.3.

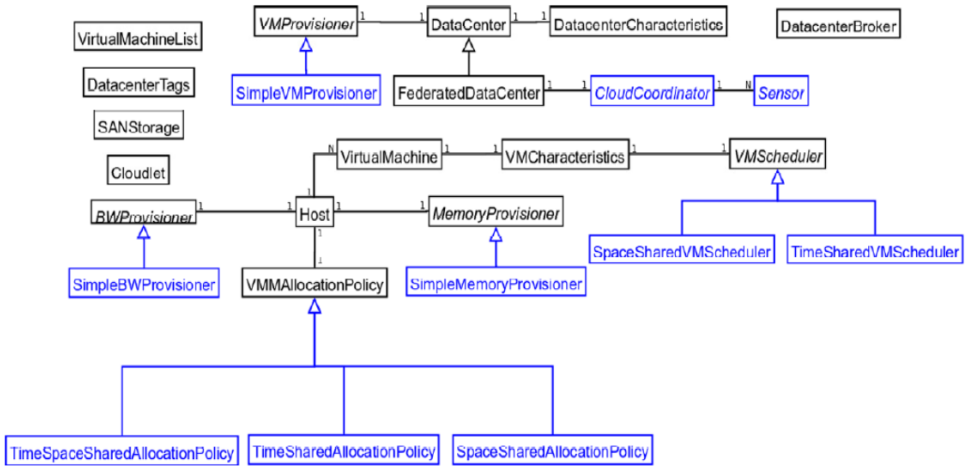

Figura 2.3 – Diagrama de Classes do *CloudSim*.

Fonte: (CALHEIROS et al., 2011)

2.4 *Docker Container*

Máquinas virtuais favoreceram bastante a evolução dos sistemas em que havia a necessidade da replicação do ambiente completo. Ao criarem todo o sistema operacional e bibliotecas instaladas no sistema original garante-se a compatibilidade de execução em qualquer *datacenter* que a VM for provisionada. Porém, essa compatibilidade tem o contraponto de precisar carregar todo o ambiente sempre que for necessário executar um novo serviço virtualizado. Essa carga gera um atraso inconveniente na disponibilização do serviço para o usuário.

Ao invés de executar um sistema operacional completo em um hardware virtualizado, a virtualização baseada em *container* utiliza o sistema operacional existente como base para modificações que oferecem um isolamento extra. Geralmente essa modificação envolve incluir o ID[5] do *container* a todos os processos sob sua responsabilidade e incluir novos controles de acessos e chamadas de sistema. Desse modo os *containers* podem ser interpretados como um outro nível de controle de acesso e de permissões ao sistema.

Historicamente os *containers* foram concebidos a nível de *kernel* para contornos de algumas dificuldades que existiam nos *clusters* de alto desempenho. Em regra o Linux implementa os escopos de Sistema de Arquivos, PID[6], redes, usuário, IPC[7] e endereçamentos, porém

[5] *IDentification* - número de identificação do serviço.

[6] *Process IDentification* - Número de identificação do processo.

[7] *Inter-Process Communication*: Modelo de comunicação entre processos.

o *container* cria um clone dos escopos já existentes e os vinculam ao processo que o requisitou, criando assim um novo contexto restrito ao criador, evitando que seus objetos possam ser vistos por outros escopos irmãos. Por serem cópias do escopo maior, as aplicações que rodam nos *containers* recebem instruções como se estivessem executando em um Linux convencional, apesar de dividirem a execução com outros contextos irmãos.

Diferentemente das máquinas virtuais, um *container* poder ser a execução virtualizada de apenas uma aplicação, opcionalmente pode ser executado um conjunto de aplicações para determinado fim onde todas vão ter acesso ao mesmo escopo, neste caso são referidos como um *system container*. Para economia de memória principal geralmente são escolhidos *containers* de aplicativo, assim não será necessário executar processos redundantes com o sistema principal. Essa é a escolha desta dissertação, como consequência da economia de memória RAM, o que possibilita uma transferência de dados mais eficiente.

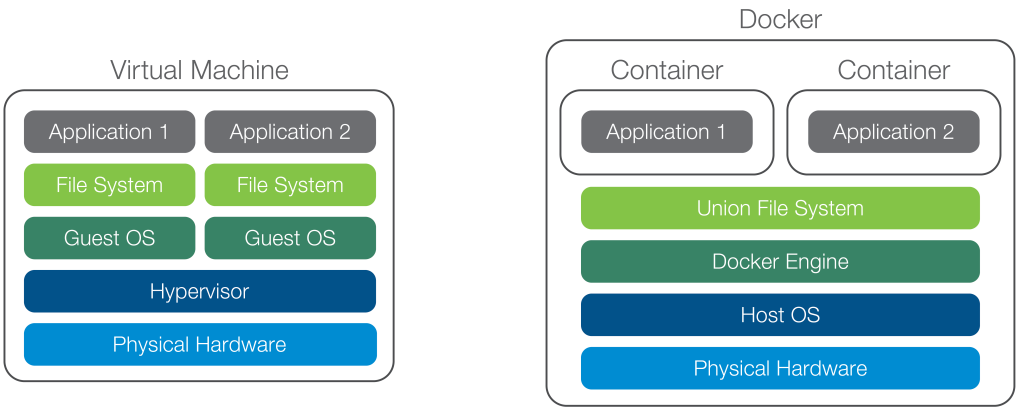

Figura 2.4 – Diagrama comparativo entre Máquinas Virtuais e *Dockers*.

Fonte: (SOLUTIONS, 2016)

Apesar de possuírem contextos diferentes, é possível compartilhar alguns aspectos de sua execução. Um sistema de arquivos poderia ter seu contexto compartilhado entre determinados *containers* para facilitar a busca de informações entre seus pares, diferentemente das máquinas virtuais onde toda sua comunicação dependeria das chamadas de sistemas dos sistemas virtualizados.

A segurança em *containers* é aplicada pelo próprio escopo em que ele se encontra, por ser tratado como processo em nível de usuário, um usuário com poder administrativo interno ao *container* não poderia ter os mesmo privilégio fora dele. Por não possuir privilégios externos, não haverá nem conhecimento de alguma informação do sistema hospedeiro.

Dentre as várias opções para *containers*, a exemplo LXC, lmctfy, Warden ou Docker foi-se escolhida essa última. Docker vem sendo bastante difundido e popularizado, utilizado inclusive em pesquisas científicas de outras áreas (GERLACH et al., 2014), (TOMMASO et al., 2015), (MERCHANT et al., 2016) devido ao seu conjunto de propriedades e facilidade de uso das ferramentas. Uma propriedade importante a se ressaltar em *Docker* é o seu sistema de arquivos distribuído entre camadas, como pode ser visto na Figura 2.4 essa característica permite o reuso do sistema de arquivos entre os *containers* com os mesmo serviços ofertados. Com isso ao se transferir um *Docker* para outro local seria transferido apenas a camada intermediária que foi modificada, mantendo o sistema base idêntico aos *containers* com o mesmo serviço, requerendo menos largura de banda, acessos a disco e espaço do que um serviço equivalente que utilize máquinas virtuais. Na Figura 2.4 tem-se um esquema explicativo do sistema de arquivo compartilhado e seu comparativo com relação a máquinas virtuais.

2.5 *OpenStack*

O *OpenStack* é uma solução de código aberto para orquestração de *Cloud Computing*, oferece uma interface intuitiva personalizável e extensível. Serve como plataforma para uma nuvem IaaS desenvolvida pela NASA em 2010 sob a licença do Apache 2. Suas principais características são:

- Escalável: Adota uma solução já implementada em empresas cujo volume de dados são medidos na casa dos *Petabytes* de informação, capaz de adotar um milhao de máquinas físicas, até 60 milhões de máquinas virtuais e bilhões de objetos armazenados;
- Flexível: Suporta a maioria das soluções de virtualização do mercado: ESX, Hyper-V, KVM, LXC, QEMU, UML, Xen e XenServer;
- Código Livre: Todo o código pode ser modificado e adaptado.

O projeto *OpenStack* tem o apoio de mais de 10.000 membros e mais de 850 organizações distribuídos em 87 países baseado no código escrito em Python pela NASA. Ele é especializado em prover à indústria de Tecnologia da Informação uma arquitetura de hospedagem em código aberto de máquinas virtuais em larga escala utilizando *hardware* distribuído.

Sua arquitetura é composta de três componentes principais que interagem entre sim conforme Figura 2.5:

- *Compute* - A plataforma básica de gerenciamento e controle das nuvens essencialmente IaaS. Responsável por controlar, manipular e criar servidores virtuais além das características de rede;

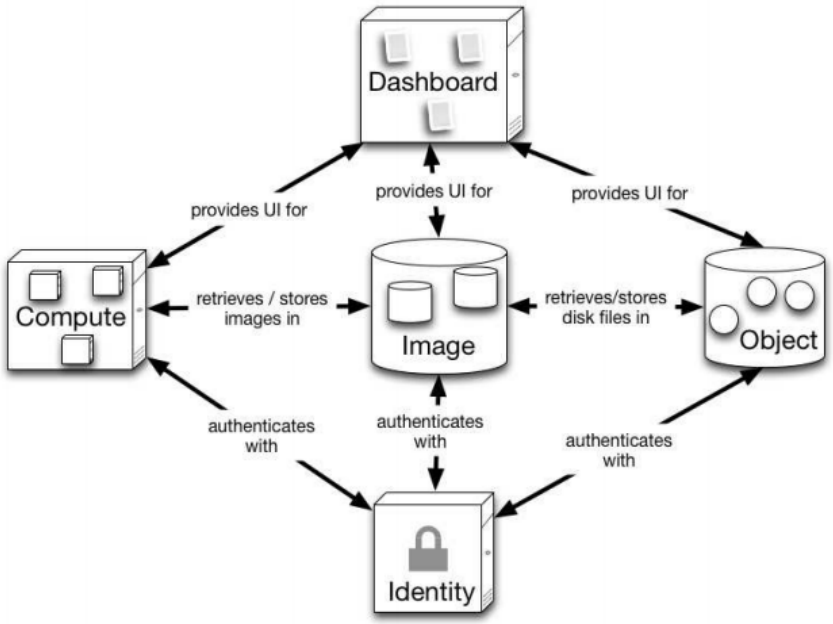

Figura 2.5 – Arquitetura do *OpenStack*.

Fonte: (SEFRAOUI; AISSAOUI; ELEULDJ, 2012)

- *Image* - Provê serviço de armazenamento de imagens, guardando e distribuindo as imagens dos servidores virtuais;
- *Object* - Provê o serviço de armazenamento, replicação e consistência dos dados trabalhados como objetos.

2.5.1 *OpenStack++*

Para execução das *Cloudlets* utilizando *VM Base* e *VM Synthesis* é necessário algumas modificações no *OpenStack* para fazer uso de suas otimizações. Essas modificações podem ser aplicadas em qualquer *OpenStack* por meio de extensões distribuídas pelos desenvolvedores, ao serem aplicadas as modificações referentes às *Cloudlets* e inclusão em seu *Dashboard* classifica-se a implementação como *OpenStack++*.

Suas principais implementações foram:

- *Import Base VM*: Importar uma *VM Base* para o gerenciador de imagem Glance, utilizado no *OpenStack*;
- *Resume Base VM*: Retoma a execução de uma VM utilizando a *VM Base* a partir de um arquivo *VM Overlay* gerado pela *VM Synthesis*;
- Criar *VM Overlay*: Cria um *VM Overlay* a partir de uma máquina virtual em execução com base na *VM Base*;

- *VM Synthesis*: Provisiona uma instância de uma máquina virtual no *OpenStack++* utilizando uma *VM Overlay*;
- *VM Handoff*: Migra uma instância de máquina virtual utilizando *VM Overlay* para um *cluster* diferente que utilize *OpenStack++*;

2.6 Considerações Finais

Neste capítulo foram apresentados os principais conceitos que envolvem a proposta deste trabalho, descrevendo *Cloud Computing* com seu histórico e principais tipos. O conceito de federação de nuvens através de sua relação de confiança e sua evolução para *Mobile Cloud Computing* onde os desafios passam a ser diferentes. Apresenta-se a definição de *Cloudlet*, foco principal desta dissertação que propõe uma otimização para o mesmo para ser avaliado através do *CloudSim* que foi estendido neste trabalho para permitir a mobilidade das *Cloudlets*. Também é apresentado as ferramentas existentes que podem implementar *Cloudlets*: *OpenStack*, *OpenStack++* e *Dockers*.

Ao final deste capítulo é possível ter o conhecimento dos conceitos que serão explorados e otimizados para avaliação.

3 Trabalhos Relacionados

Neste capítulo são discutidos alguns trabalhos referentes ao assunto de *Cloudlets*, *Mobile Cloud Computing* e *dockers*. Apesar de relativamente recente, o número satisfatório de trabalhos demonstra o quão relevante é o aspecto de processamento remoto através de *Cloudlets* e o quanto *dockers* podem ser eficientes em desempenho se comparados à máquinas virtuais. Na seção 3.1 esta exposto o modelo de implementação baseado em máquinas virtuais, na seção 3.2 é descrita a implementação das *Cloudlets* com sua preocupação com a latência e na seção 3.3 alguns estudos das vantagens dos *dockers* em detrimento das máquinas virtuais.

3.1 Modelo implementado em *Cloudlets*

A evolução para *Mobile Cloud Computing* já envolvia algumas adaptações para a nova plataforma e conceito de um modelo conveniente conforme descrito em (WOLBACH et al., 2008), neste artigo há uma proposta de decomposição do estado da máquina virtual em um sistema base (*VM Base*) e uma camada de informações particulares da máquina virtual (*VM Overlay*. Nesse modelo somente a *VM Overlay* precisaria ser entregue pelo dispositivo móvel à nuvem que o atenderia conforme descrito na Figura 3.1, minimizando o tempo de transferência e inicialização se comparado a uma máquina virtual completa.

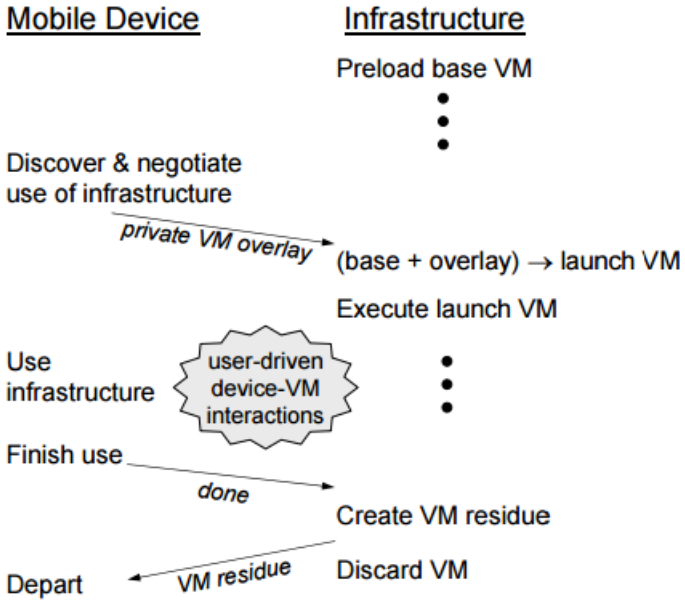

Figura 3.1 – Diagrama de transferência da *VM Overlay*.

Fonte: (WOLBACH et al., 2008)

Esse modelo utilizando máquinas virtuais foi utilizado como solução para a implementa-

ção das *Cloudlets* conforme descrito em (SATYANARAYANAN et al., 2009), essa arquitetura visava oferecer os recursos computacionais disponíveis na *Cloud Computing* aos dispositivos móveis da maneira mais transparente possível em aplicações *online* e modularizadas conforme descrito na seção 2.2. O princípio das *Cloudlets* seria executar os serviços oferecidos nos dispositivos móveis em módulos idênticos nas máquinas virtuais, que neste caso estariam o mais próximas possível aos clientes para diminuir o problema de latência e transferência da *VM Overlay*.

Para comprovação, em (HA; LEWIS; SIMANTA, 2011) ficou demonstrado através de experimentos em protótipos que a arquitetura satisfazia em termos de carga, latência e comportamento aos requisitos impostos em situações comuns a ambientes hostis que são suscetíveis ao tempo de vida da bateria dos dispositivos móveis, estado da conectividade, largura de banda, sincronismo dos dados e latência da comunicação. O protótipo em questão seguiu a Figura 3.2. A proximidade da *Cloudlet* com o dispositivo móvel foi determinado pelo estudo apresentado em (CLINCH et al., 2012).

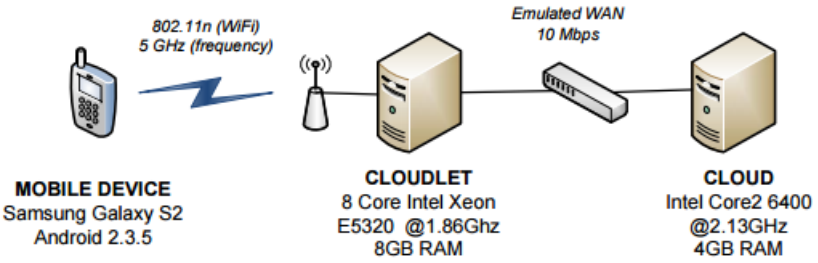

Figura 3.2 – Diagrama de experimento da *Cloudlet*.

Fonte: (HA; LEWIS; SIMANTA, 2011)

3.2 Latência e serviços

Preocupado sobre o problema de latência e carga sobre a largura de banda disponível para a transferência das máquinas virtuais, em (HA et al., 2013b), propuseram uma solução para atender a demanda em larga escala com uma latência mínima, permitindo a execução de uma *Cloudlet* baseada em máquina virtuais que conseguiria atender ao usuário com um atraso inicial médio de 46 segundos em uma aplicação de realidade aumentada. Essa proposta com otimizações de memória em execução e uso de deduplicação para economizar banda e evitar transferência desnecessárias apresentou o conceito da *VM Synthesis*, que criaria um sistema para ser transferido apenas com o que estava em execução no momento da máquina virtual e ao ser transferido se acoplaria a uma *VM Base* para receber a *Cloudlet*. No trabalho em questão ficou demonstrado que uma máquina virtual convencional de 514 MB poderia ser comprimida em uma *VM Synthesis* de 42 MB. Mesmo com essa otimização, é importante frisar que para

criação e execução da *VM Synthesis* é necessário agregar o tempo de pré e pós processamento, responsáveis por montar, compactar, descompactar e subir o serviço.

Com o conceito de *Cloudlet* amplamente disseminado no meio acadêmico e aceito na indústria, a exemplo do Google que utilizava suas características em alguns produtos como o *Google Glass*, o uso dessas *Cloudlets* apresentavam alguns inconvenientes clássicos de inicialização e *handover*. Por usar o modelo de máquinas virtuais convencionais, todos os seus problemas foram portados. Para prover uma maneira sistematica para desenvolvimento de *Cloudlets* em (HA; SATYANARAYANAN, 2015) foi apresentado o *OpenStack++*, uma extensão do já bem conhecido e difundido *OpenStack*, um ecosistema de código aberto para *Cloud Computing*. Essa implementação permitiu a criação de *Cloudlets* como GigaSight (SIMOENS et al., 2013), QuiltView (CHEN et al., 2014) e Gabriel (HA et al., 2014), possibilitando experimentos mais detalhados para análises. O *OpenStack++* apresentado em (HA; SATYANARAYANAN, 2015) implementou o que foi proposto em (HA et al., 2013b) conforme descrito na Figura 3.3, que apresenta a *VM Overlay* como o resultado compactado da diferença entre a máquina virtual original e a atual com os processos em execução.

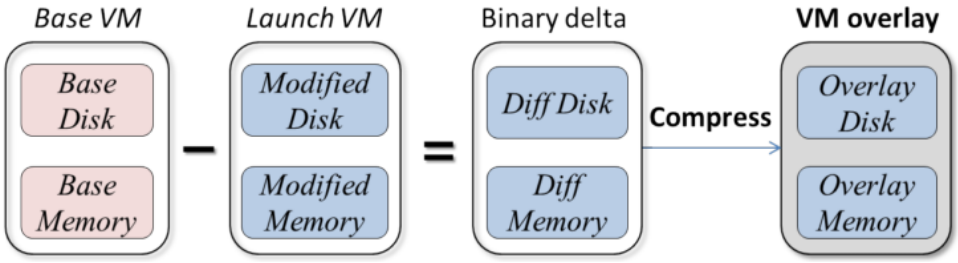

Figura 3.3 – Criação da *VM Overlay* a partir da *VM Base*.

Fonte: (HA; SATYANARAYANAN, 2015)

Implementados os ambientes para o desenvolvimento de *Cloudlets* utilizando máquinas virtuais, em particular *VM Synthesis*, Ha et al. (2015) demonstrou que o desempenho proposto tem um desempenho muito superior ao que poderia se ter em VMs sem uso da *VM Overlay*. Em seus experimentos em ambientes que exigiam baixa latência com várias aplicações móveis *online* ele demonstrou que o *handoff* das *Cloudlets* tinham o montante de dados transferidos reduzidos se comparados com as VMs, consequentemente com menos atraso e carga mais rápida do sistema. Em sua análise observou-se uma redução no tamanho do arquivo de transferência da máquina virtual entre 70% e 90% mantendo-se o tempo de inicialização do sistema, equivalente a uma máquina virtual.

3.3 *Dockers* como alternativa às VMs

Paralelamente à vertente de VMs tem se evoluído e amadurecido a tecnologia de virtualização de *Kernel* para criação de ambientes tão isolados quanto os das máquinas virtuais

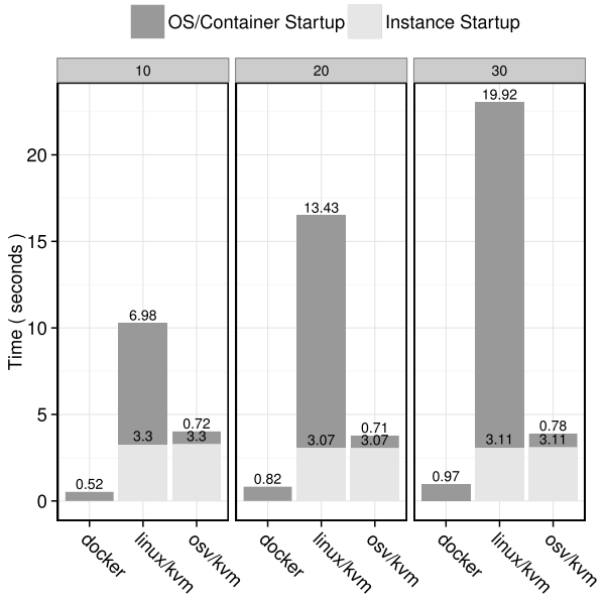

Figura 3.4 – Criação e inicialização de 10, 20 e 30 instâncias de Sistemas Operacionais/*Containers*.

Fonte: (XAVIER; FERRETO; JERSAK, 2016)

porém com uma economia de uso da memória principal e rápida carga do sistema, o LXC ou *Linux Container*. *Linux Conainer* é um método de virtualização a nível do sistema operacional que permite executar múltiplos sistemas Linux (*containers*) em um único *host* de controle. Não é implementado uma máquina virtual, mas sim um ambiente de virtualizado que possui sua própria CPU, memória, rede e espaço e mecanismo de controle, existem duas classificações de *containers*:

- OS *Container* - São ambientes virtualizados completos que compartilham o *kernel* do *host* hospedeiro mas oferece um isolamento do espaço de usuário. É possível implementar, configurar e executar diferentes aplicações, bibliotecas específicas, variáveis de ambiente próprias etc. Assim como em máquinas virtuais, o isolamento do ambiente é completo, pode executar vários serviços.

- *Application Container* - O seu propósito é isolar e executar um único serviço, o isolamento neste tipo de *container* se restringe ao processo que é executado, o kernel é compartilhado com o *host* igualmente como o OS *container*. Os principais exemplos são *Dockers* e *Rocket*.

Recentemente, uma das opções dos *Containers* de aplicação, o *Docker* tem se destacado por sua facilidade e otimização, sendo utilizado como interface de replicação de pesquisas (HUNG et al., 2016) por permitir a criação de aplicações em *containers* para sua distribuição e manutenção das informações em comum (BINET; COUTURIER, 2015) ou mesmo para

padronizar *softwares* de bioinformática para que sejam intercambiáveis (BELMANN et al., 2015). Ainda na mesma linha, algumas pesquisas como (FELTER et al., 2014) e (XAVIER; FERRETO; JERSAK, 2016) demonstram o nível de maturidade e consistência do escopo em que o *Docker* esta, além de apresentar resultados que validam a sua inicialização expressivamente mais rápida que os concorrentes como *OpenStack* que utilizam virtualizadores conforme, Figura 3.4.

Por fim, como forma de validar propostas que não estejam implementadas e possam gerar experimentos a exemplo do *OpenStack*, mostra-se necessário o uso de um simulador que possa modelar a situação proposta. O *CloudSim* proposto por Calheiros em (CALHEIROS et al., 2011) apresenta as características necessárias através de código aberto para personalização, além de ser qualificado para simulações de *Cloudlets* de monitoramento pessoal, como pode ser visto em (QUWAIDER; JARARWEH, 2015), em *Cloudlets* colaborativas, conforme em (BOHEZ et al., 2014) ou em armazenamento de nuvens móveis apresentado em (DURO et al., 2015).

3.4 Considerações Finais

Este capítulo ratifica a importância das *Cloudlets*, como foram implementadas assim como apresentando trabalhos que demonstram o desempenho mais eficiente dos *Dockers*, enfatizando a relação com a latência dos serviços. Os trabalhos citados neste capítulo, por serem bastante recentes, indicam a atualidade do assunto, demonstrando o limite atual de desenvolvimento das *Cloudlets*.

4 Mobilidade em *Cloudlet*

Nesta capítulo são discutidos os cenários em que há mobilidade de clientes e suas nuvens, são expostos os quadros em que os clientes migram e para manter a coerência com conceito de *Cloudlets* e baixa latência. Também é discutida a ausência de mobilidade no *CloudSim* e como foram incluídas novas funções para que a proposta pudesse ser avaliada através de simulação. Na seção 4.1 é justificado o porquê do deslocamento das *Cloudlets*, na seção 4.2 é descrito como essa mobilidade foi implementado no *CloudSim* e em sua subseção 4.2.1 apresentado como o atraso foi incluído.

4.1 Situações de mobilidade

Como descrito na subseção 2.2.1, a razão da *Cloudlet* é trazer os recursos da nuvem mais próximos aos clientes, garantindo uma baixa latência e a capacidade computacional necessária para execução da aplicação. Por esse motivo a existência desses recursos à distância máxima de um salto do cliente é imprescindível. Esse salto pode ser através de uma conexão por *WiFi* ou uma tecnologia de redes celulares[1].

4.1.1 *Cloudlets* em máquinas virtuais

O fato da criação da *Cloudlet* ser baseada em máquinas virtuais onerou perceptivelmente a maior premissa que os dispositivos móveis possuem: a mobilidade. A Figura 4.1 demonstra a situação inicial em que os clientes A, B, C e D estão com suas respectivas *Cloudlets* em execução em um mini-*datacenter* interno à célula da rede sem fio em que os clientes em questão estão conectados e precisam se deslocar para uma outra célula de rede sem fio. Nesta situação as *Cloudlets* estão em execução usando máquinas virtuais utilizando a *VM Base* como arquitetura e a migração dos clientes deveriam manter a execução das *Cloudlets* em máquinas virtuais.

Após o deslocamento dos clientes, sua comunicação com a *Cloudlet* ficará com maior latência pois a mudança de célula de rede sem fio aumentará ao menos um salto de distância até as máquinas virtuais. Para evitar esse salto maior, novas *Cloudlets* deverão ser criadas no mini-*datacenter* da nova célula da rede sem fio em que os clientes se associaram. Essa criação é otimizada por conta da *VM Base*, onde os clientes precisarão transferir para o servidor local a *VM Overlay*, consideravelmente menor que a máquina virtual completa que precisaria ser enviada. Porém essa transferência através da rede sem fio, que está sujeita a interferências ou intermitências, pode levar um tempo maior do que o ideal, resultando em uma experiência

[1] Tecnologias 4G ou 5G, neste caso é necessário uma velocidade e latência que as tecnologias de telefonia celulares anteriores não oferecem.

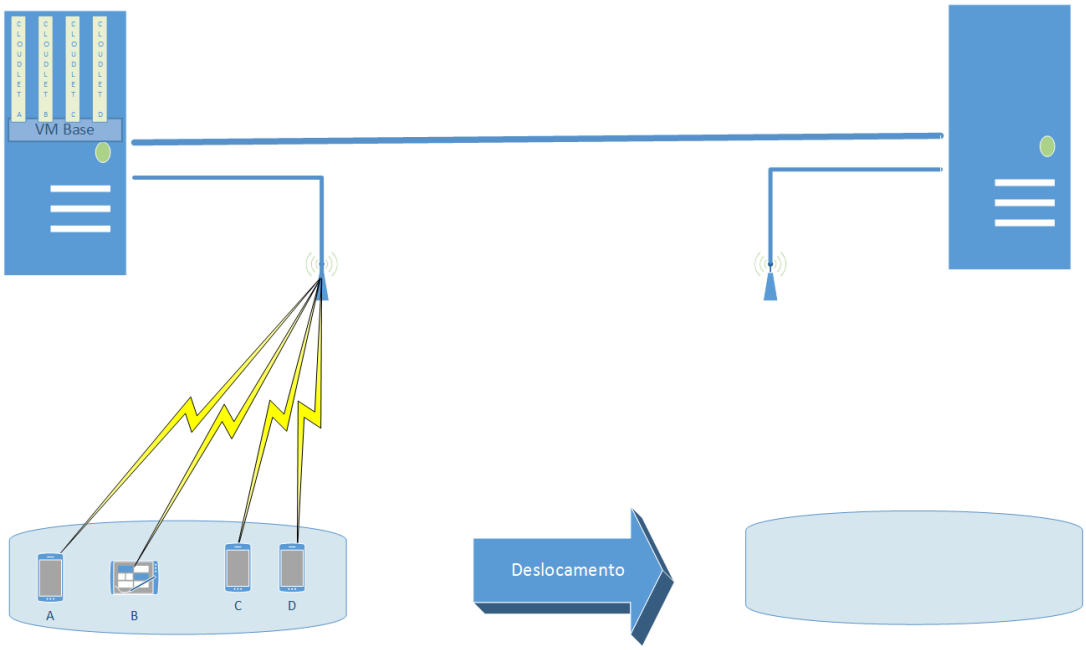

Figura 4.1 – Deslocamento dos Clientes.

Fonte: Produzido pelo autor

provavelmente desconfortável ao cliente. Na Figura 4.2 demonstra-se os clientes já associados à nova rede e transferindo a *VM Overlay* para o servidor localizado no mini-*datacenter*. Durante a transferência e instanciação, os clientes não podem fazer uso das *Cloudlets* e terão sua capacidade computacional drasticamente reduzidas, apresentando um desempenho insatisfatório e maior consumo de recursos do dispositivo móvel. Após a instanciação, novas *Cloudlets* poderão atender aos clientes, diferentes das que inicialmente faziam uso. A política de exclusão ou reuso das primeiras *Cloudlets* ficam a cargo do serviço implementado, em alguns casos elas são mantidas para reuso em caso de retorno ou associação de outros cliente, em outros casos são imediatamente excluídas para liberação de recursos a novos usuários.

Tanto na Figura 4.1 quanto na Figura 4.2 observa-se a existência de um enlace de comunicação entre os dois servidores de virtualização, ele esta presente pois presume-se que o usuário desloca-se para uma outra célula de rede sem fio do mesmo provedor de serviços, geralmente associado à operadora de telefonia celular ou empresa conveniada. Fazendo-se o reconhecimento de *Cloudlets* vizinhas através da *VM Base* torna-se possível descobrir também os mini-*datacenters* vizinhos que comportam novas *Cloudlets* e que fazem parte da célula de rede sem fio a que os seus clientes migraram. Com esse conhecimento a transferência das *Cloudlets* para o novo servidor pode fazer uso desse enlace, com uma velocidade muito superior à das rede sem fio dos clientes, e diminuir o tempo de transferência da *VM Overlay*, conforme visto na Figura 4.3. Nesta situação, as mesmas *Cloudlets* que atendiam aos clientes inicialmente migram junto com os usuários para a nova célula, levando consigo os dados na *VM Overlay* ao invés de copiar e criar outras instâncias.

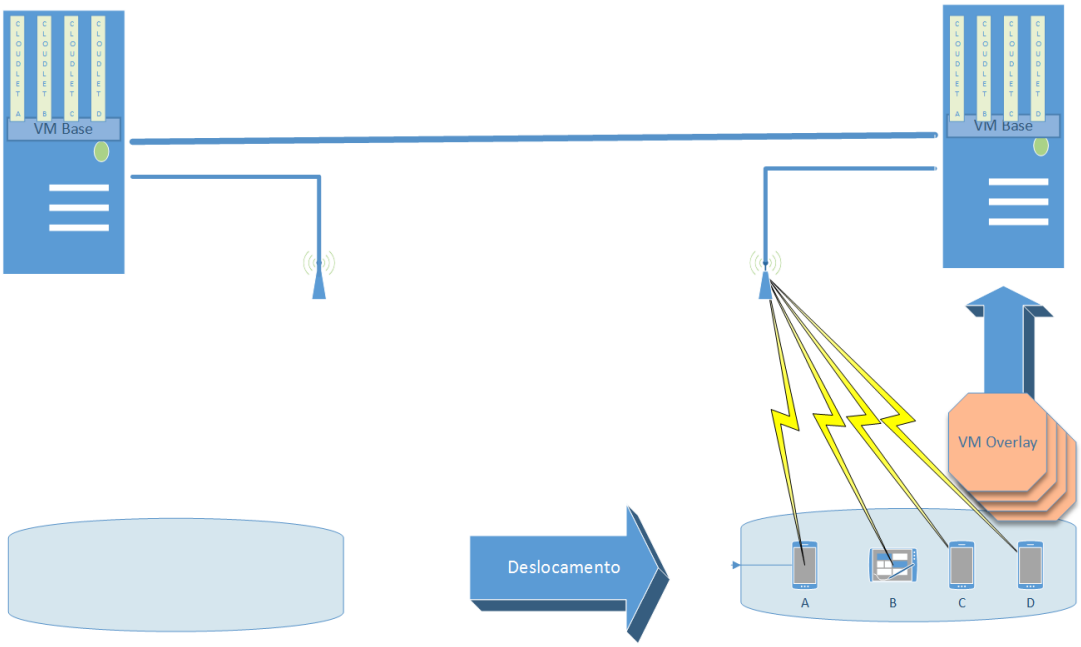

Figura 4.2 – Deslocamento da *Cloudlet*.

Fonte: Produzido pelo autor

4.1.2 Proposta de *Cloudlets* em *Dockers*

O modelo proposto nesta dissertação baseia-se igualmente na migração das *Cloudlets* para o novo mini-*datacenter* que atende aos clientes que se deslocaram, fazendo uso do conhecimento adquirido através das *Cloudlets*. Porém essa migração não utilizará máquinas virtuais, *VM Base* ou *VM Overlays*, ela se dará pelo uso de *Dockers Containers*, conforme apresentado na Figura 4.4.

Levando-se em conta sua maturidade enquanto produto conforme evidenciado em (HUNG et al., 2016) e (BINET; COUTURIER, 2015), bem como sua reduzida carga de inicialização, menor tamanho das imagens dos serviços se comparados a máquinas virtuais, menor quantidade de IOPs[2] e uso de memória principal, também em comparação a VMs demonstrados em (FELTER et al., 2014) e (XAVIER; FERRETO; JERSAK, 2016), considera-se a ferramenta *Docker* é mais adequada para solução de mobilidade das *Cloudlets*. A otimização quanto aos dados aplicados na *VM Overlay* para transferência apenas do diferencial da *VM Base* já existem de forma segurança e nativamente no sistema de arquivos dos *dockers*, sendo visível por serviços equivalentes com tamanho inferiores às opções virtualizadas. Suas chamadas de sistemas também são executadas nativamente pelo sistema operacional principal, evitando camadas de virtualização nescessárias que agregariam mais tempo de resposta a aplicações de *Cloudlets* mais exigentes.

Utilizando o *CloudSim* como simulador, visa-se, nesta dissertação, demonstrar que as

[2] Input/Output Operations Per Second - Operações de Entrada/Saaída por Segundo

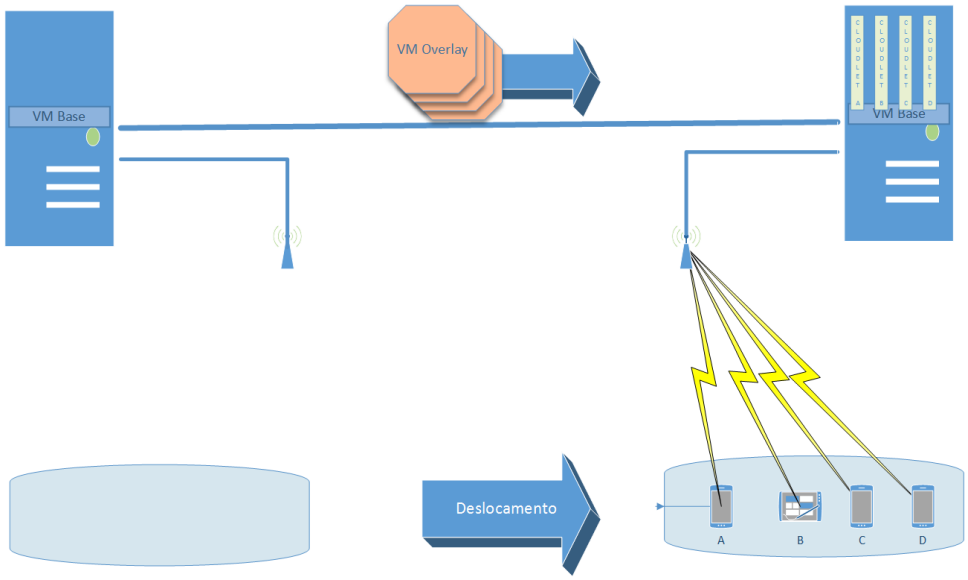

Figura 4.3 – Deslocamento da *Cloudlet*.

Fonte: Produzido pelo autor

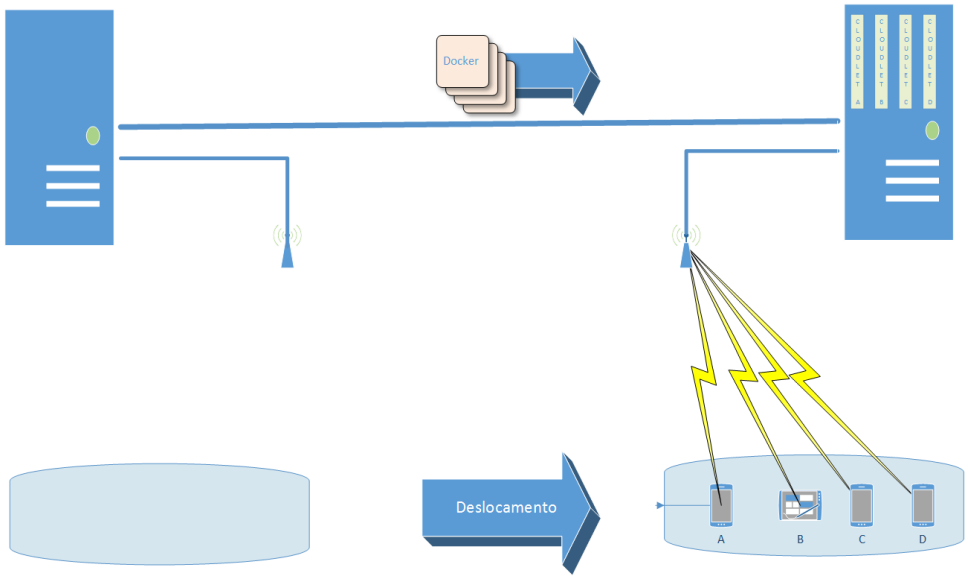

Figura 4.4 – Deslocamento da *Cloudlet*.

Fonte: Produzido pelo autor

vantagens apontadas do *Docker* são suficientes para justificar uma arquitetura que faça uso de *Containers* em detrimento das máquinas virtuais. A portabilidade das *Cloudlets* para *Containers* seriam uma atividade natural dada a voracidade pelo desempenho que as aplicações móveis necessitam.

4.2 Introduzindo Modelo de Mobilidade no *CloudSim*

O processo de deslocamento de serviços (*Cloudlets*) entre as Nuvens, ou *datacenters* não são nativos do *CloudSim*, ou seja não há uma extensão ou métodos que possibilitassem o atendimento dos objetivos propostos nesta dissertação no simulador escolhido, também não foram encontrados simuladores que fossem focados em Nuvens que permitissem a alteração requerida de forma modular. Para implementar essa funcionalidade seria necessário permitir a comunicação entre as Nuvens modeladas e em seguida estender para uma migração dos serviços que se comunicam. Por ser uma etapa fundamental para o trabalho, seria necessário um maior cuidado na inclusão da API do *CloudSim* e estender algumas funções já consolidadas no simulador.

Para o fim pretendido, as classes *DatacenterBroker* e *Datacenter*, apresentadas na Figura 2.3 foram reestruturadas, cujo resultado do processo pode ser visto na Figura 4.5. As etapas demonstradas na Figura 4.5 são descritas a seguir:

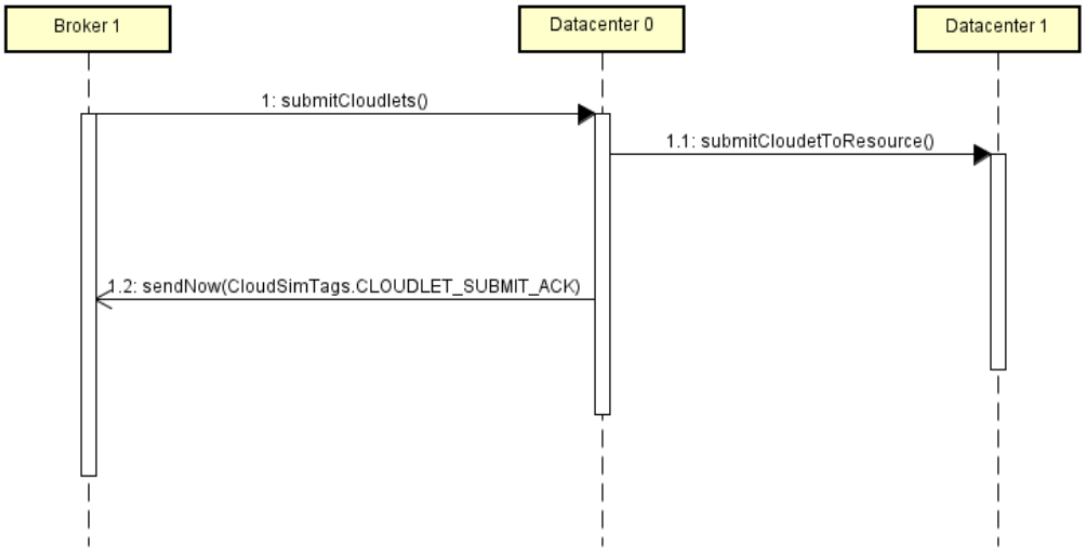

Figura 4.5 – Comunicação ente Nuvens.

Fonte: Produzido pelo autor

1. Inicialmente o *Broker 1*, gerente/operador do *datacenter* e detentor de um conjunto de *Cloudlets*, que foram incluídas na lista de *Cloudlets* do *Broker 1*, no instante em que define-se as configurações do cenário da simulação, inicia o processo que representará a *Cloudlet* no *Datacenter 0*. Para a instanciação será utilizado o método *sendNow()* com uma *tag* `CLOUDLET_STORE_ON_RESOURCE_AND_SUBMIT`, sem atraso de envio e inicialização uma vez que o intuito é avaliar a movimentação considerando que a *Cloudlet* já estava inicializada;

2. O *Datacenter 0* recebe a *Cloudlet* a partir do método *submitCloudletToResource()* e, como uma etapa do mesmo método, envia a *Cloudlet* para o *Datacenter 1*;

3. Ainda o método *submitCloudletToResource()*, em outra etapa, envia um ACK para o *Broker 1* confirmando o envio e recebimento da *Cloudlet* no *Datacenter 1*;

Conforme pode ser visto na Figura 4.6, o método *submitCloudletToResource()* executa outras operações para auxiliar no controle da transferência das *Cloudlets*. O *Datacenter 0* envia a *Cloudlet* para o *Datacenter 1* que em seguida um ACK confirmando o envio para o *Broker 1*.

```
/**
 * Move a cloudlet to any resource
 * @param ev
 */
protected void submitCloudetToResource(SimEvent ev){
    Cloudlet cl = (Cloudlet) ev.getData();
    cl.setVmId(1);
    Log.printLine(CloudSim.clock() + ": " + getName() + ": Sending cloudlet "
            + cl.getCloudletId() + " from resource: " + CloudSim.getEntityName(2) + " --> to resource: " + CloudSim.getEntityName(3));

    sendNow(3, CloudSimTags.CLOUDLET_SUBMIT, cl);

    //Método da transformação inversa da distribuição exponencial para gerar números de delay aleatórios
    // (-1/lambda)*ln(random)
    double Delay = (-0.1*(Math.log(Math.random())));
    sendNow(cl.getUserId(), CloudSimTags.CLOUDLET_SUBMIT_ACK, cl, Delay);
}
```

Figura 4.6 – Métodos de envio de um *Datacenter* a outro

.

Fonte: Produzido pelo autor

4.2.1 Latência para mobilidade de *Cloudlets*

Para o cálculo da mobilidade das *Cloudlets* no *CloudSim* é importante a inclusão da latência de rede na transferência. Para isso o *CloudSim* já oferece uma API satisfatória e bem aceita em trabalhos científicos (BUYYA; RANJAN; CALHEIROS, 2009), (SHI; JIANG; YE, 2011), (MISHRA; TONDON, 2016) e (MADHU et al., 2016) que o utilizam, a classe *InfoPacket* oferece informações como largura de banda, RTT e outras informações. Nesta customização, utiliza-se o RTT ofertado pelo *CloudSim* como base da informação da latência que a rede apresentará.

Para cálculo da latência são instanciados 5 objetos *InfoPacket*, que se comportam como pacotes ICMP. Na instanciação da simulação, esses objetos oferecem informações relativas ao tempo de entrada e saída de pacotes no *Datacenter 0* e no *Datacenter 1*. A partir desses tempos, com os RTT oferecidos nos objetos calcula-se o tempo de atraso dos pacotes entre os *datacenters*. O ciclo informado pelos 5 pacotes (objetos *InfoPacket*) podem ter sua sequência observada na Figura 4.7.

O fluxo da Figura 4.7 é repetido em cada um dos 5 pacotes e com os valores obtidos é incluída uma média representando o atraso do enlace entre os *datacenters* envolvidos.

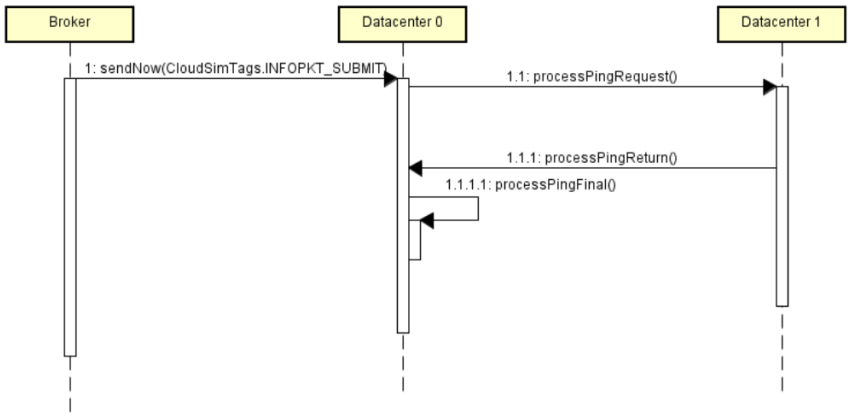

Figura 4.7 – Métodos para cálculo de latência

.

Fonte: Produzido pelo autor

4.3 Considerações Finais

Este capítulo apresenta a situação de mobilidade a que as *Cloudlets* são submetidas e localiza a proposta desta dissertação, sua implementação em *Dockers*, que permitirá um melhor desempenho dadas suas características. A implementação em máquinas virtuais gera mais sobrecarga que em *Dockers*. Também é apresentada uma notória contribuição à comunidade que é a extensão do *CloudSim* para que seja possível a avaliação das situações de mobilidade dos serviços envolvidos.

5 Simulação e Análise de Resultados

Este capítulo descreve como foi implementado o cenário para simulação com suas comparações entre as alternativas de *dockers* e máquinas virtuais. Na seção 5.1 é descrito o cenário de mobilidade proposto para esta dissertação, na seção 5.2 detalha-se a modelagem dos cenários para sua simulação através do *CloudSim* e na seção 5.3 os resultados obtidos da simulação são discutidos e comparados para uma melhor análise.

5.1 Cenário

O cenário das simulações utilizou como premissa o exposto no Capítulo 4, onde os clientes se deslocam entre as células de rede sem fio de um mesmo provedor de serviço, onde seus mini-*datacenters* encontram-se interconectados por um enlace de alta velocidade e baixa latência.

Para permitir a mobilidade das *Cloudlets* foram criados dois mini-*datacenters*, μDC1 (descrito como *Datacenter 0* na seção 4.2) e μDC2 (Descrito como *Datacenter 1* na seção 4.2), interconectados por um enlace, FO1, de largura de banda 1 Gbps em fibra ótica com uma latência de 17 ms, conforme média apresentada em (FCC's Office of Engineering and Technology and And Consumer and Governmental Affairs, 2015) para enlaces de fibra ótica dentro da mesma operadora. O diagrama da descrição pode ser observado na Figura 5.1.

Os clientes se conectaram aos mini-*datacenters* através de uma rede sem fio com velocidades assimétricas para *Downlink* (8,8 Mbps) e *Uplink* (6,2 Mbps) e latência de 118 ms conforme apresentado na Figura 5.1, distribuídos entre dois enlaces sem fio, W1 e W2. As medidas utilizadas se baseiam na média da velocidade e latência da tecnologia 4G auferidas no Brasil, apresentadas em (REPORT, 2016) e distribuídas conforme Tabela 5.1.

Tabela 5.1 – Características dos Enlaces.

Enlace	***Uplink* (Mbps)**	***Downlink* (Mbps)**	**Latência (ms)**
FO1	1000	1000	17
W1	6,2	8,8	118
W2	6,2	8,8	118

Fonte: Produzido pelo autor

Todas as máquinas que estão fisicamente alocadas nos mini-*datacenters* apresentam a mesma configuração de *hardware*, possuindo 8 GB de memória principal, 1 TB de memória secundária, um processador *Intel*^{R} i7 com 4 núcleos físicos com capacidade de 14.564 MIPS (equivalente a 2,2 Ghz). Porém, essas máquinas não estão distribuídas equitativamente entre

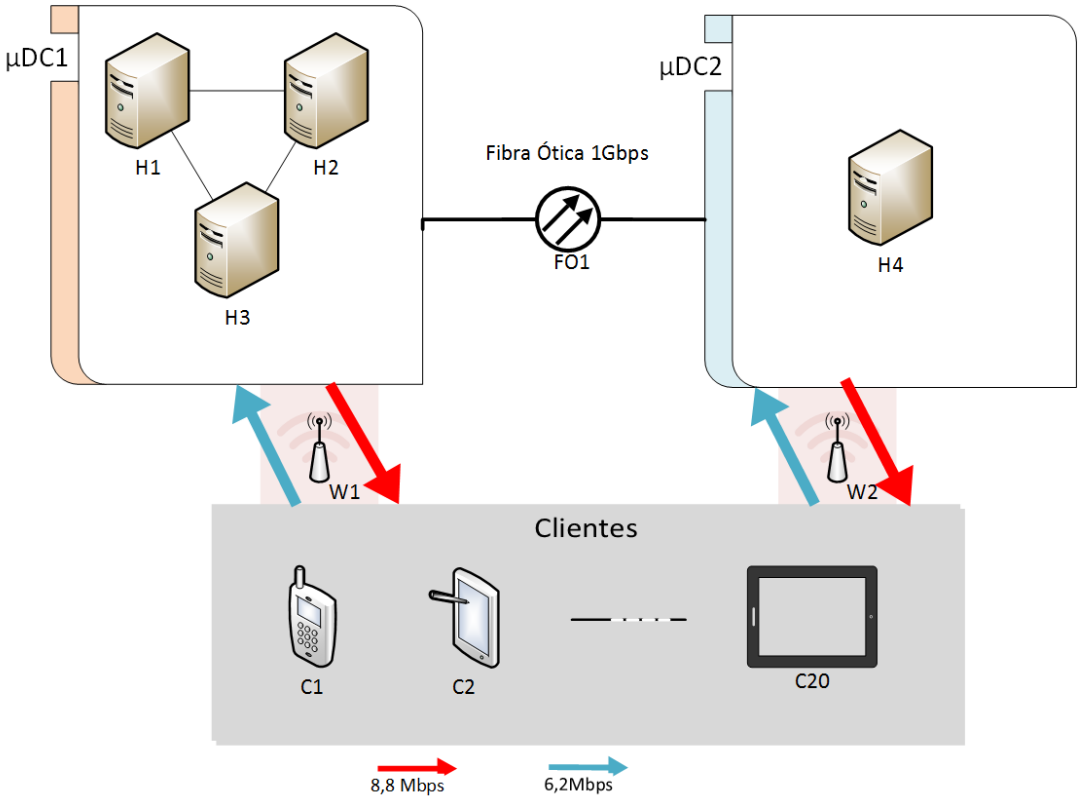

Figura 5.1 – Cenário simulado.

Fonte: Produzido pelo autor

os mini-*datacenters*, uma está alocada em μDC2 e três estão em μ DC1atuando como *cluster*, corforme detalhes da Tabela 5.2.

Tabela 5.2 – Características dos *Datacenters*.

Datacenter	***Host***	**Memória (GB)**	**Núcleos**	**MIPS**	**Armazenamento (GB)**
μDC1	H1	8	4	14564	1000
μDC1	H2	8	4	14564	1000
μDC1	H3	8	4	14564	1000
μDC2	H4	8	4	14564	1000

Fonte: Produzido pelo autor

No cenário proposto 20 clientes estão inicialmente sendo atendidos pelo μDC1 através do enlace W1 e se deslocam para o μDC2 se associando ao enlace W2, consequentemente suas *Cloudlets* alocadas devem migrar, acompanhando, do μDC1 para μDC2. Essa migração em massa deve gerar um impacto relacionado à latência e à percepção dos clientes.

A situação deve ser repetida para duas tecnologias distintas, a original proposta utilizando máquinas virtuais e suas otimizações com *VM Base* e *VM Synthesis* e uma outra utilizando suas medidas equivalentes através de *containers* em *Dockers*. Ambas utilizando para transferência o enlace FO1 que interconecta μDC1 e μDC2.

5.2 Simulação

Para simulação foram criados dois cenários, *Cenário A* representando a utilização das máquinas virtuais com *VM Synthesis* e *VM Base*, e o *Cenário B* com as *Cloudlets* executadas em *Dockers* no sistema operacional nativo. Para sua execução utilizou-se como máquina física um Intel^{R} Core^{TM} i5-2410M, com 4,00 GB de memória RAM e Sistema Operacional Ubuntu 16.04 LTS x64, utilizando Java^{TM} SE Runtime Environment 1.8.0 e IDE Eclipse Mars 2 para modificar o *CloudSim*.

O simulador utilizado foi o *CloudSim* descrito na seção 2.3, porém por não modelar a movimentação das *Cloudlets* foi necessário implementar uma extensão que garantisse o comportamento de migração dos clientes e das *Cloudlets*. Por ser de código livre utilizando a linguagem de programação Java, sua extensão baseou-se na documentação do simulador, o que demandou um considerável esforço de compreensão.

A descrição de ambos cenários foi gerada utilizando BRITE[1], uma ferramenta utilizada pelo *CloudSim*, também usada no *Network Simulator* (HECKMANN et al., 2003) e no OmNet++ (VARGA et al., 2001), responsável por descrever a topologia desejada para simulação. BRITE foi utilizada pelo *CloudSim* por ser uma ferramenta flexível, extensível, interoperável, portável e amigável, garantindo uma compreensão detalhada de características de situações de larga escala representadas em topologias físicas. A descrição da topologia dos cenários pode ser analisada no Apêndice A.

A *Cloudlet* foi descrita como um serviço com 10.000 MIPS a serem executados no μDC1 e em seguida migrado para o μDC2, onde no *Cenário A* ele seria executado em máquinas virtuais e no *Cenário B* em *Dockers*, ambas com saída de 2,4 MB, o equivalente a página web média, conforme descrito em (HTTPARCHIVE, 2016). Na simulação, essa saída é representada pela variável *outputsize*.

No *Cenário A*, quando representados por uma VM, cada máquina virtual teria como características:

- Tamanho de imagem da VM: 10.000 MB;
- Memória Principal (RAM): 1 GB;
- MIPS: 1.000;
- Largura de Banda (representada por *bw* na descrição da topologia): 1 Gbps;
- Número de Processadores (representado por *pesnumber* na descrição da topologia): 1;
- Virtualizador (representado por *vmm* na descrição da topologia): KVM[2];

[1] Boston university Representative Internet Topology gEnerator - A saída dessa ferramenta consiste em um arquivo padronizado e identado com as características que serão interpretadas pelo simulador.

[2] Virtualizador utilizado pelo OpenStack++, onde Starianmam implementou a *VM Base* e *VM Synthesis*

No *Cenário B* não há implementação de máquinas virtuais uma vez que *Dockers* tratam as *Cloudlets* como serviços do próprio sistema operacional nativo, mas ainda assim é alocado um processador virtual para o *Docker* que executa uma *Cloudlet*.

Nos dois casos um serviço com 10.000 MIPS representa uma *Cloudlet* que migra de uma máquina do μDC1 (H1, H2 ou H3) para o μDC2 (H4) utilizando o enlace FO1 descritos em Figura 5.1. O serviço que a *Cloudlet* representa é o Gabriel[3] (HA et al., 2014), cujo tamanho para transferência entre os mini-*datacenters* é de 710 MB. No *Docker* o serviço equivalente para migração tem 182 MB, essa diferença substancial se deve ao fato do modelo de sistemas de arquivos adotado pelo *Docker* que garante uma otimização de compartilhamento dos requisitos em relação ao sistema de arquivos do sistema operacional.

A migração nos cenários deverá ser iniciada pelos 20 clientes que inicialmente estão associados ao μDC1 utilizando o enlace W1 e se deslocam para o μDC2 passando a utilizar o enlace W2. O deslocamento é gradual com a quantidade de clientes e sua análise deve acompanhar o deslocamento. Os parâmetros utilizados e descritos em BRITE foram:

- *lenght*: Tamanho da *Cloudlet* (em quantidade de instruções);
- *filesize*: Tamanho do arquivo de entrada (tamanho do *Docker* ou da VM);
- *outputsize*: Saída (Uma página web média);
- *pesnumber*: Quantidade de processadores;
- *utilizationmode*: Distribuição de processamento (*full*);

Cada cenário foi executado 50 vezes com os clientes migrando gradualmente, garantindo um intervalo de confiança de 95%, julgado suficiente para garantir a qualidade dos resultados dos experimentos deste trabalho.

5.3 Análise dos Resultados

A partir das 50 simulações em cada cenário foi possível traçar um comportamento do que deve acontecer na situação proposta e na movimentação das *Cloudlets* máquinas virtuais ou *dockers*. A descrição dos gráficos dos resultados podem ser vistas a seguir com suas respectivas considerações.

[3] Protótipo de uma arquitetura de um sistema de assistência baseado no *Google Glass* para usuários com dificuldades cognitivas. Ele combina uma imagem capturada em primeira pessoa e a capacidade dos óculos executar remotamente a informação gerando uma interpretação em tempo real da imagem com retorno de resposta para o usuário.

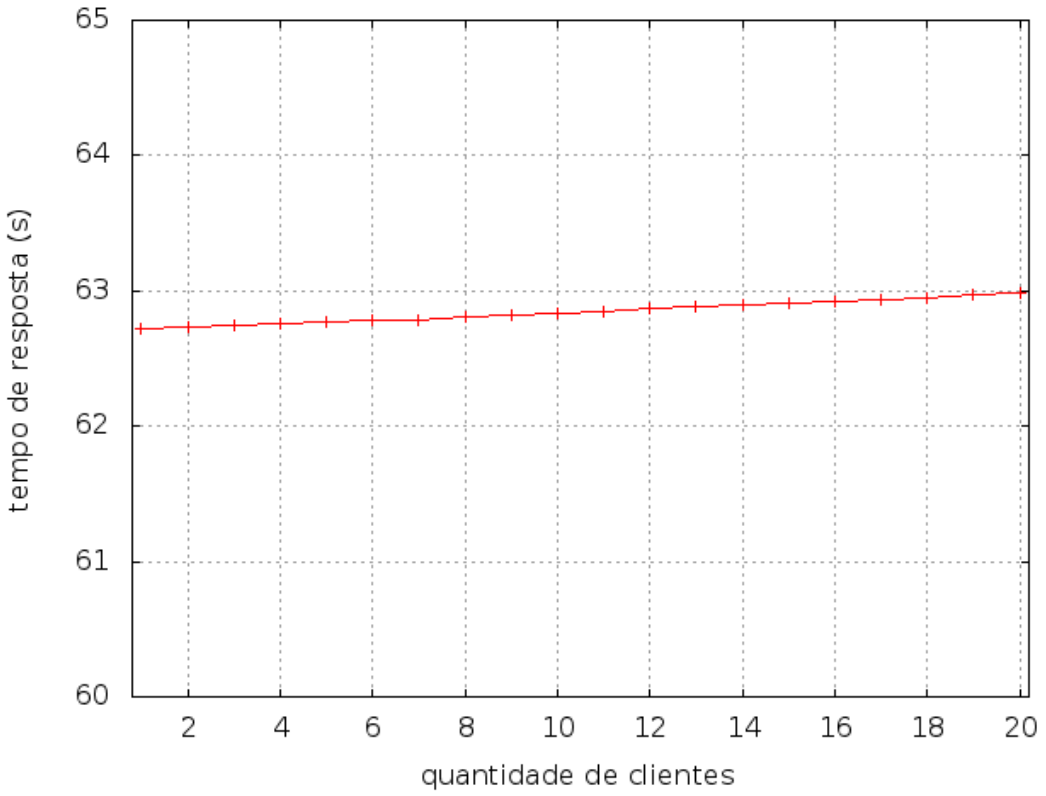

Figura 5.2 – Gráfico de inicialização das máquinas virtuais dos clientes no μDC1.

Fonte: Produzido pelo autor

O cenário propõe que as máquinas virtuais sejam inicializadas no μDC1. O tempo que cada VM leva para começar a atender o serviço é contabilizado considerando apenas um mini-*datacenter*. Na Figura 5.2 pode-se observar o tempo de resposta que as máquinas virtuais levam para instanciar, entre 1 e 20 clientes. Cada ponto demonstrado na linha do gráfico representa o tempo que uma máquina virtual leva para executar; percebe-se que não há uma diferença expressiva no tempo entre a primeira máquina virtual precisa para executar ou da vigésima. Cada VM necessita entre 62,718 e 62,988 segundos para inicializar e servir.

O experimento prossegue com a inclusão do μDC2 no cenário, com isso há a possibilidade de migração das *Cloudlets* entre os mini-*datacenters*. Nas simulações representou-se a migração de metade dos clientes para a rede sem fio atendida pelo μDC2, consequentemente o deslocamento de metade das *Cloudlets*. Na Figura 5.3 é perceptível o aumento do tempo de resposta de alguns clientes em comparação a Figura 5.2, os clientes com identificadores pares (0,2,4,....,20) representam a metade das *Cloudlets* que migraram apresentando um atraso maior, naturalmente. Neste caso as máquinas virtuais apresentam um atraso entre 62,718 *s* e 64,018 *s* na última máquina virtual inicializada, igualmente visível o maior tempo das máquinas do μDC2 causados pelo menor poder computacional do mesmo e a latência do enlace que comunica os dois mini-*datacenters*.

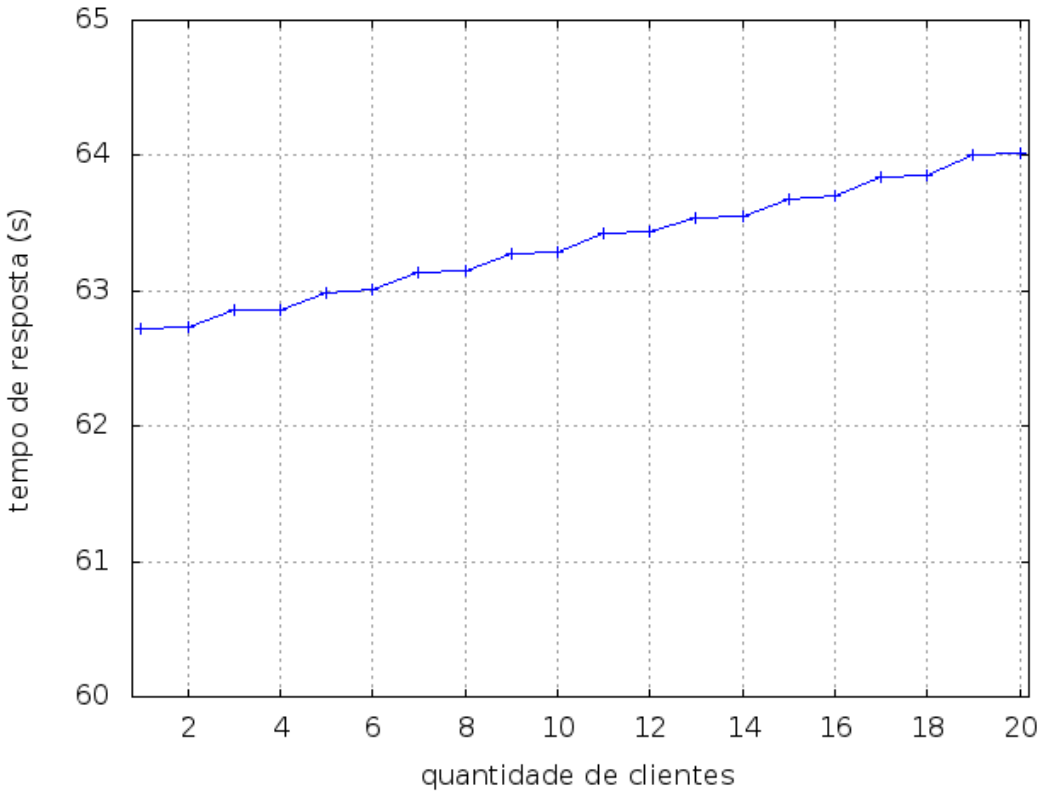

Figura 5.3 – Gráfico de inicialização das máquinas virtuais dos clientes com migração para o μDC2.

Fonte: Produzido pelo autor

Esses resultados demonstrados nos gráficos das VMs apresentam-se condizentes com os exibidos em (HA et al., 2015), onde é apresentado o resultado de mobilidade de *Cloudlets* envolvendo *OpenStack++*, utilizando a *VM Synthesis* e a *VM Base* como modelos das máquinas virtuais.

Na Figura 5.4 inicia-se a análise do uso de *dockers* em *Cloudlets*, conforme descrito na seção 5.2. O serviço de *Cloudlet* não é implementado nativamente em *dockers* por isso foi necessário sua modelagem com quantidade de instruções, tamanho da imagem, tamanho e frequência das mensagens proporcionais ou equivalentes aos das máquinas virtuais.

O tempo de inicialização no μDC1 é sensivelmente inferior ao das VMs conforme visto na Figura 5.4 e condizente com os apresentados em (XAVIER; FERRETO; JERSAK, 2016), onde cada *docker* necessita de 3,718 a 3,984 segundos para ser instanciado e responder a requisições. A diferença do tempo de resposta entre o primeiro e o vigésimo *docker* é relativamente baixa devido ao seu modelo de implementação utilizando *containers* e otimizando a necessidade de IOPs. Assim como na Figura 5.2, cada ponto representa o tempo de instanciação de um *docker* e, apesar de ser o mesmo serviço, seu tempo não permanece constante devido à carga acumulada requerida no μDC1 em cada *docker* novo solicitado.

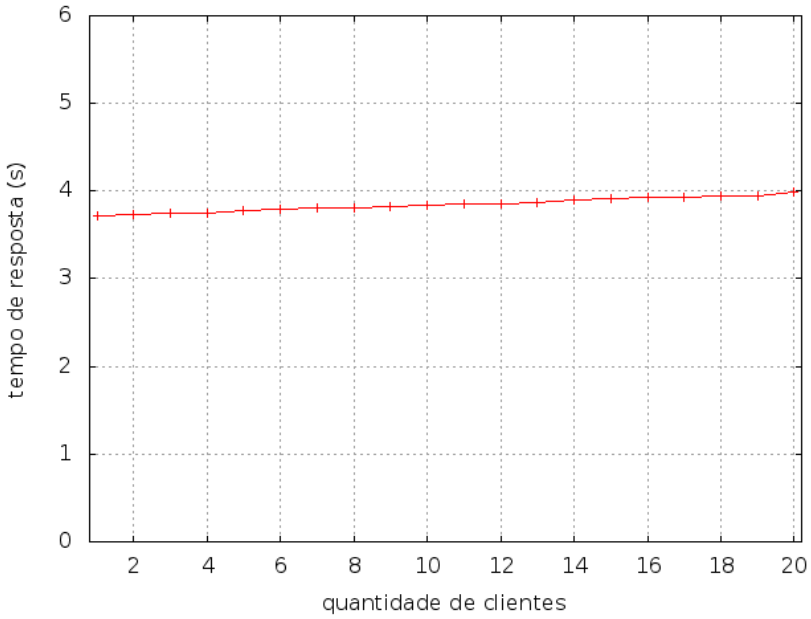

Figura 5.4 – Gráfico de inicialização dos *dockers* dos clientes no μDC1.

Fonte: Produzido pelo autor

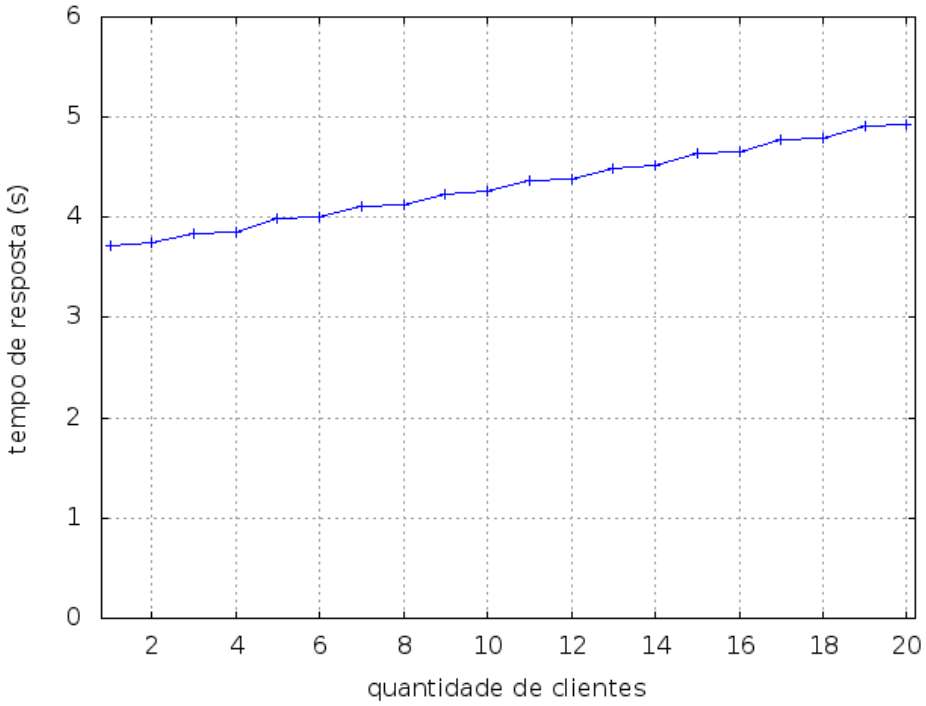

Figura 5.5 – Gráfico de inicialização dos *dockers* dos clientes com migração para o μDC2.

Fonte: Produzido pelo autor

A Figura 5.5 refere-se à inclusão de um segundo mini-*datacenter*, μDC2, na simulação onde metade dos *dockers* existentes na Figura 5.4 serão deslocados do μDC1 para o μDC2, similar ao demonstrado na Figura 5.3 para máquinas virtuais. Na Figura 5.5, os *dockers* com identificadores pares (2,4,6,...,20) são instanciados no μDC2 acumulando-se o atraso para transferência das imagens de um mini-*datacenter* para outro. Observa-se que assim como na Figura 5.3, os clientes com identificadores pares têm uma curva mais acentuada de tempo de resposta se comparados aos clientes com identificadores ímpares. Isso se deve ao fato da transferência da imagem do serviço do μDC1 para o μDC2, conforme esperado.

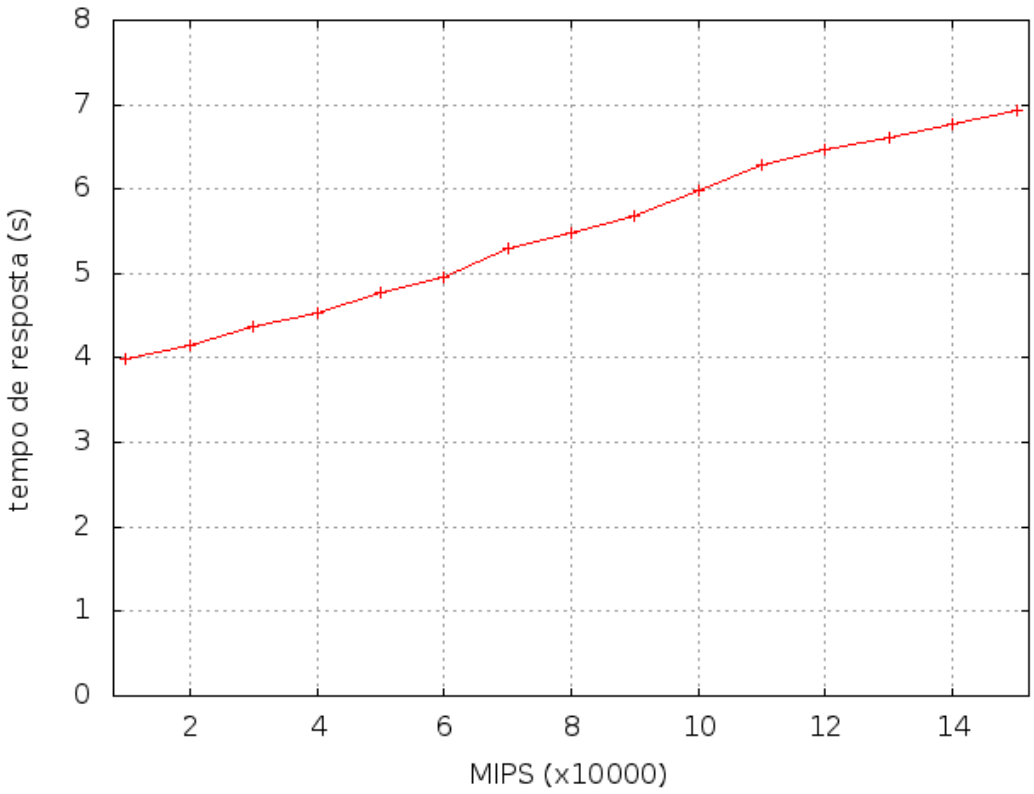

Figura 5.6 – Gráfico demonstrando aumento da carga de processamento das *Cloudlets*.

Fonte: Produzido pelo autor

O gráfico da Figura 5.6 demonstra uma outra abordagem levando em consideração a carga exigida pelo serviço. O acúmulo dessa carga deve aumentar o tempo de atraso dos serviços apresentando uma situação que o atraso gerado pelo processamento se iguale ao apresentado pela inicialização e transferência das imagens dos serviços. Nesta situação é perceptível o aumento do atraso, chegando ao pico de 6,937 segundos de atraso, suas curvas mostram-se mais acentuadas, diferentemente de quando os *dockers* exigiam apenas 10.000 MIPS[4]. A sobrecarga gerada por acréscimos de 10.000 em todos os 20 clientes mostram-se mais contundentes em apenas um mini-*datacenter*.

[4] *Millions of Instructions Per Second* - Milhões de Instruções por Sergundo.

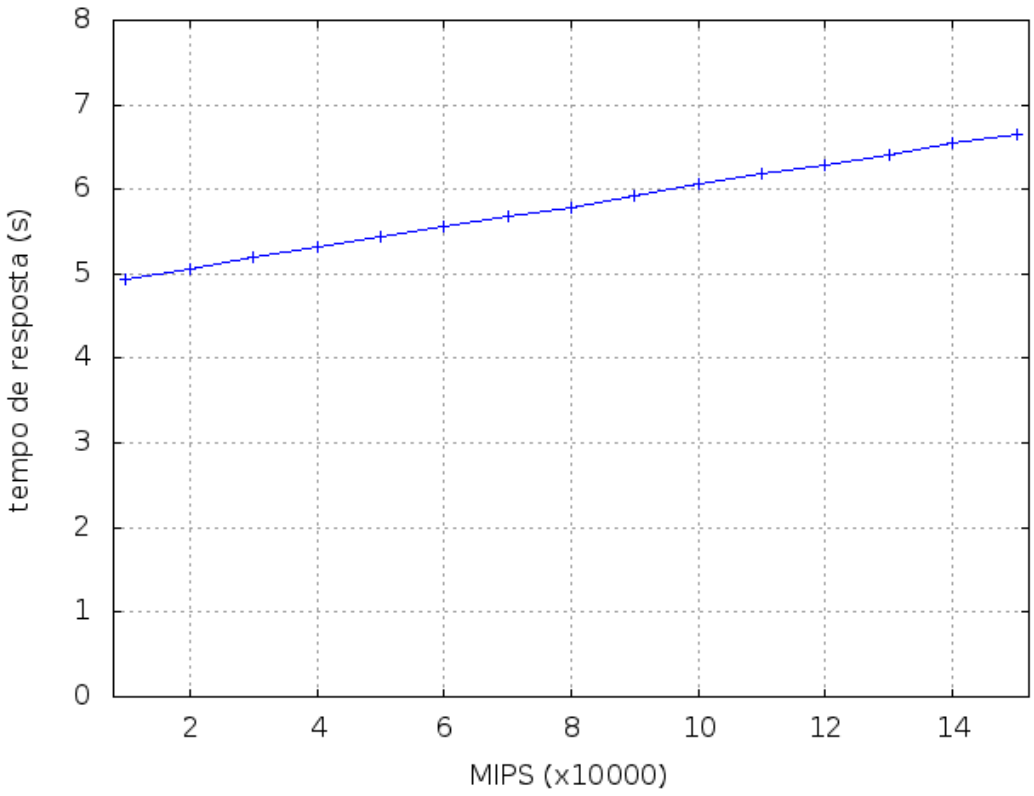

Figura 5.7 – Gráfico demonstrando aumento da carga de processamento das *Cloudlets* para transferências para o μDC2.

Fonte: Produzido pelo autor

Na Figura 5.7 apresenta-se *dockers* que migram para o μDC2, como nessa situação a carga de processamento apresenta-se relevante no montante do atraso, sua distribuição entre os mini-*dataceters* suavizou a inclinação das curvas de acréscimo de atraso das *Cloudlets*. Tal suavização deve favorecer a justificativa de migração e distribuição entre os mini-*datacenters*, situação que deve se apresentar equivalente ao acréscimo de mais *Cloudlets* exigindo mais processamento.

Esses resultados apresentados nos gráficos demonstram um comportamento condizente com os artigos relacionados a *Cloudlets* utilizando máquinas virtuais e serviços em *dockers* comparados às mesmas máquinas virtuais, validando o comportamento do *Cloudsim* com as implementações de simulação de mobilidade para *Cloudlets*, não presente no simulador originalmente distribuído. Permitindo que resultados da simulação sirvam para avaliação de cenários reais que utilizem *OpenStack++* para execução de *Cloudlets*, utilizando máquinas virtuais ou *dockers*.

Nota-se através do gráfico da Figura 5.8 que as máquinas virtuais em um mini-*datacenter* apresentam um tempo de resposta inferior ao resultante da simulação da mobilidade de metade das *Cloudlets* para o μDC2. Isso é naturalmente o esperado pois o processo de migração das

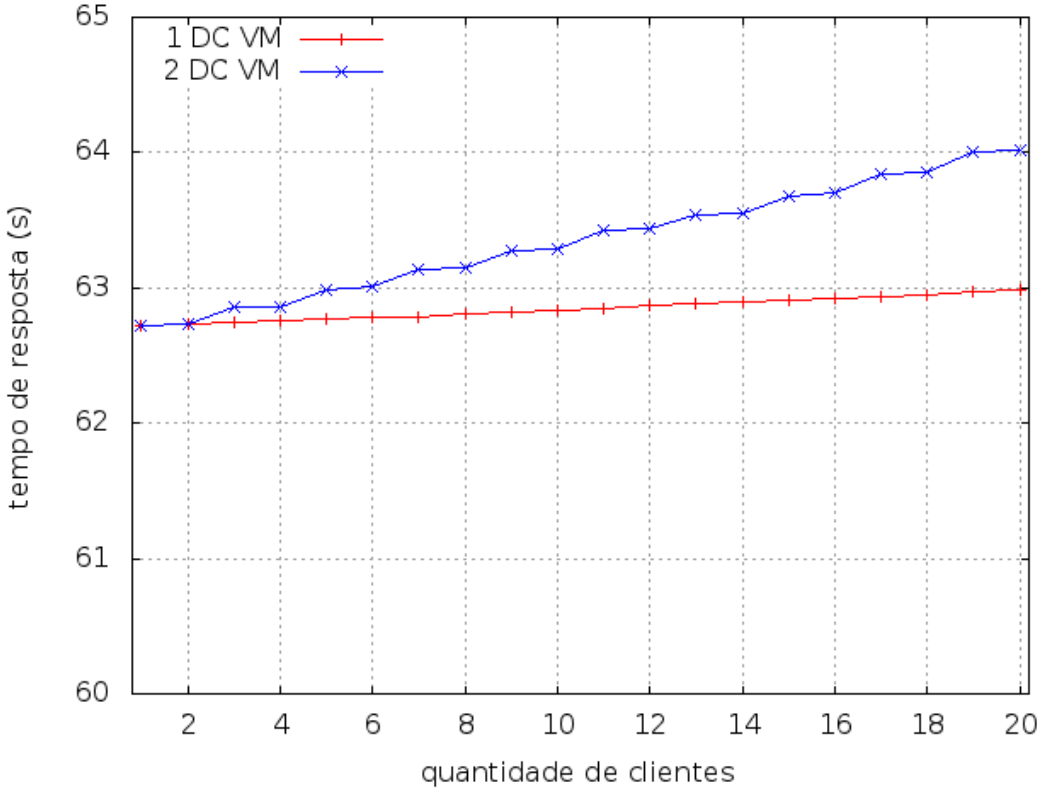

Figura 5.8 – Gráfico comparativo das máquinas virtuais em um ou dois *datacenters*, com deslocamento.

Fonte: Produzido pelo autor

Cloudlets mostrou-se oneroso, deve-se recordar que a migração envolve suspensão, transferência e retomada da execução no mini-*datacenter* destino. Se comparar a situação existente com sua mobilidade apresentada em (HA et al., 2015) utilizando *OpenStack++* e VMs percebe-se o total acordo com os resultados.

Na Figura 5.9 encontra-se o gráfico equivalente ao da Figura 5.5, porém utilizando a tecnologia de *docker container* ao invés de máquinas virtuais. Neste gráfico é possível observar, com a diferença entre as curvas, que a tecnologia de *dockers* sofre mais com a migração de mini-*datacenters* pois houve um acréscimo de aproximadamente 32% no tempo de resposta das *Cloudlets* que migraram. Porém após uma análise mais detalhada dos motivos percebe-se que a razão deste acréscimo acentuado refere-se ao tempo de transferência das imagens entre os mini-*datacenters*.

Apesar do tamanho das imagens dos *dockers* serem sensivelmente menores que os das máquinas virtuais equivalentes, sua transferência gera um impacto relativamente superior aos das VMs devido ao baixo tempo de atraso originalmente notado na inicialização dos *dockers* no μDC1, mesmo um valor pequeno de atraso, se agregado ao baixo tempo de atraso seria notado, contudo muito inferior ao agregado das máquinas virtuais.

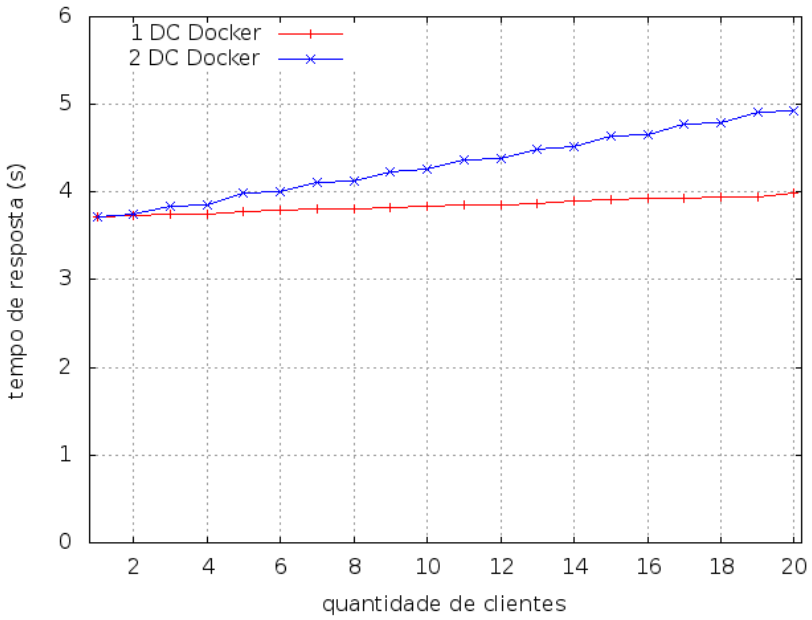

Figura 5.9 – Gráfico de movimentação dos *dockers* dos clientes entre as Nuvens.

Fonte: Produzido pelo autor

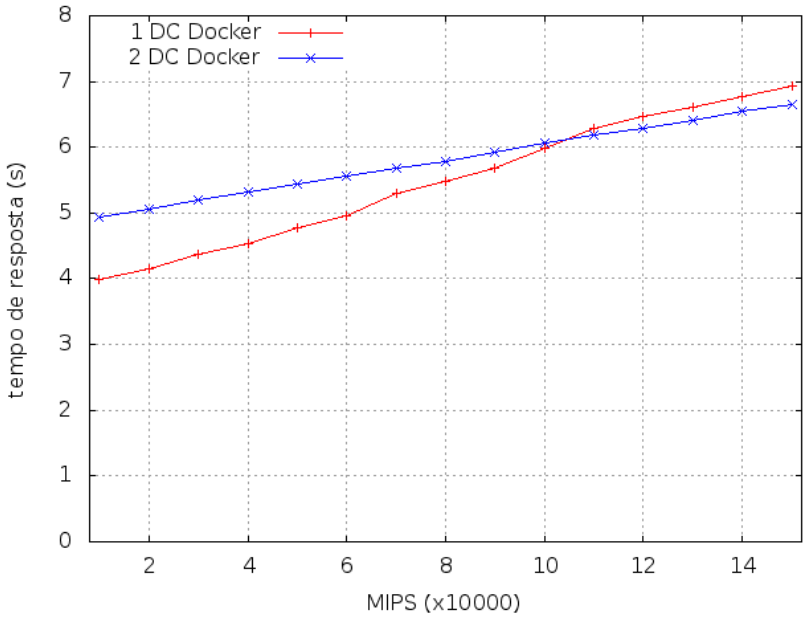

Figura 5.10 – Gráfico comparativo entre máquinas virtuais e *dockers* movimentando-se entre os mini-*datacenters* com variação da carga de processamento.

Fonte: Produzido pelo autor

A Figura 5.10 apresenta o gráfico de análise das soluções em relação à quantidade de processamento exigida por cada *Cloudlet*, após uma quantidade 110.000 MIPS de carga de processamento exigida em cada *docker*, sua interferência no comportamento da *Cloudlet* no mini-*datacenter* atinge um patamar em que sua migração se justifica igualmente pelo atraso gerado pelo próprio mini-*datacenter*. A convergência do tempo de atraso demonstrado na Figura 5.10 demonstra o momento em que passa ser mais eficiente a migração da *Cloudlet* do μDC1 para o μDC2, sendo otimizado pelo uso de *dockers*.

Ao ser considerado a taxa de processamento de 150.000 MIPS (último ponto da Figura 5.10), o atraso gerado por essa carga em um *datacenter* passa a sobrepor o atraso existente na rede, justificando o uso de dois *datacenters* e a mobilidade das *Cloudlets* conforme mostrado na Figura 5.10. A análise desse cenário com acréscimos de até 20 clientes onde metade se movimentam pode ser visto na Figura 5.11.

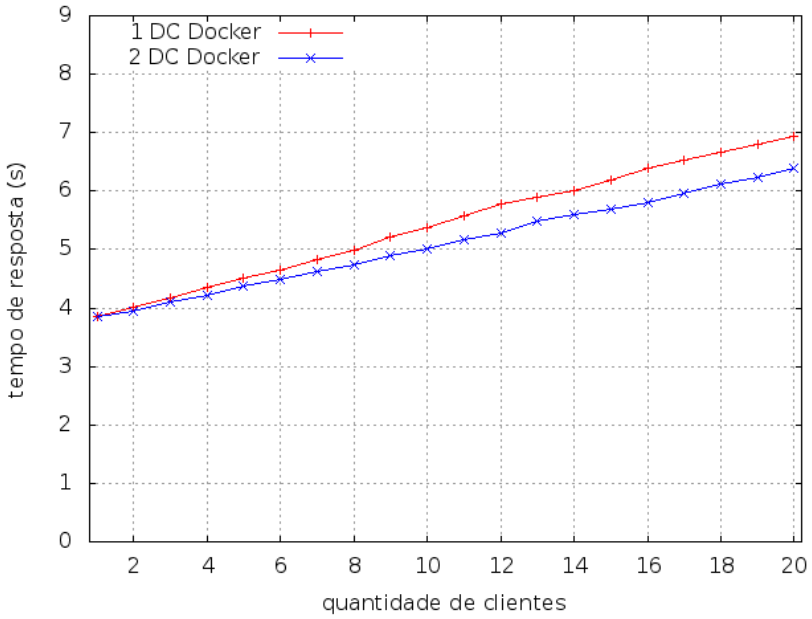

Figura 5.11 – Gráfico comparativo de *dockers* com carga de processamento de 150.000 MIPS para um ou dois *datacenters*.

Fonte: Produzido pelo autor

Na Figura 5.11 o atraso gerado para um *datacenter*, utilizando *dockers*, varia de 3,858 segundos com um cliente até 6,935 segundos com 20 clientes, porém com 2 mini-*datacenters* essa variação, com a mesma quantidade de clientes e movimentação de metade deles varia entre 3,858 segundos e 6,395 segundos. Isso demonstra um ganho de 8,44% com 20 clientes, e um ganho médio de 4,31%.

O mesmo comparativo executando 150.000 MIPS é apresentado na Figura 5.12 utilizando máquinas virtuais, nela é visível que para um mini-*datacenter* a execução varia de um atraso de 62,858 segundos com um cliente até 65,986 segundos com 20 clientes. Para dois *datacenters*

essa variação é de 62,858 segundos até 65,431 segundos com a mesma quantidade de cliente. Neste cenário, dois mini-*datacenters*, com a movimentação de metade das máquinas virtuais entre eles, é obtido um ganho de aproximadamente 1% no atraso na melhor situação.

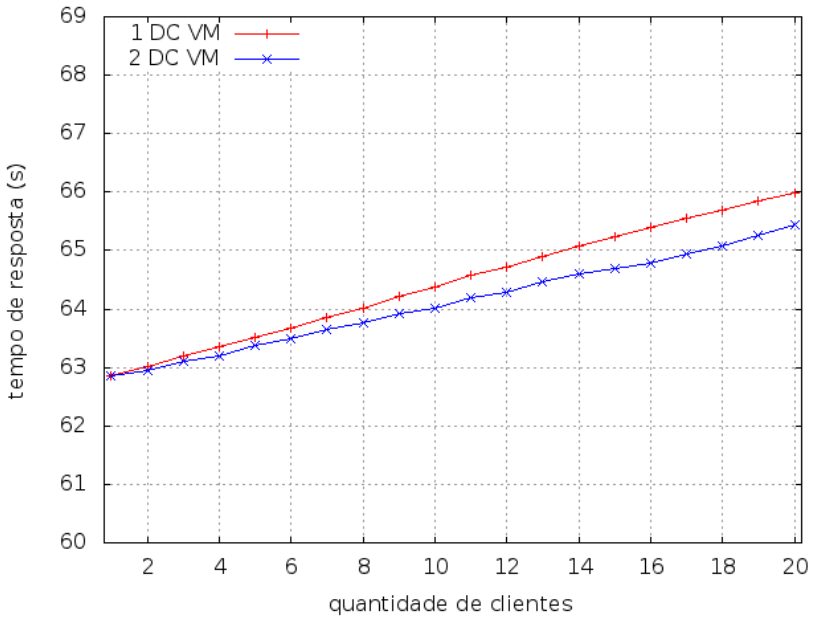

Figura 5.12 – Gráfico comparativo das máquinas virtuais com carga de processamento de 150.000 MIPS para um ou dois *datacenters*.

Fonte: Produzido pelo autor

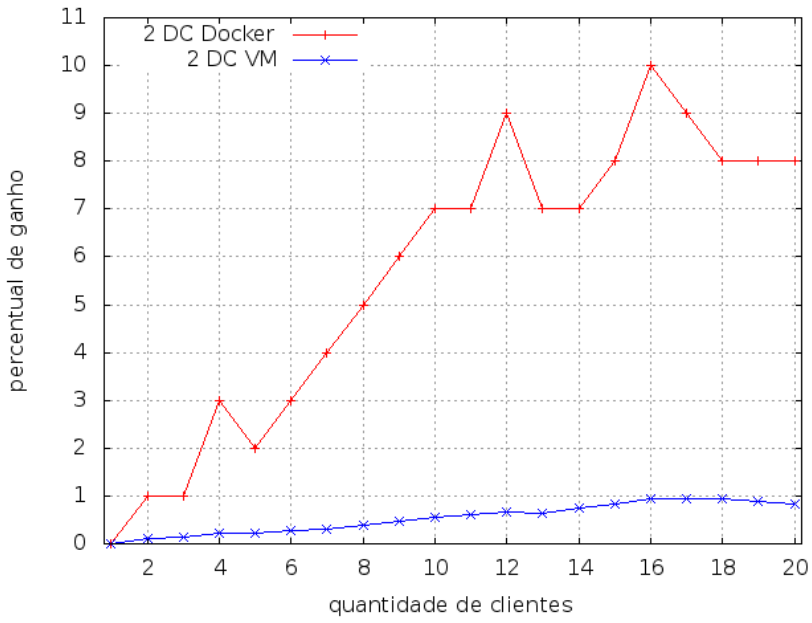

Figura 5.13 – Gráfico comparativo das máquinas virtuais e *dockers* movimentando-se entre os mini-*datacenters* com carga de processamento de 150.000 MIPS.

Fonte: Produzido pelo autor

A Figura 5.13 apresenta um gráfico de ganho percentual das duas tecnologias, *Dockers* e máquinas virtuais, utilizando o deslocamento das *Cloudlets* em relação a um *datacenter*. No primeiro ponto em ambos cenários percebe-se um ganho nulo pois só há uma *Cloudlet* em apenas um *datacenter*. Já no segundo ponto começa a apresentar um ganho pois ela será executada no segundo *datacenter*. Neste cenário observa-se que a movimentação das *Cloudlets* utilizando *dockers* obtém um ganho de até 10%, enquanto a migração das máquinas virtuais apresentam um ganho de no máximo 1%.

Ainda na Figura 5.13 percebe-se que a curva de ganho das máquinas virtuais começa uma descendente a partir de 16 clientes, esse comportamento advém da sobrecarga do μDC2 que possui menor capacidade de processamento. Por outro lado, os *dockers* mantém o aumento do percentual de ganho, mesmo que de forma mais suave. Isso é consequência da menor quantidade de IOPs que a tecnologia exige, que utiliza a capacidade de processamento do *datacenter* de maneira mais eficiente.

5.4 Considerações Finais

Com a descrição do ambiente simulado e da análise dos resultados expostos neste capítulo, é possível chegar a uma conclusão da proposta através do *Cloudsim* para a situação descrita. Os gráficos com as migrações, atrasos e processamento indicam que, para a situação descrita, a proposta tem resultados concernentes com o esperado e descrito em outros trabalhos.

6 Conclusão

Conforme apresentado na seção 5.3, as *Cloudlets* devem considerar os aspectos do ambiente para justificar sua migração para outro mini-*datacenter*. Pelo próprio conceito de *Cloudlet* ela deve ser executada o mais próximo possível de clientes, e isso deveria justificar sua migração para o mini-*datacenter* mais próximo da rede sem fio que o cliente se associa.

Porém, a migração dos serviços para acompanhar o deslocamento do cliente é uma ação bastante onerosa conforme demonstrado nos resultados alcançados. O impacto gerado a princípio pela mobilidade inviabilizaria algumas transferências de acordo com o serviço ofertado pela *Cloudlet* ou a latência do(s) enlace(s) que interconecta(m) *datacenters*. Na maior parte das aplicações demandadas pela *Cloudlet* e dos cenários existentes nas operadoras essa migração pode ser justificável, porém em aplicações com pouco uso de processamento ou poucos clientes a situação de manter-se no *datacenter* original pode ser mais promissora.

Naturalmente, o uso de *dockers containers* mostrou-se mais eficientes que o uso de máquinas virtuais, utilizado na implementação do *OpenStack++*, porém como a transferência onera o comportamento do sistema substancialmente sua implementação deve-se justificar em situações com grande carga de processamento ou com grande número de *Cloudlets* associadas. Essa sobrecarga diminuirá, relativamente, o impacto da transferência e fará uso de forma mais evidente a economia no tamanho da imagem, das chamadas de sistemas e das IOPs que os *dockers* se utilizam.

O impacto da carga de processamento das *Cloudlet* no ambiente de execução da Nuvem apresenta-se como um fator decisivo para decisão da migração, sua análise deve ser levada em algoritmos de *handoff* dos serviços por representarem alterações na percepção dos clientes.

Desta forma, na situação proposta, indica-se que de fato os *dockers* podem otimizar o comportamento das *Cloudlets*, com ou sem migração, diminuindo a latência de instanciação ou transferência das imagens do serviço dos clientes e fazendo melhor uso dos recursos das Nuvens conforme demonstrado no gráfico de ganho relativo. Seu aproveitamento será mais evidente em cenários de grande sobrecarga de processamento e quantidade de clientes.

6.1 Desafios Superados

Por se tratar de uma arquitetura relativamente recente, *Cloudlets* em *Mobile Cloud Computing* apresentou-se como um desafio para implementação e simulação. A construção de uma nuvem para avaliação necessitava de *hardwares* robustos em uma quantidade que se mostrasse possível executar os experimentos. Para contornar-se esse impeditivo optou-se por simular o ambiente de nuvem e quantificar a mobilidade das *Cloudlets*, porém não há simulador

algum capaz de gerar a situação das *Cloudlets* e tampouco de permitir a mobilidade das mesmas acompanhando seus clientes. Sendo assim tornou-se necessário a adoção de um simulador flexível e robusto o suficiente para que fosse personalizado a ponto de entregar resultados confiáveis; optou-se pelo *Cloudsim*.

A tarefa de personalizar o *Cloudsim*, introduzir o conceito de mobilidade de *Cloudlets*, ainda não abordado por trabalhos pares que utilizem simulação, e comparar seus resultados com os de uma Nuvem executando os serviços dos clientes móveis mostrou-se desafiadora. Em meados de 2015, em (HA et al., 2015) foram demonstrados resultados de transferências das *Cloudlets* bem como o conceito de mobilidade, ratificando o que seria defendido nesta dissertação ainda em produção. Apesar de retirar a classificação de inédita da proposta, a abordagem por Ha et al. (2015) da necessidade de migração também das *Cloudlets* demonstrou a relevância do estudo, gerando mais estímulo pelo assunto e novos desafios.

Ainda em meados de 2015, a aquisição de novos equipamentos pelo grupo de pesquisas permitiu a implantação de uma nuvem computacional utilizando *OpenStack* com extensões para *Cloudlets*. Porém, mesmo os novos *hardwares* se mostraram insuficientes para criação duas nuvens computacionais que garantissem a mobilidade. A medida alternativa encontrada foi virtualizar duas nuvens computacionais dentro de uma única maior, o que levou a outros desafios de virtualização e arquitetura de computadores.

Dois níveis de virtualização gerava situações em que o *hardware* virtualizado não poderia ser utilizado pela *VM Base* através do *OpenStack++* que implementava as *Cloudlets*, impossibilitando o planejado para os recursos computacionais disponíveis. Outra solução de contorno foi adotada, desta vez no *OpenStack++* que precisava de alterações para sua execução em *hardware* virtual.

Paralelamente à implantação das duas nuvens computacionais em *hardware* virtualizado para os testes, buscou-se uma otimização das máquinas virtuais para execução das *Cloudlets* por ser notório o *overhead* gerado na transferência das mesmas. Para essa questão, após estudo e análises de algumas ferramentas optou-se pelo *Docker* devido às vantagens mencionadas.

O *Docker* já tinha chegado em um nível de maturidade satisfatório sendo utilizado em outras soluções científicas, porém seria necessário também simulá-lo no *CloudSim*, que não possuía atributos nativos para modelá-los. Produziu-se uma nova adaptação da ferramenta de simulação para o problema proposto, adaptação que serviu para geração dos dados analisados nesta dissertação.

Ao final este trabalho chegou a uma conclusão através de simulação, propondo um novo modelo de execução das *Cloudlets* e garantindo que sua implementação em tecnologias atuais são exequíveis conforme alguns produtos parcialmente aplicados em *OpenStack*, *OpenStack++* e *Docker*.

6.2 Trabalhos Futuros

Percebe-se a necessidade de um estudo mais profundo com detalhes exclusivos de implementações *de facto* em ambientes de Nuvens utilizando o *hardware* característico. Como trabalho futuro, que esta em situação avançada e parcialmente concluso, pretende-se implementar o ambiente fisicamente o ambiente do *OpenStack++*, personalizando-o para que possa executar internamente em uma Nuvem, com *hardware* virtualizado e permitir o efetivo deslocamento entre duas nuvens através de um *hardware* reduzido.

A implementação das *Cloudlets* em *dockers* apresenta-se igualmente imperativo para estudos mais detalhados, permitindo dessa forma que possa ter efetivamente seu desempenho comparado com as máquinas virtuais. Esse processo encontra-se igualmente avançado no processo de *dockerização* das *Cloudlets*, isso permitirá que, além de estudos, as aplicações dos clientes possam optar por algum modelo que achem mais adequado.

A mudança de tecnologia e arquitetura proposta nesta dissertação abrirá novas perspectivas de interpretações do modelo de *Cloudlets* favorecendo sua otimização de desempenho e latência, especialmente para primeira carga ou migração.

As adaptações executadas permitirão a análise ambientes mais complexos, imaginando-se migrações de clientes e *Cloudlets* em larga escala para avaliar as implicações no desempenho percebido pelos usuários otimizando algoritmos para *handoff* de serviços críticos dada o novo modelo de arquitetura .

Referências

BAHL, P. et al. Advancing the state of mobile cloud computing. In: ACM. *Proceedings of the third ACM workshop on Mobile cloud computing and services*. [S.l.], 2012. p. 21–28.

BAYS, L. R. et al. Virtual network security: threats, countermeasures, and challenges. *Journal of Internet Services and Applications*, Springer London, v. 6, n. 1, p. 1, 2015.

BELMANN, P. et al. Bioboxes: standardised containers for interchangeable bioinformatics software. *GigaScience*, BioMed Central, v. 4, n. 1, p. 1, 2015.

BINET, S.; COUTURIER, B. docker & hep: Containerization of applications for development, distribution and preservation. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2015. v. 664, n. 2, p. 022007.

BOHEZ, S. et al. Discrete-event simulation for efficient and stable resource allocation in collaborative mobile cloudlets. *Simulation Modelling Practice and Theory*, Elsevier B.V., v. 50, p. 109–129, 2014. ISSN 1569190X. Disponível em: <http://dx.doi.org/10.1016/j.simpat.2014.05.006>.

BUYYA, R.; RANJAN, R.; CALHEIROS, R. N. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In: IEEE. *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*. [S.l.], 2009. p. 1–11.

CALHEIROS, R. N. et al. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, Wiley Online Library, v. 41, n. 1, p. 23–50, 2011.

CHELLAPPA, R. Intermediaries in cloud-computing: A new computing paradigm. *INFORMS meeting*, Univ. of TX, Ctr. for Res. on Elect. Comm., MSIS Dept., v. 19, n. 1, 1997.

CHEN, Z. et al. Quiltview: a crowd-sourced video response system. In: ACM. *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. [S.l.], 2014. p. 13.

CLINCH, S. et al. How close is close enough? Understanding the role of cloudlets in supporting display appropriation by mobile users. *2012 IEEE International Conference on Pervasive Computing and Communications, PerCom 2012*, p. 122–127, 2012.

DURO, F. R. et al. CoSMiC: A hierarchical cloudlet-based storage architecture for mobile clouds. *Simulation Modelling Practice and Theory*, Elsevier B.V., v. 50, p. 3–19, 2015. ISSN 1569190X. Disponível em: <http://dx.doi.org/10.1016/j.simpat.2014.07.007>.

FCC's Office of Engineering and Technology and And Consumer and Governmental Affairs. 2015 Measuring Broadband America Fixed Broadband Report. p. 1–67, 2015. Disponível em: <https://www.fcc.gov/reports-research/reports/measuring-broadband-america/measuring-broadband-america-2015>.

FELTER, W. et al. An Updated Performance Comparison of Virtual Machines and Linux Containers. *Technology*, v. 25482, p. 171–172, 2014. Disponível em: <http://domino.research.

ibm.com/library/CyberDig.nsf/papers/0929052195DD819C85257D2300681E7B/{\protect\
T1\textdollar}File/rc25482.>

GERLACH, W. et al. Skyport: container-based execution environment management for multi-cloud scientific workflows. In: IEEE PRESS. *Proceedings of the 5th International Workshop on Data-Intensive Computing in the Clouds*. [S.l.], 2014. p. 25–32.

HA, K. et al. Adaptive VM Handoff Across Cloudlets. Technical Report CMU-CS-15-113, CMU School of Computer Science, n. June, 2015.

HA, K. et al. Towards wearable cognitive assistance. In: ACM. *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. [S.l.], 2014. p. 68–81.

HA, K.; LEWIS, G.; SIMANTA, S. Cloud Offload in Hostile Environments. *Science And Technology*, n. December, 2011. Disponível em: <http://reports-archive.adm.cs.cmu.edu/anon/anon/usr0/ftp/2011/CMU-CS-11-146.pdf>.

HA, K. et al. The impact of mobile multimedia applications on data center consolidation. In: IEEE. *Cloud Engineering (IC2E), 2013 IEEE International Conference on*. [S.l.], 2013. p. 166–176.

HA, K. et al. Just-in-time provisioning for cyber foraging. In: ACM. *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. [S.l.], 2013. p. 153–166.

HA, K.; SATYANARAYANAN, M. OpenStack++ for Cloudlet Deployment. *School of Computer Science Carnegie Mellon University Pittsburgh*, 2015.

HECKMANN, O. et al. On realistic network topologies for simulation. In: ACM. *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*. [S.l.], 2003. p. 28–32.

HTTPARCHIVE. *HTTP Archive*. 2016. <http://httparchive.org/>. (Acessado em 05/07/2016).

HUNG, L.-H. et al. Guidock: Using docker containers with a common graphics user interface to address the reproducibility of research. *PloS one*, Public Library of Science, v. 11, n. 4, p. e0152686, 2016.

KOVACHEV, D.; CAO, Y.; KLAMMA, R. Mobile Cloud Computing : A Comparison of Application Models. *arXiv preprint arXiv:1107.4940*, 2011.

MADHU, B. et al. A comparative study of algorithms for efficient dynamic consolidation of virtual machines in cloud. *International Journal of Applied Engineering Research*, v. 11, n. 6, p. 4597–4600, 2016.

MEDINA, A. et al. Brite: An approach to universal topology generation. In: IEEE. *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*. [S.l.], 2001. p. 346–353.

MELL, P.; GRANCE, T. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg, 2011.

MERCHANT, N. et al. The iplant collaborative: Cyberinfrastructure for enabling data to discovery for the life sciences. *PLoS Biol*, Public Library of Science, v. 14, n. 1, p. e1002342, 2016.

MISHRA, S.; TONDON, R. A shared approach of dynamic load balancing in cloud computing. 2016.

PAGARE, J.; KOLI, N. Design and simulate cloud computing environment using cloudsim. *Int. Journal of Computer Technology and Applications*, v. 6, n. 1, p. 35–42, 2015.

PEREZ, S. Mobile operators - strategies for growth. *Juniper Whitepapers*, Juniper Networks, v. 16, n. 9, p. 15, 2014.

PETTEY, C.; TUDOR, B. Gartner says worldwide cloud services market to surpass 68*billionin*2010.*Gartner Inc., Stamford, Press release*, 2010.

QUWAIDER, M.; JARARWEH, Y. Cloudlet-based efficient data collection in wireless body area networks. *Simulation Modelling Practice and Theory*, Elsevier B.V., v. 50, p. 57–71, 2015. ISSN 1569190X. Disponível em: <http://dx.doi.org/10.1016/j.simpat.2014.06.015>.

RAJARAMAN, V. Cloud computing. *Resonance*, v. 19, n. 3, p. 242–258, 2014. ISSN 0973712X.

RELLERMEYER, J. S.; RIVA, O.; ALONSO, G. Alfredo: an architecture for flexible interaction with electronic devices. In: SPRINGER-VERLAG NEW YORK, INC. *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*. [S.l.], 2008. p. 22–41.

REPORT, O. *3G and 4G LTE Cell Coverage Map - OpenSignal*. 2016. <http://opensignal.com/>. (Acessado em 06/14/2016).

REZENDE, R. D. Á. et al. Monitoramento de Aplicações em Cuidados Médicos em uma Nuvem Governamental. *Simpósio Brasileiro de Telecomunicações*, p. 2–6, 2016.

RIVERA, J. Gartner identifies the top 10 strategic technology trends for 2015. *Gartner, October*, v. 8, 2014.

SATYANARAYANAN, M. et al. The Case for VM-Base Cloudlets in Mobile Computing. *Pervasive Computing*, v. 8, n. 4, p. 14–23, 2009. ISSN 1536-1268.

SATYANARAYANAN, M. et al. The role of cloudlets in hostile environments. *IEEE Pervasive Computing*, v. 12, n. 4, p. 40–49, 2013. ISSN 15361268.

SEFRAOUI, O.; AISSAOUI, M.; ELEULDJ, M. OpenStack: Toward an Open-Source Solution for Cloud Computing. *International Journal of Computer Applications*, v. 55, n. 03, p. 38–42, 2012. ISSN 09758887. Disponível em: <http://research.ijcaonline.org/volume55/number3/pxc3882991.pdf>.

SHI, Y.; JIANG, X.; YE, K. An energy-efficient scheme for cloud resource provisioning based on cloudsim. In: IEEE. *2011 IEEE International Conference on Cluster Computing*. [S.l.], 2011. p. 595–599.

SIMOENS, P. et al. Scalable crowd-sourcing of video from mobile devices. In: ACM. *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. [S.l.], 2013. p. 139–152.

SOLUTIONS, F. *Using Docker Container Technology with F5 Products and Services*. 2016. <https://f5.com/resources/white-papers/using-docker-container-technology-with-f5-products-and-services>. Acessado: 2016-08-07.

STALLINGS, W. *Arquitetura e Organização de Computadores, 8a. Edição*. [S.l.]: Prentice/Hall, 2010.

TOMMASO, P. D. et al. The impact of docker containers on the performance of genomic pipelines. *PeerJ*, PeerJ Inc., v. 3, p. e1273, 2015.

VARGA, A. et al. The omnet++ discrete event simulation system. In: SN. *Proceedings of the European simulation multiconference (ESM'2001)*. [S.l.], 2001. v. 9, n. S 185, p. 65.

WANG, Y. H.; WU, I. C. Achieving high and consistent rendering performance of java AWT/Swing on multiple platforms. *Software - Practice and Experience*, v. 39, n. 7, p. 701–736, 2009. ISSN 00380644.

WOLBACH, A. et al. Transient Customization of Mobile Computing Infrastructure. *Proceedings of the 1st Workshop on Virtualization in Mobile Computing (MobiVirt'08)*, p. 37–41, 2008. Disponível em: <http://doi.acm.org/10.1145/1622103.1622108>.

XAVIER, B.; FERRETO, T.; JERSAK, L. Time Provisioning Evaluation of KVM, Docker and Unikernels in a Cloud Platform. *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, p. 277–280, 2016. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7515699>.

Anexos

ANEXO A – Arquivo de Topologia BRITE.

Topology: (3 Nodes, 4 Edges)
Model (1 - RTWaxman): 5 5 5 1 2 0.15000000596046448 0.20000000298023224 1 1 10.0 1024.0

Nodes: (3)
0 1 3 3 3 -1 RT$_N$$ODE$
15333 – 1$RT_N ODE$
21133 – 1$RT_N ODE$

Edges: (6)
0 0 1 3.0 04 1024.0 -1 -1 E$_R$$TU$
1104.0041024.0 – 1 – 1$E_R TU$
2022.828427124746190311 86.2 – 1 – 1$E_R TU$
3203.6055512754639891188.8 – 1 – 1$E_R TU$
4123.01188.8 – 1 – 1$E_R TU$
5214.01186.2 – 1 – 1$E_R TU$