UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE QUIXADÁ

CURSO DE GRADUAÇÃO EM TECNOLOGIA EM REDES DE COMPUTADORES

CLEILSON SARAIVA RAMOS

ANÁLISE DA CAPACIDADE DE DESCARREGAMENTO DE REDES MÓVEIS DENSAS

QUIXADÁ

2017

CLEILSON SARAIVA RAMOS

ANÁLISE DA CAPACIDADE DE DESCARREGAMENTO DE REDES MÓVEIS DENSAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Tecnologia em Redes de Computadores do Centro de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Tecnologia em Redes de Computadores.

Orientador: Prof. Me. Antonio Rafael Braga

QUIXADÁ

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

R142a Ramos, Cleilson Saraiva.
Análise da capacidade de descarregamento de redes móveis densas : análise de desempenho em computação em nuvem / Cleilson Saraiva Ramos. – 2017.
55 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Redes de Computadores, Quixadá, 2017.
Orientação: Prof. Me. Antonio Rafael Braga.

1. Computação em nuvem móvel. 2. Computação em nuvem. 3. Redes locais sem fio. 4. Desempenho-análise. 5. Offloading Computacional. I. Título.

CDD 004.6

CLEILSON SARAIVA RAMOS

ANÁLISE DA CAPACIDADE DE DESCARREGAMENTO DE REDES MÓVEIS DENSAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Tecnologia em Redes de Computadores do Centro de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Tecnologia em Redes de Computadores.

Aprovada em: __/__/____

BANCA EXAMINADORA

Prof. Me. Antonio Rafael Braga (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo Antonio Leal Rego
Universidade Federal do Ceará (UFC)

Prof. Dr. Arthur de Castro Callado
Universidade Federal do Ceará (UFC)

Este trabalho é dedicado a todos aqueles que me fizeram acreditar no impossível e tornaram possível que eu chegasse até aqui.

AGRADECIMENTOS

Gostaria de agradecer primeiramente ao Professor Rafael Braga, pela ótima orientação, paciência nos momentos difíceis e pela motivação me fazendo acreditar que era possível. Ao professor Paulo Rego pelas ótimas ideias desde o inicio e pelos puxões de orelha quando foram necessários. A toda minha família, que me apoiaram bastante nesses últimos anos de diversas maneiras. A toda a galera da RH, principalmente meus grandes amigos Alisson Lima e Roseli Rocha que proporcionaram momentos inesquecíveis e minha namorada Sara Cibelle por todo apoio. A todos do 2013.1 (melhor turma *ever*). Gostaria de agradecer também a todos os alunos e professores que colaboraram com os experimentos e tornaram possível esse trabalho.

"Os maiores acontecimentos e pensamentos são os que mais tardiamente são compreendidos."

(Friedrich Nietzsche)

()

RESUMO

Com o surgimento dos *smartphones* e com o aumento do número de aplicativos para eles desenvolvidos é difícil explorar todo o seu potencial devido a suas limitações energéticas e de hardware. Com o objetivo de solucionar os problemas a eles relacionados, temos a Computação em nuvem móvel, do inglês - Mobile Cloud Computing (MCC), que estende a visão da computação utilitária da computação em nuvem para aumentar os recursos computacionais e energéticos dos dispositivos móveis. Esse aumento é realizado através do *offloading*, que é uma técnica que permite que os processos das aplicações sejam realizados em provedores de recursos externos. Em meio as diversas soluções propostas em um ambiente de MCC, muitos problemas são enfrentados, dentre eles estão a latência e a limitação dos recursos da nuvem devido ao alto custo para sua aquisição. Neste trabalho realizamos uma análise e avaliação de como esses problemas afetam o desempenho dos ambientes propostos, mostrando qual modelo de *offloading* sé mais apropriada para determinada configuração, a análise foi realizada levando em consideração diversos fatores, tais quais, a quantidade de dispositivos conectados, execução realizada em nuvem local ou remota, e quantidade de dados processados.

Palavras-chaves: Mobile Cloud Computing. Nuvem. *Offloading* de Processamento.

ABSTRACT

With the emergence of smartphones and the increasing number of applications developed for them, it is difficult to exploit their full potential due to their energy and hardware limitations. In order to solve the related problems, we have the Mobile Cloud Computing (MCC), which extends the vision of the utilitarian computing of cloud computing to increase the computational and energetic resources of the mobile devices. This increase is achieved through offloading, which is a technique that allows application processes to be performed in external resource providers. Among the various solutions proposed in an MCC environment, many problems are faced, among them are the latency and limitation of the cloud's resources due to the high cost of its acquisition. In this work we perform an analysis and evaluation of how these problems affect the performance of the proposed environments, showing which offloading model is most appropriate for a particular configuration, the analysis was carried out taking into account several factors, such as the number of connected devices, performed in local or remote cloud, and amount of data processed.

Key-words: Mobile Cloud Computing. Cloud . Offloading de Processamento.

LISTA DE FIGURAS

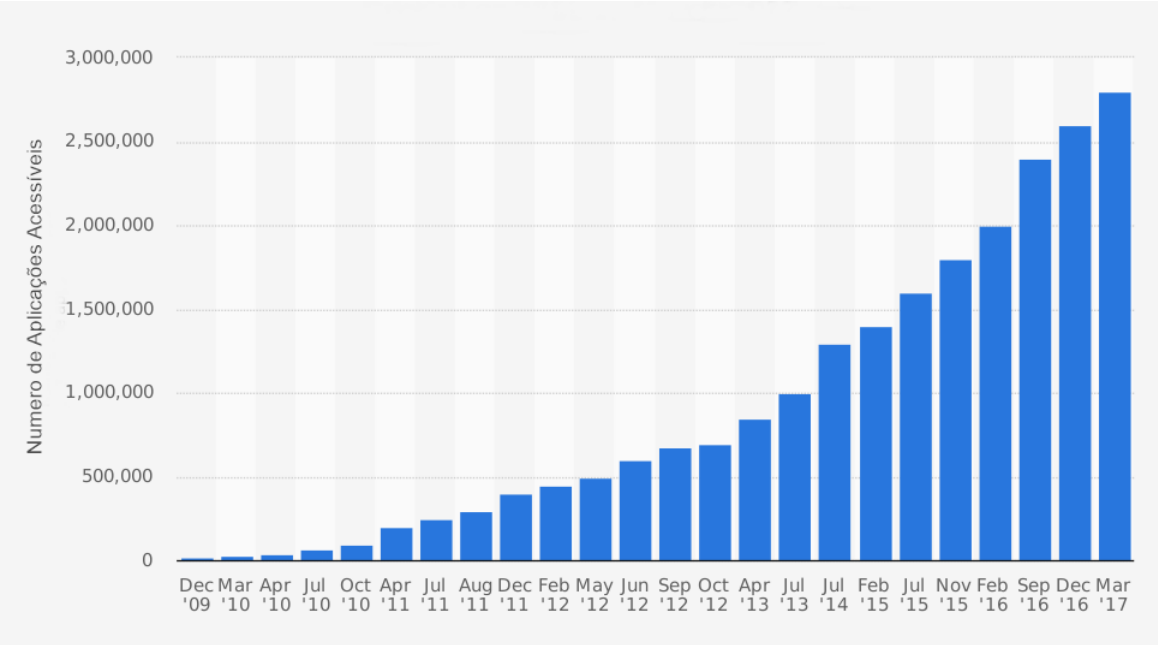
LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, com o surgimento dos *smarthphones*, o desenvolvimento de aplicativos para dispositivos móveis tem crescido bastante. Tal crescimento pode ser observado na Figura 1. Todas essas aplicações já fazem parte do dia-a-dia das pessoas em diversas áreas e setores, tais como: entretenimento, jogos, saúde, viagens e notícias. Mas, apesar de todo seu crescimento, é difícil explorar todo seu potencial devido aos diversos problemas inerentes à área, causados principalmente pelas limitações energéticas e de hardware (BAHL *et al.*, 2012)(FERNANDO *et al.*, 2013).

Figura 1 – Crescimento do Número de Aplicações na PlayStore



Fonte: www.statista.com

De acordo com isso, as limitações dos recursos dos dispositivos móveis interferem no crescimento dos mesmos. Por serem móveis, suas baterias são limitadas e seus hardwares, apesar de terem melhorado nos últimos anos e hoje conseguirem executar aplicações com uma alta demanda de processamento, ainda não são suficientes para atender completamente algumas aplicações. Essas limitações fazem com que os desenvolvedores se restrinjam a desenvolver aplicações com grande preocupação em relação ao hardware e, principalmente, com o consumo energético (SATYANARAYANAN *et al.*, 2009)(FERNANDO *et al.*, 2013).

Devido a todo o potencial crescimento e como uma tentativa de busca para soluções

dos problemas já citados, surge um novo paradigma de redes, a Computação em Nuvem Móvel, do inglês - *Mobile Cloud Computing* (MCC), que em (SHIRAZ *et al.*, 2013) é definida como o mais recente paradigma prático que estende a visão da computação utilitária da computação em nuvem para aumentar os recursos computacionais e energéticos dos dispositivos móveis através da utilização das técnicas de *offloading*, enquanto em (COSTA *et al.*, 2014) é definido como uma combinação de diversas áreas e tecnologias, sendo as principais delas a computação em nuvem, a computação móvel e as redes sem fio.

No contexto do paradigma de MCC, aparece uma operação denominada *offloading*, que é uma técnica que permite que o processamento de aplicações (ou parte dele) seja feito em provedores de recursos externos. Esses provedores devem possuir recursos suficientes para um processamento mais rápido e eficiente, tendo como objetivo diminuir o consumo energético, o processamento local e tempo total de resposta da aplicação (CUERVO *et al.*, 2010). Tal processo pode ser executado em uma nuvem remota, por meio da Internet, ou em uma nuvem local, através de uma rede sem fio.

No primeiro caso, onde o servidor de *offloading* é hospedado na nuvem remota, tal operação pode não ser uma boa opção em determinados cenários, pelo fato de que os clientes utilizarão uma rede que pode possuir grande tráfego para conectarem-se ao servidor, dessa forma os dispositivos cliente acabariam sofrendo com a latência, um dos maiores problemas das aplicações sensíveis ao tempo (FERNANDO *et al.*, 2013). No segundo caso, que Satyanarayanan *et al.* (2009) denomina como *cloudlet*, a utilização do serviço de *offloading* é feita em servidor hospedado em uma nuvem local, sendo essa uma melhor opção para a maioria das aplicações sensíveis ao tempo.

Partindo disso, o conceito de *cloudlets* nasce como uma solução para o problema de latência nas aplicações de *offloading* enfrentado na rede pública, uma vez que a quantidade de tráfego em uma rede local é bem menor. A principal ideia que motiva o uso de *cloudlets* é a utilização das redes sem fio, que por sua vez possuem menor latência e menos congestionamento, em sua maioria, se comparadas à rede pública. Como consequência disso, um *cloudlet* torna-se uma boa opção em cenários com aplicações que são mais sensíveis a latência, tais como aplicativos de processamento de imagem, visão computacional, realidade aumentada, reconhecimento de voz e jogos (BAHL *et al.*, 2012).

Seguindo esse contexto, surge um outro desafio, que seria a utilização dos *cloudlets* em Redes de Área de Campus (do inglês, *Campus Area Network* - CAN), que seria uma rede

que interliga máquinas situadas em diferentes edificações de um mesmo complexo institucional. Como é definido em (JIANG; SHAN, 2008), uma rede de campus tem sido uma plataforma de divulgação e transmissão de importantes informações. Nas redes de campus, muitas aplicações também se deparam com problemas de latência, já que o tráfego gerado por uma rede como essa pode ser superior ao de uma rede local de uso comum, uma vez que, diariamente, muitos estudantes e profissionais se deslocam e usam os mais diversos tipos de serviços. Nas Redes de Campus, há ainda as redes cabeadas que conectam computadores localizados em uma variedade de espaços, incluindo os prédios administrativos, acadêmicos, bibliotecas, laboratórios, auditórios e centrais de processamento de dados que podem ser utilizados, oportunisticamente, como servidores de *cloudlets* (KUSAKABE *et al.*, 2014).

Considerando todos os desafios apresentados, como o de grande latência nas redes públicas e do poder computacional reduzido nos dispositivos móveis, diversos trabalhos como (CUERVO *et al.*, 2010) e (KOSTA *et al.*, 2012) mostraram a eficiência e o ganho de qualidade na utilização de servidores de *offloading* em redes sem fio. Foi possível observar também que o ganho foi ainda maior ao se utilizar utilizar *cloudlets*, se comparado ao uso de recursos da nuvem pública acessados através da Internet. Mas seriam os *cloudlets* melhores em quaisquer cenários? Em quais cenários uma nuvem pública seria vantajosa em uma redes de campus? Como esses tipos de servidores de *offloading* se comportam diante de um aumento significativo de requisições?

1.1 Motivação

Em (COSTA *et al.*, 2015) é realizada uma análise do desempenho da técnica de *offloading* ao ser utilizada em uma nuvem privada local (*cloudlet*) e em uma nuvem pública remota. Nessa análise, utilizando apenas uma única requisição por vez em seus testes, concluíram que os *cloudlets* são melhores nesse tipo de cenário. Este trabalho, dando sequência a isso, pretende ir além, realizando uma análise e uma comparação dos resultados de desempenho da técnica quando se considera a variação da quantidade de requisições simultâneas. Para tal análise, planejamos utilizar a ferramenta MpOS (*A Multiplatform Offloading System*) que foi proposta e utilizada em Costa *et al.* (2015).

Nesse contexto, com o intuito de contribuir com a comunidade acadêmica da Universidade Federal do Ceará no campus de Quixadá, esse trabalho propõe a realização de uma análise de desempenho comparativa entre o uso do servidor de *offloading* local e o remoto em

cenários com aumento significativo de requisições e em ambientes com grande grande número de pessoas, como por exemplo, nas já citadas Redes de Campus. Através dessa análise esperamos também traçar políticas de seleção automática do melhor servidor a ser utilizado na operação de *offloading*.

1.2 Objetivos

O objetivo geral desse trabalho é realizar uma análise de desempenho da técnica de *offloading* ao ser utilizada em uma nuvem privada local e em uma nuvem pública remota, a fim de mostrar qual das soluções é mais eficiente quando ocorre uma grande variação da quantidade de requisições simultâneas.

Os seguintes objetivos específicos foram levados em conta para alcançar a meta do objetivo geral.

a) Definir métricas para comparação dos cenários definidos;
b) Definir a melhor ferramenta para execução da comparação;
c) Identificar possíveis problemas de desempenho.

1.3 Organização do Trabalho

O restante do trabalho está dividido da seguinte forma, fundamentação teórica, onde apresentamos os principais termos presentes nesse trabalho, como *Mobile Cloud Computing*, *Cloudlet*, *Offloading* e *Cloud Computing*. Depois disso, apresentamos a proposta de nosso trabalho e em seguida a configuração do nosso ambiente de testes que descreve todas as ferramentas utilizadas e como ocorreu todo o processo configuração e/ou produção das mesmas. Por fim, temos a validação do sistemas com nossos resultados seguida de nossas considerações finais.

2 TRABALHOS RELACIONADOS

Nos últimos anos, vários trabalhos baseados no conceito de *Mobile Cloud Computing* foram propostos. Nessa seção são descritos alguns deles, relacionando-os ao nosso, de forma a apresentar em que se assemelham e em que diferem. Apresentamos ainda os principais pontos que tornaram-os base para a criação deste. Ao fim da sessão mostramos ainda um comparativo sintetizado das características apresentadas.

Em (KOSTA *et al.*, 2012) é descrita a ferramenta *ThinkAir*, que tem como principal função proporcionar a descarga de parte do processamento de um dispositivo móvel para um servidor externo, a fim de melhorar seu desempenho e economizar recursos (que são limitados). Sua arquitetura básica consiste em três componentes principais: o ambiente de execução, a aplicação servidor e os *profilers*.

O primeiro desses componentes é o ambiente de execução, que tem como principal função decidir, a partir de informações coletadas e/ou configurações pré-estabelecidas, se o *offloading* de um método deve ser feito. O segundo é a aplicação servidor, que irá executar esses métodos na máquina na qual estará funcionando o *cloudlet* e por fim, temos os *profilers*, que são responsáveis por colher informações que serão utilizadas futuramente em decisões da própria aplicação e para a análise e gerência dos dados.

O *ThinkAir* é uma ferramenta com fortes pontos positivos, pois como foi citado no próprio artigo, houve uma preocupação com pontos específicos, como o paralelismo na execução de atividades no servidor e o aproveitamento de uma mesma plataforma por múltiplos usuários com atividades em comum. Essas características economizam de maneira notável a quantidade de recursos necessários para cada usuário, e como complemento a isso, a ferramenta possui também um bom escalonamento do recursos para aproveitar da melhor forma possível o hardware disponível.

Como já citado anteriormente, o *ThinkAir* pode ser utilizado em um ambiente com múltiplos usuários simultâneos, o que se assemelha bastante à ideia desse trabalho que é de observar e analisar como o desempenho dos provedores se comporta de acordo com a variação da quantidade de requisições. Outro ponto relevante é o fato dele conseguir um desempenho razoável mesmo sem um *hardware* muito robusto.

No trabalho de Sergio *et al.* (2015) é proposta uma forma de modelagem e a simulação de uma arquitetura típica de computação em nuvem móvel. Nessa proposta são contemplados aspectos desde componentes físicos do ambiente de simulação até as tomadas de decisões para a

realização de *offloading*, podendo elas serem estáticas ou dinâmicas (como descrito na seção 4.1.2). Tais decisões podem produzir um melhor gerenciamento dos recursos, tornando possível a execução de várias atividades sem necessidade de uma infraestrutura computacional real em grande escala.

Semelhante ao nosso, o trabalho de Sergio *et al.* (2015) buscou uma maneira de executar as atividades de *offloading* de forma que não fosse necessária a utilização de um *hardware* muito potente no provedor externo. Um outro fator importante são as métricas que foram utilizadas para a validação dos testes, que também são utilizadas neste trabalho, são elas: tempo de *upload*, tempo de processamento e tempo de *download*, sendo a soma dela o tempo total de resposta.

Por fim, no trabalho de Costa *et al.* (2015) é apresentada uma ferramenta de *offloading* chamada MpOS (Multiplataform Offloading System), que permite que parte do processamento de um dispositivo móvel seja feito em um provedor externo. Frente aos vários desafios de MCC, o MpOS surgiu para ajudar os desenvolvedores de aplicativos móveis a lidar com questões relativas à descarga em múltiplas plataformas, dando suporte às operações de *offloading* nas plataformas *Android* e *Windows Phone*. Sua arquitetura e seus requisitos foram inspirados em características de diferentes *frameworks* de *offloading*, tais como *ThinkAir* (KOSTA *et al.*, 2012), MAUI (CUERVO *et al.*, 2010), *Cuckoo* (KEMP *et al.*, 2010) e *Scavenger* (TYUKIN *et al.*, 2015).

No trabalho de Costa *et al.* (2015) também é realizada uma análise e comparação do desempenho da utilização da plataforma em ambiente de nuvem local e nuvem pública. Nessa análise, os autores consideraram que apenas um único usuário estaria utilizando o *MpOS* em ambos os cenários, o que pode pode não tornar os resultados tão precisos, de maneira que, com a variação da quantidade de requisições os resultados apresentados devem ser mais próximos do real. Diferente disso, o nosso trabalho pretende buscar dados mais precisos, partindo da ideia de que a variação da quantidade de requisições vai afetar o desempenho da *cloudlet* de forma bem mais expressiva. Assim, esperamos encontrar até que ponto a *cloudlet* é a melhor opção se comparado com um servidor hospedado em rede pública.

As principais diferenças entre os trabalhos citados podem ser observados na Tabela 1. Nela apresentamos algumas características utilizados nos experimentos de cada trabalho, tais como, a realização em ambiente real ou simulado, presença ou ausência de redes densas e utilização de múltiplos usuários assim como a análise do comportamento das plataforma com a

variação dessa quantidade de requisições.

Tabela 1 – Comparativo dos trabalhos relacionados

	Ambiente real	**Ambiente simulado**	**Múltiplos usuários**	**Redes densas**
(KOSTA *et al.*, 2012)	Sim	Não	Não	Não
(COSTA *et al.*, 2015)	Sim	Não	Não	Não
(SERGIO *et al.*, 2015)	Não	Sim	Sim	Não
Este Trabalho	Sim	Não	Sim	Sim

3 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, é apresentada a fundamentação teórica do trabalho, onde são descritos de forma detalhada os conceitos-chave importantes para um melhor entendimento da problemática que está sendo tratada. Tal fundamentação tem início com o conceito de redes sem fio. Na sequência, apresenta-se o conceito de *cloud computing* com suas características, modelos de serviço e de implementação e, por fim, o conceito *Mobile Cloud Computing* que abrange os conceitos de *cloudlets*, *offloading* e a plataforma de *offloading* MpOS.

3.1 Redes Sem Fio

Os avanços nas comunicações nos últimos anos possibilitaram o surgimento de várias tecnologias, que desde então procuram atender a real necessidade de seus usuários e com a melhor qualidade possível. Nos últimos anos, a comunicação sem fio ganhou um espaço considerável nas tecnologias de transmissão de dados, deixando de existir apenas nas comunicações de longa distância (feitas através de satélite e rádios de alta capacidade), para fazer parte de ambientes locais. Essa tendência foi fortalecida pelo investimento de instituições e empresas no sentido de aplicar a transmissão sem fio em redes de computadores.(WELCH; LATHROP, 2003)

As redes locais sem fio (WLAN) estão crescendo em popularidade. Eles estão sendo utilizadas por empresas de todos os tipos, instituições de ensino, órgãos governamentais e forças armadas. A razão é que as WLANs fornecem, aos usuários, o acesso a suas informações em diversos locais.

3.1.1 Principais Características

Como é apresentado em (WELCH; LATHROP, 2003), as redes sem fio apresentam varias características que podem motivar o seu uso, de acordo com o cenário e as necessidades de quem as utiliza. Nas próximas seções, essa características serão apresentadas, sendo divididas em vantagens e desvantagens.

3.1.1.1 Vantagens

- Mobilidade: Oferecem a liberdade de deslocamento mantendo-se a conexão. Simplicidade - configuração fácil e rápida e simples da rede, sem cabos a serem

instalados.

- Flexibilidade: Podem ser instaladas em locais praticamente impossíveis para cabos e facilitam configurações temporárias e remanejamentos.
- Baixo Custo: Se considerado o custo global da rede e não o preço individual dos equipamentos, as WLANs reduzem os custos de instalação porque dispensam cabeamento. Por isso, a economia é ainda maior em ambientes sujeitos a mudanças frequentes.
- Fácil expansão: Basta instalar o adaptador de LAN sem fio no dispositivo cliente.

3.1.1.2 Desvantagens

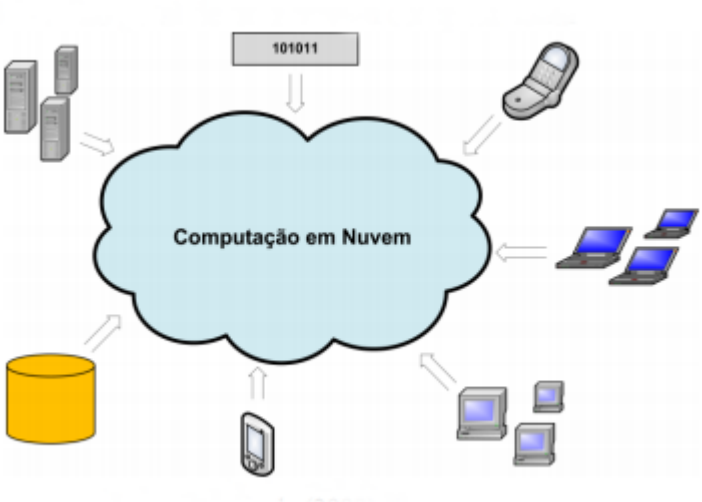
- Qualidade de serviço: A qualidade do serviço provido ainda é menor que a das redes cabeadas. A principal razão para isso é a alta taxa de erro devido a interferências.
- Interferência: As interferências no sinal nas redes sem fio sempre serão aspectos de preocupação perante os usuários, afinal elas podem sofrer interferências tanto de outros sistemas/equipamentos como os telefones sem fio e alarmes de segurança que operam na frequência de 2.4GHz, como também podem ter interferências de outros APs que utilizam o mesmo canal ou canais adjacentes.
- Distorção por percursos múltiplos (*Multipath*): Devido às reflexões do sinal transmitido provocado por diferentes superfícies ao longo do trajeto, o sinal percebido pelo receptor é distorcido, piorando a qualidade do sinal.
- Segurança: Em uma rede sem fio é mais difícil garantir a segurança, uma vez que o meio de transmissão é aberto a qualquer um que esteja no perímetro geográfico do transmissor.
- Consumo de energia: Os dispositivos sem fio devem ser bastante eficientes em relação ao consumo de energia, uma vez que nem sempre terão disponibilidade fácil de recarga. Esse problema hoje já foi minimizado.
- *Handoff*: devido à possibilidade de deslocamento do terminal sem fio, o sistema deve garantir a conectividade conciliando o *handoff* entre as fronteiras de transmissão e o roteamento do tráfego, principalmente em uma rede com múltiplos pontos de acesso.

3.2 Computação em Nuvem

O mundo computacional está se tornando muito grande e complexo. A computação em nuvem tem emergido como um modelo de computação popular para suportar o processamento de grandes volumes de dados usando *clusters* de computadores como provedores externos (RIMAL *et al.*, 2009). A exigência crescente de recursos computacionais em áreas como computação científica abriu o caminho para o desenvolvimento da computação paralela e distribuída. Depois do sucesso generalizado dos *clusters* e a adoção mais lenta das grades computacionais, o modelo de computação em nuvem é cada vez mais adotado em soluções distribuídas (EVOY; SCHULZE; GARCIA, 2011).

De acordo com SOUSA *et al.* (2009), a nuvem é uma metáfora para a Internet ou infraestrutura de comunicação entre os componentes arquiteturais, baseada em uma abstração que oculta a complexidade de infraestrutura. Cada parte dessa infraestrutura é provida como um serviço, que normalmente são alocados em *datacenters*, que utilizam o hardware compartilhado para computação e armazenamento. Foster *et al.* (2008) define computação em nuvem como um grande paradigma de computação distribuída, em que os provedores de nuvem possam fornecer vários tipos de serviços aos seus clientes usando máquinas virtuais que são hospedadas nesses provedores. Ele inclui algumas técnicas de utilização, o que melhora a eficiência do sistema. A Figura 2 apresenta uma visão geral de uma nuvem computacional, com os vários equipamentos que podem compor essa infraestrutura diversificada.

Figura 2 – Visão geral de uma nuvem computacional



Fonte: SOUSA *et al.* (2009)

O NIST (*National Institute of Standards and Technology*) define computação em nuvem como um modelo que permite acesso conveniente, sob demanda, através das redes, a um conjunto de recursos computacionais configuráveis, como por exemplo, redes, servidores, armazenamento, aplicações e serviços, que podem ser rapidamente fornecidos e liberados com o mínimo esforço de gerenciamento do usuário e com pouca (ou nenhuma) interação com o provedor de serviço. Este modelo de nuvem é composto por cinco características essenciais, três modelos de serviço, e quatro modelos de implementação que serão apresentados a seguir.

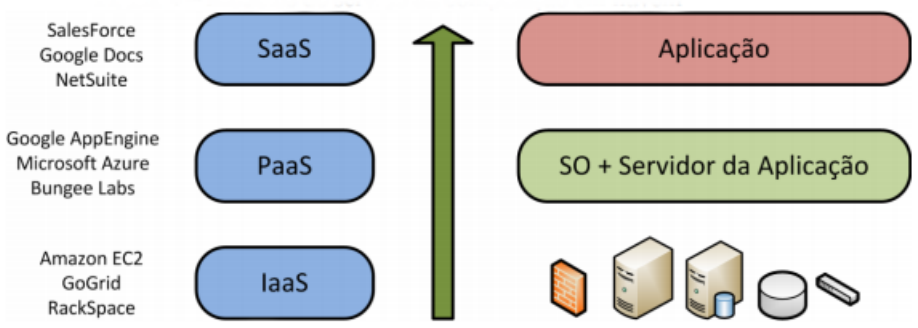
3.2.1 Características essenciais

a) Serviço sob demanda: o usuário pode adquirir unilateralmente recurso computacional na medida em que necessite e sem precisar de interação humana com os provedores de serviço.
b) Amplo acesso: os recursos computacionais são disponibilizados por meio da Internet e acessados através de mecanismos padronizados que possibilitam o uso por dispositivos móveis, computadores, etc.
c) *Pooling* de recursos: os recursos computacionais do provedor são utilizados para servir múltiplos usuários usando um modelo multi-inquilino (*multi-tenant*), com diferentes recursos físicos e virtuais, dinamicamente atribuídos e ajustados de acordo com a demanda dos usuários. Estes usuários não precisam ter conhecimento da localização física dos recursos computacionais, podendo somente especificar a localização em um nível mais alto de abstração, tais como o país, estado ou centro de dados.
d) Elasticidade: recursos podem ser adquiridos de forma rápida e elástica, em alguns casos automaticamente, caso haja a necessidade de escalar com o aumento da demanda, e liberados, na retração dessa demanda. Para os usuários, os recursos disponíveis para uso parecem ser ilimitados e podem ser adquiridos em qualquer quantidade e a qualquer momento.
e) Medição dos serviços: os sistemas de gerenciamento utilizados para computação em nuvem controlam e monitoram automaticamente os recursos para cada tipo de serviço (armazenamento, processamento e largura de banda). Este monitoramento do uso dos recursos deve ser transparente para o provedor do serviço, assim como para o consumidor do serviço utilizado.

3.2.1.1 Modelos de Serviço

Nessa sessão apresentamos e definimos os modelos de serviço de uma nuvem, que podem ser a de *Software* como um Serviço, Plataforma como um Serviço e a de Infraestrutura como um Serviço. Ao fim da sessão temos esses modelos ilustrados na figura 3.

Figura 3 – Modelos de Serviço



Fonte: REGO (2012)

3.2.2 Modelos de Implementação

a) *Software* como um Serviço (*Software as a Service* - SaaS): este modelo proporciona sistemas de software com propósitos específicos, que são disponíveis para os usuários por meio da Internet e acessíveis a partir de aplicações como o navegador Web. No SaaS, o usuário não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistema operacional, armazenamento ou mesmo as características individuais da aplicação, exceto configurações específicas.

b) Plataforma como um Serviço (*Platform as a Service* - PaaS): este modelo fornece sistema operacional, linguagens de programação e ambientes de desenvolvimento para as aplicações, auxiliando a implementação de sistemas de software. Assim como no SaaS, o usuário não administra ou controla a infraestrutura subjacente, mas tem controle sobre as aplicações implantadas e, possivelmente, as configurações de aplicações hospedadas nesta infraestrutura.

c) Infraestrutura como um Serviço (*Infrastructure as a Service* - IaaS): este modelo torna mais fácil e acessível o fornecimento de recursos tais como servidores, rede,

armazenamento e outros recursos de computação fundamentais para construir um ambiente de aplicação sob demanda, que podem incluir sistemas operacionais e aplicativos. Em geral, o usuário não administra ou controla a infraestrutura da nuvem, mas tem controle sobre os sistemas operacionais, armazenamento, aplicativos implantados e, eventualmente, seleciona componentes de rede, tais como *firewalls*.

Apresentamos agora os modelos de implantação da nuvem, que podem ser Privadas, Públicas ou Comunitárias, podendo existir nuvens híbridas que são uma solução mais elaborada que unifica mais de um desses modelos apresentados para conseguir utilizar melhor seus serviço. Esses modelos também podem ser observados visualmente na figura 4.

Figura 4 – Modelos de Implementação

Fonte: Criado pelo próprio autor

a) Nuvem privada (*Private Cloud*): A infraestrutura de nuvem é provisionada para uso exclusivo por uma única organização que agrupa vários consumidores. Pode ser local ou remota e administrada pela própria empresa ou por terceiros.
b) Nuvem comunidade (*Community Cloud*): Fornece uma infraestrutura compartilhada por uma comunidade de organizações com interesses em comum.
c) Nuvem pública (*Public Cloud*): A infraestrutura de nuvem é disponibilizada para uso aberto do público em geral. A nuvem é disponibilizada publicamente através do modelo pague pelo uso (*pay-per-use*) e tipicamente é oferecida por companhias que possuem grandes capacidades de armazenamento e de processamento.
d) Nuvem híbrida (*Hybrid Cloud*): A infraestrutura é composta por duas ou mais nuvens distintas (privada, comunitária, ou pública) que permanecem entidades únicas, mas conectadas através de tecnologia proprietária ou padronizada.

3.3 *Mobile Cloud Computing*

Existem várias definições de Computação em Nuvem Móvel, entre elas destacamos a de Mei *et al.* (2008), que afirma que *mobile cloud computing* é baseada nos conceitos básicos de *cloud computing*, de modo que existem requisitos básicos como a adaptabilidade, escalabilidade, disponibilidade e autoconsciência que precisam ser atendidos em uma nuvem e que também são requisitos válidos para a *mobile cloud computing*. Por exemplo, uma nuvem de computação móvel também precisa oferecer disponibilidade e qualidade de serviço, além de permitir que diversas entidades possam se conectar a ela, para que os usuários móveis possam ter bom proveito da nuvem.

Todavia, além dos requisitos semelhantes, uma nuvem móvel precisa considerar aspectos específicos da MCC tais como a mobilidade, baixa conectividade e uma fonte finita de energia. Como é definido em (FERNANDO *et al.*, 2013), o termo MCC refere-se a uma técnica para executar aplicações (como o Google tradutor) em um servidor remoto com mais recurso (neste caso, os servidores do Google), enquanto o dispositivo móvel funciona apenas como um cliente, conectando-se ao servidor remoto através de uma rede móvel, como é apresentado na Figura 5. Outros exemplos destes tipos de serviços para dispositivos móveis são o Facebook, Twitter e *widgets* meteorológicos.

Como é relatado em (FERNANDO *et al.*, 2013), várias abordagens já foram propostas, como exemplo, quando outros dispositivos móveis atuam como provedores de recursos da nuvem, formando uma rede *peer-to-peer* (MARINELLI, 2009). Nessa abordagem, os recursos coletivos dos vários dispositivos móveis conectados, e outros dispositivos estacionários, se disponíveis, são utilizados como provedores de recursos da nuvem. Essa abordagem apoia a mobilidade do usuário e reconhece o grande potencial de nuvens móveis. Uma outra abordagem, um pouco diferente da anterior, seria através da utilização de *cloudlets* como provedores do serviço de *offloading*.

3.3.1 Cloudlets

Um *cloudlet* é semelhante a um pequeno centro de dados que está situado em uma área bem determinada ligado a um servidor de nuvem maior através da Internet. Internamente, eles se assemelham a um *cluster* de computadores *multi-core* com alta conectividade interna e uma alta largura de banda através de uma rede sem fios. Estes pequenos servidores podem ser

Figura 5 – Visão geral de MCC

Fonte: Sergio *et al.* (2015)

utilizados em ambiente comuns como lanchonete, praças e shoppings ou até mesmo ambiente de empresas ou universidades, para que os dispositivos móveis possam se conectar e funcionar como clientes de um servidor local, em oposição a um servidor de nuvem remota cujo acesso iria apresentar problemas de latência e largura de banda.

Nessa abordagem, que foi proposta por Satyanarayanan *et al.* (2009), o dispositivo móvel envia parte do processamento a um servidor local, que é denominado de *cloudlet*, podendo ele ser composto de vários computadores *multi-core* com conectividade para os servidores em nuvem remota. Esse provedor de recursos permite trazer as funções de *offloading* para serem executadas em máquinas locais, utilizando as redes sem fio para conectar os dispositivos móveis à elas. Os principais motivos para se utilizar esse tipo de rede seria uma maior facilidade de gerenciamento e uma menor latência por serem bem menos congestionadas do que as redes públicas.

Como complemento ao trabalho de Satyanarayanan *et al.* (2009), que sugere que é mais eficiente o uso de pequenas nuvens locais ao invés de se conectar a uma nuvem distante como uma solução aos problemas de desempenho e de alto consumo energético dos dispositivos móveis, temos o trabalho de Bahl *et al.* (2012). Esse trabalho ressalta a superioridade dos resultados apresentados no uso de *cloudlets*, de modo que eles conseguem entregar um melhor serviço, principalmente se utilizados em determinadas categorias de aplicações que são sensíveis à percepção da latência.

Dando sequência a isso, temos o trabalho de Fernando *et al.* (2013) que ressalta a importância dos dispositivos móveis estarem em proximidade física com as *cloudlets*. Nesse cenário, os dispositivos móveis estariam funcionando apenas como clientes, enquanto todo o processamento dos dados estaria acontecendo dentro da *cloudlet*. O dispositivo móvel seria ligado ao *cloudlet* por uma conexão sem fio de baixa latência e com largura de banda elevada, garantindo assim a resposta em tempo real da requisições fazendo que os resultados sejam sempre retornados rapidamente.

3.3.2 Offloading

Em (SATYANARAYANAN *et al.*, 2009) o *offloading* é descrito com a principal operação realizada em uma nuvem móvel. Essa técnica viabiliza o descarregamento do processamento de um dispositivo móvel para um provedor de recursos externos. Semelhante a isso, no trabalho de Cuervo *et al.* (2010) o *offloading* é descrito como uma técnica que permite a execução de tarefas computacionais de dispositivos móveis em infraestruturas mais robustas, com maior poder processamento e maior espaço de armazenamento, podendo essas máquinas estarem hospedadas em uma nuvem local ou em uma pública.

Tal processo, como descrito em (ZHANG; LI, 2016) pode ser dividido em duas formas, estático ou dinâmico. Enquanto o modo estático tem a vantagem de possuir um baixo custo operacional durante a execução e ser uma abordagem válida apenas quando os parâmetros pode ser previstos com precisão e antecedência, o modo dinâmico pode se adaptar a diferentes condições de tempo de execução, tais como variações de latência, porém tem um custo computacional muito mais alto que a forma anterior.

Abordagens dinâmicas também podem utilizar mecanismos de previsão para tomada de decisão. Tais decisões, quando são mais dinâmicas, acabam por terem uma maior sobrecarga de processamento, afinal, o programa tem que monitorar as condições em tempo de execução. Mesmo para programas com decisões dinâmicas, as tarefas que podem potencialmente ser descarregadas são identificadas durante o desenvolvimento do programa (ZHANG; LI, 2016).

3.3.3 MpOS

Em (COSTA *et al.*, 2015) é realizada uma descrição do *MpOS*, uma ferramenta de *offloading* que permite que parte do processamento de um dispositivo móvel seja feito em um provedor de recursos externos. Essa ferramenta surgiu em frente a vários desafios de MCC, tendo

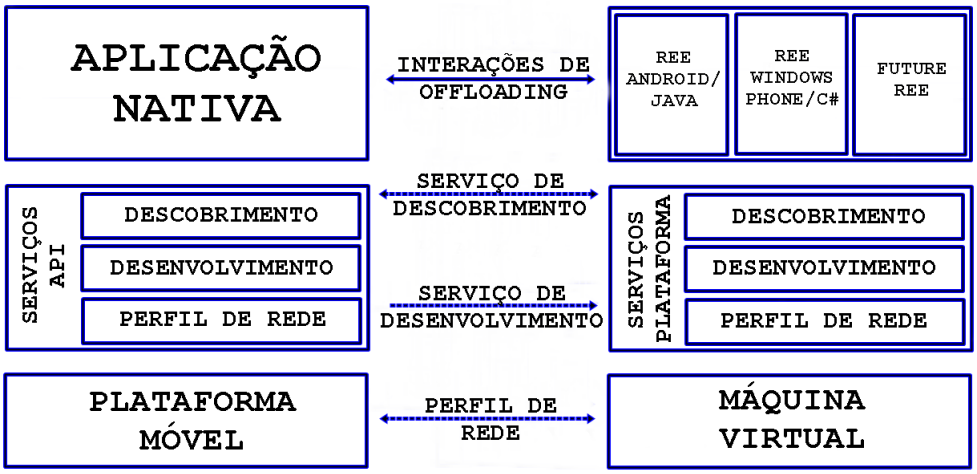
como principal objetivo ajudar os desenvolvedores de aplicativos móveis a lidar com questões relativas à falta de compatibilidade de diferentes plataformas (*Android e Windows Phone*) com uma única plataforma.

3.3.3.1 Arquitetura do MpOS

A arquitetura do *MpOS* é composta por três camadas, como é detalhado na Figura 6, sendo a primeira subdividida pelo ambientes de execução remota (do inglês *remote execution environments* - REE) no lado do servidor e pelas aplicações nativas no lado do cliente, além de possuir o serviço de *offloading*, que por sua vez é responsável por interceptar, analisar e enviar o descarregamento de operações para serem executadas nas REE.

A segunda camada é composta por duas partes, a primeira é a plataforma *MpOS* no lado servidor e a segunda é a *MpOS API* no lado cliente, que por sua vez, possui quatro serviços distintos: o serviço de descoberta, o serviço de implementação, o serviço de rede *profiler* e serviço de *offloading*.

Figura 6 – Arquitetura do MpOS



Fonte: Costa *et al.* (2015)

O primeiro serviço, de descoberta, é responsável por encontrar um servidor rodando em uma plataforma *MpOS* na rede local. O segundo é o serviço de implementação, que é usado apenas quando o anterior não encontra nenhum servidor rodando o *MpOS* e tem como função enviar todas as dependências necessárias para fazer um pedido para ser criado um novo serviço de desembarque para um aplicativo móvel.

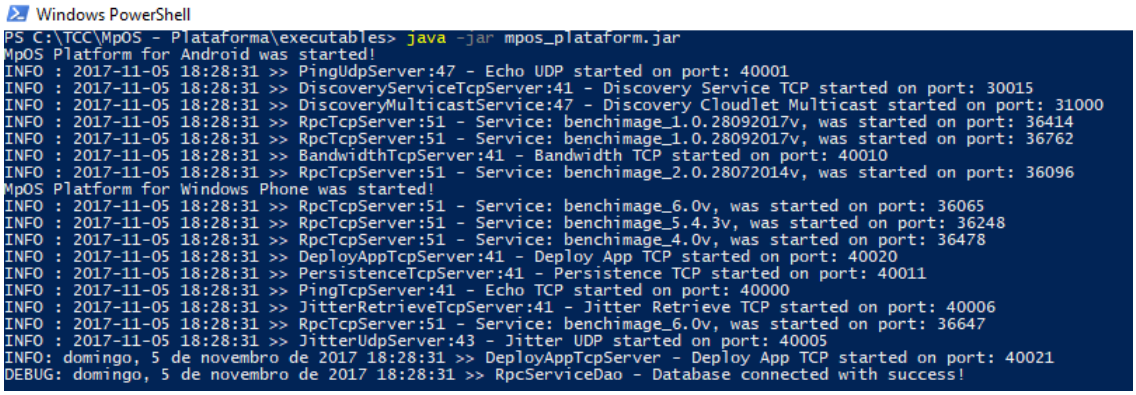
No terceiro, tem-se o serviço de rede *profiler*, que por sua vez tem como objetivo medir a qualidade da conexão entre dispositivo móvel e o servidor remoto, para que a partir dessa informação, possa atuar o último dos serviços, o serviço de *offloading*. Esse último serviço é responsável por decidir se os métodos das aplicações devem ser executados localmente, ou em um servidor remoto, e por fim, a última camada, que é representada por plataformas móveis no lado cliente e por um sistema operacional no lado servidor.

3.3.3.2 Funcionamento do MpOS

Após feita a conexão inicial, o usuário, que já iniciou a aplicação, pode realizar a aplicação do filtro da imagem localmente, na *cloudlet* ou na nuvem, como preferir. Para nossos testes foi definida uma ordem dessas execuções e quantidade fixa de testes para uma melhor análise dos dados, esses fatores serão detalhados na próxima subseção, que define melhor a aplicação cliente utilizada nessa análise.

O funcionamento da plataforma ocorre da seguinte forma: após iniciar a execução, no caso da *cloudlet*, o serviço de *Discovery* é realizado primeiramente, este, por sua vez divulga na rede a localização (endereço IP) da plataforma para os possíveis usuários. Feito isso, os usuários que estiverem na mesma rede da máquina que executa a plataforma e que estiverem executando o *benchImage2* já poderão se conectar à plataforma, como pode ser observado na figura 7, essa é um *print* dos *logs* da plataforma que apresenta o IP e Porta que o usuário se conectou.

Figura 7 – Logs de Conexão do MPoS na Cloudlet

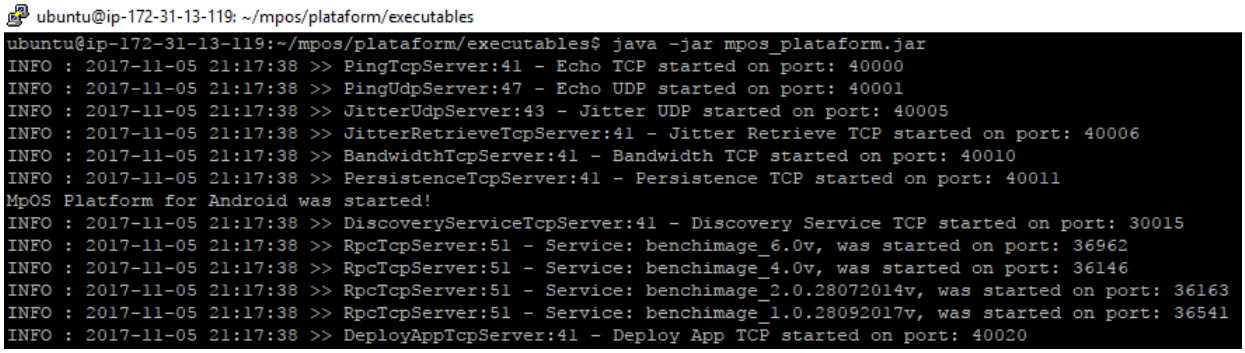


Fonte: Próprio autor Baseado Costa *et al.* (2015)

Para realizar o *offloading* na plataforma em execução na nuvem, o processo é um

pouco diferente. Primeiramente o recurso de *Discovery* é desabilitado, pelo fato de não estar trabalhando em uma rede local, então torna-se inviável a divulgação da localização de todos os possíveis usuários. Dessa forma, já que não tem a divulgação do endereço IP, deve ser definido na própria aplicação cliente o endereço do servidor para ser possível realizar a conexão. Os *logs* das conexões de novos clientes também são mostrados quando a plataforma está executando na nuvem e poder ser observados na figura 8.

Figura 8 – Logs de Conexão do MPoS na Nuvem



Fonte: Próprio autor Baseado Costa *et al.* (2015)

3.3.4 BenchImage2

Do lado do usuário, onde se origina o processamento, vários dispositivos foram configurados, dispositivos esses pertencentes aos alunos, professores e servidores do campus UFC Quixadá que se dispuseram a colaborar com o experimento. Nestes foram instalado uma variação da aplicação BenchImage2 desenvolvida por Costa *et al.* (2015), que se encontra disponível no repositório [1].

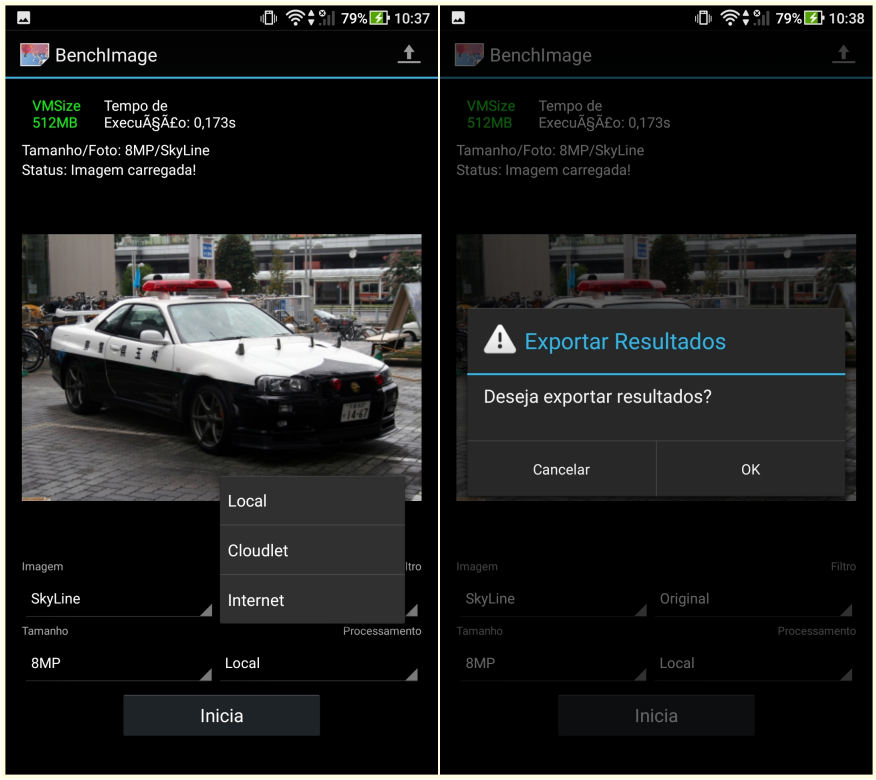
As especificações dos dispositivos utilizados são das mais variadas, caracterizando nossa ideia de mostrar o funcionamento e os nossos testes em ambiente real, com imprevisibilidade de tráfego na rede e dos dispositivos que utilizam a aplicação. O levantamento dos dispositivos que utilizaram nossa aplicação e participaram do nosso experimento pode ser observada no 1.

[1] https://github.com/ufc-great/mpos

3.3.4.1 Descrição da aplicação

A aplicação utilizada em nossos testes foi baseada na mesma utilizada no trabalho de (COSTA *et al.*, 2014), com alguma alterações para atender melhor às nossas necessidades. Tais adaptações são definidas detalhadamente na próxima subseção, assim como o motivo de cada uma delas. A figura 9 mostra como era a aplicação original. Como pode ser observado, ela possui um interface um pouco complexa com diversas opções de entrada para iniciar o teste.

Figura 9 – Interfaces do Aplicativo BenchImage2



Fonte: Próprio autor Baseado Costa *et al.* (2015)

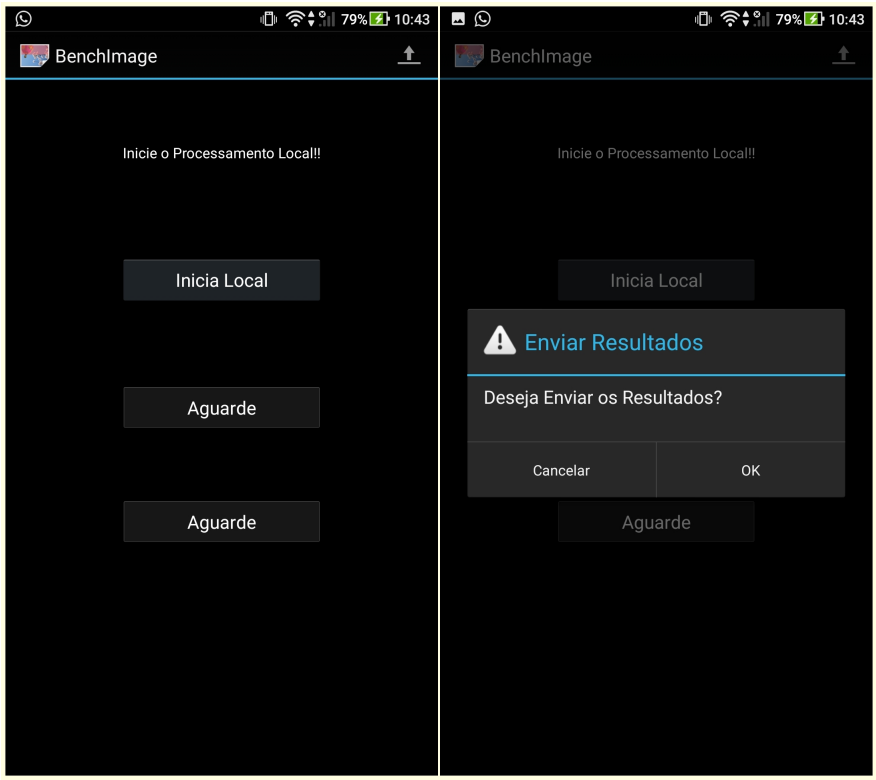
O BenchImage2 aplica um filtro em uma imagem, também, além dessa funcionalidade, os tamanhos da imagem podem ser variados (0,3MP, 1MP, 2MP, 4MP ou 8MP), o local do processamento (local, na *cloudlet* ou na nuvem) e por fim, pode ser feita a seleção da própria imagem à ser processada.

3.3.4.2 Adaptações da aplicação

Para a aplicação mobile foram realizadas adaptações no código do benchImage2. Algumas funcionalidades foram removidas, alteradas ou acrescentadas e se fizeram necessárias

para que a aplicação atendesse à nossas necessidades de uso com muito usuários. Tais adaptações podem ser observadas na figura 10 que contém a interface nova, pós adaptações.

Figura 10 – Interfaces do Aplicativo BenchImage2



Fonte: Próprio autor Baseado Costa *et al.* (2015)

- Seleção de Filtro – Removida: Na nova versão, o usuário não precisa selecionar o filtro que deseja utilizar, essa escolha é feita pela própria aplicação. Essa adaptação se fez necessária pelo fato de termos feito testes com vários usuários, dessa forma, se eles selecionassem filtros diferentes poderia ocasionar alteração e imprevisibilidade nos resultados, já que filtros diferentes geram processamento e tráfego diferentes.
- Seleção do Local de Processamento – Alterada: Na nova versão, o usuário só pode selecionar um local por vez, e a ordem é pré definida. Quando o usuário deve ser executado na cloudlet, por exemplo, os outros dois ficam bloqueados, impedindo assim a execução incorreta. Essa adaptação foi feita pela necessidade de realizar os testes em todos os cenários e em determinada ordem, como uma maneira de controlar, fazendo com que os diferentes usuários estejam sempre na mesma fase de execução.

- Seleção de Tamanho da Imagem – Removida: Na nova versão, o usuário não precisa selecionar o tamanho da imagem que ele deseja utilizar, essa escolha é feita pela própria aplicação. Essa adaptação se fez necessária pelo fato dos tamanhos de imagem diferentes ocasionarem um aumento ou diminuição de processamento e tráfego, trazendo uma imprevisibilidade nos resultados.
- Seleção de Imagem – Removida: Na nova versão, o usuário não seleciona a imagem que deseja utilizar, essa escolha é feita pela própria aplicação. Essa adaptação foi feita por que todos os usuários devem utilizar a mesma imagem para a aplicação do filtro, já que imagens diferentes ocasionaria em aumento ou diminuição de processamento e tráfego, ocasionando uma imprevisibilidade nos resultados.
- Exportação de Informações – Removida: Na nova versão, o usuário não tem a opção de fazer a exportação dos dados em arquivo no próprio dispositivo, como podia ser feito anteriormente. A função de exportação foi substituída pelo envio de informações, detalhada no próximo ponto.
- Envio de Informações – Adicionada: Na nova versão, o usuário pode enviar as informações geradas pela aplicação para a plataforma. Essa adaptação foi adicionada pois na versão anterior, a aplicação apenas criava um arquivo no próprio dispositivo móvel com as informações do teste, dificultando assim a coleta dos dados, principalmente em um ambiente com muitos usuários.
- Coleta de Informação dos Dispositivos - Adicionada:Na nova versão,o aplicativo coleta algumas informações do hardware do aparelho. Essas informações foram utilizadas em uma análise, para estimar os dispositivos mais utilizados na ambiente proposto, assim como o poder de processamento e armazenamento disponíveis nos mesmos.

3.3.4.3 Funcionamento da aplicação

Quando o usuário inicia a aplicação, já realiza em segundo plano duas coisas cruciais para a o teste. Primeiro, o descobrimento da plataforma que está configurada na mesma rede, por meio do método Discovery, através de requisições multcast, segundo, a conexão com a plataforma configurada na Instancia AWS (nuvem). Feito isso, o o processamento já pode ser iniciado, aplicando em determinada imagem.

4 PROPOSTA

Para a avaliação da técnica de *offloading* em ambientes densos em relação ao número de usuários, propomos nesse trabalho a execução de um experimento que busca testar cenários típicos de execução de aplicações móveis. Para a realização dos testes foi necessária a configuração do lado servidor e cliente. A parte do servidor também é dividida em duas, *cloudlet* que foi configurada em uma máquina de uso comum da própria Universidade Federal do Ceará e a nuvem que foi configurada em uma instância EC2 da Amazon.

No nosso servidor, em ambos os cenários (local e remoto), foi configurada a aplicação MpOS, executando em uma máquina com o sistema operacional Linux, para atuar como a plataforma de *offloading*, essa plataforma recebe as informações passadas pelos usuários, faz o processamento e retorna aos mesmos os resultados.

O outro cenário a ser configurado foi o do usuário, que fez a utilização da aplicação BenchImage2 rodando em dispositivos Android. Para utilizar essa aplicação em nosso trabalho, foi necessário realizar algumas customizações, adicionando e removendo algumas funcionalidades para que essa atendesse a todas as nossas necessidades. Tais modificações serão apresentadas na subseção de configuração do usuário.

Nas subseções seguintes também apresentamos o ambiente configurado para a realização dos testes e mostraremos de forma detalhada a configuração da plataforma MpOS no cenário local e remoto. Em seguida, detalhamos a configuração do cliente móvel - Android através da ferramenta BenchImage2, aplicação que utilizamos no lado cliente.

4.1 Servidor

O lado servidor, que está executando a plataforma MpOS, foi configurado de duas maneiras. A primeira dessas foi realizada em uma instância AWS que caracteriza o uso da plataforma em uma nuvem pública, enquanto a segunda, foi em um computador de uso comum localizado na mesma rede do dispositivo cliente, caracterizando o uso de uma *cloudlet*.

O servidor local foi configurado em uma máquina pessoal do próprio autor. A mesma conta com um processador Intel i5, memória RAM de 8 GB e armazenamento de 1 TB e tem como sistema operacional o Windows 10. Nessa máquina, está em execução a plataforma *MPoS*, que é melhor descrita na próxima subseção. Ela é responsável por realizar parte do processamento da aplicação *BenchImage2* que será melhor detalhada na parte cliente.

O servidor de descarregamento remoto foi configurado em uma instância m4.4xlarge da nuvem *Amazon*(AWS), que tem como sistema operacional o Ubuntu 16.04 LTS, com 64Gb de memória RAM e 16 vCPUs. Nesta máquina também está configurada a mesma plataforma de *offloading*(MPoS) para a realização dos testes.

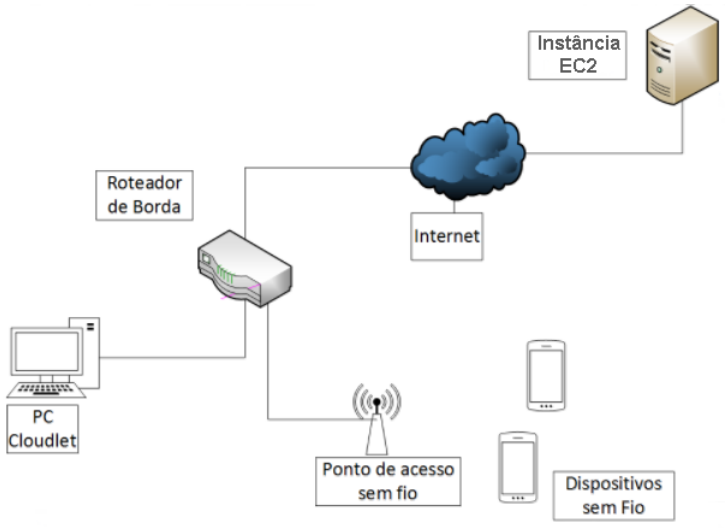
4.2 Usuário

Do lado do usuário, onde se origina o processamento, vários dispositivos foram configurados, dispositivos esses pertencentes aos alunos, professores e servidores do campus UFC Quixadá que se dispuseram a colaborar com o experimento. Nestes foram instalado uma variação da aplicação *BenchImage2* desenvolvida por (COSTA *et al.*, 2015), que se encontra disponível no repositório GitHub[1].

As especificações dos dispositivos utilizados são das mais variadas, caracterizando nossa ideia de mostrar o funcionamento e os nossos testes em ambiente real, com imprevisibilidade de tráfego na rede e dos dispositivos que utilizam a aplicação. O levantamento dos dispositivos que utilizaram nossa aplicação e participaram do nosso experimento pode ser observada no Apêndice A.

Os elementos da arquitetura proposta podem ser observados na figura 11

Figura 11 – Arquitetura do Sistema



Fonte: Criada pelo próprio autor

[1] https://github.com/ufc-great/mpos

4.2.1 Análise dos Dados

Para realizar a análise dos dados gerados pela nossa aplicação definimos as métricas, os parâmetros da aplicação e os fatores com seus níveis. Iniciando com as métricas, como já foi observado na seção de trabalhos relacionados as utilizadas em nosso trabalho semelhantes as utilizadas no trabalho de Sergio *et al.* (2015), são ela: tempo de cpu, tempo de download, tempo de upload e o tempo total de processamento que é a soma dos três anteriores com o tempo de processamento local.

- Tempo de processamento remoto - Tempo do processamento da requisição enviada pelo usuário para o servidor ser finalizada.
- Tempo de Upload - Tempo resultado desde o envio dos dados para o *offloading* do cliente até ser recebido pelo servidor.
- Tempo de Donwload: Tempo resultado desde o envio dos resultados do servidor até ser recebido pelo cliente.
- Tempo total de processamento: Tempo total de resposta desde o envio da requisição do dispositivo móvel para a nuvem até o recebimento da resposta pelo mesmo. A fórmula que será utilizada para o cálculo desse tempo é apresentada a seguir, onde o tempo total de resposta(T_{total}) é igual a soma do tempo de processamento remoto (T_{Tcpu}), do tempo de *upload* (T_{up}) e o tempo de *download*(T_{dw}).

$$T_{total} = T_{cpu} + T_{up} + T_{dw}$$

O restante das informações pode ser observada na Tabela 2, que além da métricas apresenta também os parâmetros do sistema, são eles: CPU, memória e carga de processamento do servidor. Além disso, pode ser observado na tabela os fatores utilizados nos testes, assim como seu níveis que são eles: Quantidade de requisições simultâneas, tamanho da imagem processada e o local do processamento.

Tabela 2 – Resumo da execução da avaliação de desempenho

Características	**Descrições**
Ferramenta de análise	MpOS
Parâmetros do Sistema	CPU Memória Carga de processamento
Métricas	Tempo de CPU Tempo de upload Tempo de download Tempo total de processamento
Fatores	Tamanho da imagem: 1MP, 2MP e 4MP Local de Processamento: Cloudlet e Nuvem Numero de usuários simultâneos: 1, 2,4, 8, 16 e32
Tecnicas de avaliação	Experimentação
Apresentação dos resultados	Gráficos e tabelas

5 EXPERIMENTAÇÃO

Com os testes executados e a análise dos dados realizada, conseguimos resultados satisfatórios e significativos, apesar de não terem sido como esperávamos de início. No início dos experimentos, esperávamos que quando aumentássemos a quantidade de requisições em ambos os cenários, o que deveria acontecer era o seguinte, quando o experimento fosse realizado apenas com um usuário, o tempo total de processamento na nuvem seria superior ao da cloudlet, pois apesar dela tem um processamento melhor e um tempo de CPU bem menor, a latência impedia que o tempo final fosse melhor. Dessa forma, a *cloudlet* seria uma melhor opção nesse cenário, quando observado apenas o tempo total do processamento.

Esperávamos que, quando aumentássemos a quantidade de requisições, a nuvem manteria seu tempo de processamento estável, ou seja, não iria se elevar de maneira significativa, enquanto a *cloudlet* por ter menos recurso iria sofre com a perda de processamento e iria aumentar o tempo final. Dessa forma, acreditávamos que quando aumentássemos a quantidade de requisições em uma quantidade X, a nuvem iria superar a *cloudlet* e ter um resultado melhor em relação ao tempo total de processamento.

Porém, a latência superou as expectativas e foi totalmente impactante nos resultados, independente da quantidade de usuários simultâneos. Dessa forma, a nuvem, apesar de ter um processamento superior em todos os casos em relação a *cloudlet*, obteve um desempenho final inferior, pois como já mencionado, a latência impediu que ela tivesse um resultado melhor. Todavia, em nosso experimentos os recursos da cloudlet se esgotaram mais rapidamente, de modo que só foi possível a realização de testes com 8 usuários simultâneos.

Para o desenvolvimento dos experimentos deste trabalho, utilizou-se a proposta descrita no capítulo anterior. Neste capítulo, apresentaremos de maneira mais aprofundada a proposta de nosso trabalho, detalhando nossos ambientes de teste, bem como suas configurações e as ferramentas utilizadas para construção dos mesmos. Iniciamos com uma breve contextualização da proposta e em seguida com a própria configuração dos ambientes.

Este trabalho apresenta uma análise de desempenho do processo de *offloading* ao ser realizado em um ambiente de rede real. O ambiente escolhido foi a rede do campus da Universidade Federal do Ceara, de Quixadá, por diversos motivos, são alguns deles: grande quantidade de dados circulando na rede com tráfego de diversos usuários e aplicações, uma rede bem configurada com divisões de vlans e sub-redes que facilita o controle e, por fim, por conhecermos a arquitetura e os usuários necessários para a realização de nossos testes.

Como já foi citado no capítulo dos trabalhos relacionados, este trabalho tem grande semelhança com o trabalho de Costa *et al.* (2015), onde realizaram uma análise do desemprenho do serviço de *offloading* quando utilizado em uma nuvem local e em nuvem remota, porém, eles não levaram em consideração alguns pontos, que utilizamos em nosso trabalho como diferencial, para que, através de uma análise de desempenho, pudéssemos mostrar se esses pontos são ou não relevantes em um ambiente de MCC.

O primeiro ponto que utilizamos como diferencial em nossos experimentos foi a variação na quantidade de usuários simultâneos que utilizam os serviços da nuvem. Este fator foi escolhido pelo fato de que apesar de alguns trabalhos como (COSTA *et al.*, 2015) e (COSTA *et al.*, 2014) já terem mostrado que a *cloudlet* pode ser superior a nuvem remota em algumas aplicações, acreditamos que, quando aumentarmos a carga de processamento devido às várias requisições de diversos usuários simultâneos, ela deve começar a perder desempenho e a nuvem remota (com mais recursos) pode até superar a *cloudlet* em termos de desempenho.

O segundo ponto que levamos em consideração foi a realização dos testes em ambiente de rede real e compartilhada com outros usuários e aplicações, caracterizando a utilização de redes densas. Esse tipo de ambiente tornaram os testes mais próximos ao real, de modo que uma rede dedicada não atenderia aos nossos objetivos. Neste ponto pretende-se mostrar como o processo de *offloading* se comporta em uma rede com concorrência de banda com o tráfego de outras aplicações.

A validação desta proposta foi realizada buscando-se verificar o funcionamento das funcionalidades de ambas as ferramentas utilizadas em nossa análise, tais como o tempo necessário para o *offloading* pela aplicação cliente e pela execução remota da *cloudlet* e da nuvem. Essa seção é divida em planejamento, experimentos e resultados. A seção de planejamento detalha todo material utilizado e configurações realizadas para criação do sistema. A seção de experimentos detalha todos os passos para a implementação do sistema no ambiente de estudo e por fim, na seção de resultados mostramos a análise dos dados coletados pelo sistema

5.1 Planejamento

As subseções abaixo detalham todo o material e configurações utilizadas para a execução da plataforma e do aplicativos móvel descrito na seção 4.

5.1.1 Ambientes de Execução

Na parte servidor, para a *cloudlet*, foi configurada a plataforma MPoS em uma máquina de um laboratório da Universidade Federal do Ceará - Campus Quixadá. Esta máquina conta com processador 64 bits Intel de 5º geração de core I5 de 3.10 GHz, com memória RAM de 8 GB e armazenamento de 1 TB e o Ubuntu 16.04 LTS como sistema operacional. Esta máquina está conectada diretamente à rede da instituição a qual os usuário utilizarão para acessá-la.

Ainda na parte servidor, temos a configuração da plataforma na nuvem. Esta, por sua vez, está configurada em uma instancia EC2 da *Amazon* que é do tipo t2.2xlarge e conta com processador Intel *Broadwell* E5-2686v4 de 2,30 GHz, 8 vCPUs, 32 GB de memória RAM e com uma performance de rede moderada. Ela fica disponível para os usuários por meio da *Internet*, podendo ser acessada por qualquer usuário em qualquer lugar. Para os nossos testes, todos os usuários utilizaram a rede sem fio do Campus para a realização dos testes.

Foi necessária ainda uma terceira máquina na parte servidor. Nela foi configurado o servidor de armazenamento, para que fosse possível para o usuário a exportação dos resultados das execuções para o servidor sem interferir no desempenho da plataforma, que agora executa em uma máquina exclusiva para o serviço de *offloading*. Como essa máquina tem a função apenas de receber os resultados após o fim da aplicação, não serão definidas suas especificações, já que isso não interfere nos resultados da análise.

5.2 Experimentos

Após finalizada a edição do aplicativo usuário, bem como a configuração do servidor de armazenamento e monitoramento dos recursos, foram iniciados os experimentos.

5.2.1 Metodologia de Execução dos Testes

A execução dos testes ocorreu no campus da Universidade Federal do Ceará em Quixadá e contou com a participação e colaboração de mais de 100 alunos de diferentes cursos e turmas. Foram realizadas diversas execuções entres os dias 03 de Outubro e 21 de Novembro em várias áreas do campus, como o restaurante universitário, área de convivência, laboratórios e salas de aula.

O experimento ocorreu da seguinte forma, durantes os dias citados anteriormente, a plataforma estava executando nas máquinas configuradas, como também foi descrito nas seções

anteriores, e com isso, alguns grupos de alunos foram formados, esses fizeram o download da aplicação e a executaram simultaneamente. Esses grupos variavam de tamanho a cada bateria de testes, de modo que um grupo sempre deveria ter o dobro de alunos da quantidade do grupo anterior.

5.2.2 Tratamento dos Dados

Após finalizar os testes, foi iniciada a parte do tratamentos dos dados. Como já foi descrito anteriormente, a nova versão do *BenchImage2* tem a função de exportar os resultados para o servidor de armazenamento, que por sua vez tem como função armazenar as informações em um banco de dados *postgresql* e em seguida exportá-lo em planilhas, tornando mais fácil o tratamento dos dados.

Depois dos dados já estarem nas planilhas, é iniciada a segunda etapa, onde um script criado pelo autor tem o papel de organizar e agrupar os dados de acordo com os diferentes fatores para que então possam ser gerados os gráficos e assim ser realizada uma melhor análise dos dados. Por fim, para a criação dos gráficos e o cálculos de médias e intervalos de confiança foi utilizada a ferramenta R [1]. Vale destacar que em nossos experimentos foram considerados todos os dados coletados, inclusive alguns *outliers*, pois como foram utilizados diversos dispositivos, com diferentes configurações, optou-se por considerar tais dados já que a ideia é de experimentação em ambiente real, com essas diversidades.

5.3 Resultados

Apresentamos os resultados de nosso trabalho da seguinte forma, mostramos os gráficos agrupados dos resultados de nossa análise, comparando sempre as informações do processamento nos dois cenários principais de nosso experimento (*cloudlet* e nuvem), na esquerda pode-se observar o gráfico referente ao processamento na *cloudlet* e ao lado direito temos o gráfico com o resultado do processamento em nuvem.

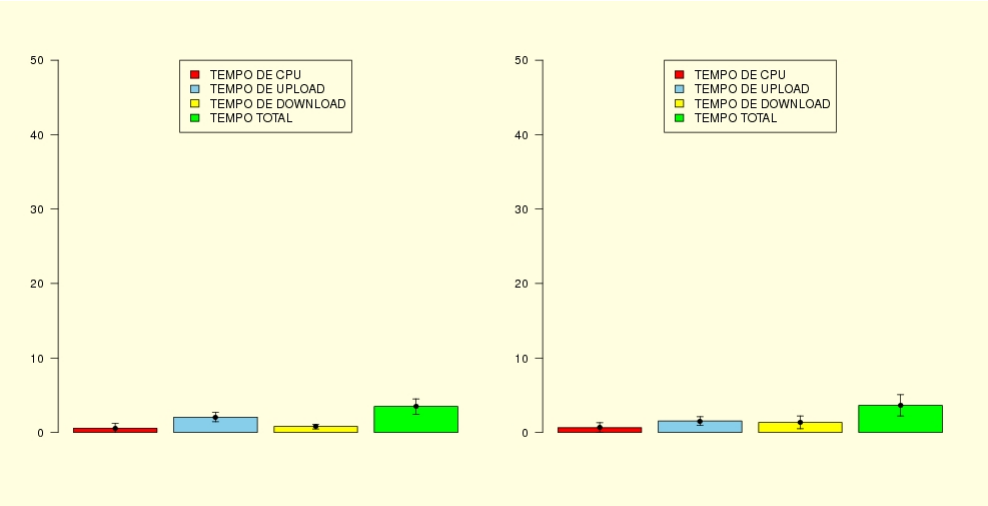
Em cada gráfico as informações são apresentadas da seguinte forma: o tempo de CPU pode ser observado na cor vermelha, o tempo de *upload* é apresentado na cor azul claro, o tempo de *download* está na cor amarela e por fim o tempo total de resposta é apresentado na cor verde. Mostramos um grupo de gráfico para cada tamanho de imagem (1MP, 2MP e 4MP).

Os primeiros testes ocorreram da seguinte forma, o *BenchImage2* foi instalado apenas

[1] https://www.r-project.org/

em um dispositivo móvel e executou a aplicação nos três cenários do experimento. Foi utilizada a imagem de 1MP e como resultados obtivemos o que pode ser visualizado na figura 12. Em seguida, foi realizada a execução com a mesma imagem, mas com uma resolução maior, a fim de gerar mais tráfego na rede e maior processamento. As informações dessas outras etapas podem ser observadas nas figuras seguintes.

Figura 12 – Resultados Para Imagem de 1MP com 1 Usuário

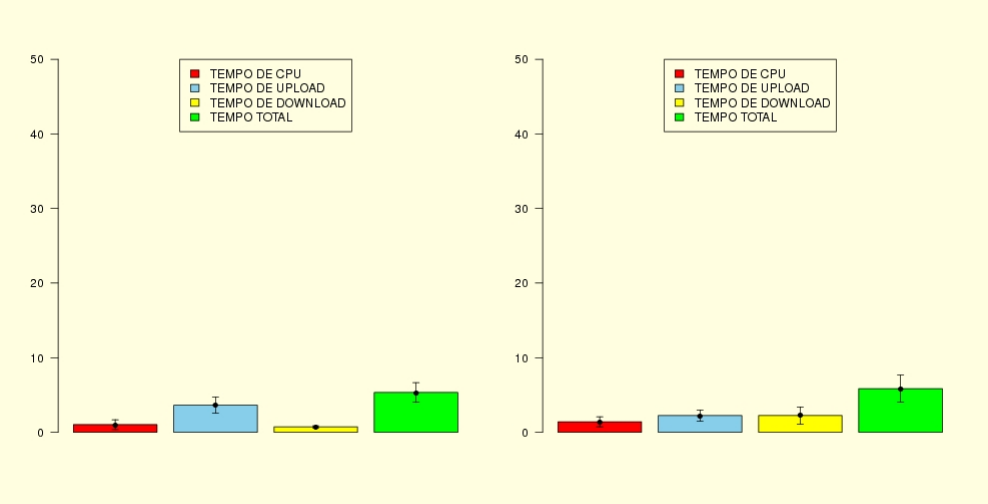


Fonte: Criada pelo próprio autor

Na figura 12 podemos observar que os resultados de nossas análise comprovam a teoria relatada anteriormente, onde a latência tem grande peso no resultado final. No primeiro caso, com a imagem de 1MP e apenas um usuário podemos observar que o tempo de CPU da cloudlet possui média inferior ao da nuvem, já que a mesma possui menos recursos que sua concorrente. Porém, quando observamos o tempo total de processamento, que é o somatório de todas as fases do processo, vemos que a *cloudlet* acaba tendo um resultado melhor, pois o tempo de *upload* e *download* ficou muito mais alto quando o *offloading* é realizado na nuvem por conta da alta latência.

Em seguida realizamos o experimento com uma imagem de 2MP, a fim de observar o comportamento das plataforma e ver como isso afetaria seus desempenhos. Os resultados apresentados foram semelhantes aos mostrados anteriormente. A latência, mais uma vez, teve papel significativo nos resultados, de modo a fazer com que a *cloudlet* tivesse o desempenho final ainda melhor, já que seu tempo de *download* e *upload* são bem mais baixos, mesmo não possuindo um tempo de CPU tão bom quanto ao da nuvem. Isso pode ser observado na figura 13.

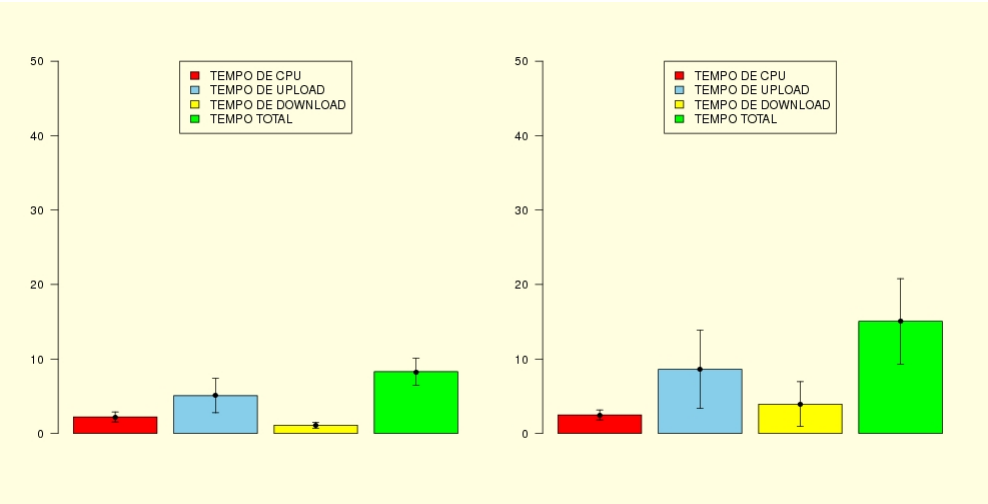
Figura 13 – Resultados Para Imagem de 2MP com 1 Usuário



Fonte: Criada pelo próprio autor

Para finalizar os testes apenas com um usuário, realizamos experimento com uma imagem de 4MP, e os resultados continuaram semelhantes aos anteriores. Após finalizar a primeira bateria de testes podemos concluir que mesmo que a técnica de *offloading* seja realizada com apenas um usuário, a latência ainda é um grande vilão para esse tipo de abordagem. na figura 14 podemos observar esses resultados.

Figura 14 – Resultados Para Imagem de 4MP com 1 Usuário

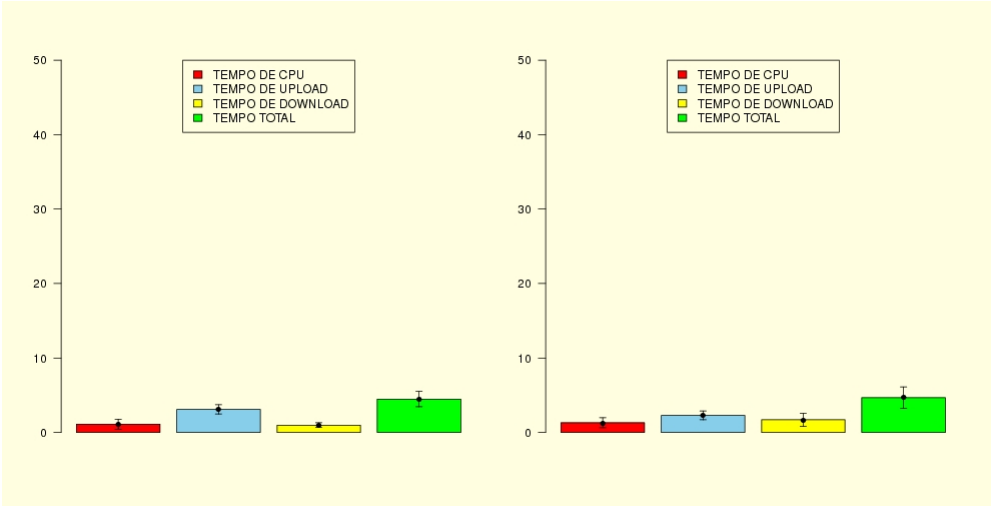


Fonte: Criada pelo próprio autor

A segunda parte de nossos testes foi a realização do experimento com o dobro de usuários da rodada anterior, ou seja, com dois usuários utilizando a aplicação simultaneamente.

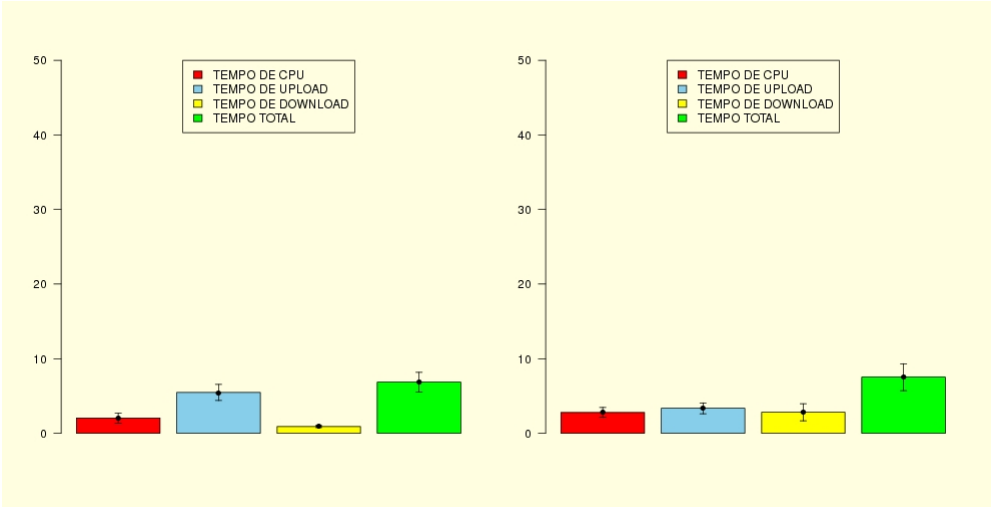
Os resultado obtidos não foram muito diferentes dos anteriores, de modo que a nuvem mostrou melhor desempenho, porém a latência fez com que ela, mais uma vez, obtivesse tempo final maior que o da *cloudlet*. Os resultados para a execução com 2 usuários e a imagem de 1MP, que podem ser observados nos gráficos abaixo, na figura 15. Já na figura 16 temos o resultados para a execução com 2 usuários e imagem de 2MP, por fim, a figura 17 apresenta os resultados de 2 usuários com imagem de 4MP.

Figura 15 – Resultados Para Imagem de 1MP com 2 Usuários



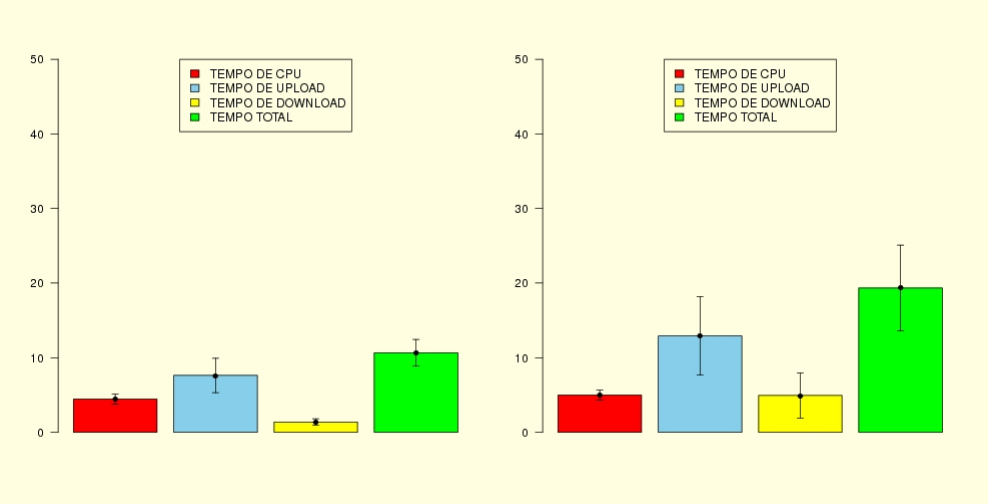
Fonte: Criada pelo próprio autor

Figura 16 – Resultados Para Imagem de 2MP com 2 Usuários



Fonte: Criada pelo próprio autor

Figura 17 – Resultados Para Imagem de 4MP com 2 Usuários



Fonte: Criada pelo próprio autor

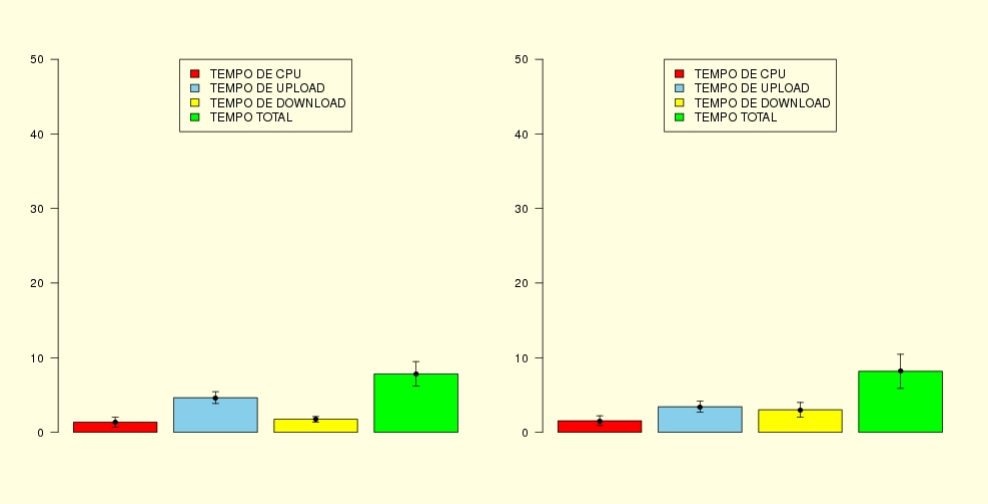
Como pode ser observado, a variação dos valores ainda não foi tão grande. Algo que pode ser observado é o fato do processamento local não variar muito, afinal, variar a quantidade de usuários utilizando a aplicação não vai afetar em nada o processamento local do dispositivo móvel, uma vez que, ele vai estar processando apenas a sua aplicação e não a dos demais, dessa forma, a variação ocorre apenas quando alteramos o tamanho da imagem a ser processada. Outro ponto que deve ser lembrado é o fato de que os valores tempo de CPU, tempo de *upload* e tempo de *download* são zerados por ser uma execução local, dessa forma não tem envio, recebimento ou processamento remoto.

Finalizada essa etapa, aumentamos mais um vez a quantidade de usuários, novamente dobramos, o que antes eram dois, agora são quarto dispositivos realizando o *offloading* para a *cloudlet* e nuvem. Com o aumento da quantidade de usuário já podemos notar um variação um pouco maior, comparada com a anterior.

Abaixo, na figura 18 temos os resultados para a execução com 4 usuários e imagem de 1MP, na figura 19 temos o resultados para a execução com 4 usuários com imagem de 2MP e por fim, a figura 20 apresenta os resultados de 4 usuários com imagem de 4MP.

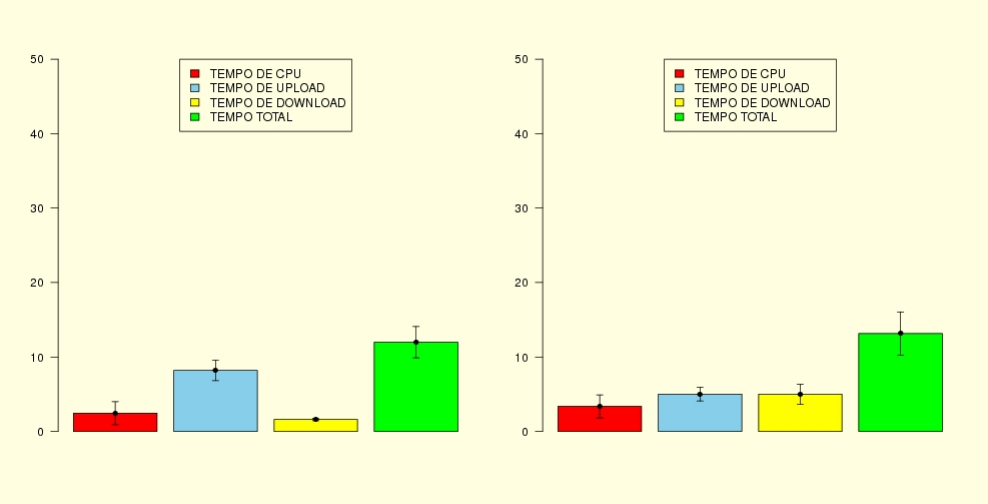
Seguindo com os experimentos, dobramos mais uma vez essa quantidade de aplicações utilizando o *BenchImage2*, realizando os teste com 8 usuários simultâneos. Com esse aumento, podemos notar algumas diferenças mais expressivas, se comparadas com as anteriores. Um dos fatos que tornaram-se mais notórios com esse aumento de processamento foi a perda de desempenho da *cloudlet* enquanto a nuvem se manteve sem muitas variações.

Figura 18 – Resultados Para Imagem de 1MP com 4 Usuários



Fonte: Criada pelo próprio autor

Figura 19 – Resultados Para Imagem de 2MP com 4 Usuários

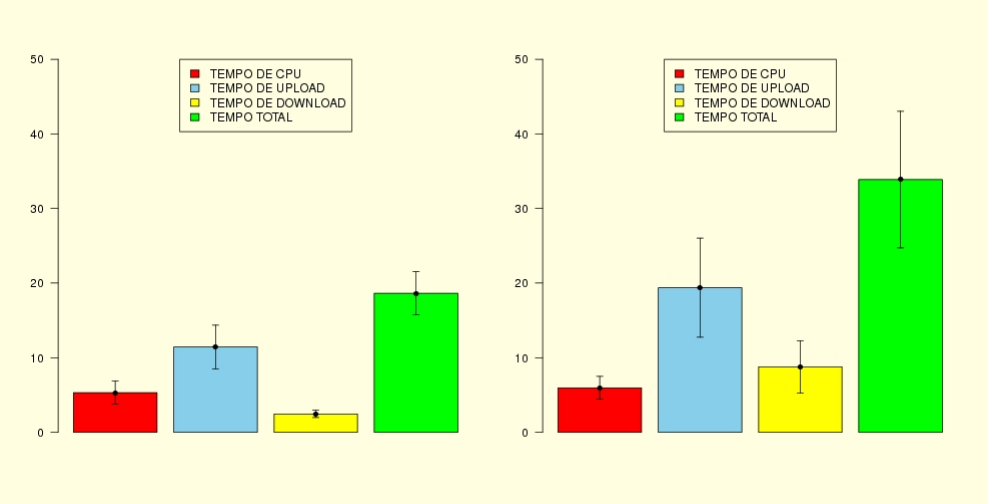


Fonte: Criada pelo próprio autor

Os resultados podem ser observados nas figuras abaixo, na figura 21 temos resultado para a execução com 8 usuários e a imagem de 1MP, na figura 22 temos o resultados para a execução com 8 usuários com imagem de 2MP e por fim, a figura 23 apresenta os resultados de 8 usuários com imagem de 4MP.

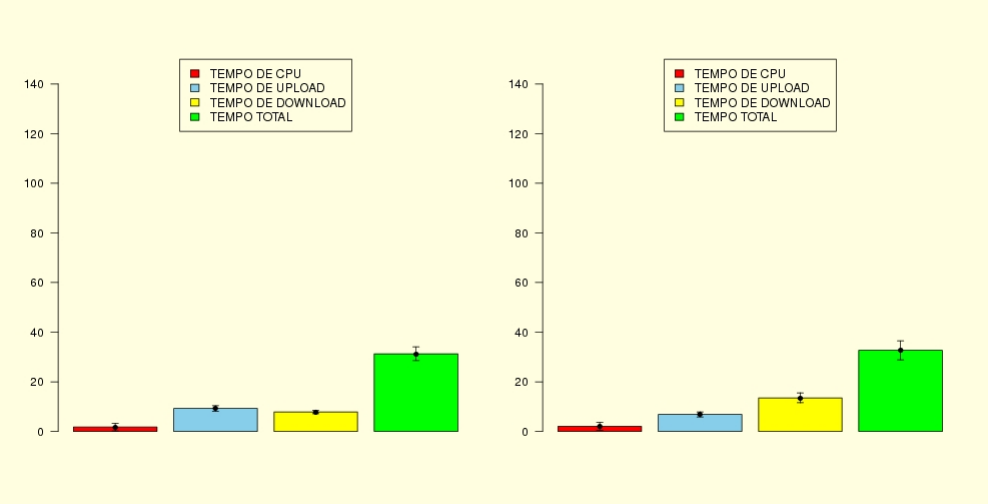
Após o teste com 8 usuários, tentamos ainda a execução com ainda mais, dobrando mais uma vez essa quantidade. Porém, quando tentamos realizar esse aumento, os testes foram interrompidos na maioria das tentativas quando estavam executando na *cloudlet*, porém na nuvem, apesar dos tempos terem sido bem altos, ainda conseguimos finalizar alguns testes sem

Figura 20 – Resultados Para Imagem de 4MP com 4 Usuários



Fonte: Criada pelo próprio autor

Figura 21 – Resultados Para Imagem de 1MP com 8 Usuários

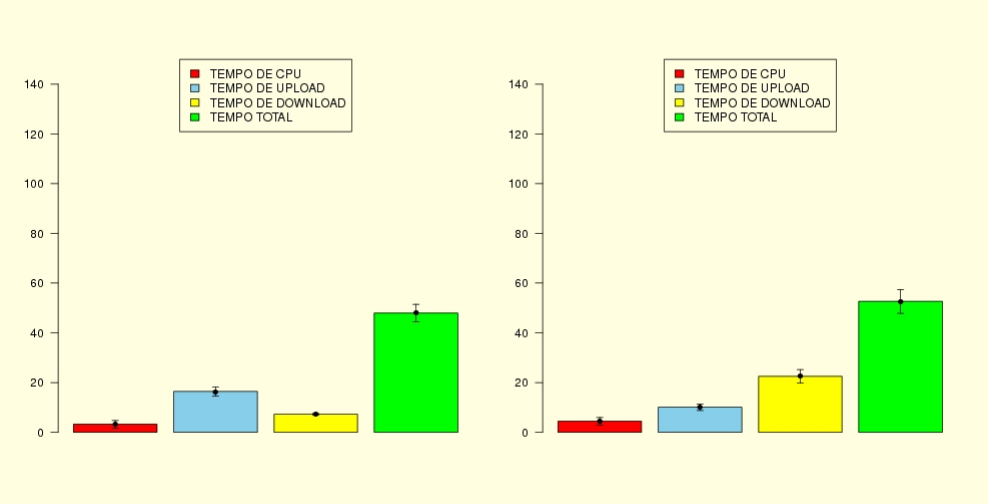


Fonte: Criada pelo próprio autor

erros em algumas vezes. Todavia quando os dados eram enviados para o servidor ocorria erro nos parâmetros. Através do monitoramento dos recursos identificamos que nesse último teste os recursos da *cloudlet* foram quase que totalmente consumidos, de modo que a mesma não possuiu *hardware* suficiente para tal execução. A nuvem, também teve um aumento expressivo na utilização dos recursos, mas como possui um *hardware* bem mais robusto, conseguiu finalizar o experimento, mesmo tendo uma queda significativa do desempenho.

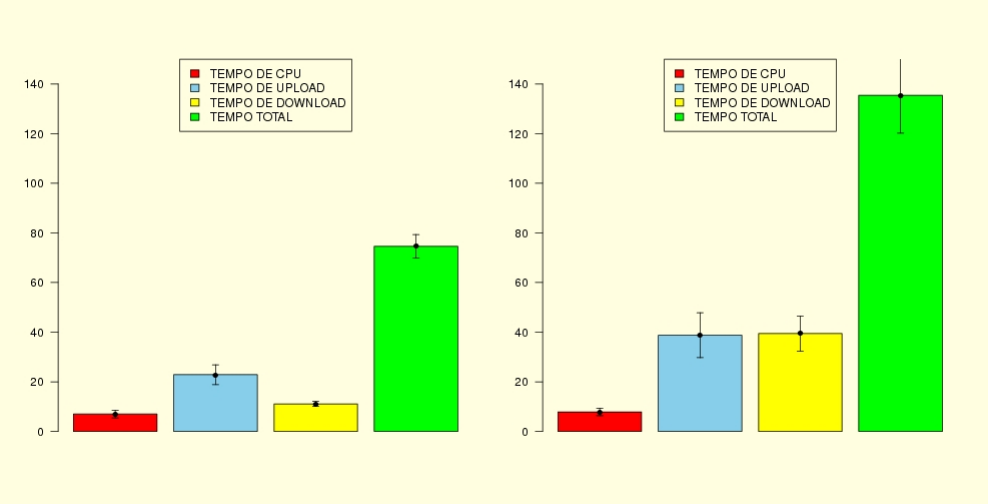
Para finalizar, tentamos um teste final com 32 usuários, e dessa vez nada foi concluído, pois por conta da demora, muito usuários tiveram que sair ou fecharam a aplicação por algum

Figura 22 – Resultados Para Imagem de 2MP com 8 Usuários



Fonte: Criada pelo próprio autor

Figura 23 – Resultados Para Imagem de 4MP com 8 Usuários



Fonte: Criada pelo próprio autor

motivo. Ocorreu também de alguns diapositivos mais simples não suportarem a carga de processamento e fecharem a aplicação após algum tempo. Nos dois últimos testes a maioria dos casos não tivemos dados suficientes para análise por serem interrompidos de maneira indevida.

No início de nossos estudos não esperávamos que a latência seria tão significativa nos resultados, esperávamos que a nuvem iria ultrapassar rapidamente a cloudlet por conta do seu melhor desempenho e maior quantidade de recurso. Porém, não foi isso que aconteceu, ao menos não no tempo total, já que apesar do tempo de processamento da nuvem ter sido bem melhor se comparado a *cloudlet*, o tempo de *upload* e *download* elevaram muito o tempo total

final.

Sendo que com o aumento significativo do nível de processamento, a nuvem conseguiu finalizar o experimento, diferente da *cloudlet*. Isso mostrou que para esse tipo de cenário, a *cloudlet* precisaria de muito mais recursos para continuar mantendo seu desempenho próximo ao da nuvem quando fôssemos aumentando cada vez mais o número de requisições. Enquanto a nuvem, com esse aumento contínuo ia perdendo cada vez mais desempenho por conta da latência, que em nossos testes, mostrou ter grande significância no tempo final por conta dos atrasos de *upload* e *download*.

Ao fim de nosso experimento, ficou nítido que no ambiente proposto a latência chegou a ser tão significativa quanto a quantidade de recursos computacionais em cenários com poucos usuários, porém, para uma quantidade muito grande a quantidade de recursos tem papel fundamental. Como já foi tratado anteriormente, era esperado que, com o aumento do número de requisições a *cloudlet* iria perder desempenho mais rápido e de maneira mais significativa, sendo ultrapassada pela nuvem em relação ao desempenho.

6 CONSIDERAÇÕES FINAIS

Após o fim do estudo e através de nossas análises, pode-se concluir que a utilização da computação em nuvem móvel para melhorar o desempenho de dispositivos menos robustos foi muito eficaz e deve ser tornar tendência de mercado, pois a mesma oferece grandes possibilidades de crescimento e um vasto campo de pesquisa. A computação em nuvem móvel, além de todos as outras características já citadas anteriormente, dispõe de inúmeros motivos aos quais está levando diversas organizações a adotá-la dentro de seu setor de TI.

Este trabalho apresentou dois modelos de utilização da técnica de *offloading* em um ambiente de computação em nuvem móvel. Nessa análise pudemos concluir que a latência de rede é ainda um dos principais vilões na utilização de uma nuvem remota, mesmo que ela tem muito mais recursos que um nuvem local, pois ela acaba elevando muito o seu tempo de resposta final por conta dos tempos de *upload* e de *download*. Dessa forma deve-se antes de adotar qualquer um dos modelos deve-se levar em consideração alguns fatores, como o a rede à ser utilizada pelos os dispositivos finais e a quantidade de recurso que a plataforma terá para utilizar. Assim, após um levantamento de todas a possíveis variáveis, pode-se tomar uma decisão que tenha um melhor custo benefício.

É importante ressaltar que tentamos aproximar ao máximo os nossos ambientes de testes à um ambiente real de uma rede de uso comum. Utilizamos uma rede com diversos usuários e aplicações, concorrendo o meio com nossa aplicação, a fim de obter resultados próximo ao de uma possível utilização por uma empresa ou instituição de um serviço semelhante. Utilizamos também dispositivos aleatórios, com o objetivo de mostrar essa heterogeneidade de um ambiente real e com isso podemos observar que a maioria dos dispositivos utilizados possuem recursos e configurações bem semelhantes, tanto em processamento, memória e armazenamento. Esse resultado de certa forma foi importante para nosso experimento, pois mostra que os dispositivos utilizados não foram um fator decisivo nos resultados, mas sim o uso de rede e a quantidade de usuários utilizando a plataforma simultaneamente.

Para finalizar, todos os códigos desenvolvidos e utilizados nesse trabalho estão disponíveis para download através de github[1]. A lista dos dispositivos utilizados em nosso experimentos podem ser observados no 3. Podemos observar em nossos experimentos que a maioria dos dispositivos utilizados possuem recursos e configurações bem semelhantes, tanto em processamento, memória e armazenamento. Esse resultado de certa forma foi importante

[1] https://github.com/CSRmos/TCC_Cleilson

para nosso experimento, de modo que os dispositivos utilizados não foi um fator decisivo nos resultados, mas sim o uso de rede e a quantidade de usuários utilizando a plataforma simultaneamente.

6.1 Recomendações de Uso

Com os resultados obtidos podemos observar que ao realizarmos um aumento da quantidade de usuários o desempenho da plataforma em ambos os cenário foi afetado drasticamente. Podemos observar também que a latência, assim como a baixa quantidade de recursos pode ser também um grande vilão nesse tipo de experimento. Outro fato importante que deve ser relatado é o fato de que, apesar de todos os testes e resultados obtidos, não é possível dizer qual dos ambientes é o melhor, de modo que cada um possui suas especificidades e se adéquam melhor a determinados cenários.

Por exemplo, um cenário que utilize uma aplicação que tenha uma grande quantidade de dados sendo transmitida do cliente para o servidor e ao fim outra grande quantidade de informações enviada de volta ao cliente mas que tenha apenas uma pequena quantidade de processamento vai demandar muito mais de uma baixa latência do que até mesmo um alto poder de processamento, sendo dessa forma mais viável a utilização de um *cloudlet*.

Todavia, uma aplicação que tenha pequenas quantidades de dados sendo trocados entre cliente para o servidor mas que tenha um processamento muito alto no servidor, vai necessitar de um poder de processamento bem maior para conseguir atender a demanda e manter um bom desempenho e a latência provavelmente não será tão significante, de modo que a quantidade de dados trafegando em rede seriam muito baixos.

Dessa forma, o modelo a ser utilizado sempre vai depender do ambiente em que a mesma será implantado, da aplicação e principalmente da utilização.

REFERÊNCIAS

BAHL, P.; HAN, R. Y.; LI, L. E.; SATYANARAYANAN, M. 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.

COSTA, P. B.; REGO, P. A.; COUTINHO, E. F.; TRINTA, F. A.; SOUZA, J. N. de. Uma análise do impacto da qualidade da internet móvel na utilização de cloudlets. **Conference: XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2014)**, 2014.

COSTA, P. B.; REGO, P. A.; ROCHA, L. S.; TRINTA, F. A.; SOUZA, J. N. de. Mpos: a multiplatform offloading system. In: ACM. **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. [S.l.], 2015. p. 577–584.

CUERVO, E.; BALASUBRAMANIAN, A.; CHO, D.-k.; WOLMAN, A.; SAROIU, S.; CHANDRA, R.; BAHL, P. Maui: making smartphones last longer with code offload. In: ACM. **Proceedings of the 8th international conference on Mobile systems, applications, and services**. [S.l.], 2010. p. 49–62.

FERNANDO, N.; LOKE, S. W.; RAHAYU, W. Mobile cloud computing: A survey. **Future Generation Computer Systems**, Elsevier, v. 29, n. 1, p. 84–106, 2013.

FOSTER, I.; ZHAO, Y.; RAICU, I.; LU, S. Cloud computing and grid computing 360-degree compared. **Grid Computing Environments Workshop**, p. 1–10, 2008.

JIANG, N.; SHAN, L. Application of mac-based vlans for mobile office in campus area network. In: IEEE. **Intelligent Information Hiding and Multimedia Signal Processing, 2008. IIHMSP'08 International Conference on**. [S.l.], 2008. p. 1029–1032.

KEMP, R.; PALMER, N.; KIELMANN, T.; BAL, H. Cuckoo: a computation offloading framework for smartphones. In: SPRINGER. **International Conference on Mobile Computing, Applications, and Services**. [S.l.], 2010. p. 59–79.

KOSTA, S.; AUCINAS, A.; HUI, P.; MORTIER, R.; ZHANG, X. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. **INFOCOM, 2012 Proceedings IEEE**, p. 945–953, 2012.

KUSAKABE, S.; LIN, H. H.; OMORI, Y.; ARAKI, K. Requirements development of energy management system for a unit in smart campus. In: IEEE. **Advanced Applied Informatics (IIAIAAI), 2014 IIAI 3rd International Conference on**. [S.l.], 2014. p. 405–410.

MARINELLI, E. E. **Hyrax: cloud computing on mobile devices using MapReduce**. [S.l.]: School of computer science. Pittsburgh, PA., 2009.

MEI, L.; CHAN, W.; TSE, T. A tale of clouds: paradigm comparisons and some thoughts on research issues. **Proceedings of the Asia-Pacific Services Computing Conference**, 2008.

REGO, P. A. L. Faircpu : Uma arquitetura para provisionamento de máquinas virtuais utilizando características de processamento. **Departamento de Computação, Universidade Federal do Ceará, Fortaleza**, 2012.

RIMAL, B. P.; CHOI, E.; LUMB, I. A taxonomy and survey of cloud computing systems. **INC, IMS and IDC**, p. 44–51, 2009.

SATYANARAYANAN, M.; BAHL, P.; CACERES, R.; DAVIES, N. The case for vm-based cloudlets in mobile computing. **IEEE pervasive Computing**, IEEE, v. 8, n. 4, p. 14–23, 2009.

SERGIO, L.; V., M. D.; G, G. D. Modelagem e simulação de offloading para computação móvel em nuvem. **7º SBCUP -Simpósio Brasileiro de Computação Ubíqua e Pervasiva**, 2015.

SHIRAZ, M.; GANI, A.; KHOKHAR, R. H.; BUYYA, R. A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. **IEEE Communications Surveys & Tutorials**, IEEE, v. 15, n. 3, p. 1294–1313, 2013.

SOUSA, F. R.; MOREIRA, L. O.; MACHADO, J. C. Computação em nuvem : Conceitos, tecnologias, aplicações e desafios. **II Escola Regional de Computação Ceará, Maranhão e Piauí**, 2009.

TYUKIN, A.; KRAMER, S.; WICKER, J. Scavenger–a framework for efficient evaluation of dynamic and modular algorithms. In: SPRINGER. **Joint European Conference on Machine Learning and Knowledge Discovery in Databases**. [S.l.], 2015. p. 325–328.

WELCH, D. J.; LATHROP, S. A survey of 802.11 a wireless security threats and security mechanisms. **United States Military Academy West Point**, 2003.

ZHANG, Z.; LI, S. A survey of computational offloading in mobile cloud computing. In: **4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)**. [S.l.: s.n.], 2016. p. 81–82.

APÊNDICE A – APÊNDICE A

Tabela 3 – Levantamento dos dispositivos

DISPOSITIVO	QUANTIDADE DE VEZES
ALCATEL A3	1
GALAXY A5 E SUAS VARIAÇÕES	5
GALAXY A7 E SUAS VARIAÇÕES	3
GALAXY J3 E SUA VARIAÇÕES	7
GALAXY J5 E SUAS VARIAÇÕES	5
GALAXY J7 E SUAS VARIAÇÕES	4
GALAXY S5 E SUAS VARIAÇÕES	6
GALAXY S6 E SUAS VARIAÇÕES	2
GALAXY S7 E SUAS VARIAÇÕES	1
LENOVO VIBE K5	3
LENOVO VIBE K6	2
LG G3	4
LG K10	5
LG L90	2
MOTO G E SUAS VARIAÇÕES	1
MOTO G 2 E SUAS VARIAÇÕES	1
MOTO G 3 E SUAS VARIAÇÕES	2
MOTO G 4 E SUAS VARIAÇÕES	4
MOTO G 5 E SUAS VARIAÇÕES	12
MOTO X E SUAS VARIAÇÕES	2
MOTO Z2	1
QUANTUN GO	1
SONY XPERIA E SUAS VARIAÇÕES	4
ZENFONE 5	1
ZENFONE 2	1
ZENFONE 3 E SUAS VARIAÇÕES	3
OUTROS NÃO IDENTIFICADOS	18