Avaliação de Desempenho da Técnica de *Offloading* Computacional em Nuvens Móveis

Warley Junior, Adriano Henrique, Kelvin Lopes

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 15.064 – 91.501-970 –Recife – PE – Brasil

{wmvj,ahmf,kld}@cin.ufpe.br

Abstract. *Despite the advances in hardware for mobile devices, it is well-known that they are still limited with regard to processing, storage and energy resources as compared to desktops and servers. The Mobile Cloud Computing (MCC) paradigm provides an opportunity to extend the computational and energy resources of mobile devices through the use of offloading techniques. However, the MCC offloading does not always benefits energy consumption, but may depend on the wireless access technology. This paper presents a performance evaluation of computational offloading technique in a mobile cloud environment. The study highlights the impact of MCC use via Wi-Fi networks (Wireless Fidelity) and 4G LTE (Long Term Evolution) on battery lifetime and the mobile device processor occupancy time by using computational offloading.*

Resumo. *Apesar dos avanços de hardware nos dispositivos móveis, é sabido que estes ainda são limitados em termos de processamento, armazenamento e energia quando comparados aos desktops e servidores. O paradigma de Computação em Nuvem Móvel (MCC – Mobile Cloud Computing) permite estender os recursos computacionais e energéticos de dispositivos móveis através da utilização das técnicas de offloading. Porém, o offloading nem sempre trás ganhos de energia, uma vez que isso pode depender da tecnologia de acesso sem fio empregada. Este artigo apresenta uma avaliação de desempenho da técnica de offloading computacional em um ambiente de nuvem móvel. O estudo destaca o impacto na utilização da MCC via redes Wi-Fi (Wireless Fidelity) e 4G LTE (Long Term Evolution) na autonomia da bateria e no tempo de ocupação do processador do dispositivo móvel, ao utilizar offloading computacional.*

1. Introdução

Nos últimos anos, tem-se assistido a um enorme crescimento do mercado de telefonia móvel. De acordo com o relatório da *Portio Research*[1], o número de assinantes de telefonia móvel em todo o mundo deverá crescer a uma taxa de 7,3 % ao ano e atingirá aproximadamente 8,5 bilhões até o final de 2016. Outros dados afirmam que o mercado mundial de aplicações móveis deverá atingir 63,5 bilhões de dólares com mais de 200 bilhões de *downloads* por ano até 2017. Esta informação indica que os dispositivos móveis estão se tornando rapidamente uma plataforma computacional dominante (Ha et al. 2013).

[1] http://www.portioresearch.com

Como a capacidade dos dispositivos móveis cresce a cada ano (potência da CPU, memória, bateria e conectividade), usuários tendem a utilizá-los na execução de diversas atividades, que antes eram consideradas incomuns. Essas atividades envolvem aplicações tais como, *streaming* de vídeo, jogos on-line, *e-commerce*, reconhecimento de face, realidade aumentada, aprendizagem de máquina e redes sociais on-line. Estas são apenas algumas das várias aplicações móveis que requerem altos níveis de capacidade de processamento que, por sua vez, exigem recursos computacionais poderosos e largura de banda adequada.

Em contrapartida, a maioria dos dispositivos móveis disponíveis no mercado, possuem ainda limitações computacionais e de conectividade, por causa de diversos fatores relacionados principalmente com suas dimensões físicas e restrições técnicas (Bahtovski and Gusev 2014).

Neste âmbito, uma infraestrutura adequada para permitir o armazenamento e processamento de dados de aplicações móveis em provedores de recursos, e não no próprio dispositivo móvel, denomina-se Computação em Nuvem Móvel ou MCC (*Mobile Cloud Computing*). Ao explorar a capacidade de computação e armazenamento de uma nuvem móvel, aplicações que fazem o uso intensivo da computação, podem ser executadas por dispositivos móveis de baixo custo (Gao, Gruhn, and Roussos 2013).

A MCC é um paradigma que está ainda em fase de consolidação, tanto na academia quanto na indústria. É definida pela disponibilidade de serviços de computação em nuvem em um ecossistema móvel. A MCC permite que a computação, armazenamento de dados e processamento de informações em massa, possam ser transferidos para servidores de nuvem para melhorar a confiabilidade e disponbilidade de serviços, enquanto minimiza a energia e os requisitos computacionais em dispositivos móveis (Li et al. 2015).

No contexto de MCC, (Satyanarayanan et al. 2009) introduziu o conceito de uma pequena nuvem que fica localizada próxima dos usuários móveis, para que estes possam rapidamente instanciar máquinas virtuais personalizadas que permitam executar o aplicativo requerido em um dispositivo móvel. Esta pequena nuvem é denominada cloudlet. A cloudlet é uma simples infraestrutura de nuvem, onde dispositivos móveis fazem *offloading* de sua carga de trabalho para um servidor (ou desktop, ou cluster de computadores multi-core) conectado a uma rede local sem fio de baixo custo que mantém conectividade com nuvens remotas. A técnica de *offloading* é uma operação que visa migrar o processamento e os dados de um dispositivo móvel para computadores com mais recursos, com o intuito de aumentar o desempenho da aplicação e economizar energia do dispositivo móvel (Gani et al. 2014).

Com base nestes conceitos, (Bahtovski and Gusev 2014) fomenta uma discussão sobre o problema de escolher se é melhor processar uma aplicação no próprio dispositivo móvel, ou na cloudlet via rede Wi-Fi, ou na nuvem pública via rede 4G. Esta decisão é fruto de muita investigação por considerar vários fatores, tais como o fato de saber que a cada ano o hardware móvel está cada vez mais robusto e também que aplicações móveis estão utilizando algoritmos mais complexos e informações em massa.

Baseado nessa discussão devemos responder às seguintes perguntas: Realizar *offloading* computacional para uma cloudlet é sempre benéfico? Quando uma nuvem pública é a melhor opção? Quando o *offloading* computacional pode ser utilizado para reduzir o consumo de energia e o tempo de processamento?

Visando responder estas questões, este trabalho apresenta a avaliação de desempenho da técnica de *offloading* em diferentes ambientes de execução, tais como em uma cloudlet, nuvem pública e no próprio dispositivo móvel. O artigo destaca a avaliação do impacto de redes Wi-Fi (*Wireless Fidelity*) e 4G LTE (*Long Term Evolution*), na autonomia da bateria e no tempo de ocupação do processador do dispositivo móvel, ao utilizar *offloading* computacional.

O trabalho está organizado da seguinte maneira: a Seção 2 apresenta os conceitos de *offloading*, do processo de decisão e que entidades impactam na decisão de quando e onde fazer *offloading*. Na Seção 3, são apresentados os trabalhos relacionados; a Seção 4 apresenta os experimentos que foram realizados para avaliar o *offloading* computacional em diferentes ambientes e seu impacto em relação ao consumo de energia e desempenho de processamento nos dispositivos móveis. Por fim, a conclusão e trabalhos futuros são apresentados na Seção 5.

2. O processo de *Offloading* Computacional

O conceito de *offloading* não é algo novo. Desde a década de 70, conceitos similares, tais como balanceamento de carga em sistemas distribuídos, já tinham sido propostos. O *offloading* computacional evoluiu do paradigma cliente-servidor para sistemas móveis e computação em nuvem. O paradigma cliente-servidor ainda é parte de uma nuvem. No entanto, a computação em nuvem implica em negócios, armazenamento de dados, e outros recursos que estão hospedados remotamente (Ma et al. 2012).

O conceito de *offloading* ainda existe hoje de várias formas. O trabalho de (Ha et al. 2013) apresentou o conceito de "*cyber foraging*" para dispositivos móveis e descreve-o como um mecanismo para aumentar a capacidade computacional e de armazenamento dos dispositivos móveis através da distribuição de tarefas.

Segundo (Enzai and Tang 2014), a técnica de *offloading* computacional é uma operação que visa migrar o processamento e dados de um dispositivo móvel para computadores com mais recursos, com o intuito de aumentar o desempenho e economia de energia do dispositivo móvel.

Uma aplicação móvel, essencialmente necessita de um algoritmo de decisão de *offloading* para poder escolher de forma inteligente sobre o "se" e "onde" fazer *offloading* dos dados e dos códigos, considerando entidades contextuais que estão em torno do usuário. A Figura 1 apresenta o fluxo básico das atividades do processo de *offloading* computacional.

Portanto, conforme (Li et al. 2015), um aplicativo que roda em nuvem móvel, deve passar pelas seguintes etapas antes de realizar o *offloading* computacional para um recurso de nuvem.

1ª etapa: O fluxo inicia com a execução de um aplicativo seguido pela descoberta e requisição de serviços e recursos computacionais via rede sem fio. Este recurso pode ser uma nuvem pública ou uma cloudlet. Caso tenha poder de processamento e armazenamento suficientes para atender os requisitos da aplicação, então o dispositivo móvel recebe uma resposta favorável;

2ª etapa: Neste momento cabe ao dispositivo móvel coletar informações contextuais, tais como o estado atual de consumo de energia, o tempo de processamento

da CPU e taxa de transferência do enlace sem fio. Após coletados, os mesmos devem alimentar a máquina de decisão de *offloading* para que haja maior acurácia;

3ª etapa: Neste momento é decidido se realizar *offloading* é favorável ou não, o qual, depende do objetivo do usuário, do estado atual do hardware móvel e da aplicação em execução. Se for favorável, a atividade de particionamento da aplicação é executada. Caso contrário, a aplicação é executada no próprio dispositivo móvel;

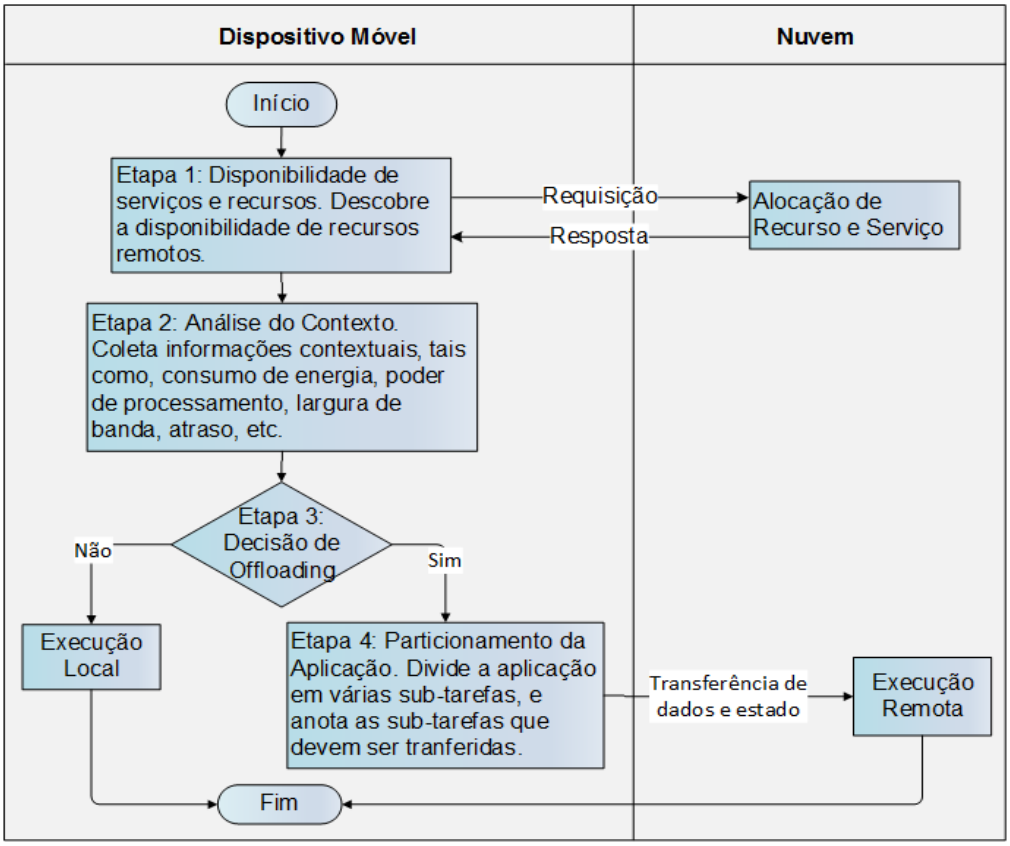

Figura 1 - O processo de *offloading* computacional. Fonte: (Li et al. 2015)

4ª etapa: Com a decisão de realizar *offloading*, a aplicação deve ser particionada em tempo de execução, de modo que suas sub-tarefas, previamente anotadas por um programador, sejam transferidas para um recurso de nuvem anteriomente requisitado. Este recurso deve processar todas as sub-tarefas e seus respectivos dados para em seguida devolver os resultados para a aplicação que roda no dispositivo móvel.

A decisão de realizar *offloading* ou não, é um processo extremamente complexo e é afetado por diferentes entidades, tais como, o usuário, a conectividade com a rede, a aplicação, e o serviço de nuvem.

3. Trabalhos Relacionados

Existem na literatura vários trabalhos que focam no desenvolvimento de sistemas de *offloading* computacional com o objetivo de reduzir o consumo de bateria e aumentar o desempenho das aplicações. Cada uma destas pesquisas será abordada a seguir.

O trabalho apresentado por (Costa et al. 2014) visa realizar uma avaliação da qualidade da Internet móvel em três capitais do Brasil, e uma prova de conceitos do uso de cloudlets comparado à nuvem pública para a realização de *offloading* computacional.

Os autores desenvolveram duas aplicações de *benchmark*, uma para medir a qualidade da Internet móvel e outra para medir o desempenho da aplicação quando se faz *offloading* para uma cloudlet e nuvem pública via rede Wi-Fi e 4G. Diferente desta proposta, o presente trabalho avalia no dispositivo, o tempo de ocupação de recursos computacionais e consumo de energia quando faz ou não *offloading*.

Alguns trabalhos projetaram algoritmos de decisão de *offloading* mais sensíveis ao comportamento da rede e ao consumo energético do dispositivo. Os autores de (Lin et al. 2013), apresentaram o algoritmo CADA (*Context-Aware Decision Algorithm*), responsável pela coleta de informações contextuais, tais como: localização, consumo de energia, RSS (*Received Signal Strength*) e vazão da rede. Estas informações são utilizadas para auxiliar na tomada de decisão de *offloading*. Para avaliação da proposta, os autores desenvolveram quatro aplicações de *benchmark* com diferentes demandas de comunicação e computação; em seguida comparam a execução local destas com execuções na nuvem, via Wi-Fi e 3G. Em (Magurawalage et al. 2014), foi projetado um algoritmo que leva em consideração o consumo de energia necessária para a execução de uma tarefa e o estado atual da rede sem fio. Estas informações são utilizadas por um tomador de decisão para fazer *offloading* ou não. O autor comparou a execução de sua proposta localmente no dispositivo móvel, quando faz-se *offloading* para uma nuvem via rede 3G e cloudlet via rede Wi-Fi.

O trabalho de (Barbera et al. 2013), apresenta uma arquitetura que permite com que cada dispositivo móvel tenha seu respectivo clone na nuvem. Assim os autores avaliaram a viabilidade e os custos da implementação de duas técnicas de clonagem, a *off-clone*, cujo o propósito é suportar *offloading* computacional, e o *back-clone*, que visa a restauração de dados e aplicativos do usuário quando requisitado.

Em (Flores and Srirama 2013), os autores implementaram uma máquina de decisão que utiliza lógica fuzzy para o *offloading* de código. Esta leva em consideração variáveis tanto do dispositivo móvel quanto da nuvem. Para sua avaliação, utilizou-se o serviço GCM (*Google Cloud Messaging*), para enviar mensagens assíncronas de um servidor GCM para o dispositivo móvel.

Grande parte das propostas visam aumentar o desempenho da aplicação e economizar energia do dispositivo, pois são pontos críticos para qualquer cenário da MCC. Alguns dos trabalhos apresentados, ou fazem uso de redes Wi-Fi com cloudlets e nuvem pública, ou utilizam redes 3G e 4G, para avaliar o impacto do *offloading* em nuvens móveis. Vários aplicativos de *benchmark* foram desenvolvidos com o objetivo de avaliar esta técnica em diferentes perspectivas. Baseado nestes dados, podemos concluir que nenhuma das propostas contempla a avaliação do impacto das redes Wi-Fi e 4G no consumo de energia e no tempo de ocupação do processador do dispositivo, quando faz-se *offloading* para cloudlets e nuvem pública.

4. Experimentos de Avaliação

Nesta seção, são apresentados os experimentos realizados para avaliar o consumo de recursos computacionais e o desempenho da aplicação no dispositivo, e quando faz-se *offloading* para uma cloudlet e nuvem pública. É avaliado também o quanto de bateria é consumido quando se faz ou não *offloading*. Deste modo a Subseção 4.1 descreve a metodologia utilizada para a experimentação; as Subseções 4.2, 4.3 e 4.4 apresentam os

experimentos e seus respectivos resultados; e a Subseção 4.4 discute os resultados obtidos.

4.1. Metodologia

Para a realização deste trabalho, duas aplicações para a plataforma Android foram utilizadas e estão disponíveis para fins acadêmicos[2].

BenchImage: Esta aplicação de processamento de imagens, aplica efeitos em fotos de variados tamanhos. O aplicativo tem como opções, executar totalmente no dispositivo móvel ou parcialmente, fazendo *offloading* da tarefa de processar o efeito para uma cloudlet ou para uma nuvem pública. Se o usuário optar por executar fora do dispositivo, a aplicação transfere a foto para o servidor remoto (uma máquina virtual na nuvem ou cloudlet) e a recebe após o efeito ser aplicado.

BatteryMonitor: Esta aplicação de monitoramento do consumo de bateria, permite o usuário monitorar a porcentagem e o fluxo de bateria em mAh (miliAmpere-hora), configurar o tempo de coleta dos dados e gerar estatísticas a partir de seu banco de dados. É uma aplicação que roda em *background* e portanto pode-se executar outras tarefas sem qualquer intervenção em seu funcionamento.

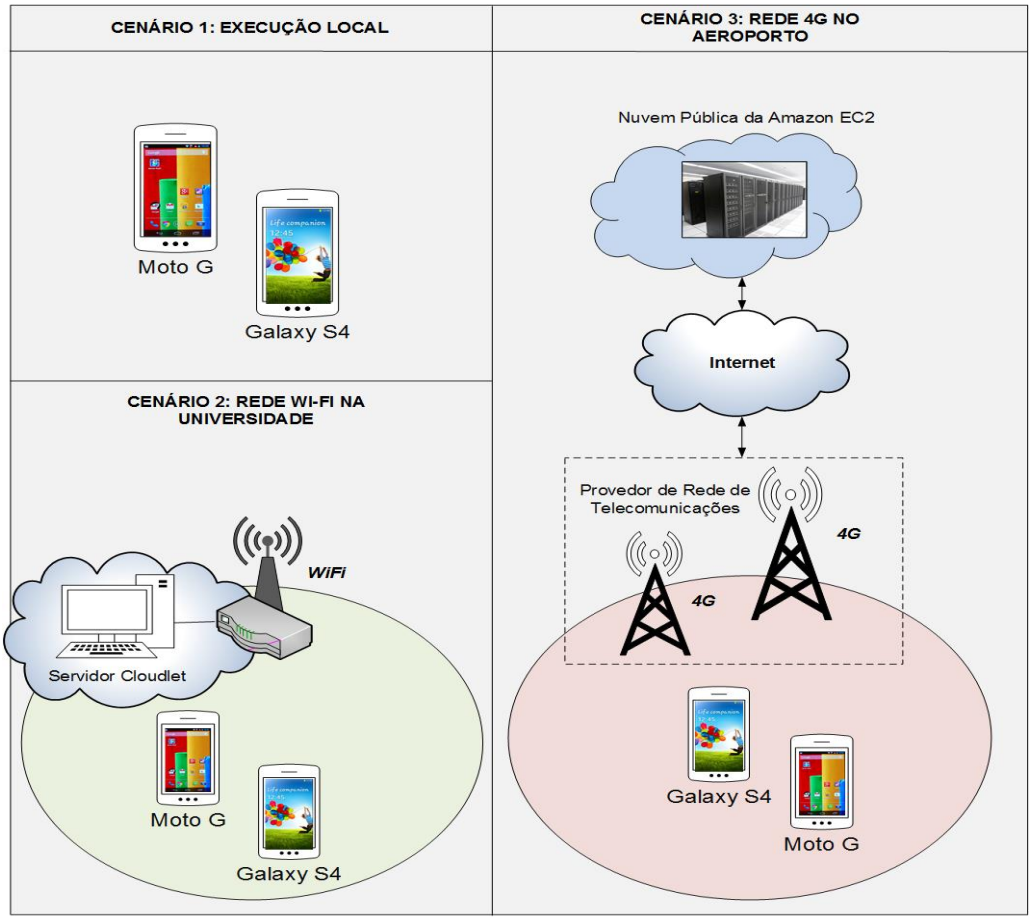

Figura 2 – Cenários dos ambientes de execução dos experimentos.

[2] A aplicação *BenchImage* desenvolvida por (Costa et al. 2014), está disponível em http://github.com/ufc-great/apps. A aplicação *BatteryMonitor*, desenvolvida pelo autor, está disponível em https://github.com/adrianohenrique/ProjetoStatusBateria

Conforme ilustrado na Figura 2, os experimentos foram realizados em três cenários distintos.

No cenário 1, o aplicativo *BenchImage* foi utilizado para processar os efeitos nas fotos no próprio dispositivo móvel. Vale resaltar que em todos os cenários, dois *smartphone*s de modelos diferentes foram utilizados para aplicar o efeito requisitado pela aplicação.

No cenário 2, o mesmo aplicativo é utilizado para delegar a uma cloudlet a tarefa de processar os efeitos nas fotos enviadas via rede Wi-Fi. Após o processamento, a instância da aplicação na cloudlet, devolve o resultado para o dispositivo móvel. Esta infraestrutura localiza-se em um laboratório de pesquisa no Centro de Informática da UFPE.

No cenário 3, o aplicativo envia fotos via rede 4G para instâncias de máquinas virtuais (VM – *Virtual Machines*) da Amazon EC2. Para que houvesse a viabilidade de utilizar uma rede 4G, foi necessário entrar em uma área de cobertura com qualidade de sinal aceitável, para isto foi escolhido o Aeroporto da cidade. Após o processamento do efeito, a máquina virtual deve retornar o resultado para o respectivo aplicativo no *smartphone*.

Em todos os cenários, o aplicativo *BatteryMonitor*, monitora o consumo de bateria do dispositivo móvel, enquanto o mesmo realiza o processamento, com ou sem *offloading* computacional.

A Tabela 1 detalha as configurações de hardware e software do experimento. Um destaque para a necessidade da instalação e configuração de uma cloudlet, pois necessita de um servidor de baixo custo atuando como uma nuvem privada, que é gerenciada pelo software de infraestrutura de nuvem, OpenStack[3] e, em cima deste, uma VM que roda uma instância do aplicativo *BenchImage*, para atender às solicitações do mesmo.

Tabela 1 - Configuração de hardware e software nos ambientes de execução

Ambiente de Execução	Hadware Utilizado	Software Utilizado
Local	Samsung Galaxy S4 GT-I9505, Processador 4 Quad-Core, 1.9 GHz; Memória 2GB; Capacidade da Bateria 2600 mAh	Android 4.2, Aplicativo BenchImage, Aplicativo BatteryMonitor
	Motorola Moto G com 4G, Processador 4 Quad-Core, 1.2 GHz; Memória 1GB; Capacidade da Bateria 2070 mAh	Android 4.4, Aplicativo BenchImage, Aplicativo BatteryMonitor
Cloudlet	Servidor Processador Intel(R) Xeon(R) CPU E5645, 4 Quad-Core, 2.40GHz, Cache de 12 MB e Memória RAM de 8GB Tipo de instância: c1.medium, 2 VCPUs, 5 ECUs e 2 GB de memória	Ubuntu Desktop 12.04 LTS 64 bits, OpenStack-Grizzly, Instância BenchImage
	Roteador Wireless Linksys WRT54G	Firmware LinkSys Version: v4.30.16
Nuvem Pública	Nuvem Pública da Amazon EC2, Tipo de instância: c1.medium, 2 VCPUs, 5 ECUs e 2 GB de memória	Ubuntu Desktop 12.04 LTS 64-bits Instância BenchImage

[3] OpenStack - http://www.openstack.org/software/

O *offloading* na cloudlet, foi realizado utilizando uma rede Wi-Fi dedicada (ponto de acesso LinkSys WRT54G) com capacidade de transferência de 54 Mbps.

No caso da nuvem pública, uma VM com configurações mínimas de um servidor foi contratada junto ao serviço da Amazon EC2. Esta VM roda também a instância do aplicativo *BenchImage*.

4.2. Experimento – Medição do Tempo de Execução do Aplicativo

Neste experimento, a aplicação de processamento de imagens utilizada foi executada 30 vezes para cada tamanho de imagem (0,3 MP, 1 MP, 4 MP e 8 MP), aplicando-se o efeito *Cartoonizer* sobre uma imagem que ilustra uma Cidade, nos três ambientes de execução. A Tabela 2, mostra com mais detalhes os parâmetros utilizados.

A aplicação foi executada totalmente no dispositivo móvel e fazendo *offloading* na cloudlet e na nuvem pública. Vale lembrar que todo o procedimento mencionado, foi realizado em dois *smartphone*s distintos, um aparelho Moto G e outro Galaxy S4, para que houvesse uma melhor margem de confiança na conclusão a ser retirada do experimento. Estes dispositivos foram configurados para estarem com condições semelhantes de processamento, memória e armazenamento.

Tabela 2 - Parâmetros do Experimento.

Ambiente de Execução	Filtro	Tamanho	Imagem	Número de Execuções
Local	Cartoonizer	0,3 MP 1,0 MP 4,0 MP 8,0 MP	Cidade	30 Vezes
Cloudlet				
Nuvem Pública				

A Figura 3 apresenta os resultados para o tempo total de execução do aplicativo (sua média), para cada tamanho de foto e ambiente de execução. Para os casos em que foram realizados *offloading*, o cálculo do tempo total considera o tempo de *upload* e *download* da imagem, seu processamento na VM e a leitura da mesma no dispositivo. Já para o processamento local, o cálculo considera somente o tempo em que o dispositivo acessa a imagem, processa o efeito e apresenta o resultado na tela.

Podemos observar na Figura 3, referente ao *smartphone* Moto G (figura da esquerda), que para todos os tamanhos de imagem, a execução local foi a mais demorada. Ou seja, para este tipo de aplicação, fazer *offloading* em todos os casos (cloudlet ou nuvem pública) é a melhor opção, mesmo para a imagem de 0,3 MP.

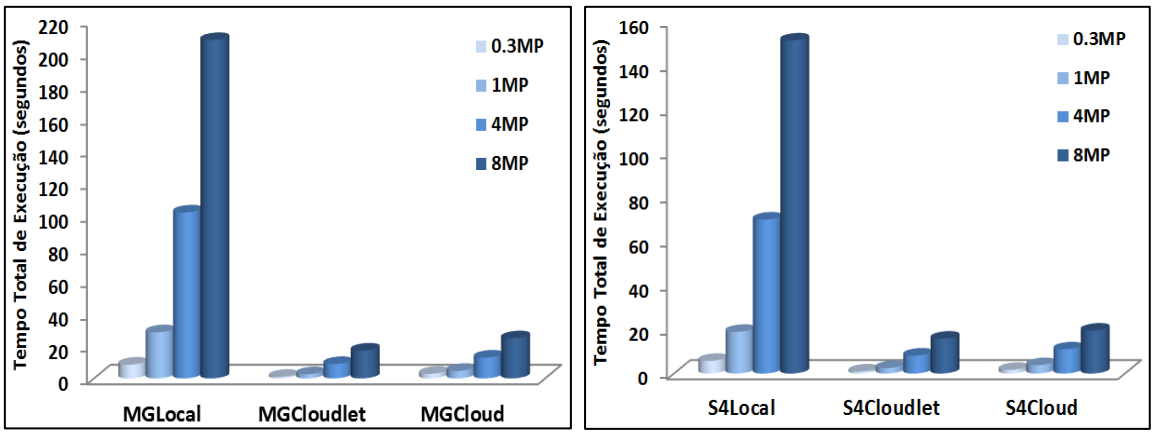

Figura 3 - Tempo total de execução do aplicativo no moto G e galaxy S4.

A mesma conclusão vale para o resultado referente ao Galaxy S4 (figura da direita). Com o detalhe de que por se tratar de um *smartphone* com melhor capacidade de processamento, o mesmo concluiu a tarefa localmente em menor tempo (média de 151,22 segundos para imagem de 8 MP) comparado ao *smartphone* da figura à esquerda.

Para ambos *smartphone*s, a melhor opção de *offloading* é a cloudlet, por apresentar menor tempo de processamento do efeito em todos os tamanhos de fotos. Mesmo para a menor foto, de 0,3 MP, a cloudlet é 2,97 vezes mais rápida que a nuvem pública, para o Moto G; e 1,78 vezes mais rápida que a nuvem pública, para o Galaxy S4. Este efeito é conquistado devido à baixa latência e a alta largura de banda que a rede Wi-Fi dispõe, além do tráfego pela rede ser de um único salto, já que o usuário e a cloudlet estão na última milha da rede.

A Tabela 3 apresenta o tempo de execução do aplicativo *BenchImage* no dispositivo (média e intervalo de confiança), para cada tamanho de foto e ambiente de execução. Neste caso, o tempo de transferência das fotos e o processamento remoto não foram considerados, pois o foco é avaliar somente o processamento local.

Tabela 3 - Tempo de execução no dispositivo (segundos)

	Ambiente de Execução	**0,3MP**	**1MP**	**4MP**	**8MP**
Moto G	Local	8,33 ± 0,04	27,98 ± 0,06	100,72 ± 0,14	205,66 ± 1,25
	Cloudlet	0,53 ± 0,005	1,76 ± 0,01	6,47 ± 0,16	12,39 ± 0,08
	Nuvem	0,52 ± 0,006	1,75 ± 0,02	6,25 ± 0,03	12,41 ± 0,05
Galaxy S4	Local	5,70 ± 0,15	18,80 ± 0,10	69,43 ± 0,60	150,25 ± 2,14
	Cloudlet	0,53 ± 0,01	1,82 ± 0,06	6,32 ± 0,07	12,36 ±0,04
	Nuvem	0,52 ± 0,009	1,76 ± 0,01	6,37 ± 0,07	12,45 ± 0,08

A computação do efeito local quando se utiliza nuvem pública é aproximadamente 11 vezes mais rápida que o *smartphone* mais potente, Galaxy S4 (que foi de 5,70 segundos), para imagem de 0,3 MP; e aproximadamente 1,05 vezes mais rápida do que na cloudlet (que foi de 1,82 segundos), para imagem de 1 MP. Para as imagens de 4 MP e 8 MP, o processamento via cloudlet demonstra ser mais veloz e, portanto, a melhor opção.

Os resultados mostram que há mais ganho de desempenho para a aplicação, quando se faz *offloading* de pequenas imagens para uma nuvem pública, e grandes imagens para uma cloudlet. No caso das pequenas imagens, o resultado é justificado pelo fato de os protocolos de roteamento em infraestruturas de redes de telecomunicações serem mais eficientes quando tratam de pequenos pacotes de dados. Já para as grandes imagens, o uso de um enlace sem fio de alta vazão e com único salto, justifica a rápida taxa de transferência de grande quantidade de dados.

4.3. Experimento – Medição da Utilização e *Service Demand*

Para obter a medição da utilização da CPU, foi utilizado o conceito da *Utilization Law*, que pode ser descrito como a média da utilização de qualquer componente do sistema ser igual ao *throughput* do sistema multiplicado pelo tempo que cada processo consome no componente (Jain 1991).

A Figura 4 apresenta o resultado obtido ao multiplicar o tempo médio de serviço com o *throughput* do sistema, para cada ambiente de teste, utilizando os dois dispositivos móveis. Os resultados nos mostra que o Moto G e o Galaxy S4 passaram

81,52% e 95,51%, respectivamente, do seu tempo total do teste local utilizando a CPU, diferente dos testes realizados utilizando a nuvem e cloudlet, pois nesses, a CPU do dispositivo foi utilizada em aproximadamente 50% do tempo total. Esse tempo total se refere ao intervalo de tempo entre a solicitação e a resposta com a imagem já tratada e apresentada na tela do dispositivo.

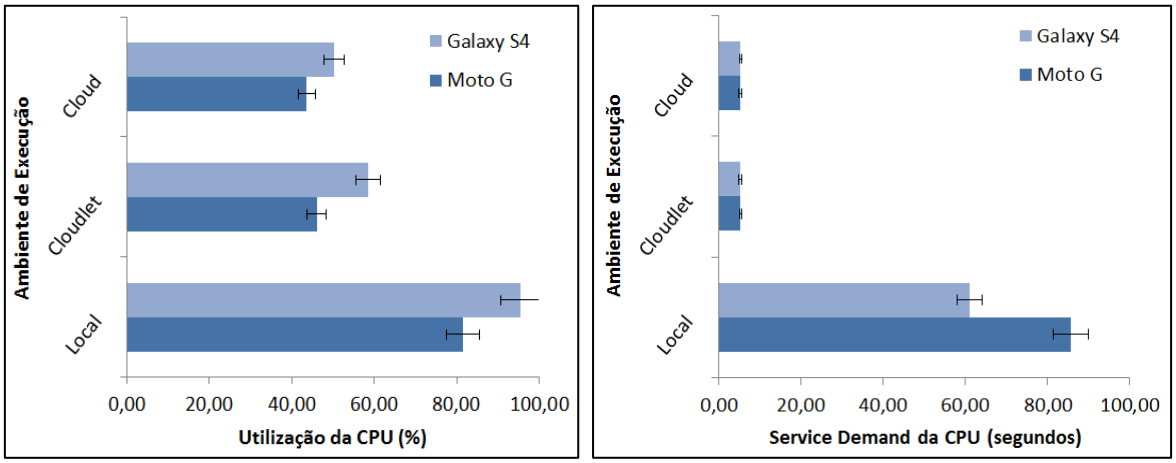

Figura 4 - Utilização da CPU do moto G e galaxy S4.

A lei *service demand* é utilizada como ferramenta de análise operacional de sistemas computacionais. Através do *service demand* é possível conhecermos o tempo médio total da utilização de um recurso computacional no dispositivo. O *service demand* representa o tempo médio total gasto por uma requisição ao longo de sua existência, levando em conta que a mesma solicitação pode utilizar vários recursos computacionais e o mesmo recurso varias vezes, entretanto, para uma solicitação, o seu *service demand* é a soma de todos os tempos de utilização do serviço durante todas as visitas a um recurso especifico (Jain 1991).

A Figura 4 ilustra os resultados do *service demand* da CPU durante os testes. O *service demand* foi utilizado como base para calcular o tempo de utilização da CPU em segundos. Observando os resultados podemos verificar que a CPU foi mais utilizada durante o teste local, dessa forma houve um maior consumo de energia por parte do processador. Enquanto que nos testes utilizando a nuvem e a cloudlet, a utilização da CPU foi baixa, pois basicamente o dispositivo utiliza a CPU apenas para enviar, receber e apresentar a imagem, deixando a cargo do servidor remoto o processamento da imagem.

4.4. Experimento – Medição do Consumo de Bateria

Durante os testes de aplicação de filtros nas imagens, o aplicativo *BatteryMonitor* foi utilizado para monitorar o consumo da bateria a cada 5 segundos. Os resultados desse monitoramento permitiu a criação dos gráficos de consumo da bateria (veja as Figuras 5 e 6) tanto para o Moto G quanto para o Galaxy S4.

Ambos os *smartphones* executaram o total de 120 vezes a aplicação do filtro, em 30 execuções repetidas para cada tamanho de imagem, ou seja, no intervalo de 0 a 30 a imagem de tamanho, 0,3 MP foi executada 30 vezes, e assim sucessivamente para as demais imagens.

Os resultados mostram a redução do consumo de energia da bateria quando a aplicação é processada na cloudlet ou na nuvem, em comparação com o processamento local. O Moto G registrou uma economia de bateria 12,25 vezes menor quando a aplicação é executada via cloudlet, comparada a utilização do recurso local. Similarmente, o Galaxy S4 quando executado via cloudlet registrou uma economia de bateria 16,67 vezes menor do que a execução local.

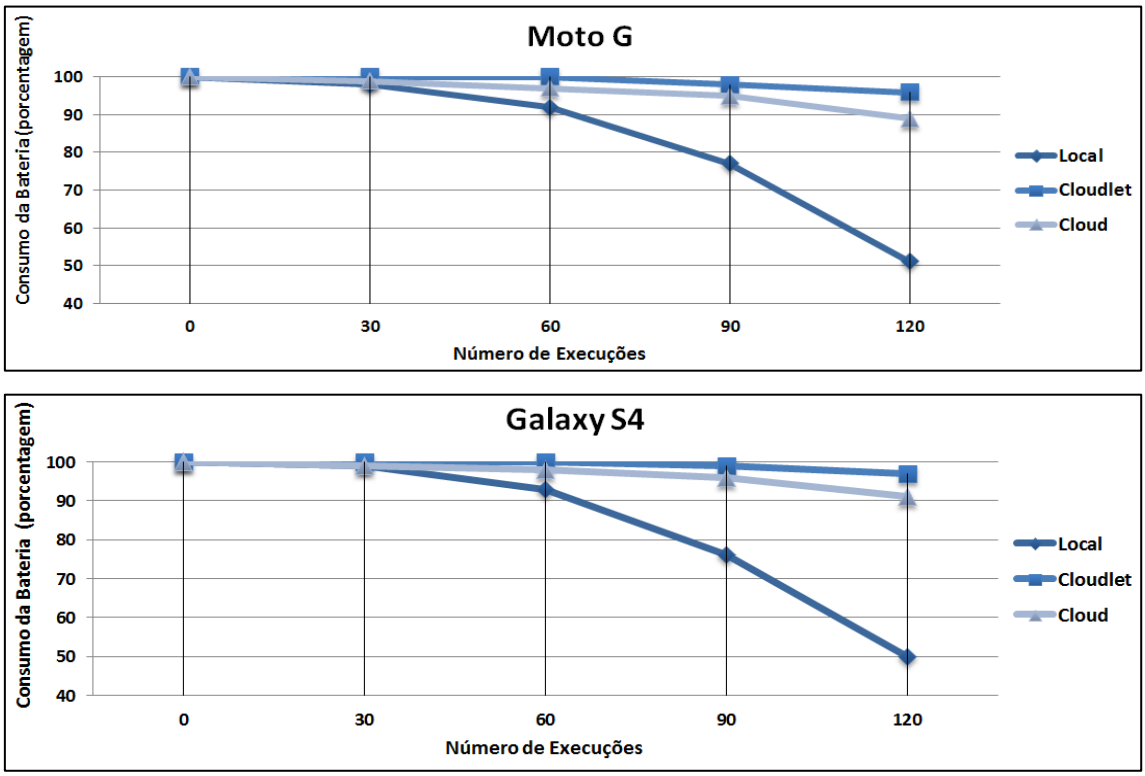

Figura 5 - Consumo de bateria do moto G e galaxy S4 para cada imagem.

Do total de 100% de bateria, utilizando o ambiente cloudlet, os *smartphones* Moto G e Galaxy S4, consumiram apenas 8% e 6% da bateria, respectivamente. Não muito distante ficou o consumo de bateria pelos dois *smartphone*s utilizando o ambiente em nuvem, que consumiram respectivamente 22% e 18%.

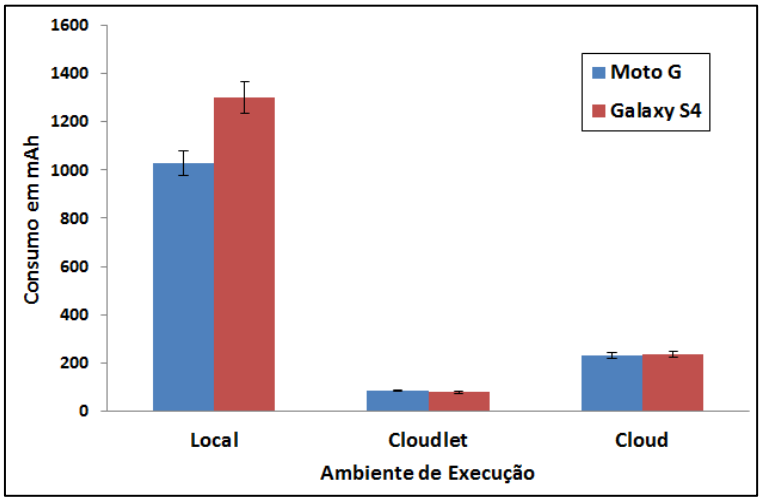

Figura 6 - Consumo de bateria do moto G e galaxy S4.

Em relação à média de consumo em mAh, o Moto G consumiu 1.029 mAh e o Galaxy S4, 1.300 mAh durante o teste local (veja a Figura 6), ou seja, o Moto G consumiu 4,45 vezes a mais ao ser executado localmente em relação à nuvem, da mesma forma que o Galaxy S4 consumiu 5,56 a mais quando executado localmente, também em relação a nuvem.

Com relação à Figura 5, o resultado é justificado pelo fato do dispositivo móvel transferir toda a carga de trabalho para uma cloudlet ou nuvem pública, deixando parte de seu poder computacional livre para execução de tarefas mais simples.

No caso da Figura 6, o Moto G consumiu menos energia que o Galaxy S4, por ter capacidade energética menor e processador com frequência de trabalho pequena.

Isto nos leva a acreditar que em um futuro próximo, teremos dispositivos móveis com hardware de baixíssimo custo, que fazem *offloading* de código e dados em larga escala para cloudlets e nuvens públicas com grande capacidade de processamento e armazenamento. Com a consolidação de infraestruturas de redes sem fio heterogêneas estes dispositivos poderão usufruir ao máximo da mobilidade e realizar *handover* transparente de *offloading* computacional.

4.5. Discussão dos Resultados

Os resultados apresentados provam os benefícios de usar a técnica de *offloading*, via cloudlet ou nuvem pública, uma vez que garante o melhor desempenho da aplicação e reduz o consumo de energia do dispositivo móvel.

Entende-se também que quando há necessidade de realizar *offloading* de uma carga de trabalho pequena, uma nuvem pública acessada via 4G se torna uma boa opção, uma vez, que apresenta largura de banda alta, latência mediana e mobilidade que se estende a milhares de quilômetros, que são características que pouco interferem na qualidade de serviço (QoS – *Quality of Service*) de aplicações leves ou que demandam pouco processamento de dados.

Em contrapartida, quando se faz *offloading* de uma carga trabalho muito grande, uma cloudlet se torna uma opção mais benéfica, uma vez que a largura de banda é alta, o roteamento é de um único salto e a latência é baixa, que são requisitos essenciais para aplicações complexas com requisitos estritos de QoS.

Os testes também comprovam a eficiência em utilizar a MCC quando o foco for a redução do consumo de energia. Os resultados colocam em destaque o ambiente cloudlet que demonstrou sua eficiência energética, mesmo em *smartphones* com hardwares diferentes e aplicações que fazem o uso de arquivos de tamanhos variados.

5. Conclusão e Trabalhos Futuros

Este artigo apresentou a avaliação de desempenho do dispositivo móvel quando faz-se *offloading* computacional para uma cloudlet e nuvem pública, via redes Wi-Fi e 4G.

O trabalho mostrou que a qualidade do enlace sem fio, a capacidade de processamento do dispositivo e a complexidade das tarefas da aplicação, são fatores que mais impactam na decisão de *offloading*. As tecnologias Wi-Fi e 4G se mostraram bastante promissoras como meios de acesso sem fio para a realização do *offloading* em futuros cenários da computação ubíqua.

Entre as contribuições deste trabalho estão a avaliação do tempo de processamento total e local, por meio de um aplicativo de processamento de imagem; o desenvolvimento de um aplicativo de monitoramento de consumo de bateria; e, por último, a medição do consumo de bateria do dispositivo quanto ao uso da técnica de *offloading*.

Como trabalhos futuros pretende-se desenvolver um *middleware* adaptativo de modo a permitir que o dispositivo móvel possa coletar e filtrar uma grande quantidade de informações contextuais e tomar decisões de *offloading* de acordo com as necessidades do usuário e as reais limitações do dispositivo.

Referências

Bahtovski, Aleksandar, and Marjan Gusev. 2014. "Cloudlet Challenges." *Procedia Engineering* 69: 704–11. http://linkinghub.elsevier.com/retrieve/pii/S1877705814002914 (June 13, 2014).

Barbera, Marco V., Sokol Kosta, Alessandro Mei, and Julinda Stefa. 2013. "To Offload or Not to Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing." *Proceedings - IEEE INFOCOM*: 1285–93.

Costa, Philipp B, Paulo A L Rego, Emanuel F Coutinho, and Fernando A M Trinta. 2014. "Uma Análise Do Impacto Da Qualidade Da Internet Móvel Na Utilização de Cloudlets." In *Simpósio Brasileiro de Redes de Computadores E Sistemas Distribuídos*, Florianópolis, 223–36. http://sbrc2014.ufsc.br/anais/files/trilha/ST06-1.pdf.

Enzai, Nur Idawati Md, and Maolin Tang. 2014. "A Taxonomy of Computation *Offloading* in Mobile Cloud Computing." *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*: 19–28. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6834942 (September 19, 2014).

Flores, Huber, and Satish Srirama. 2013. "Adaptive Code *Offloading* for Mobile Cloud Applications: Exploiting Fuzzy Sets and Evidence-Based Learning." *MCS '13*: 9–16. http://dl.acm.org/citation.cfm?id=2482984.

Gani, Abdullah et al. 2014. "A Review on Interworking and Mobility Techniques for Seamless Connectivity in Mobile Cloud Computing." *Journal of Network and Computer Applications* 43: 84–102. http://linkinghub.elsevier.com/retrieve/pii/S1084804514000927 (June 15, 2014).

Gao, Jerry, V. Gruhn, and G. Roussos. 2013. "Mobile Cloud Computing Research - Issues, Challenges and Needs." *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*: 442–53. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6525561 (December 14, 2014).

Ha, Kiryong et al. 2013. “Just-in-Time Provisioning for Cyber Foraging.” *Proceeding of the 11th annual international conference on Mobile systems, applications, and services - MobiSys ’13*: 153. http://dl.acm.org/citation.cfm?doid=2462456.2464451.

Jain, Raj. 1991. *Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling*. ed. Inc. Wiley Computer Publishing, John Wiley & Sons.

Li, Wenzhong, Yanchao Zhao, Sanglu Lu, and Daoxu Chen. 2015. “Mechanisms and Challenges on Mobility-Augmented Service Provisioning for Mobile Cloud Computing.” (March): 89–97.

Lin, Ting-yi, Ting-an Lin, Cheng-hsin Hsu, and Chung-ta King. 2013. “Context-Aware Decision Engine for Mobile Cloud *Offloading*.” *2013 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*: 111–16. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6533324.

Ma, Xiao, Yong Cui, Lian Wang, and Ivan Stojmenovic. 2012. “Energy Optimizations for Mobile Terminals via Computation *Offloading*.” : 236–41.

Magurawalage, Chathura, Kun Yang, Liang Hu, and Jianming Zhang. 2014. “Energy-Efficient and Network-Aware *Offloading* Algorithm for Mobile Cloud Computing.” *Computer Networks*. http://linkinghub.elsevier.com/retrieve/pii/S1389128614003193 (October 30, 2014).

Satyanarayanan, M., P. Bahl, R. Caceres, and Nigel Davies. 2009. “The Case for VM-Based Cloudlets in Mobile Computing.” *IEEE Pervasive Computing* 8(4): 14–23. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5280678 (August 4, 2014).