

See discussions, stats, and author profiles for this publication at: <http://www.researchgate.net/publication/281150386>

Modelagem e Simulação de Offloading para Computação Móvel em Nuvem

CONFERENCE PAPER · JULY 2015

3 AUTHORS, INCLUDING:



Danielo Gonçalves Gomes

Universidade Federal do Ceará

54 PUBLICATIONS 73 CITATIONS

SEE PROFILE

Modelagem e Simulação de Offloading para Computação Móvel em Nuvem

Luis Sérgio da Silva Júnior, Deborah V. Magalhães, Danielo G. Gomes

¹Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
Programa de Pós-Graduação em Engenharia de Teleinformática (PPGETI)
Universidade Federal do Ceará (UFC)
Campus do Pici – Fortaleza, CE – Brasil

{luissilva,deborah,dgomes}@great.ufc.br

Abstract. *Mobile Cloud Computing (MCC) uses cloud services to spread out the energy and computing resources of mobile devices. Despite these devices have grown in use, we observe a lack of models and simulation tools to study and analyse mobile devices resource limitation using cloud computing as solution. In this paper, we model a typical MCC architecture which focuses on an offloading perspective. Our model provides a development environment to validate offloading strategies without using a real cloud infrastructure. To validate our approach, we simulate the execution time of a task on a local (mobile device) and in a remote cloud. Results show that our models represent MCC scenarios with an maximum accuracy of 98.9%. Moreover, we verified the workload algorithm complexity impacts on the offloading decisions.*

Resumo. *Computação Móvel em Nuvem (Mobile Cloud Computing, MCC) utiliza os serviços de nuvens a fim de estender os recursos energéticos e computacionais de dispositivos móveis. Apesar da sua presença em nosso cotidiano, observamos uma carência de modelos e simuladores para análise de problemas relacionados à escassez de recursos dos dispositivos móveis tendo as nuvens computacionais como solução em perspectiva. Neste artigo, propomos a modelagem e simulação de uma arquitetura típica de MCC compreendendo diferentes abordagens para execução de tarefas computacionais em servidores remotos (offloading). A modelagem foi concebida com vistas à provisão de um ambiente controlado para avaliação e ajuste de políticas de offloading sem a necessidade da implantação física de uma infraestrutura computacional de larga escala. A validação da proposta foi conduzida através de simulações, a partir de experimentos com imagens entre 0.3 e 8 megapixels, e cujos resultados indicam uma acurácia de até 98.9%. Concluimos ainda que a ordem de complexidade algorítmica da carga é diretamente proporcional à probabilidade de offloading.*

1. Introdução

Computação móvel em nuvem (*Mobile Cloud Computing*, MCC) baseia-se nas características de computação em nuvem [Mell and Grance 2011] para otimizar e expandir os recursos dos dispositivos móveis. Apesar do poder computacional crescente de tais dispositivos, sua natureza móvel e compacta limita-os sob o ponto de vista da autonomia energética [Justino and Buyya 2014].

Além da limitação de recursos, existem outros desafios em sistemas computacionais móveis (e.g. qualidade da comunicação móvel, mobilidade, sensoriamento de ambientes) [Dinh et al. 2013]. Estratégias como *offloading*¹ e melhorias na largura de banda em função da mobilidade tentam mitigar essas dificuldades. O termo *offloading* refere-se ao uso de uma infraestrutura remota para execução de tarefas oriundas de um dispositivo com recursos limitados [Cuervo et al. 2010].

Apesar da vasta literatura sobre Computação Móvel, observamos uma carência de modelos e simuladores para estudo e análise dos problemas relacionados à escassez de hardware tendo as nuvens computacionais como solução em perspectiva [Ahmed 2014], [Dinh et al. 2013], [Qi and Gani 2012], [Satyanarayanan et al. 2009], [Li et al. 2013], [Cuervo et al. 2010], [Chun et al. 2011], [Kumar et al. 2013].

O objetivo geral deste artigo é modelar e simular um ambiente real de MCC. A simulação dos modelos propostos permite o ajuste, validação de políticas de gerenciamento de recursos e ideias de comportamentos de elementos de arquitetura sem a necessidade de uma infraestrutura computacional real em escala. Tendo em vista esse cenário, criamos modelos para entidades de uma arquitetura típica de MCC. Estes modelos englobam desde a representação de componentes físicos de um ambiente de MCC, tais como dispositivos móveis, enlaces e nuvens computacionais, até as tomadas de decisão para realização de *offloading* estático e dinâmico.

Para validação da proposta, dois experimentos foram projetados e executados. O primeiro objetivou a simulação do modelo de *offloading* estático e respectiva comparação com cenário real descrito em [Costa et al. 2014]; o segundo orientou-se a tomadas de decisão de *offloading* dinâmico baseadas em critérios estabelecidos por [Kumar et al. 2013].

As principais contribuições deste artigo são (i) uma modelagem original de entidades de uma arquitetura de MCC, com respectiva validação em simulação, (ii) uma implementação de duas estratégias de *offloading* (estático e dinâmico), (iii) detecção de associações de interferência entre complexidade algorítmica e procedimentos de *offloading*.

2. Trabalhos Relacionados

Vários trabalhos sobre Computação Móvel em Nuvem têm sido propostos nos últimos 5 anos. Alguns destes trabalhos são brevemente discutidos a seguir, os quais estão organizadas em propostas de arquiteturas, *frameworks* e modelagem analítica para MCC.

Em uma visão geral típica de MCC, os dispositivos móveis utilizam-se de nuvens computacionais como extensões de sua capacidade de processamento e armazenamento [Qi and Gani 2012], [Dinh et al. 2013] (vide Figura 1). Nela existem componentes e operações essenciais em um contexto MCC: (i) dispositivos móveis (ii) envio e recebimento de dados (iii) mecanismo de *offloading* (iv) nuvem computacional.

Satyanarayanan et al. 2009 apresentaram o conceito de *Cloudlet*, o qual pode ser entendido como uma nuvem computacional disponível através de uma rede local sem fio. O uso de *Cloudlets* aplica-se a ambientes de conectividade limitada ou inexistente

¹Outros termos equivalentes: *Augmented execution*, *cyber foraging* ou *remote execution*

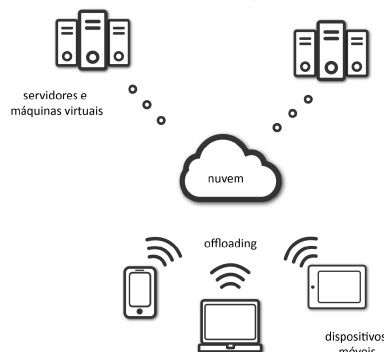


Figura 1. Visão geral MCC. Adaptada de [Qi and Gani 2012].

(e.g. ambientes fechados ou distantes de redes móveis) e no uso de aplicações sensíveis à latência.

Fernando et al. 2013 propuseram uma arquitetura que explora a utilização dos próprios dispositivos móveis disponíveis em rede para a alocação de tarefas. [Li et al. 2013] propuseram uma arquitetura com aspectos de mobilidade e cuja tomada de decisão para *offloading* leva em conta a mudança de localização de um dispositivo ao longo do tempo.

Alguns exemplos de *frameworks* foram propostos por [Cuervo et al. 2010] e [Chun et al. 2011]. Estes trabalhos utilizam ferramentas (e.g. MAUI, Clonecloud) que implementam características encontradas em MCC, tais como tomada de decisão de *offloading*. Essas ferramentas gerenciam recursos os quais vão desde a virtualização de uma sistema operacional em uma máquina virtual remota até a reescrita ou edição de um aplicativo móvel. Assim, para o(a) leitor(a) interessado(a) em reusar esses frameworks ou replicar os resultados apresentados nos respectivos artigos, é necessário conhecer ferramentas de manuseio de máquinas virtuais (Virtual Machines, VM) e/ou de edição de aplicativos para dispositivos móveis.

Kumar et al. 2013 propuseram uma modelagem de *offloading* dinâmico em que a largura de banda e a quantidade de computação das tarefas envolvidas são métricas para a tomada de decisão.

3. Proposta

Neste artigo propomos a modelagem e simulação de uma arquitetura típica de MCC (vide Figura 1). A modelagem proposta contempla os aspectos de *offloading* dos tipos estático e dinâmico [Kumar et al. 2013] [Dinh et al. 2013]. Neste sentido, propomos a criação de cinco modelos: (i) dispositivo móvel em contexto MCC (ii) tarefa passível de *offloading*, (iii) mecanismo de *offloading* estático e dinâmico, (iv) banda entre dispositivos e nuvem e, (v) infraestrutura de nuvem como um *pool* de servidores e VMs para execução remota de procedimentos. Esses modelos representam os componentes ilustrados na Figura 1. Sua associação com a modelagem proposta é ilustrada na Figura 2, na qual é possível distinguir quais classes representam quais modelos.

O modelo de dispositivos móveis é representado pelas classes *Device* e *DeviceTaskExecutor* (Figura 2(a)). *Device* contém informações sobre um dado dispositivo

como: modelo (atributo *DeviceHardware*) e identificador. *DeviceTaskExecutor* possui uma operação chamada de *executeTasksOnDevice* a qual é responsável por simular a execução de uma dada tarefa em um dispositivo móvel. Atrrelada a um *Device*, a estrutura *taskList* contém uma lista de tarefas.

As classes *Offloader*, *StaticOffloader* e *DynamicOffloader* (Figura 2(b)) possuem informações sobre a natureza de *offloading* a que uma tarefa é submetida. Essas classes implementam e representam os modelos de mecanismos de tomada de decisão de *offloading* estático e dinâmico. *DynamicOffloader* é uma especialização com implementação de *offloading* dinâmico em termos da proposta de [Kumar et al. 2013] e simplificada por [Costa et al. 2014]. Seu comportamento segue a Equação 1.

$$\frac{W}{P_m} > \frac{D_u}{V_u} + \frac{D_d}{V_d}, \quad (1)$$

em que W representa a quantidade de processamento realizado por uma dada tarefa, P_m representa o poder computacional de um dispositivo móvel, D_u é a quantidade de dados enviados à nuvem, D_d é quantidade de dados recebidos, V_u significa a taxa de *upload* e, de maneira análoga, V_d é a taxa de *download*. Outra especialização de

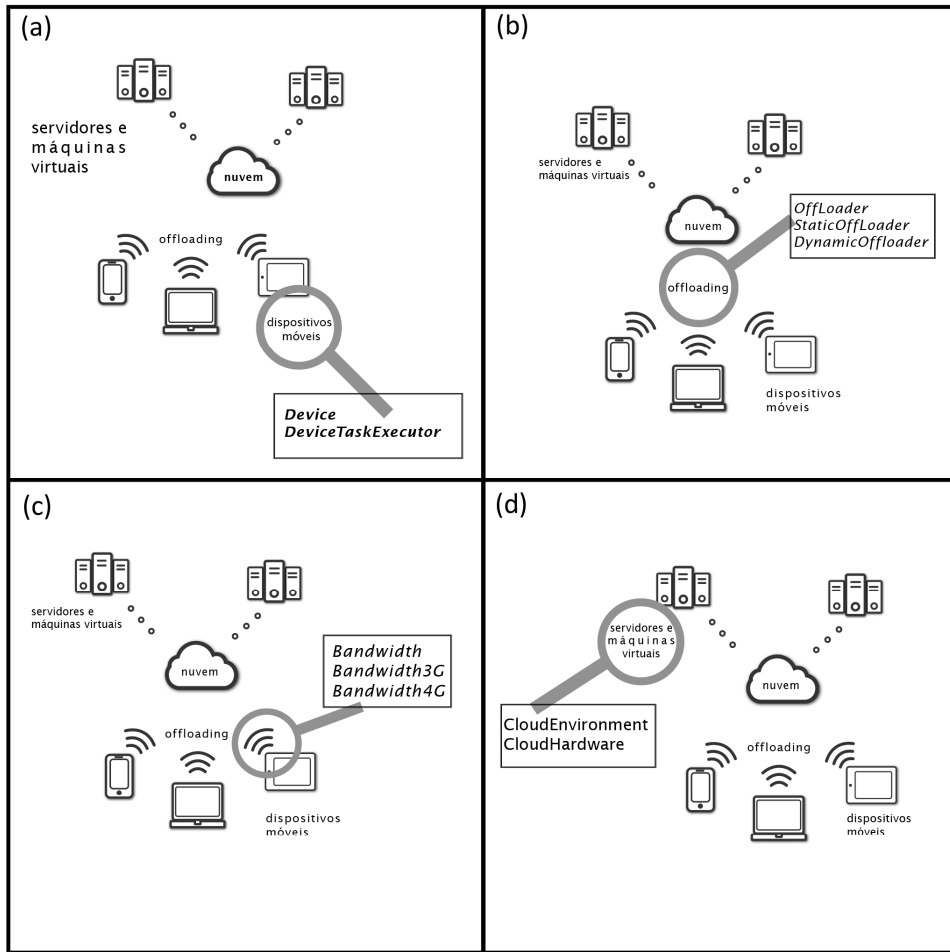


Figura 2. Associação da modelagem proposta entre as respectivas classes e (a) Dispositivos móveis, (b) Tomada de decisão de offloading, (c) Banda sem fio, (d) Infraestrutura de nuvem.

Offloader, a classe *StaticOffloader* implementa uma marcação na tarefa para informar que a mesma sofrerá *offloading*. Ambas as especializações da classe *Offloader* precisam ser referenciadas por um objeto do tipo *Task* e isso é necessário para o cálculo do tempo total de execução do processo de *offloading*, independente de sua especialização, o qual é expresso pela Equação 2:

$$T_{Total} = T_{Upload} + T_{Execucao} + T_{Download}, \quad (2)$$

na qual (T_{Upload} e $T_{Download}$) expressam o tempo de envio de informações necessárias ao *offloading* e recebimento de resultados oriundos da nuvem, respectivamente. $T_{Execucao}$ representa o tempo de uma tarefa executada local ou remotamente. Notar que na ausência de *offloading*, T_{Upload} e $T_{Download}$ são nulos e somente $T_{Execucao}$ representa o tempo para execução da tarefa no dispositivo móvel.

Os modelos de banda são representados pelas entidades *Bandwidth*, *Bandwidth3G* e *Bandwidth4G* (Figura 2(c)) as quais são baseadas nos dados oriundos de taxas de transmissão de dados 3G e 4G [Costa et al. 2014]. Ainda, a classe *Device* possui um atributo do tipo *Bandwidth* (*networkData*) e, com sua inicialização, o dispositivo consegue calcular o tempo que uma tarefa leva ao ser realizado seu *download* e *upload*. Essa informação é utilizada durante a simulação para cálculo do tempo final de execução.

A modelagem de uma infraestrutura de nuvem é necessária para a execução de uma tarefa em uma nuvem computacional. *CloudHardware* e *CloudEnvironment* implementam esse modelo e possuem informações sobre o tempo de execução remota do procedimento e sobre a quantidade de VMs (Figura 2(d)). Neste artigo, a informação mais relevante no lado servidor é o tempo de execução remota da tarefa. Essa informação varia de acordo com o tipo de nuvem a ser considerada em simulação conforme [Costa et al. 2014]. Desta forma, nossos modelos contemplam os casos de execução em nuvens e em dispositivos móveis.

Outro aspecto a ser mencionado é a possibilidade de extensão e customização de componentes de nossa proposta como a inclusão de novos mecanismos de *offloading* e novas modelagens de banda e de outros dispositivos móveis.

4. Material e Métodos

As entidades descritas na Seção 3 foram implementadas como extensão do simulador CloudSim [Calheiros et al. 2011]. Devido à característica de representar e simular o comportamento de infraestrutura de nuvem e sua alocação de tarefas, estendemos seus componentes. Não obstante, nosso modelo está desacoplado de ferramentas e soluções externas, uma vez que nenhum componente interno do CloudSim teve seu comportamento original alterado. As entidades criadas para representar o modelo da visão geral de MCC (Figura 2) *Device*, *Offloader*, *DeviceTaskExecutor*, *CloudEnvironment*, *Bandwidth* e *Task* precisam ser instanciadas e relacionadas entre si para podermos executar a simulação e experimentos.

O primeiro passo consiste na instanciação do dispositivo móvel. Em seguida, a tarefa a ser executada na nuvem deve ser definida e instanciada. De posse desta, é necessário definir uma instância responsável pela banda que é utilizada como referência na simulação.

Imediatamente o mecanismo de *offloading* (estático ou dinâmico) é definido. Depois é necessária a inclusão de informações sobre a nuvem e sua relação com a tarefa a ser executada (relação *CloudEnvironment* e *Task*). Entre elas destacamos tempo de execução da tarefa na nuvem.

Após todas as entidades serem instanciadas e devidamente relacionadas entre si a execução da simulação é realizada. Dependendo da instância de *offloading* inicializada, a entidade faz a simulação de um procedimento executando remota ou localmente. Finalmente, os dados gerados são passados por saídas comuns em linha de comando. Essa saída possui o tempo total de execução e os tempos intermediários descritos na Equação 2.

Para validação da proposta, dois experimentos foram projetados e executados em distintos contextos de realização de *offloading*. O Experimento #1 foi projetado para validação da precisão no cálculo do tempo total de execução tratando exclusivamente de *offloading* estático. O Experimento #2 foi projeto com o intuito de validar a estratégia adotada neste trabalho para tomada de decisão em *offloading* dinâmico. Ambos estão descritos nas duas subseções seguintes.

4.1. Experimento #1: Tempo Total de Execução em *Offloading* Estático

O Experimento #1 utilizou uma simulação de *offloading* estático para delegar a execução da aplicação BenchImage² à nuvem. Essa execução foi delegada a partir de dois modelos de smartphones para dois tipos de instância da Amazon EC2 configuradas com o Ubuntu Server 12.04 64-bit. O modelo de largura de banda de rede 4G foi utilizado no experimento para simular a comunicação entre o dispositivo móvel e a nuvem. Neste modelo, as taxas de *upload* e *download* são geradas aleatoriamente.

Neste experimento, simulamos uma aplicação de filtro de imagens *Cartoonizer* a um conjunto de imagens com tamanhos distintos. Os fatores dos experimentos e seus valores (níveis) estão nas Tabelas 1 e 2. Durante a execução dos cenários, o tempo total de execução da aplicação (Equação 2) foi calculado para fins de comparação com os resultados alcançados em [Costa et al. 2014].

4.2. Experimento #2: Tomada de Decisão em *Offloading* Dinâmico

O Experimento #2 consistiu na tomada de decisão do *offloading* dinâmico com base nos seguintes fatores: quantidade de computação realizada e quantidade de comunicação dispositivo-nuvem [Kumar et al. 2013] para cada algoritmo (*Binary Search*, *Bubble Sort*, *Matrix Multiplication*). A comunicação entre o *smartphone* e a nuvem foi modelada considerando redes 3G e 4G. Nestes modelos de largura de banda as taxas de *upload* e *download* são geradas aleatoriamente. Ao adicionarmos modelos com taxa de transmissão compatível com redes 3G, consideramos que a variação desta taxa traz maior veracidade em cenários de decisão corriqueiros para os usuário de aplicativos, sobretudo no Brasil (Tabelas 1 e 2).

5. Resultados e Discussões

Os resultados do *Experimento #1* são apresentados na Figura 3. A Figura 3(a) ilustra o tempo total para execução de uma tarefa de processamento de imagem em uma instância

²<https://play.google.com/store/apps/details?id=br.ufc.mdcc.benchimage>

Tabela 1. Critérios para o projeto dos experimentos

Itens	Descrição
Métricas	Experimento 1: tempo total de execução; Experimento 2: ocorreu/não ocorreu migração
Parâmetros	Experimento 1: filtro da imagem, tipo de <i>offloading</i> , tipo da rede; Experimento 2: tipo de <i>offloading</i> , modelo do dispositivo móvel, tipo da instância, taxa de download e upload
Fatores	Experimento 1: tamanho da imagem, modelo do dispositivo móvel, tipo da instância; Experimento 2: quantidade de comunicação, quantidade de computação, tipo da rede
Técnica de análise	Simulação via CloudSim
Carga de Trabalho	Experimento 1: BenchImage; Experimento 2: <i>Binary Search</i> , <i>Bubble Sort</i> e <i>Matrix Multiplication</i>

Tabela 2. Parâmetros de carga

Parâmetros	Configuração
Tamanho da imagem	0.3MP, 1MP, 2MP, 4MP e 8MP
Modelo do smartphone	LG Optimus G E977 e Samsung GT-i8190 Galaxy SIII Mini
Tipo da instância	Amazon EC2 Medium e EC2 Micro
Tipo de Rede	Experimento 1: 4G; Experimento 2: 3G e 4G
Tipo de <i>offloading</i>	Experimento 1: estático; Experimento 2: dinâmico
Experimento 2: Dados enviados (em KBytes)	[200, 400, 600, 800, 1000] <i>Matrix Multiplication</i> [2000, 4000, 6000, 8000] <i>Binary Search</i> e <i>Bubble Sort</i>

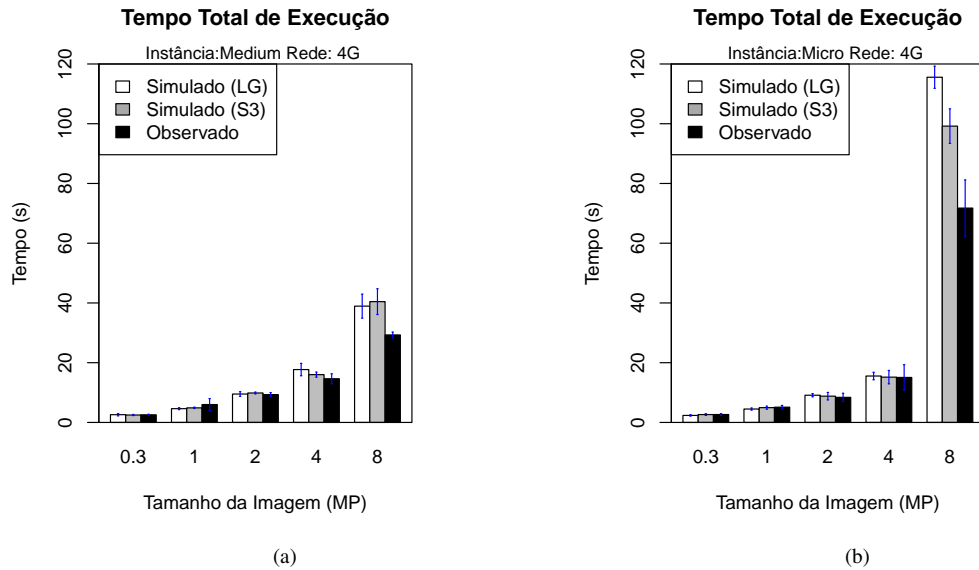


Figura 3. Experimento #1: Tempos totais de execução (ambiente real vs. simulado), banda 4G, com diferentes tamanhos de imagem e *smartphone*, (a) instância Medium; (b) instância Micro.

do tipo *Medium* da Amazon no ambiente real e simulado. O intervalo de confiança utilizados nas simulações foi de 95%.

Os resultados do experimento possuem uma acurácia de até 98.9% em comparação com valores reais obtidos em [Costa et al. 2014]. No entanto, observamos que para a imagem de 8 Megapixels, existe uma diferença de até 84% no tempo total observado e simulado. O mesmo comportamento é encontrado para a instância Micro (Figura 3(b)). Tal discrepância é explicada através da Equação 1, onde o tempo total de execução é calculado com base no tempo de *download* e *upload* das imagens. Considerando que, imagens maiores podem sofrer uma distorção maior no tempo de *download* e *upload* em relação

a imagens menores, a perda de precisão no cálculo que pode ocorrer será maior. Outro aspecto a ser considerado é a compressão de diferentes formatos de imagem. Taxas de transmissão de dados podem sofrer influência considerando esse aspecto e, portanto, produzindo resultados ligeiramente diferentes do observado em *frameworks* de MCC reais.

Outro aspecto observado foi que o tempo total de execução para imagens de 8 Megapixels é bastante diferente entre os dois casos. Isso ocorre devido ao fato da capacidade de processamento da instância *Medium* ser maior que a da instância *Micro*. Essa diferença traduz-se em uma simulação que considera maior tempo de processamento para realização da tarefa em imagens maiores.

O Experimento #2 apresenta a tomada de decisão do *offloading* para três algoritmos clássicos (Figuras 4 e 5), de implementações triviais e com ordens de complexidade distintas. O algoritmo *Binary Search* ($\theta(\log n)$) pode ser utilizado em programas de buscas de dados (e.g. aplicativo de armazenamento de contatos). O algoritmo *Bubble Sort* ($\theta(n^2)$) pode ser utilizado em programas que realizam ordenação de dados, como gerenciadores de arquivos em dispositivos móveis. Já o algoritmo *Matrix Multiplication* ($\theta(n^3)$) pode ser utilizado em *softwares* de processamento de imagens e sinais de uma maneira geral.

Na Figura 4(a) observamos que, dada a mesma quantidade de comunicação, o algoritmo *Bubble Sort* consome um pouco mais de instruções na sua execução do que o *Binary Search*. Também notamos que é necessária uma quantidade de dados menor para promover a execução do algoritmo *Bubble* na nuvem. Portanto, observamos que a complexidade dos algoritmos impacta na tomada de decisão de *offloading* entre esses dois procedimentos. Tal observação é confirmada pelas Figuras 5(a) e 5(b), nas quais é apresentada a decisão de *offloading* do algoritmo *Matrix Multiplication* para as bandas 3G e 4G. Tais figuras mostram que a tomada de decisão em favor do *offloading* não ocorreu em apenas um caso (*Matrix* (local), quadrado branco, Figura 5(a)), mesmo com uma variação considerável da carga.

Apesar da escala diferir entre as Figuras 4 e 5, o modelo de tomada de decisão mostrou que o algoritmo *Matrix Multiplication* é bem mais suscetível a sofrer migração para execução em nuvem. Realizando uma comparação entre esses resultados e aqueles apresentados pelos algoritmos *Binary Search* e *Bubble Sort*, observamos que para uma quantidade de processamento maior, o algoritmo de *offloading* opta pela execução remota com pouca influência da banda da rede 3G ou 4G.

Baseados nesses resultados, inferimos que algoritmos de maior grau complexidade (e.g. *Matrix Multiplication*) são mais suscetíveis ao *offloading* quando comparados a algoritmos de menor complexidade (e.g. *Binary Search*). Podemos ainda afirmar que o modelo produziu resultados esperados com a descrição e variação de fatores encontrados na Equação 1.

6. Conclusões

Este artigo apresenta uma proposta de modelagem e simulação de uma arquitetura típica de MCC³. Foram simulados dois mecanismos de decisão de *offloading* e, a partir dos resultados obtidos, mostrou-se que a modelagem proposta é capaz de representar cenários

³Disponível em: <https://github.com/UFC-GREAt-PPGETI/MCCSimulator>

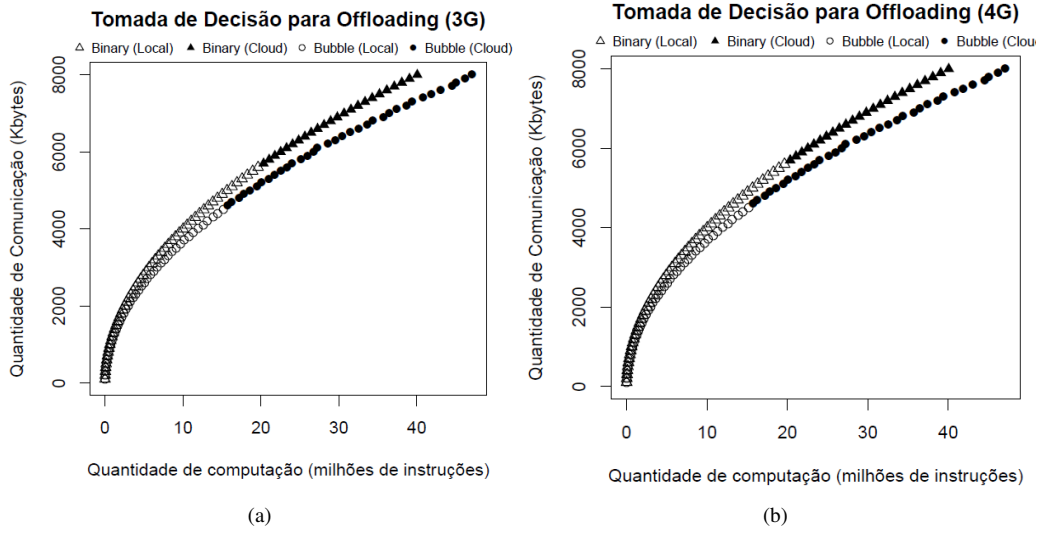


Figura 4. Experimento #2: Decisão de *offloading* levando-se em conta os algoritmos *Binary Search* e *Bubble Sort*, (a) banda 3G; (b) banda 4G.

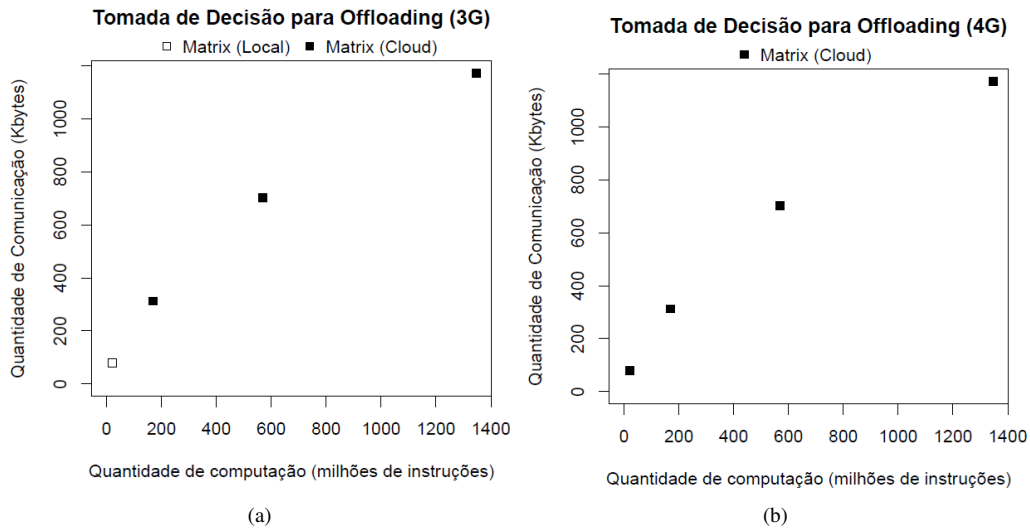


Figura 5. Experimento #2: Decisão de *offloading* levando-se em conta o algoritmo *Matrix Multiplication*, (a) banda 3G; (b) banda 4G.

reais de *offloading* estático com uma acurácia de até 98.9%. Com relação ao *offloading* dinâmico, foram realizadas tomadas de decisão baseadas no custo computacional e na quantidade de dados a serem transmitidos, a partir das quais concluímos que a ordem de complexidade algorítmica da carga é diretamente proporcional à probabilidade de *offloading*. Portanto, diante desta associação interferente, podemos concluir que a ordem de complexidade pode servir de critério de decisão em algoritmos de *offloading*.

Como sequência deste trabalho, podemos sugerir a inclusão de *Cloudlets* e de novos mecanismos de tomadas de decisão de *offloading* baseados no consumo energético dos dispositivos móveis.

Referências

- Ahmed, A. e Sabyasachi, A. (2014). Cloud computing simulators: A detailed survey and future direction. In *Advance Computing Conference (IACC), 2014 IEEE International*, pages 866–872.
- Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., and Buyya, R. (2011). Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, 41(1):23–50.
- Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., and Patti, A. (2011). Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 301–314, New York, NY, USA. ACM.
- Costa, P. B., Rego, P. A. L., Coutinho, E. F., Trinta, F. A. M., and de Souza, J. N. (2014). Uma análise do impacto da qualidade da internet móvel na utilização de cloudlets. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2014)*, pages 223–236, Florianópolis. SBC.
- Cuervo, E., Balasubramanian, A., Cho, D.-k., Wolman, A., Saroiu, S., Chandra, R., and Bahl, P. (2010). Maui: Making smartphones last longer with code offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 49–62, New York, NY, USA. ACM.
- Dinh, H. T., Lee, C., Niyato, D., and Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless Communications and Mobile Computing*.
- Fernando, N., Loke, S. W., and Rahayu, W. (2013). Mobile cloud computing: A survey. *Future Gener. Comput. Syst.*, 29(1):84–106.
- Justino, T. and Buyya, R. (2014). Outsourcing resource-intensive tasks from mobile apps to clouds: Android and aneka integration. In *Cloud Computing in Emerging Markets (CCEM), 2014 IEEE International Conference on*, pages 1–8.
- Kumar, K., Liu, J., Lu, Y.-H., and Bhargava, B. (2013). A survey of computation offloading for mobile systems. *Mob. Netw. Appl.*, 18(1):129–140.
- Li, J., Bu, K., Liu, X., and Xiao, B. (2013). Enda: Embracing network inconsistency for dynamic application offloading in mobile cloud computing. In *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, MCC '13, pages 39–44, New York, NY, USA. ACM.
- Mell, P. and Grance, T. (2011). The nist definition of cloud computing. Technical Report 800-145, National Institute of Standards and Technology (NIST), Gaithersburg, MD.
- Qi and Gani, A. (2012). Research on mobile cloud computing: Review, trend and perspectives. In *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on*, pages 195–202.
- Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23.