UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Avaliação Mista de Aplicações do Tipo *Bag of Tasks* sobre Infraestruturas de Nuvem Física Limitada e Virtual Escalada com a Utilização do *OpenStack* e do *CloudSim*

Fernando Angelin

Pelotas, 2017

Fernando Angelin

Avaliação Mista de Aplicações do Tipo *Bag of Tasks* sobre Infraestruturas de Nuvem Física Limitada e Virtual Escalada com a Utilização do *OpenStack* e do *CloudSim*

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Mestre em Ciência da Computação

Orientador: Prof. Dr. Gerson Geraldo H. Cavalheiro

Pelotas, 2017

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

A111a Angelin, Fernando

Avaliação mista de aplicações do tipo Bag of Tasks sobre infraestruturas de nuvem física limitada e virtual escalada com a utilização do Openstack e do Cloudsim / Fernando Angelin ; Gerson Geraldo Homrich Cavalheiro, orientador. — Pelotas, 2017.

71 f.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2017.

1. Cloudsim. 2. Bag of tasks. 3. Computação em nuvem. 4. Escalonamento. I. Cavalheiro, Gerson Geraldo Homrich, orient. II. Título.

CDD : 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Universidade Federal de Pelotas
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

ATESTADO

ATESTO para os devidos fins, que **FERNANDO ANGELIN** foi aprovado pela Banca Examinadora da Defesa de Dissertação realizada em 30 de Agosto de 2017, intitulada: "Avaliação Mista de Aplicações do Tipo Bag of Tasks sobre Infraestruturas de Nuvem Física Limitada e Virtual Escalada com a Utilização do OpenStack e do CloudSim", sob a orientação do Prof. Dr. Gerson Cavalheiro. Realizadas as correções recomendadas pela Comissão Examinadora, o trabalho será encaminhado para homologação pelo Colegiado do Programa de Pós-Graduação em Computação. Igualmente, informamos que os demais requisitos necessários para a obtenção do grau de **Mestre em Ciência da Computação** pela Universidade Federal de Pelotas foram cumpridos.

Pelotas, 30 de Agosto de 2017

Gerson Cavalheiro
Coordenador do Programa de Pós-Graduação em Computação

Programa de Pós-Graduação em Computação - Universidade Federal de Pelotas
E-mail: ppgc@inf.ufpel.edu.br Site: http://inf.ufpel.edu.br/ppgc Telefone:+55 (53) 3921-1327

Dedico este esforço ao meus pais João e Salete e à minha noiva Fernanda.

AGRADECIMENTOS

Gostaria de agradecer minha família, em especial, meus pais João Carlos e Salete e meus irmãos, Dudu e Léo, pela base sempre forte e por sempre me apoiarem no caminho dos estudos. Um agradecimento especial à minha noiva Fernanda, por ter me apoiado em todos os momentos (bons e ruins), sempre motivando. O mundo será nosso! Te amo muito!

Um abraço especial aos meus amigos pela motivação de todos os dias: as risadas e filosofadas com o pessoal do LUPS, Dionisíacos e DeMorgan. Obrigado pela força e motivação! Um obrigado pro meu manolo Vini. Valeu por tudo!

Ao meu querido orientador, professor Gerson, muito obrigado pela paciência, pela ajuda e pelo compartilhamento do teu conhecimento para comigo.

Agradecimento também à UFPel, por ter me proporcionado a base da minha vida acadêmica. Obrigado à CAPES pela bolsa que me auxiliou a permanecer estudando e me aperfeiçoando.

A evolução do homem passa, necessariamente, pela busca do conhecimento.
— SUN TZU

RESUMO

ANGELIN, Fernando. **Avaliação Mista de Aplicações do Tipo *Bag of Tasks* sobre Infraestruturas de Nuvem Física Limitada e Virtual Escalada com a Utilização do *OpenStack* e do *CloudSim***. 2017. 71 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2017.

A Computação em Nuvem vem apresentando um crescimento extraordinário nos últimos anos, em questão de quantidade e variedade de serviços oferecidos, estes, tomando uma forma onipresente no cotidiano. Com isso, usuários que necessitam, geralmente, de alta disponibilidade de processamento, buscam na Nuvem soluções que diminuam custos pontuais, como construção e manutenção de uma infraestrutura privada. A saída para tal é alugar infraestrutura em uma Nuvem ou até mesmo utilizar a Nuvem para dimensionar uma infraestrutura própria que supra sua demanda, sem sub ou superdimensionamento. Este trabalho apresenta um modelo de simulação mista, o qual busca comparar uma infraestrutura física limitada à uma infraestrutura virtual simulada com as mesmas características. Para isto, foram executados testes em uma infraestrutura física limitada e testes de simulação utilizando o *CloudSim*, escalando o tamanho das tarefas do tipo *Bag of Tasks* (BoT) e o número de *hosts* e núcleos computacionais. Para tais testes foram implementados algoritmos que realizam a transformação da entrada BoT para a execução na infraestrutura física e na simulada. Também, foram prototipadas classes para complementação do *CloudSim*, tanto para leitura dos BoTs transformados quanto para a criação da infraestrutura simulada. Com os testes realizados, notamos a estabilidade do sistema, quando simulados testes com BoT pequenos, médios e grandes em infraestruturas que, para nosso caso, foram classificadas como pequena, média e grande. Outra observação importante realizada foi a de que quando a infraestrutura oferece carga externa à execução desejada (utilização por outro usuário, por exemplo), o tempo final de execução dos BoTs aumenta proporcionalmente à quanto a infraestrutura está em utilização. Também percebemos que a granularidade das tarefas impacta na execução. Com relação à escalabilidade, foi percebido que BoTs classificados como grandes para infraestruturas categorizadas como pequenas foram agrupados como pequenos para infraestruturas identificadas como grandes.

Palavras-Chave: cloudsim; bag of tasks; escalonamento; computação em nuvem

ABSTRACT

ANGELIN, Fernando. **Mixed Evaluation of Bag of Tasks Applications over Limited Physical and Virtual Scheduled Cloud Infrastructures with OpenStack and CloudSim Utilization**. 2017. 71 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2017.

Cloud Computing has been showing extraordinary growth in recent years, in terms of the quantity and variety of services offered, these, taking a ubiquitous form in everyday life. As a result, users who generally require high availability of processing, search on cloud solutions that reduce specific costs, such as building and maintaining a private infrastructure. The way out is to rent infrastructure in a Cloud or even use the Cloud to size an infrastructure that suits your demand, without sub or oversize. This dissertation presents a mixed simulation model, which seeks to compare a limited physical infrastructure to a simulated virtual infrastructure with the same characteristics. For this, tests were performed on a limited physical infrastructure and simulation tests using CloudSim, scaling the size of Bag of Tasks (BoT) tasks and the number of hosts and processing cores. For such tests, were implemented algorithms that perform the transformation of the BoT input for execution in real infrastructure and simulation. Also, classes to complement CloudSim were prototyped, both for reading the transformed BoTs and for creating the simulated infrastructure. With the tests carried out, we noticed the stability of the system when simulated small, medium and large BoT tests in infrastructures that, in our case, were classified as small, medium and large. Another important observation was that when the infrastructure offers external load to the desired execution (use by another user, for example), the final execution time of the BoTs increases proportionally to how much the infrastructure is in use. We also realize that the granularity of tasks impacts execution. With regard to scalability, it was noticed that BoTs classified as large for infrastructures categorized as small were grouped as small for infrastructures identified as large.

Key-words: cloudsim; bag of tasks; scheduling; cloud computing

LISTA DE FIGURAS

1	Camadas da Computação em Nuvem.	21
2	Relacionamento entre as camadas do *CloudSim*	25
3	Diagrama de funcionamento do *OpenStack*	26
4	Comunicação entre os componentes *OpenStack.*	28
5	Modelo de controle centralizado para BoT.	30
6	Diagrama de desenvolvimento	40
7	Comparação da Hipótese 1	46
8	Comparação da Hipótese 2	47
9	Comparação dos usos em produção	48
10	Conjunto de gráficos que mostram a inviabilidade da comparação variando o intervalo de chegada e duração das tarefas do BoT	49
11	Custo por tarefa em relação à utilização de CPU	50

LISTA DE TABELAS

1 Comparação entre os tipos de Nuvem 22
2 Métricas de escalonamento consideradas pelas políticas 34
3 Parâmetros para análise de BoTs 50
4 Tempo de execução dos BoTs com sobrecarga externa (9, 100 e 1.000 nós) 51
5 Tempos de execução de um BoT em 9 *hosts* físicos e simulados; tempo de execução deste BoT escalado em BoT' para execução em 125 *hosts*; tempo dado pelo consolidação de tarefas para BoT e BoT' (tempo em minutos) 52

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
APIs	*Application Programming Interfaces*
BoT	*Bag of Tasks*
BoTs	*Bags of Tasks*
BSP	*Bulk Synchronous Parallel*
FCFS	*First Come First Served*
GB	*Gigabyte*
IAAS	*Infrastructure as a Service*
JSON	*JavaScript Object Notation*
LCFPE	*Longest Cloudlet Fastest Processing Element*
MV	Máquina Virtual
MVs	Máquinas Virtuais
NIST	*National Institute of Standards and Technology*
PAAS	*Plataform as a Service*
PaJFit	*Parallel Job Fit*
PPGC	Programa de Pós-Graduação em Computação
RAM	*Random Access Memory*
RR	*Round-Robin*
SAAS	*Software as a Service*
SCFPE	*Shortest Cloudlet Fastest Processing Element*

SUMÁRIO

1 INTRODUÇÃO 14
1.1 Objetivos e Resultados Esperados 16
1.2 Contribuições 16
1.3 Metodologia 17
1.4 Organização do Texto 17

2 CONCEITOS RELACIONADOS 18
2.1 Computação em Nuvem 18
2.1.1 Conceito 18
2.1.2 Características Essenciais 19
2.1.3 Arquitetura e Modelos de Serviço 20
2.1.4 Tipos de Nuvem 21
2.1.5 Desafios de Pesquisa em Computação em Nuvem 22
2.2 *CloudSim* 23
2.2.1 Arquitetura do *CloudSim* 24
2.3 *OpenStack* 24
2.3.1 Caracterização e Funcionamento 25
2.3.2 Arquitetura Conceitual do *OpenStack* e seus Componentes 26
2.4 *Bag of Tasks* 28
2.4.1 Características Gerais 29
2.5 Escalonamento em Nuvens 30
2.5.1 Otimização de Colônia de Formigas 31
2.5.2 Modelo de Berger 31
2.5.3 Algoritmo de Escalonamento Dinâmico 32
2.5.4 Algoritmo Min-min 32
2.5.5 Otimização de Enxame de Partículas 32
2.5.6 Algoritmo Baseado em Prioridade 33
2.5.7 Algoritmo *Round-Robin* 33
2.5.8 Abordagem com o Método Estocástico de Subida de Encosta 33
2.5.9 Considerações sobre Escalonamento em Nuvens 34

3 TRABALHOS RELACIONADOS 35

4 AVALIAÇÃO MISTA DE BOT EM AMBIENTE DE NUVEM 39
4.1 Arquitetura do Método de Avaliação Mista 39
4.2 Etapas e Módulos Desenvolvidos 42

5 EXECUÇÃO E RESULTADOS 44
5.1 Metodologia de Execução e Extração dos Resultados 44
5.2 Infraestrutura Real e Infraestrutura Simulada 44
5.3 Especificação de BoTs 45
5.4 Validação do Modelo de Simulação 45
5.5 Escalabilidade de Tarefas 47
5.6 Estudo de Caso 51
5.7 Síntese dos Resultados e Considerações sobre o Capítulo 52

6 CONCLUSÃO 54

REFERÊNCIAS 56

ANEXO A BOTS UTILIZADOS NO USO EM PRODUÇÃO 61

APÊNDICE A BOTS GERADOS E UTILIZADOS COM VARIAÇÃO NO INTERVALO DE CHEGADA, DURAÇÃO DE TAREFAS, TAMANHO DE TAREFAS E UTILIZAÇÃO DE CPU 63

APÊNDICE B BOT E BOT' (BOT ESCALADO) GERADOS E UTILIZADOS NO ESTUDO DE CASO 71

1 INTRODUÇÃO

A Computação em Nuvem revolucionou a indústria de tecnologia da informação e comunicação possibilitando o provisionamento sob demanda de recursos computacionais de forma elástica com uma política "pague pelo utilizado" (BELOGLAZOV, 2013). Desta forma uma organização pode terceirizar suas necessidades computacionais para a Nuvem e, assim, evitar investimentos em infraestrutura computacional privada. Já as empresas que fornecem este tipo de serviço podem adotar algumas políticas para economia de seus recursos e ao mesmo tempo gerar um maior lucro em suas atividades, viabilizando desta forma sua operação.

O emprego de serviços disponíveis na Nuvem, tanto em variedade quanto em número, vem crescendo exponencialmente nos últimos anos, oferecendo acesso a computadores de uma forma remota e eficiente. Entre tais serviços encontram-se aqueles que disponibilizam processamento de alto desempenho e capacidade. Neste tipo de modelo computacional, políticas empregadas nos provedores buscam otimizar o uso dos recursos físicos, com uso de virtualização e de estratégias de escalonamento, alocando e realocando de forma dinâmica os recursos físicos disponíveis aos usuários, de forma responsiva à variação de suas demandas. Este processo contínuo de adequação na alocação dos recursos, realizados em tempo real, permite configurar sistemas provedores de serviços mais eficientes evitando o super e/ou subdimensionamento dos recursos (ARMBRUST et al., 2010).

A importância da Computação em Nuvem é reforçada pela participação efetiva de muitas empresas que ditam a maioria dos rumos da tecnologia, como *Microsoft*[1], *Google*[2], *Amazon*[3] e *Red Hat*[4], que patrocinam pesquisas e desenvolvimento na área. Estas empresas oferecem serviços na Nuvem, que incluem: *Infrastructure as a Service* (IAAS), *Plataform as a Service* (PAAS) e *Software as a Service* (SAAS) (KIM, 2011).

É observado que, antes de colocar em produção uma determinada técnica ou nova

[1]Microsoft Azure, disponível em: https://azure.microsoft.com
[2]Coogle Cloud Plataform, disponível em: https://cloud.google.com/
[3]Amazon Web Services, disponível em: https://aws.amazon.com/
[4]Red Hat Rackspace Cloud, disponível em: https://www.rackspace.com/openstack/private/redhat

política para um melhor gerenciamento de recursos, é necessário verificar se estas apresentam melhorias significativas à toda a estrutura já instalada e se o novo funcionamento/comportamento reflete em alguma vantagem operacional, como redução do tempo de processamento e/ou dos custos de manutenção, por exemplo. Esta verificação pode ser feita experimentalmente ou empregando alguma técnica de avaliação de desempenho, por exemplo, analítica ou simulação.

O processo de avaliação experimental possui um alto custo associado, por requerer a instalação de um novo sistema previamente, o que pode requerer duplicação de recursos ou interrupção da planta operacional, para realizar uma etapa de testes de aferição. Este custo praticamente inviabiliza este método, restando a estas empresas os métodos analítico ou de simulação. No entanto, não é raro que ambientes para prototipação estejam disponíveis, usualmente tendo sido montados em escala menor ao utilizado em produção com equipamentos substituídos nos processos de modernização de infraestrutura.

Métodos analíticos são bastante satisfatórios, pois são capazes de produzir resultados bastante precisos. Infelizmente, na prática, tornam-se inaplicáveis rapidamente, devido ao tamanho e complexidade da concepção dos modelos. Nesses casos, métodos de aproximação são vistos como alternativas. Mesmo esses métodos de aproximação podem tornar-se ineficientes na maioria dos casos e, então, soluções empregando métodos de simulação tornam-se uma alternativa interessante (TUFFIN et al., 2007).

Os métodos de simulação também requerem a especificação de um modelo do sistema a ser avaliado. No entanto, o uso de simuladores é uma prática na área de processamento distribuído e de redes. Os modelos geralmente adotados refletem a estrutura funcional do sistema avaliado, reproduzindo seu comportamento. Desta forma, a especificação do modelo possui uma complexidade menor, pois o projetista da avaliação já está habituado a utilizá-lo. Exemplos da literatura de tais ferramentas são o *Network Simulator* (ISSARIYAKUL; HOSSAIN, 2008) para simulação de redes, o *SimGrid* (CASANOVA, 2001) para simulação de ambientes em Computação em Grade e o *CloudSim* (CALHEIROS et al., 2011) para simulação de ambientes em Nuvem. O problema é como certificar-se que o modelo de simulação representa adequadamente o sistema real.

Neste trabalho, é abordada uma metodologia mista para avaliação de desempenho de uma estratégia de escalonamento em Nuvem aplicada à aplicações do tipo *Bag of Tasks* (BoT), envolvendo experimentação e simulação. Em (SANTOS, 2016) foi apresentada uma estratégia de escalonamento em nível aplicativo de aplicações do tipo BoT e um *framework* para especificação de diferentes políticas de alocação de recursos representados por máquinas virtuais (MVs). A validação dos resultados da aplicação deste *framework* ficou restrita à escala dos recursos físicos existentes

no Programa de Pós-Graduação em Computação (PPGC) à ocasião, ou seja, dez nós computacionais *quad*-processados. A necessidade de validar o *framework* em escala apontou como alternativa a utilização de ferramentas de simulação. Desta forma, foi concebido um modelo de simulação para um provedor de serviço de processamento em Nuvem e reproduzidos os mesmos experimentos de forma a analisar a escalabilidade dos BoTs e da infraestrutura a ser utilizada.

No ambiente físico, representando um ambiente de prototipação, são realizados experimentos sobre a plataforma *OpenStack*. O método de simulação faz uso do *CloudSim*, em que é modelada a mesma infraestrutura real que fornece serviços de processamento em Nuvem.

Para auxiliar no tratamento da entrada para os testes, que são um ou uma série de BoTs, e criação do ambiente virtual de Nuvem, é apresentada uma modelagem de entrada por meio de arquivos *JavaScript Object Notation* (JSON). Primeiro para transformar a entrada do teste em ambiente físico e ambiente virtual e o segundo para descrever a infraestrutura física disponível para o *CloudSim*, eliminando a necessidade de alterações no código para a realização de cada teste.

1.1 Objetivos e Resultados Esperados

Este trabalho tem como objetivo apresentar uma metodologia de avaliação mista que permita avaliar o comportamento de uma estratégia de escalonamento aplicada à aplicações do tipo BoT em ambientes de Nuvem de larga escala, partindo de uma infraestrutura física limitada e, posteriormente, escalar tanto esta infraestrutura quanto as aplicações, com o auxílio do simulador *CloudSim*.

Com isso, espera-se replicar e estender os resultados apresentados por SANTOS em (SANTOS, 2016) escalando a infraestrutura e o tamanho das aplicações no simulador.

1.2 Contribuições

A principal contribuição desta dissertação é o oferecimento, aos desenvolvedores de aplicações para Nuvens, uma ferramenta que permita dimensionar uma infraestrutura para ser alugada ou construída que atenda as demandas dos usuários.

Como outras contribuições, se destacam:

- Modificação e operacionalização de um simulador de Nuvem computacional.
- Modelagem de entrada para o simulador em formato JSON tanto para a modelagem e inicialização da infraestrutura virtual como os BoTs a serem simulados.

- Relacionar o tamanho das tarefas com o número de nós de processamento disponíveis e ao tempo necessário para computação.

1.3 Metodologia

Para este trabalho ser realizado, foram seguidos os subsequentes passos:

1. Levantamento de informações e estudo teórico dos conceitos envolvidos neste trabalho (Computação em Nuvem, *CloudSim*, *OpenStack*, aplicações do tipo BoT e escalonamentos utilizados em Nuvem).

2. A proposição de um problema no qual um usuário que tem acesso a uma infraestrutura de pequeno porte, poder simular, em uma infraestrutura virtual com características semelhantes à real, porém escalada, ou seja, aumentada em números de nós de processamento, por exemplo.

3. Modelar e validar a infraestrutura física no *CloudSim*, criando tal estrutura no simulador, transformando a entrada BoT da infraestrutura real em um arquivo .JSON para a entrada do simulador, comparando as execuções.

4. Escalar a infraestrutura física no simulador, escalando também a entrada para tal, tentando encontrar uma relação entre tamanho de tarefas e número de nós de processamento disponíveis e analisar o tempo necessário para a realização da computação.

1.4 Organização do Texto

O texto desta dissertação está organizado como segue. O Capítulo 2 faz um apanhado geral sobre os conceitos utilizados para a realização deste trabalho. Traz informações relevantes sobre Computação em Nuvem, *CloudSim*, *OpenStack*, BoT e escalonamentos utilizados em Nuvens. No Capítulo 3 são apresentados os trabalhos relacionados que buscam justificar a metodologia utilizada neste trabalho. O Capítulo 4 fala sobre a avaliação mista, desenvolvida para este trabalho. O Capítulo 5 apresenta a organização e como foram realizadas as execuções para validação da avaliação mista e ademais objetivos deste trabalho. E, por fim, tem-se o Capítulo 6 que traz as considerações finais sobre o trabalho e a projeção de trabalhos futuros nesta linha de pesquisa.

2 CONCEITOS RELACIONADOS

A seguir, são elencados os conceitos para melhor compreensão do trabalho. Sobre Computação em Nuvem, são apresentados os principais conceitos, características essenciais, arquitetura e modelos de serviço, tipos de Nuvem e alguns desafios de pesquisa sobre Computação em Nuvem. Com relação ao *CloudSim*, são vistos os principais conceitos, o funcionamento do simulador e o relacionamento entre suas entidades. A respeito do *OpenStack*, são levantadas informações sobre suas características e funcionamento bem como sua arquitetura conceitual e seus principais componentes. Também, são abordadas as características gerais do padrão BoT. E por fim, são vistos alguns algoritmos de escalonamento em Nuvem para contrastar com o utilizado neste trabalho.

2.1 Computação em Nuvem

A Computação em Nuvem revolucionou a indústria da Tecnologia da Informação e Comunicação possibilitando o provisionamento sob demanda de recursos computacionais de forma elástica com uma política "pague pelo o que utilizar". Desta forma, uma organização pode terceirizar suas necessidades computacionais para a Nuvem e assim, evitar investimentos em infraestrutura computacional privada (BELOGLAZOV, 2013).

O termo Computação em Nuvem não é novo, tampouco suas tecnologias (ARMBRUST et al., 2010). John McCarty referiu-se à habilidade de prover e organizar computação como mercadoria (PREVE, 2011).

2.1.1 Conceito

Computação em Nuvem já foi definida de diferentes formas. Foram rastreadas vinte e duas definições de Computação em Nuvem e, a partir delas, foi proposta uma (ZHANG; CHENG; BOUTABA, 2010).

Porém, a mais aceita e difundida é a definição proposta pelo *National Institute of Standards and Technology* (NIST), que é a seguinte (traduzida): "Computação em Nu-

vem é um modelo que habilita o acesso por rede ubíquo, conveniente e sob demanda a um conjunto compartilhado de recursos de computação (como redes, servidores, armazenamento, aplicações e serviços) que possam ser rapidamente provisionados e liberados com o mínimo esforço de gerenciamento ou interação com o provedor de serviços (MELL; GRANCE, 2011)".

O NIST também define as cinco características essenciais, os três modelos de serviço e as quatro tipos de Nuvem, que são apresentadas a seguir.

2.1.2 Características Essenciais

O NIST define que a Computação em Nuvem é composta por cinco características essenciais, que são:

- **Autosserviço sob demanda**: o consumidor pode provisionar por conta própria recursos de computação, como tempo de servidor e armazenamento em rede, automaticamente e conforme necessário, sem demandar intervenção humana dos provedores de serviço.

- **Amplo acesso por rede**: os recursos estão disponíveis por meio da rede (*internet*) e são acessados por meio de mecanismos padronizados usando dispositivos clientes heterogêneos leves ou mais pesados (como *smartphones*, *tablets*, *laptops* ou *desktops*).

- **Agrupamento de recursos**: os recursos de computação do provedor são agrupados para atender a múltiplos consumidores em modalidade multi-inquilinos, com recursos físicos e virtuais diferentes dinamicamente atribuídos e reatribuídos conforme a demanda dos consumidores. Há uma certa independência de localização geográfica, uma vez que o consumidor em geral não controla ou conhece a localização exata dos recursos fornecidos (como armazenamento, processamento, memória e comunicação de rede), mas pode ser capaz de especificar a localização em um nível de abstração mais alto (como país, estado, *datacenter*).

- **Elasticidade rápida**: os recursos podem ser provisionados e liberados elasticamente, em alguns casos automaticamente, para rapidamente aumentar ou diminuir de acordo com a demanda. Para o consumidor, os recursos disponíveis para provisionamento muitas vezes parecem ser ilimitados e podem ser alocados em qualquer quantidade e a qualquer tempo.

- **Serviço mensurado**: os sistemas em Nuvem automaticamente controlam e otimizam o uso dos recursos por meio de mediações em um nível de abstração apropriado para o tipo de serviço (como armazenamento, processamento, comunicação de rede e contas de usuários ativas). A utilização de recursos pode

ser monitorada, controlada e informada, gerando transparência tanto para o fornecedor como para o consumidor do serviço utilizado.

2.1.3 Arquitetura e Modelos de Serviço

A arquitetura de Nuvem é descrita em quatro camadas, que são:

- **Camada de *hardware***: esta camada é responsável pela gestão dos recursos físicos como: servidores, roteadores, *switches*, sistemas de resfriamento e energia. Tipicamente é implementada em um *datacenter*.
- **Camada de infraestrutura**: Conhecida como camada de virtualização, esta camada oferece um centro de armazenamento e recursos computacionais por meio de particionamento de recursos físicos, utilizando tecnologias de virtualização como o *Xen*[1], o *KVM*[2] ou o *VMWare*[3].
- **Camada de plataforma**: esta camada é construída sobre a camada de infraestrutura e consiste em sistemas operacionais e *frameworks*. O principal objetivo desta camada é facilitar o desenvolvimento e lançamento de aplicações (em MVs). Por exemplo, o *Google App Engine*[4] opera na camada de plataforma, oferecendo uma *Application Programming Interface* (API) com suporte para armazenamento e lógica de negócio para aplicações *web* típicas.
- **Camada de aplicação**: está no nível mais alto da hierarquia. Consiste nas aplicações da Nuvem. Diferentemente das aplicações comuns, estas aplicações fazem o uso do escalonamento automático disponível nesta Nuvem para alcançar um melhor desempenho, disponibilidade e baixo custo de operação.

A Figura 1 ilustra as quatro camadas, bem como as suas sobreposições.

A Computação em Nuvem implementa um modelo de negócios orientado a serviços (ZHANG; CHENG; BOUTABA, 2010). O NIST define os modelos de serviço, que são:

- **IAAS**: oferece recursos de infraestrutura sob demanda que, normalmente são MVs (KAVIS, 2014). O proprietário da Nuvem que oferece IAAS é chamado de provedor IAAS. Exemplos: *Amazon EC2*[1] e *GoGrid*[2].

[1]Xen: `http://www.xenproject.org/`
[2]KVM: `http://www.linux-kvm.org/page/Main_Page`
[3]VMware: `http://www.vmware.com/br`
[4]Google App Engine: `https://cloud.google.com/appengine/docs`
[1]Amazon EC2: `https://aws.amazon.com/pt/ec2/`
[2]GoGrid: `https://my.gogrid.com/`

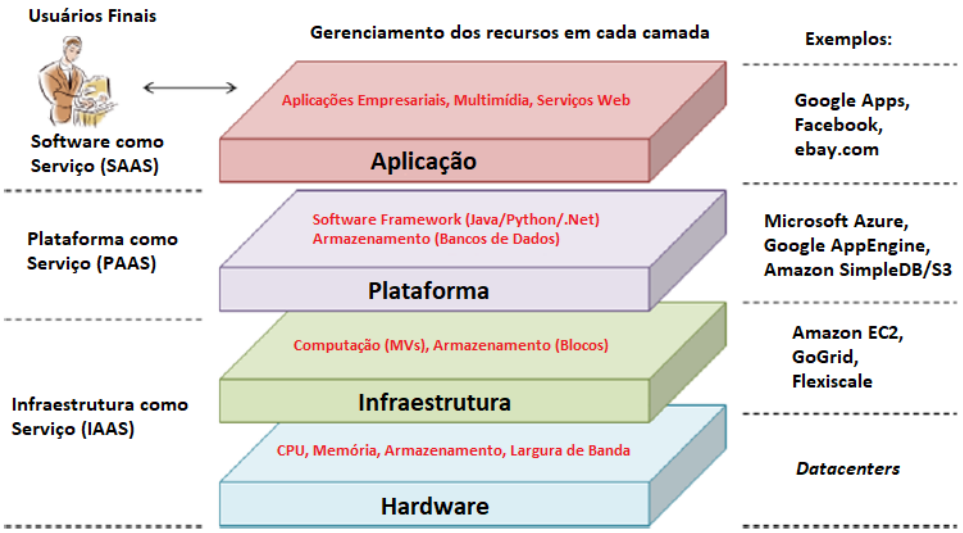

Figura 1 – Camadas da Computação em Nuvem (ZHANG; CHENG; BOUTABA, 2010).

- **PAAS**: oferece recursos de plataforma, como sistemas operacionais e *frameworks* de desenvolvimento de *software* (KAVIS, 2014). Exemplos: *Google App Engine*[3] e *Windows Azure*[4].
- **SAAS**: oferece aplicações sob demanda (KAVIS, 2014). Exemplos: *Rackspace*[5] e *Outlook Mail*[6].

2.1.4 Tipos de Nuvem

O NIST também define quatro tipos de Nuvem, que seguem:

- **Nuvem privada**: a infraestrutura na Nuvem é provisionada para uso exclusivo por uma única organização composta de diversos consumidores (como unidade de negócio). A sua propriedade, gerenciamento e operação podem ser da organização, de terceiros ou de uma combinação mista. Pode estar dentro ou fora das instalações da organização.
- **Nuvem pública**: a infraestrutura na Nuvem é provisionada para uso aberto ao público em geral. A sua propriedade, gerenciamento e operação podem ser de uma empresa, instituição acadêmica, organização governamental ou de combinação mista. Ela fica nas instalações do fornecedor.
- **Nuvem comunitária**: a infraestrutura na Nuvem é provisionada para uso exclusivo por uma determinada comunidade de consumidores de organizações que

[3]Google App Engine: `https://cloud.google.com/appengine/docs`
[4]Windows Azure: `https://azure.microsoft.com/pt-br/`
[5]Rackspace: `https://www.rackspace.com/pt`
[6]Outlook Mail: `http://outlook.com`

têm interesses em comum (de missão, requisitos de segurança, políticas, observância de regulamentações). A sua propriedade, gerenciamento e operação podem ser de uma ou mais organizações da comunidade, de terceiros ou de uma combinação mista. Pode estar dentro ou fora das instalações das organizações participantes.

- **Nuvem híbrida**: a infraestrutura na Nuvem é uma composição de duas ou mais infraestruturas na Nuvem (privadas, comunitárias ou públicas) que permanecem entidades distintas, mas são interligadas por tecnologia padronizada ou proprietária que permite a comunicação dos dados e portabilidade de aplicações, como transferência de processamento para a Nuvem para balanceamento de carga entre Nuvens.

A Tabela 1 ilustra as diferenças entre os tipos de Nuvem (gerência, propriedade, localização e segurança). As principais diferenças se encontram entre as Nuvens Pública e Privada, visto que a segunda têm a segurança mais alta e a gerência é própria.

Tabela 1 – Comparação entre os tipos de Nuvem

Implantação	*Gerência*	*Propriedade*	*Localização*	*Segurança*
Pública	Terceiros	Terceiros	Externa/Interna	Baixa
Privada	Própria	Própria/Terceiros	Interna	Alta
Comunitária	Própria/Terceiros	Própria/Terceiros	Externa/Interna	Média
Híbrida	Própria/Terceiros	Própria/Terceiros	Externa/Interna	Média

2.1.5 Desafios de Pesquisa em Computação em Nuvem

Apesar da Computação em Nuvem estar amplamente presente na indústria, as pesquisas nessa área ainda estão em fase inicial. Muitos problemas existentes ainda não foram resolvidos, enquanto novos desafios continuam surgindo das aplicações nas indústrias (ZHANG; CHENG; BOUTABA, 2010). Os principais desafios de pesquisa da Computação em Nuvem são:

- **Provisionamento automático de serviço**: é a capacidade de adquirir e liberar recursos sob demanda, alocar e desalocar recursos da Nuvem conforme a necessidade do contratante enquanto minimiza o custo operacional. Para alcançar alta agilidade para responder as flutuações de demanda, as decisões de provisionamento devem ser feitas em tempo real.

- **Migração de MVs**: é a capacidade de evitar a sobrecarga de pontos específicos da infraestrutura de servidores. Algumas ferramentas já implementam o conceito

de *live-migration*, apresentado por BELOGLAZOV em (BELOGLAZOV, 2013), onde as MVs são movidas em um processo que envolve paradas extremamente curtas que variam de dezenas de milissegundos a um segundo.

- **Consolidação de servidores**: é a capacidade de maximizar a utilização de recursos enquanto minimiza o consumo de energia. Geralmente emprega migrações de MVs.
- **Gerenciamento de energia**: tentar reduzir o consumo energético para se adequar a algumas normas governamentais e reduzir custos.
- **Segurança de dados**: serve para garantir a auditabilidade, que visa atestar se as configurações de segurança de uma aplicação foram, ou não, modificadas e a confiabilidade, que é alcançada geralmente alcançada pela utilização da criptografia.

2.2 *CloudSim*

O *CloudSim* (CALHEIROS et al., 2011) é um *framework* implementado em *Java* que fornece abstrações para diversos componentes de um ambiente de Computação em Nuvem. Seu principal propósito é o de simular ambientes de Computação em Nuvem, os quais seriam muito onerosos colocar em prática em um ambiente real. Ele foi projetado para preencher uma lacuna que faltava para simular cenários de Computação em Nuvem, pois os simuladores disponíveis eram voltados para Computação em Grade, não oferecendo funcionalidades de um ambiente de Computação em Nuvem, como o isolamento das camadas de serviço (SAAS, PAAS e IAAS) e a modelagem de ambiente de centro de dados que consiste em centenas de milhares de máquinas.

A escolha do *CloudSim* deu-se a partir do momento em que houve a necessidade de simular serviços baseados Computação em Nuvem. Também por ser altamente difundido na academia, pois é a ferramenta que, hoje, melhor reflete a simulação de um ambiente de Nuvem.

O ambiente a ser simulado com o *CloudSim* pode ser composto por um ou mais *datacenters* os quais podem possuir características como a arquitetura usada, sistema operacional, *hypervisor*, políticas de alocação de MVs, custos por unidade de processamento, memória e rede, etc.

O *datacenter* é composto por diversos *hosts*, que representam as máquinas físicas, que incorporam uma quantidade arbitrária de processadores, um escalonador de MVs e capacidades de processamento, memória e rede.

Um *host* possui um ou mais processadores que por sua vez podem ser parcialmente ou totalmente destinados a uma ou mais MVs, dependendo da política de alocação implementada.

Uma MV é modelada pela classe *Vm*, possuindo capacidades de processamento, memória e rede, número de processadores e um escalonador de tarefas a serem executadas.

O usuário é modelado pela classe *Broker*, encapsulando o gerenciamento das MVs, tais como criação, submissão de tarefas e posterior destruição dessas MVs.

Uma tarefa a ser executada em uma MV chama-se *Cloudlet*. Ela possui um número de instruções a ser executada, um tamanho de entrada e saída em *bytes* e um modelo de utilização, que define qual a porcentagem dos recursos da MV serão requisitados pela tarefa.

Ao final de uma execução é gerada uma saída/relatório onde são apresentadas informações como: quantas *cloudlets* foram executadas e o tempo de duração, em quais MVs, em quais *hosts* e em qual *datacenter*.

2.2.1 Arquitetura do *CloudSim*

O *CloudSim* pode ser dividido em três camadas principais:

- **Camada do núcleo da simulação**: onde é realizado o controle da simulação propriamente dito, faz a gerência da execução dos eventos que acontecem na simulação e faz o controle da troca de mensagens entre as entidades presentes na simulação.
- **Camada principal**: é a camada onde fica todo o controle da simulação em se tratando da Nuvem. Também é onde são definidos os objetos nos quais vão ser simulados, como rede, recursos e serviços da Nuvem, etc.
- **Camada de usuário**: é onde define-se o ambiente que será simulado, bem como algumas configurações e políticas do lado do usuário do *datacenter*.

Na Figura 2 podemos ver o relacionamento destas três camadas que compõe o *CloudSim* e suas principais funções.

2.3 *OpenStack*

Nesta seção é apresentado o *OpenStack*. São relacionados os conceitos necessários à compreensão, tanto de sua estrutura quanto de sua operação básica.

A utilização do *OpenStack* neste trabalho se dá devido a continuidade de trabalhos apresentados, realizados sobre esta plataforma (trabalhos de conclusão e dissertações) e continuados pelo grupo de pesquisa. Também, por ser uma ferramenta amplamente utilizada na academia bem como por empresas especializadas em oferecimento de serviços em Nuvem.

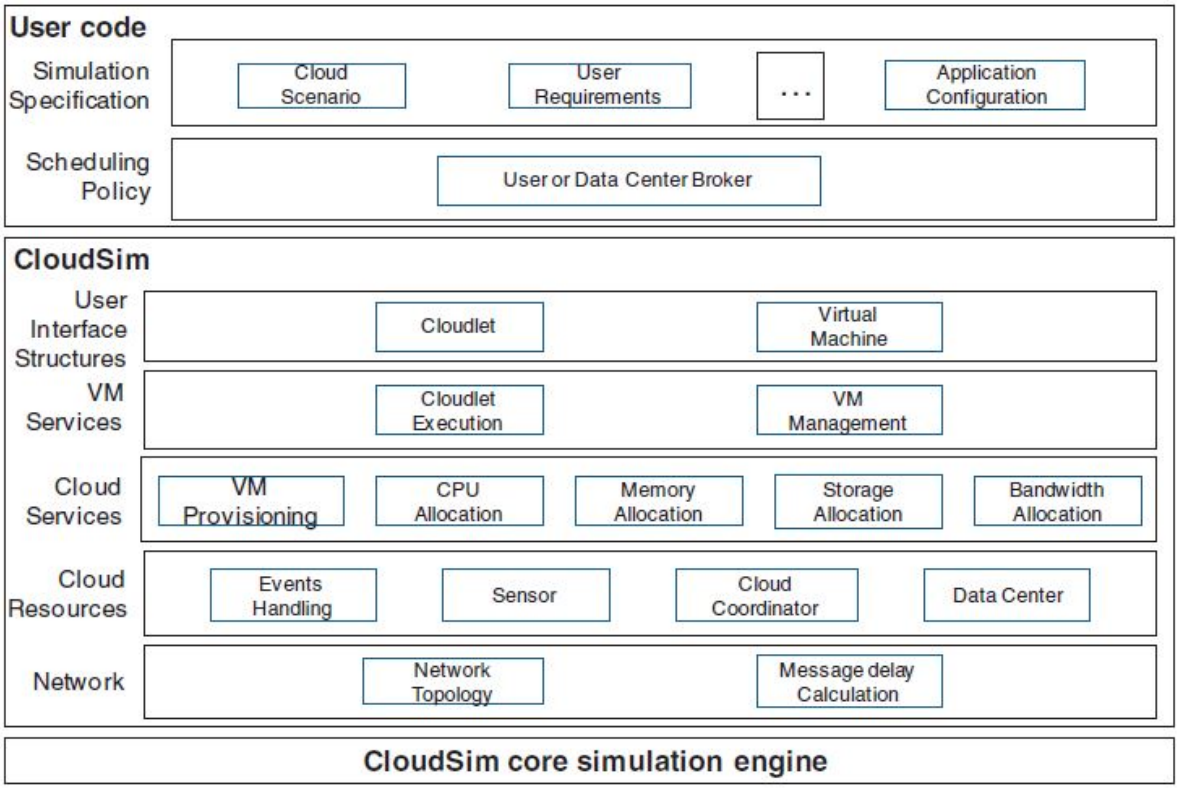

Figura 2 – Relacionamento entre as camadas do *CloudSim* (CALHEIROS et al., 2011)

2.3.1 Caracterização e Funcionamento

O *OpenStack* é um sistema operacional para o gerenciamento de forma simplificada de uma Nuvem (OPENSTACK, 2015). Ele permite controlar grandes centros de computação, armazenamento e recursos de rede (BARKAT; SANTOS; IKKEN, 2015) em um *site* provedor de recursos por meio de um painel administrativo, de onde são controlados os recursos que são fornecidos aos seus usuários, por meio de uma interface *web* (OPENSTACK, 2015). O projeto *OpenStack* é *open source*, ou seja, várias pessoas/desenvolvedores ao redor do mundo podem contribuir na sua construção, pois seu código é aberto.

O principal objetivo do *OpenStack* é prover escalabilidade e disponibilidade dos recursos de hardware às aplicações submetidas. Assim, o *OpenStack* visa fornecer Computação em Nuvem de forma elástica tanto para Nuvens privadas quanto para Nuvens públicas, independente de seu tamanho, apresentando uma implementação simples e um vasto conjunto de recursos.

Uma nuvem gerida pelo *OpenStack* provê, principalmente, uma solução de IAAS por meio de uma variedade de serviços complementares. Cada serviço oferece uma API que facilita esta integração. A Figura 3 apresenta o diagrama da estrutura de relacionamento entre os módulos do *OpenStack*.

O *OpenStack* é instalado sobre um Sistema Operacional que, por sua vez, está

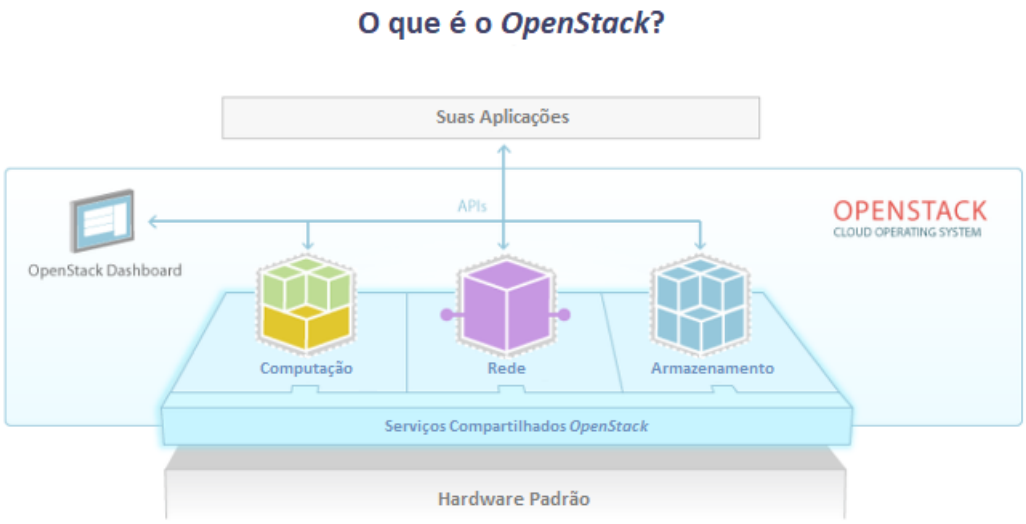

Figura 3 – Diagrama de funcionamento do *OpenStack* (OPENSTACK, 2015)

instalado no *hardware* base, que pode variar desde computadores convencionais ligados em rede até grandes *datacenters* empresariais. Os três pilares do *OpenStack* são: *OpenStack Compute*, que tem como objetivo principal prover e gerenciar computação ou processamento; *OpenStack Storage*, que fornece armazenamento; e *OpenStack Networking*, que realiza a gestão de endereços de *Internet Protocol* (IP) e rede orientada a APIs.

Estes serviços são controlados pelo *OpenStack Dashboard*, que consiste em uma interface *web* que facilita o controle de todos os outros serviços e a interação com as aplicações de usuário. Além disso, são executados sobre o *OpenStack Shared Services*, em tradução livre, Serviços Compartilhados do *OpenStack*, que são: *OpenStack Identity*, que é responsável pela autenticação dos usuários com seus referidos serviços e o *OpenStack Image*, responsável pelo armazenamento e recuperação das imagens de disco de MVs, muito úteis durante a criação de novas instâncias de computação.

Estes serviços facilitam a implementação e o controle de uma Nuvem e integram os componentes *OpenStack* uns aos outros para prover uma experiência unificada para os usuários que interagirem com os diferentes recursos da Nuvem.

2.3.2 Arquitetura Conceitual do *OpenStack* e seus Componentes

Uma das principais características do *OpenStack* é a de apresentar uma arquitetura modular, onde cada módulo ou componente é responsável por provisionar um serviço específico. Isto permite que o administrador da Nuvem possa remover ou adicionar funcionalidades em sua Nuvem sem a necessidade de reinstalação total do ambiente.

Estes componentes são conhecidos pelo codinome do projeto, para facilitar a di-

ferenciação entre eles. Cada componente é responsável por oferecer uma funcionalidade e, os componentes que estão disponíveis por padrão na versão distribuição atual, chamada *OpenStack Ocata*[1], são:

- *Horizon*: componente responsável pelo serviço *OpenStack Dashboard*. Provê aos administradores e usuários da Nuvem, uma interface gráfica de acesso, fornecimento e automatização de recursos baseados na Nuvem gerida pelo *OpenStack*. Seu *design* é extensível, ou seja, pode facilmente ligar-se aos outros componentes, como por exemplo, monitoramento, cobrança e ferramentas de gestão adicionais.

- *Nova*: componente responsável pelo serviço *OpenStack Compute*. Fornece recursos de computação sob demanda pelo provisionamento e gerenciamento do ciclo de vida das MVs usando algum tipo de monitor de MV.

- *Neutron*: componente responsável pelo serviço *OpenStack Networking*. Fornece um sistema de conexão automática, escalável e API orientada para a gestão de rede e endereços IP.

- *Cinder*: componente responsável pelo serviço *OpenStack Block Storage*. Fornece a capacidade de realizar o armazenamento em bloco com mapeamento persistente para instâncias de computação com suporte para uma variedade de soluções de armazenamento.

- *Glance*: componente responsável pelo serviço *OpenStack Image Service*. Fornece um registro de cópias de disco de *boot* e um serviço para armazenamento dessas imagens de sistema.

- *Swift*: componente responsável pelo serviço *OpenStack Object Storage*. Fornece o crescimento distribuído dos dados, armazenamento acessível pelas APIs que pode ser integrado diretamente em aplicações ou utilizados para *backup*, arquivamento e retenção de dados.

- *Keystone*: componente responsável pelo serviço *OpenStack Identity*. Fornece um diretório central dos usuários mapeados para os serviços *OpenStack*. Atua como um sistema de autenticação comum em todo o sistema operacional em Nuvem e pode integrar-se com os serviços de diretório de *back-end* existentes.

- *Ceilometer*: componente responsável pelo serviço *OpenStack Telemetry*. Fornece infraestrutura comum para a coleta de medições de uso e desempenho dentro de uma Nuvem gerida pelo *OpenStack*. A partir destes dados coletados,

[1] OpenStack Ocata Overview: `https://releases.openstack.org/ocata/index.html`

pode-se calcular, por exemplo, o quanto será cobrado pelo serviço que certo usuário utilizou.

- *Heat*: componente responsável pelo serviço *OpenStack Orchestration*. Implementa um serviço para orquestrar múltiplas aplicações em Nuvens compostas, usando o modelo *Amazon Web Service*.

Na Figura 4, pode-se analisar o fluxo de informações e comunicação entre os componentes.

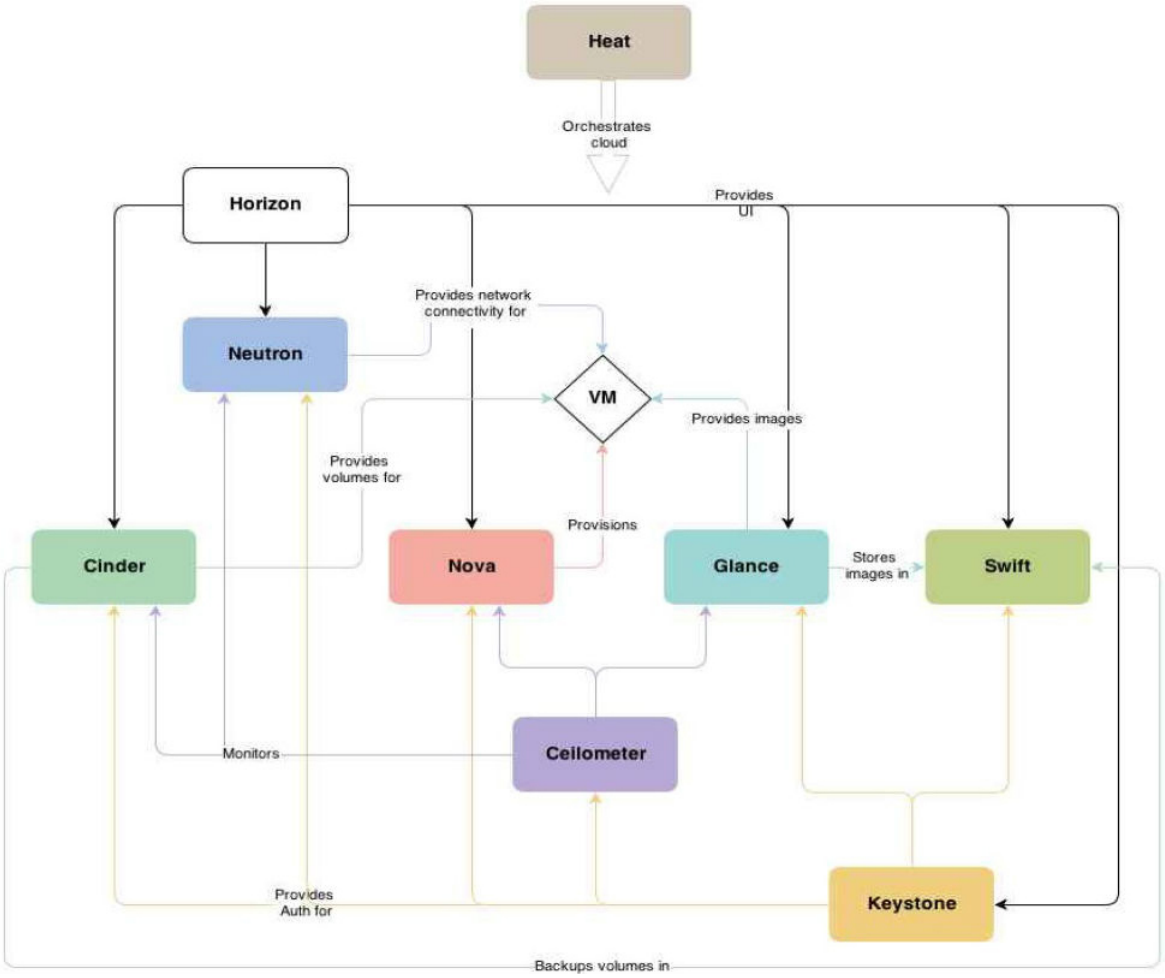

Figura 4 – Comunicação entre os componentes *OpenStack*. (LIBERTY, 2015)

2.4 *Bag of Tasks*

Nesta seção são apresentadas as principais características deste tipo de aplicação e de um modelo que as represente quando submetido a um ambiente computacional.

O principal objetivo de um programa ou algoritmo paralelo é transformar um único fluxo de instruções em múltiplos, para que possam ser executados simultaneamente em diferentes unidades de processamento. Embora isto possa parecer simples à primeira vista, a dependência de instruções sobre resultados de computações anteriores

dificulta a capacidade de distribuir instruções para processamento. Programas paralelos distinguem-se dos seus homólogos sequenciais na forma de como são incluídas algum tipo de descrição que, facilite, de certa forma, esta distribuição. No entanto, não há uma única abordagem para descrever ou mesmo conceber algoritmos paralelos.

De fato, há uma ampla classe de perspectivas, ou padrões de projeto, que representam o paralelismo de forma satisfatória para determinadas classes de problemas, mas que falham em casos mais gerais. No núcleo dos padrões de projeto dos algoritmos paralelos, estão diferentes estratégias para o particionamento do fluxo de instruções e recursos associados para a alocação de processadores e memória. Um destes padrões é o BoT, que é muito utilizado em aplicações científicas com alta demanda de processamento.

Aplicações do tipo BoT são descritas por um conjunto de tarefas independentes (CIRNE et al., 2003). Não raro, estas tarefas possuem, isoladamente, alto custo computacional. Para evitar grandes investimentos em recursos computacionais, usuários podem submeter à Nuvem, de forma oportunista, suas demandas de processamento para este tipo de aplicação.

O padrão BoT foi escolhido para ser utilizado neste trabalho, devido a utilização do mesmo em trabalhos (trabalhos de conclusão e dissertações) desenvolvidos e apresentados pelo grupo de pesquisa no decorrer dos últimos anos.

2.4.1 Características Gerais

BoT é uma estratégia de processamento paralelo em que as tarefas são colocadas em um saco de tarefas que é compartilhado pelos processadores clientes, chamados de *workers*. Cada *worker*, repetitivamente, pega uma tarefa e executa-a. É possível, e permitido, que uma tarefa crie novas tarefas durante a execução e, estas novas tarefas, são retornadas ao saco de tarefas para serem processadas. A computação encerra quando o saco de tarefas estiver vazio e os *workers* estiverem ociosos (ANDREWS, 2000).

O padrão BoT é aplicável nos casos em que os algoritmos possam ser decompostos em vários subprogramas independentes, ou seja, nenhum subprograma depende da computação de outro subprograma para ter sua execução realizada, como por exemplo, aplicações que envolvam buscas ou pesquisas por força bruta (BENEDICTIS et al., 2016). BoT também pode ser útil em estudos paramétricos (PACINI et al., 2012) ou para a avaliação da função *fitness* em algoritmos genéticos (PINGALE; MOGAL, 2015). Um fator em comum dentre estas aplicações é que a parte dominante do trabalho pode ser decomposta em tarefas que têm tempos de execução significativamente maiores do que o tempo necessário para a transferência de dados da tarefa.

A arquitetura geral de uma abordagem BoT consiste em ter um servidor central de tarefas, como ilustrado na Figura 5.

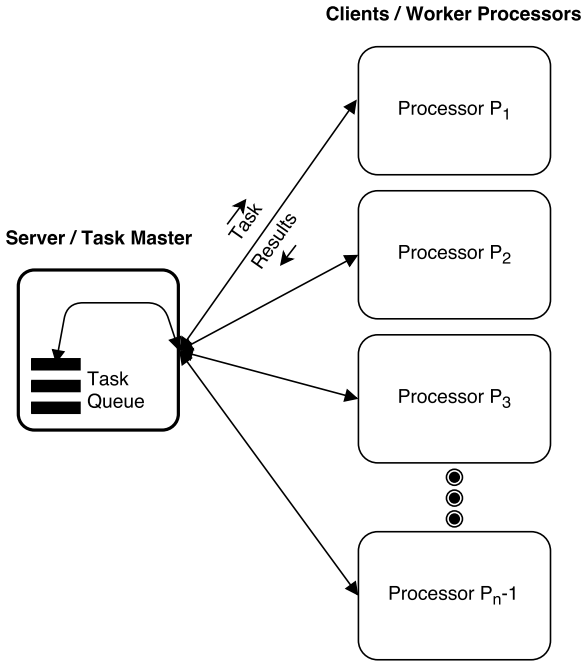

Figura 5 – Modelo de controle centralizado para BoT. (LUKE, 2003)

Este servidor gerencia o saco de tarefas e a distribuição das mesmas para os processadores que estiverem disponíveis. Além de servir os processadores clientes com as tarefas, o servidor é responsável por integrar os vários resultados de todas as computações em uma solução final.

2.5 Escalonamento em Nuvens

O escalonamento é a atividade vital no *background* da Computação em Nuvem. Sem ele, não seria possível ter uma melhora na utilização dos recursos disponíveis nos servidores e, consequentemente, ter uma melhora em sua performance (TOHIDIRAD et al., 2015). A utilização de algum tipo de escalonamento proporciona às empresas prestadoras de serviços em Nuvem, um melhor gerenciamento de seus recursos e, consequentemente, lucro em suas operações, viabilizando suas operações.

O escalonamento de tarefas é o processo de alocação de recursos para determinada tarefa em um tempo específico. O principal objetivo do escalonamento é maximizar a utilização desses recursos além de minimizar o tempo de espera. Um bom algoritmo de escalonamento proporciona um bom desempenho ao sistema. Na Nuvem existem inúmeros e distintos recursos disponíveis. O custo de executar tarefas na Nuvem depende de quais recursos estão sendo alocados no ambiente e do tipo de escalonamento (WADHONKAR; THENG, 2016).

Existe uma gama elevada de políticas e algoritmos de escalonamento. Nesta sessão são elencadas algumas destas políticas amplamente utilizadas em Computação em Nuvem.

2.5.1 Otimização de Colônia de Formigas

Um escalonador para Nuvem baseado em otimização de colônia de formigas é apresentado por MATEOS; PACINI; GARINO em (MATEOS; PACINI; GARINO, 2013). O objetivo deste escalonador é minimizar o tempo de fluxo ponderado de um conjunto de experimentos de varredura de parâmetros, enquanto ao mesmo tempo minimiza o fluxo de tempo ao usar uma Nuvem. Neste algoritmo, a carga é calculada em cada *host* levando em consideração a utilização de CPU feita por todas as MVs que estão sendo executadas em cada *host*. Esta métrica é útil para que uma formiga escolha o *host* menos carregado para alocar a sua MV.

Experimentos de varredura de parâmetros é um tipo de simulação numérica que envolve executar uma grande quantidade de tarefas independentes e tipicamente requerem muito poder computacional. Essas tarefas devem ser processadas eficientemente nos diferentes recursos de computação de um ambiente distribuído, como os fornecidos por uma Nuvem. Consequentemente, o escalonamento de tarefas neste contexto desempenha um papel fundamental. Neste algoritmo, o fluxo de tempo e tempo de execução são avaliados. Os resultados da avaliação destas métricas mostram que o desempenho desta otimização é melhor que os algoritmos randômico e melhor esforço.

2.5.2 Modelo de Berger

Em (XU et al., 2011) é proposto pela primeira vez um algoritmo de escalonamento de tarefas baseado no modelo de Berger (BERGER et al., 1966) em ambiente de Nuvem. O modelo de Berger de distribuição justa é baseado em estados de expectativa. É uma série de teorias da distribuição da riqueza social. Os estados de expectativa são formados por uma série de teorias e são usados para estudar atores e avaliar o impacto de seu comportamento. Em um breve discurso, a expectativa afirma que as teorias são para estudar as duas seguintes questões. Primeiro, como o ator pode gerar expectativas de si mesmo e de outros indivíduos de acordo com as informações (como *status*, recompensa e diferenças de desempenho) em todos; e em segundo lugar, essas expectativas de como afetar o comportamento (como participativo e decisório) dos atores e outros. As teorias dos estados da expectativa foram expandidas e aplicadas extensamente.

A teoria do valor de Estado de distribuição justa é uma base teórica importante do artigo. Ela descreve ao alocador como usar comparações referenciais para estabelecer a expectativa de alocação de recompensa. As expectativas são usadas para avaliar a justiça ou injustiça de distribuição em uma variedade de circunstâncias. Por meio de uma expansão para o *CloudSim*, o algoritmo de escalonamento de tarefas baseado no modelo de Berger foi testado. A validação do algoritmo é verificada na plataforma de simulação estendida. Ao comparar os resultados da simulação com

o algoritmo de tempo de conclusão ótimo, o algoritmo proposto é a implementação efetiva das tarefas do usuário e com maior igualdade.

2.5.3 Algoritmo de Escalonamento Dinâmico

No trabalho de (WANG et al., 2012) o algoritmo de escalonamento dinâmico tradicional é estendido considerando a confiabilidade dos nós com recursos. Este algoritmo atende à exigência de tarefas dos usuários e torna o escalonamento destas tarefas mais razoável baseado em um grafo acíclico dirigido. A principal contribuição deste estudo para sistemas de escalonamento é a extensão da formulação tradicional do problema de escalonamento de forma que tanto o tempo de execução quanto a confiabilidade das aplicações sejam considerados simultaneamente. Avaliação de confiança de MVs num ambiente de Nuvem pelo algoritmo proposto mostra diminuir a probabilidade de falha na atribuição das tarefas, e a garantia da execução de tarefas em um ambiente seguro.

2.5.4 Algoritmo Min-min

A principal ideia do algoritmo Min-min (BRAUN et al., 2001) é de enviar o mais rápido possível cada tarefa para as MVs com recursos que podem completar a tarefa no tempo mais curto possível. Este algoritmo executa tarefas curtas em paralelo e as tarefas longas são executadas depois. O defeito deste algoritmo é de escalonar primeiro as tarefas curtas, até que as MVs estejam livres para escalonar e executar tarefas longas. Min-min pode causar tanto um tempo de execução mais longo para tarefas executadas em lotes quanto uma carga desbalanceada.

Comparado com o algoritmo Min-min tradicional, o algoritmo melhorado adiciona três restrições (qualidade de serviço, o modelo de prioridade dinâmico e o custo do serviço) à estratégia que podem mudar esta condição. Os resultados experimentais do algoritmo Min-min melhorado mostram que ele pode aumentar a taxa de utilização de recursos, tarefas longas podem ser executadas em um tempo razoável e atender às necessidades dos usuários.

2.5.5 Otimização de Enxame de Partículas

Otimização de enxame de partículas como uma meta-heurística é uma técnica de otimização baseada na busca global auto-adaptável introduzida no trabalho de (KENNEDY, 2011). Este algoritmo é semelhante a outros algoritmos baseados na população como algoritmos genéticos, mas, não há recombinação direta de indivíduos da população (PANDEY et al., 2010).

O ponto central deste algoritmo é minimizar o custo total de computação de um fluxo de trabalho da uma aplicação. Como uma medida de desempenho, os autores usam o custo para a execução completa de uma aplicação como métrica. O objetivo

é minimizar o custo total do fluxo de execução da aplicação em ambientes de Nuvem. Os resultados mostram que o mapeamento de recursos para tarefas baseado neste algoritmo pode alcançar economia de pelo menos três vezes o custo em comparação com o mapeamento baseado em melhor seleção de recursos para um fluxo de execução. Em adição, a otimização de enxame de partículas realiza o balanceamento de carga nos recursos computacionais distribuindo as tarefas para recursos disponíveis.

2.5.6 Algoritmo Baseado em Prioridade

Prioridade de tarefas é uma questão importante sobre escalonamento pelo fato de que algumas tarefas devem ser atendidas antes de outras e não podem ficar por muito tempo no sistema. Em Computação em Nuvem, o principal objetivo deste algoritmo é de propor um novo algoritmo de escalonamento de tarefas baseado em prioridades chamado algoritmo de escalonamento de tarefas baseado em prioridades em Nuvem. O algoritmo proposto é baseado na teoria de Hierarquia Analítica de Processos. A Hierarquia analítica dos processos é uma tomada de decisão multicritério e um modelo de tomada de decisão multiatributo (GHANBARIA; OTHMANA, 2012). O algoritmo de escalonamento baseado em prioridades em Nuvem baseia-se no modelo de decisão de critérios múltiplos.

2.5.7 Algoritmo *Round-Robin*

O algoritmo RR concentra-se principalmente em distribuir a carga igualmente para todos os recursos (SAMAL; MISHRA, 2013). Usando este algoritmo, o *broker* aloca uma MV para um nó computacional de maneira cíclica. O escalonamento RR na Computação em Nuvem é muito similar ao escalonamento RR usado no escalonamento de processos. O escalonador começa com um nó e segue para um próximo nó, depois que uma MV é atribuída a este nó. Isto irá repetir-se até que todos os nós estarão alocados com pelo menos uma MV e então o escalonador retorna novamente ao primeiro nó. Consequentemente, neste caso, o escalonador não espera pelo esgotamento dos recursos de um nó antes de seguir para o próximo. Apesar dos algoritmos RR serem baseados em uma regra simples, mais carga é concebida nos servidores e, portanto, desbalanceando o tráfego. O resultado de algoritmos RR apresentam melhor tempo de resposta e balanceamento de carga em relação a outros algoritmos.

2.5.8 Abordagem com o Método Estocástico de Subida de Encosta

Em (MONDAL; DASGUPTA; DUTTA, 2012) é apresentada uma otimização local baseada numa abordagem com o método estocástico de subida de encosta. Uma variante deste algoritmo é uma das abordagens incompletas para resolução de problemas de otimização. Os métodos incompletos podem não garantir respostas ótimas para todas as entradas. Ao invés disso, esses métodos encontram respostas satisfa-

tórias para problemas solucionáveis com alta probabilidade. Com o cenário e configuração, o tempo médio geral de tempo de resposta é calculado. Os resultados mostram que na maioria dos casos, a utilização do algoritmo estocástico de subida de encosta diminui o tempo de resposta.

2.5.9 Considerações sobre Escalonamento em Nuvens

Para sumarizar, na Tabela 2, os principais algoritmos utilizados para escalonamento em Nuvens computacionais e as métricas consideradas por cada um.

	Métricas de escalonamento considerados											
Políticas	Tempo de execução	Tempo de resposta	Confiança	Tempo de Processamento	Utilização	Fluxo de Tempo	Custo	Execução bem sucedida	Confiabilidade	Equidade	Balanceamento de carga	Complexidade razoável
Otimização de Colônia de Formigas				✓		✓						
Modelo de Berger	✓									✓		
Algoritmo de escalonamento dinâmico	✓		✓					✓	✓			
Algoritmo Min-Min	✓				✓							
Otimização de enxame de partículas							✓				✓	
Algoritmo baseado em prioridade				✓								✓
Algoritmo *Round-Robin*		✓									✓	
Método estocástico subida de encosta		✓									✓	

Tabela 2 – Métricas de escalonamento consideradas pelas políticas (MOHAMMADI; JAMALI; BEKRAVI, 2014)

Percebe-se que apenas um algoritmo não consegue tangenciar todas as métricas, por exemplo, um algoritmo RR pode otimizar tempo de resposta e balanceamento de carga, mas não se espera uma otimização no custo.

Atualmente na academia, se trabalha principalmente com heurísticas acopladas com algum algoritmo escolhido, visto que o problema de escalonamento está associado ao problema da mochila, que é um NP-Completo (MOHAMMADI; JAMALI; BEKRAVI, 2014), as heurísticas auxiliam a diminuir o custo computacional e a complexidade desta tarefa. Porém, o desenvolvimento de heurísticas é muito custoso e, para certos problemas, não se faz necessário sua utilização.

Neste trabalho, foi utilizado o algoritmo RR para o escalonamento, tanto das *cloudlets* no *CloudSim* quanto nas MVs no *OpenStack*, para replicação dos testes de SANTOS apresentados em (SANTOS, 2016) e manter a infraestrutura virtual o mais semelhante à infraestrutura real disponível.

3 TRABALHOS RELACIONADOS

Como trabalhos relacionados, são citados alguns que utilizam métodos de execução em infraestrutura real e de simulação para obtenção dos resultados, tendo como caso de estudo aplicações BoT. Nestes trabalhos, as ferramentas utilizadas são as mesmas empregadas no desenvolvimento da presente dissertação: *OpenStack* e *CloudSim*.

Segundo IOSUP; EPEMA em (IOSUP; EPEMA, 2011), aplicações do tipo BoT são as que produzem um maior número de tarefas e que consomem mais recursos em Grades computacionais. Desta forma, justifica-se os trabalhos na literatura que abordam o problema de avaliação deste tipo de aplicação em Nuvens Computacionais, devido ao fato da necessidade de ajustar a infraestrutura disponível para a execução conforme a demanda do usuário. No artigo citado e, de forma complementar em (IOSUP et al., 2008), são apresentadas diferentes propriedades de aplicações BoT, permitindo a especificação de um modelo para esta classe de problemas. Estas propriedades, tais como número e custo computacional de tarefas e tempo entre submissões de BoTs, são apresentados em termos de suas distribuições de probabilidade.

Em (SANTOS, 2016), um modelo de aplicação BoT foi desenvolvido conforme as propriedades destacadas nos trabalhos de IOSUP; EPEMA citados anteriormente. Este modelo permitiu construir casos de estudo, na forma de BoTs, para avaliar uma estratégia de escalonamento em linha proposta para ambiente de Nuvem. O objetivo do trabalho consistiu em fornecer uma dinâmica em que pudesse ser dimensionada a necessidade de recursos em uma infraestrutura em função de aplicações respondendo a uma determinada classe de aplicações BoT.

A dinâmica proposta consiste em aplicar diferentes estratégias de escalonamento sobre um BoT submetido a uma infraestrutura conhecida e simular a execução dos custos computacionais gerados por suas tarefas, obtendo o tempo total de sua execução. A validação desta dinâmica se deu pela comparação dos resultados obtidos pelo processo de simulação, denominado Consolidação de Tarefas (SANTOS; BOIS; CAVALHEIRO, 2016), com a execução dos custos computacionais em uma infraestrutura real, gerida pelo *OpenStack*. Pelas aferições realizadas, a diferença entre o tempo

projetado daquele mensurado pela execução ficou em torno de 16% (SANTOS; BOIS; CAVALHEIRO, 2016).

No trabalho abordado por GUTIERREZ-GARCIA; SIM (GUTIERREZ-GARCIA; SIM, 2013), o problema estudado faz referência a maximizar a utilização dos recursos de processamento disponíveis para a execução de tarefas geradas por BoTs em uma Nuvem heterogênea. Para tanto, foi proposto um conjunto de quatorze (14) heurísticas para escalonamento, onde são executadas duas operações básicas: ordenação das tarefas e o mapeamento destas tarefas sobre os recursos disponíveis. Neste mesmo trabalho, o modelo de BoT considera que as tarefas possuem atributos como tempo de chegada e custo computacional expresso em termos de número de instruções. Para realizar a avaliação foram utilizadas quatro (4) combinações para determinar o tamanho das tarefas em cada BoT. Os resultados foram obtidos por meio de simulação em uma máquina utilizando o *framework* JADE (BELLIFEMINE; POGGI; RIMASSA, 1999).

No trabalho de NETTO; BUYYA (NETTO; BUYYA, 2009), é considerado um ambiente no qual o provedor não disponibiliza informações sobre a carga de processamento disponível, negociando com o usuário, uma data limite para o término do processamento. O escalonamento é realizado de forma dinâmica e, em função da carga assumida pelo provedor, é definida sua capacidade para receber novos BoTs. A avaliação de desempenho foi realizada em um ambiente composto por múltiplos *clusters* de larga escala simulados, assumindo que 50% da carga submetida determinava a execução em um *cluster* específico e os outros 50% podendo ser executados em quaisquer *cluster* do ambiente. Para tais testes foi utilizado um simulador baseado em eventos *Parallel Job Fit* (PaJFit) (NETTO; BUYYA, 2008).

Em (ER. SHIMPY, 2014), ER. SHIMPY apresenta uma comparação entre algoritmos de escalonamento, seus respectivos parâmetros, finalidade principal, em que ferramenta se encontra e se é utilizada preferencialmente em Computação em Nuvem ou Computação em Grade, concluindo que os algoritmos que usam heurísticas para a realização do escalonamento são mais apropriados. A conclusão foi obtida por meio de análise de outros artigos, apresentando um formato de *survey*. WADHONKAR; THENG em (WADHONKAR; THENG, 2016) também apresentam um comparativo entre algoritmos de escalonamento para ambientes de Nuvem, concluindo que o problema de escalonamento é heurístico. Portanto, mais pesquisa deve ser realizada na área para se encontrar uma solução ótima. A avaliação foi realizada por meio do *CloudSim*.

No trabalho de KONG; LEI; MA (KONG; LEI; MA, 2016) foi proposto um algoritmo adaptativo, que leva em consideração a banda de rede, o tempo em que se deve terminar a tarefa e o preço da execução, para escalonamento de MVs em ambiente de Nuvem. Este algoritmo foi comparado com outros três (3) algoritmos (alocação

dinâmica baseada em leilão de recursos, alocação de recursos baseado em preço fixo, com $0,45 por minuto e com $0,30 por minuto, e algoritmo de escalonamento de recursos dinâmicos). A execução foi realizada no *CloudSim* e o algoritmo proposto teve melhores resultados.

LOGANATHAN; MUKHERJEE propuseram um algoritmo de escalonamento baseado em uma modificação do melhor encaixe com capacidade (LOGANATHAN; MUKHERJEE, 2015). A avaliação foi realizada em duas etapas: experimento em infraestrutura física, utilizando quatro (4) nós, cada um com quatro (4) núcleos e um (1) GB de memória RAM; e por simulação utilizando o *CloudSim*, para escalar a arquitetura real, pois era pequena. A comparação do algoritmo proposto se deu com um algoritmo de tempo mínimo para completar a tarefa e um algoritmo de prioridade e, se mostrou mais eficiente nos casos de número de sucesso em relação ao número de requisições de tarefas e na utilização de recursos também em relação ao número de requisições de tarefas.

TOHIDIRAD et al. apresenta uma avaliação de três (3) algoritmos de escalonamentos de MVs em ambientes de Nuvem (TOHIDIRAD et al., 2015), por meio de simulação com a utilização do *CloudSim*. Os algoritmos comparados foram o algoritmo de prioridade, *First Come First Served* (FCFS) e *Round-Robin* (RR). O algoritmo de prioridade teve melhor desempenho enquanto o FCFS e RR tiveram desempenhos proporcionais.

SINDHU; MUKHERJEE apresenta em (SINDHU; MUKHERJEE, 2011) uma comparação de três (3) algoritmos de escalonamento de tarefas para Computação em Nuvem (FCFS, *Longest Cloudlet Fastest Processing Element* (LCFPE) e *Shortest Cloudlet Fastest Processing Element* (SCFPE)). A avaliação se deu por meio de simulação no *CloudSim*, sendo que quando o número de tarefas aumenta, o LCFPE exibe uma performance melhor quando comparado com o SCFPE e FCFS, desde que tarefas maiores completem antes, assim, reduzindo o tempo entre o início e o final da execução.

Com exceção do trabalho apresentado por NETTO; BUYYA em (NETTO; BUYYA, 2009), as metodologias usualmente empregadas para avaliação de estratégias de escalonamento ocorrem utilizando ferramentas de simulação. A precisão obtida utilizando tal metodologia é altamente dependente da qualidade do modelo de simulação implementado, sendo que a validação da ferramenta de simulação propriamente dita extrapola o contexto dos trabalhos citados. Em (SANTOS, 2016), a estratégia de validação dos resultados se deu pela comparação entre os resultados obtidos pelo estratégia de Consolidação de Tarefas com a execução real das cargas de trabalhos consolidadas em MVs sobre uma infraestrutura real. Esta validação foi limitada tanto pelo número de nós disponíveis na infraestrutura empregada nos experimentos quanto ao tempo a ser utilizado.

Neste trabalho, são reutilizados testes na infraestrutura física apresentado por SANTOS em (SANTOS, 2016) e, a partir da modelagem da infraestrutura no simulador, escalar-se as tarefas BoT para um número maior de nós de computação simulados, assim, pode-se representar de forma mista o método de execução real e o de simulação em um mesmo gráfico. Desta forma, este trabalho se aproxima de (LOGANATHAN; MUKHERJEE, 2015) no que diz respeito a utilização de métodos de execução real e de simulação mas diferenciando-se em relação ao tamanho e como a infraestrutura é utilizada com o método de execução real e como serão apresentados os resultados dos métodos de execução real e de simulação.

4 AVALIAÇÃO MISTA DE BOT EM AMBIENTE DE NUVEM

Na maioria dos casos, o usuário não tem acesso a grandes infraestruturas de computação no momento de realizar um projeto de implementação de uma aplicação em ambiente de Nuvem. No entanto, não raro, desenvolvedores dispõem de uma estrutura de pequeno porte para prototipação e experimentações preliminares. A dificuldade está em transpor os resultados nesta infraestrutura limitada para uma de maior porte, onde se dará a execução real da aplicação e assim identificar a real necessidade de recursos de processamento.

O cenário delineado, portanto, é aquele em que uma plataforma física limitada sirva como base para a experimentação, sendo a escalabilidade da aplicação avaliada por um processo complementar baseado em simulação. A proposta deste trabalho delimita-se em prover a utilização de um modelo de aplicação em que o usuário consiga descrever seu problema e, com o apoio de mecanismos de simulação, realizar o dimensionamento das necessidades de processamento de sua aplicação.

4.1 Arquitetura do Método de Avaliação Mista

O estudo proposto para dimensionamento da infraestrutura necessária a uma aplicação encontra-se esquematizado na Figura 6. Em um primeiro momento, o desenvolvedor dimensiona sua aplicação no modelo de descrição de BoTs proposto em (SANTOS, 2016). Esta descrição de aplicação serve como entrada para os três métodos de avaliação explorados neste trabalho:

- **Dimensionamento pela consolidação das tarefas em MVs.** Em (SANTOS, 2016), é apresentado um escalonador em nível aplicativo (consolidação do BoT em MVs), que escalona previamente as tarefas antes de submeter para execução na Nuvem. Este método tem como entrada, além da descrição do BoT, o número de recursos de processamento, nós e/ou núcleos, disponíveis para cálculo e o número de MVs a serem utilizadas. Como saída, é fornecida a consolidação (ou agrupamento) das tarefas nas MVs de forma a ocupar os recursos de processamento disponíveis conforme a estratégia de escalonamento aplicada e o

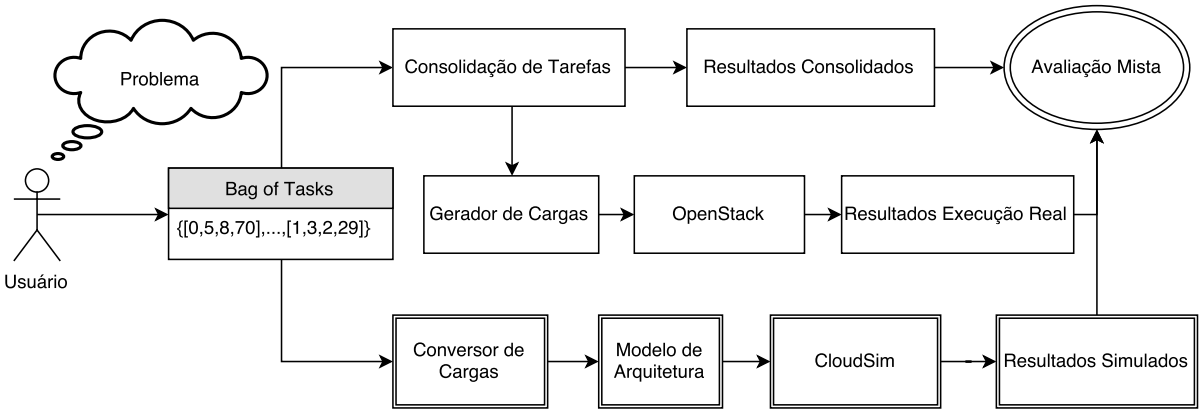

Figura 6 – Diagrama de desenvolvimento.

tempo de término da execução (*makespan*). Dada a rapidez promovida pelo método, diferentes estratégias de escalonamento podem ser aplicadas à diferentes combinações de quantidades de recursos de processamento e MVs.

- **Execução na infraestrutura física de prototipação.** A saída do processo de consolidação de tarefas em MVs gera um arquivo de registros (*log* de execução) de cargas a serem executadas pelas diferentes MVs. Este arquivo permite a geração de MVs responsáveis por executar as cargas computacionais conforme descritas no *log* produzido pelo método de consolidação de MVs. Estas MVs são submetidas à execução na Nuvem, no caso deste trabalho, gerenciado pelo *OpenStack*. A saída obtida é também o tempo total de execução. Como trata-se de uma execução real das cargas computacionais geradas, o tempo de execução é aquele determinado pelo BoT. Este aspecto limita a quantidade de estudos de caso que podem ser de realizados neste método.

- **Execução na infraestrutura simulada.** De forma semelhante ao método que explora a execução em ambiente de prototipação, o método de obtenção de resultados de desempenho por simulação realiza a geração de cargas computacionais para MVs. Estas MVs são instanciadas e geridas em uma Nuvem simulada. Neste trabalho foi utilizado o *CloudSim* como ferramenta de simulação. Este método permite obter resultados de desempenho, refletidos como tempo de execução, com grande dinamicidade.

Os resultados de desempenho providos pelos diferentes métodos oferecem diferentes perspectivas de análise, permitindo ao desenvolvedor caracterizar as necessidades de infraestrutura em função do estudo realizado sobre os dados disponíveis. Um cuidado a ser tomado é que qualquer BoT modelado pode ser submetido à qualquer um destes métodos. No entanto, como a infraestrutura disponível é restrita em termos de capacidade, a este método não devem ser submetidos BoTs que ultrapassem sua capacidade de processamento. Desta forma, o desenvolvedor pode dispor

apenas do conjunto de dados completos para os casos em que o BoT descrito requeira uma carga computacional compatível com a disponível na infraestrutura real. Nos casos em que a demanda computacional do BoT suplantar a oferecida, o desenvolvedor poderá dispor apenas dos resultados de desempenho fornecidos pelos métodos de consolidação em MVs e por simulação.

O problema, então, recai em como equalizar os dados obtidos de forma que o desenvolvedor tenha garantias que os resultados de desempenho fornecidos pelos métodos de consolidação de tarefas e de simulação venham a ser refletidos em uma arquitetura real de maiores dimensões do que aquela utilizada na prototipação. O estudo de desempenho do BoT deve contemplar uma análise de escalabilidade da aplicação.

Na condução do processo de estudo do dimensionamento de uma infraestrutura para execução de uma determinada aplicação, os custos computacionais do BoT que a descreve devem ser escalados para infraestrutura de prototipação disponível. O estudo deve considerar os diferentes métodos de avaliação descritos, compatibilizando o tamanho do BoT a infraestrutura de prototipação disponível. O conjunto de passos a serem seguidos no estudo de dimensionamento dos recursos de processamento são os seguintes:

1. Modelar a aplicação em termos de um BoT.
2. Identificar a configuração ideal, consolidando as tarefas do BoT em diferentes quantidades de recursos de processamento (número de MVs e núcleos/nós) de forma à identificar aquela que resulte no menor tempo total de execução.
3. Proceder a coleta de dados de desempenho por simulação do BoT em uma infraestrutura que represente aquela identificada no Passo 2.
4. Desenvolver um novo BoT, chamado BoT' que reproduza a demanda de processamento do BoT original proporcional à ofertada pela infraestrutura de prototipação.
5. Consolidar as tarefas do BoT' em uma configuração que reproduza a capacidade de processamento disponível na infraestrutura de prototipação.
6. Obter o tempo total de execução de BoT' na infraestrutura de prototipação.
7. Proceder a coleta de dados de desempenho por simulação do BoT' em uma infraestrutura que represente aquela identificada no Passo 2.
8. Proceder a coleta de dados de desempenho por simulação do BoT' em uma infraestrutura que represente aquela identificada no Passo 4.

O estudo do dimensionamento da infraestrutura ideal passa pela análise dos resultados obtidos para as consolidações e simulações do BoT original e do BoT' e da execução de BoT'. Neste ponto, diferentes fatores, além do tempo de execução, devem ser considerados pelo desenvolvedor, como o custo de aquisição e aluguel de máquinas. O aspecto tratado neste trabalho é como obter o BoT', conforme o Passo 4, representando os custos computacionais BoT original de forma proporcional aquelas disponíveis na infraestrutura de prototipação.

Considere que um determinado BoT A, descrito por n quádruplas no formato $A = \{[a_1, d_1, b_1, c_1], [a_2, d_2, b_2, c_2], \ldots [a_n, d_n, b_n, c_n]\}$, onde:

- ***a***: é o instante de tempo no qual o conjunto de tarefas descrito pela quádrupla chega para o processamento;
- ***d***: é o tempo de duração do grupo de tarefas;
- ***b***: é a quantidade de tarefas descritas por quádrupla correspondente; e
- ***c***: é a taxa de utilização de CPU para cada tarefa da quádrupla. O valor informado para o custo deve estar entre 1 e 100, indicando o percentual de uso da CPU de cada uma das tarefas quando em execução.

Dado que A executa em tempo mínimo t_m em uma determinada infraestrutura M_p, composta por p unidades de processamento idênticas, na qual cada unidade de processamento é capaz de executar I instruções por segundo, é possível obter um BoT A' tal que $A' = \{[a_1, d_1, f(b_1, p), g(c_1, I)], [a_2, d_2, f(b_2, p), g(c_2, I)], \ldots [a_n, d_n, f(b_n, p), g(c_n, I)]\}$. A expectativa para tempo de execução t'_m previsto para A' é $t'_m \approx t_m$.

4.2 Etapas e Módulos Desenvolvidos

A Figura 6 também apresenta os módulos que foram desenvolvidos neste presente trabalho, destacados com linhas de borda dupla. A partir do momento em que o usuário consegue modelar seu problema para o padrão BoT, temos a seguinte sequência de passos e implementação:

- **Conversor de Cargas**. De posse do BoT modelado, o *script* desenvolvido foi o para transformação desse BoT em um arquivo JSON que serve para utilização na entrada do simulador. Como o *CloudSim* necessita que as cargas computacionais das MVs sejam descritos em termos de número de instruções a serem executadas, este módulo converte para tal, a descrição dos custos computacionais das tarefas, descritos em termos de tempo de execução e percentual de

uso de CPU. A duração da tarefa em unidades de instruções é obtido pela multiplicação da duração das tarefas (número de passos de execução $\times$ número de segundos definidos para o passo) pelo valor de bogoMIPS que o processador disponibiliza por segundo; o número de tarefas é mantido.

- **Modelo de arquitetura**. Este módulo representa a saída do Conversor de Carga. Esta saída é composta por dois arquivos JSON. O primeiro arquivo contém a descrição do BoT com os custos computacionais requeridos para sua execução sobre o *CloudSim*. O segundo arquivo consiste na descrição da infraestrutura física sobre a qual a simulação deve ser realizada no *CloudSim*. Este arquivo contém todas as informações necessárias para se criar a infraestrutura virtual, como, informações de quantos *hosts* (e suas características como número de núcleos, memória RAM, etc), informações de quantas MVs serão lançadas no *datacenter* também com suas características etc.

- ***CloudSim***. Para a execução com o *CloudSim*, foram criadas algumas classes para o tratamento do BoT. Uma dessas classes transforma os termos do BoT em *cloudlets*, as quais serão lançadas nas MVs dentro da infraestrutura simulada. Cada rodada de simulação utiliza arquivos JSON com características de infraestrutura e BoTs diferentes, conforme o caso simulado. Toda simulação gera um relatório de execução para verificar se todas as MVs foram criadas, para qual MV foi cada *cloudet* e o tempo final de execução.

- **Avaliação Mista**. De posse dos resultados consolidados, dos resultados da execução real e dos resultados simulados, partimos para a avaliação mista. Nessa etapa os resultados recolhidos são comparados e analisados, de forma a verificar se a execução na infraestrutura simulada é compatível com a execução na infraestrutura real.

Importante observar que a transformação do tempo em uma unidade de execução representada pelo número de instruções também é realizada na simulação da carga sobre a infraestrutura real. O módulo Gerador de Carga, utilizado por SANTOS em (SANTOS, 2016) e indicado na Figura 6, é responsável por realizar esta transformação, também descrevendo a carga computacional das MVs em termos de bogoMIPS e indicando a taxa de ocupação da CPU para execução do número de instruções encontrado de forma que a MV ocupe a CPU física durante o tempo determinado pela tarefa com a taxa de ocupação que a ela foi definida.

A partir da validação do modelo no *CloudSim*, podemos escalar a infraestrutura, saindo dos nove (9) nós, passando para cem (100) nós até chegarmos em mil (1.000) nós de processamento. Os resultados comparados e a avaliação mista bem como um teste de escalabilidade de infraestrutura e BoT, estão presentes no Capítulo 5.

5 EXECUÇÃO E RESULTADOS

O presente capítulo apresenta uma avaliação mista, o qual se utiliza de experimentação, apresentada por SANTOS em (SANTOS, 2016) e simulação.

5.1 Metodologia de Execução e Extração dos Resultados

A obtenção dos resultados utilizados/apresentados neste trabalho se deram nas seguintes etapas:

- Modelagem da infraestrutura real, apresentada em por SANTOS em (SANTOS, 2016), sendo esta descrição para ser implementada no *CloudSim*.
- Validação do modelo de infraestrutura comparando os resultados obtidos com aqueles apresentados em (SANTOS, 2016) por SANTOS. Mais precisamente reexecução dos testes na infraestrutura física e execução na infraestrutura simulada.
- Análise da escalabilidade dos BoTs em função do incremento de recursos de processamento por simulação.
- Comparação, por meio de gráficos, dos resultados obtidos como produto da execução da simulação executada pelo *CloudSim* e da execução no ambiente real gerido pelo *OpenStack*.

5.2 Infraestrutura Real e Infraestrutura Simulada

Para a execução dos experimentos, foi utilizado um *cluster* formado por dez (10) nós físicos, um dos quais dedicado ao controle, necessário em função do uso de *OpenStack* e nove (9) nós para processamento. Cada nó possui um processador *Intel Core* i5-2310@2.90GHz com quatro (4) núcleos físicos, 16GB de memória RAM e uma interface de rede um (1) GbE; o nó controlador possui duas interfaces de rede. A capacidade de processamento de cada núcleo foi aferida, com base em consulta do arquivo "cpuinfo", em 5.786 bogoMIPS. A versão do *OpenStack* utilizada para testes

foi a *Kilo*, com os seguintes módulos instalados: *Keystone* (serviço de identidade e autenticação), *Glance* (serviço de imagens de sistema e sistemas operacionais), *Nova-compute* (serviço de computação e instanciamento de MVs), *Nova-network* (serviços de rede) e *Horizon* (interface *web* para administração). O *OpenStack* foi configurado para realizar o escalonamento padrão, o *Filter*, que atribui aos nós físicos as MVs no momento em que estas são criadas considerando a carga nos nós, funcionando de forma semelhante ao RR.

Para a execução da simulação, foi utilizado o *CloudSim*, em sua versão 4.0.4. Ele foi executado sobre uma máquina com as seguintes configurações: *Intel Core* i3-M380@2.53GHz com dois (2) núcleos físicos e 8GB de memória RAM, rodando o *Windows* 10 e utilizando o *Java* 8 *Update* 121, ambos na arquitetura x64. A arquitetura simulada é composta por nós idênticos àqueles disponíveis na infraestrutura física disponível, ou seja, nós *quad-core*, 16GB de memória RAM, e provendo 5768 bogoMIPS por núcleo.

5.3 Especificação de BoTs

Os BoTs utilizados à fim de validar a infraestrutura simulada frente à infraestrutura física foram os mesmos utilizados por SANTOS em (SANTOS, 2016), são especificados no Capítulo 4.

Para a escalabilidade das tarefas, estipulou-se BoT com cargas de CPU pequenas, médias e grandes, no mesmo formato utilizado em (SANTOS, 2016).

Mesmo que o modelo de BoT preva a reinjeção de tarefas nos BoTs, neste trabalho não é empregada.

5.4 Validação do Modelo de Simulação

Para validar o modelo de simulação concebido, foram replicados experimentos apresentados em (SANTOS, 2016) e realizados experimentos no simulador, utilizando como entrada o mesmo conjunto de BoTs. O objetivo é verificar se os tempos de simulação são compatíveis com a execução real, no *cluster* disponível e com o tempo que foi obtido por meio do Consolidação de Tarefas.

As mesmas hipóteses de análise apresentadas em (SANTOS, 2016) foram utilizadas, e são:

Hipótese 1 : Dado um BoT descrito por tarefas com baixo consumo de CPU, havendo uma quantidade suficiente de recursos físicos de processamento, o tempo total de execução real deve ser igual ao tempo de execução em uma arquitetura não contingenciada.

Os BoTs gerados para validar esta hipótese estão apresentados na sequência:

- $H1$-1 = {[0,5,3,24],[3,6,7,30]}
- $H1$-2 = {[0,9,4,38],[1,14,3,23],[4,8,5,19]}
- $H1$-3 = {[0,11,5,18],[2,4,8,15],[5,6,4,29],[6,4,2,25]}

Percebe-se que, de acordo com a Figura 7, os resultados obtidos por meio de simulação (barras verdes) confirmam a Hipótese 1 pois ficaram dentro do esperado, na comparação com os resultados obtidos pelo escalonamento em nível aplicativo (barras azuis) e executados na infraestrutura real (barras laranjas). Importante observar que, os resultados obtidos na execução real indicam tempos maiores àqueles previstos pelo modelo de escalonamento apresentado em (SANTOS, 2016) e os obtidos no modelo simulado neste trabalho. Este resultado é o esperado, uma vez que a execução real adiciona sobrecustos de operações de escalonamento, como por exemplo, a migração de MVs, não consideradas nos modelos de escalonamento da simulação.

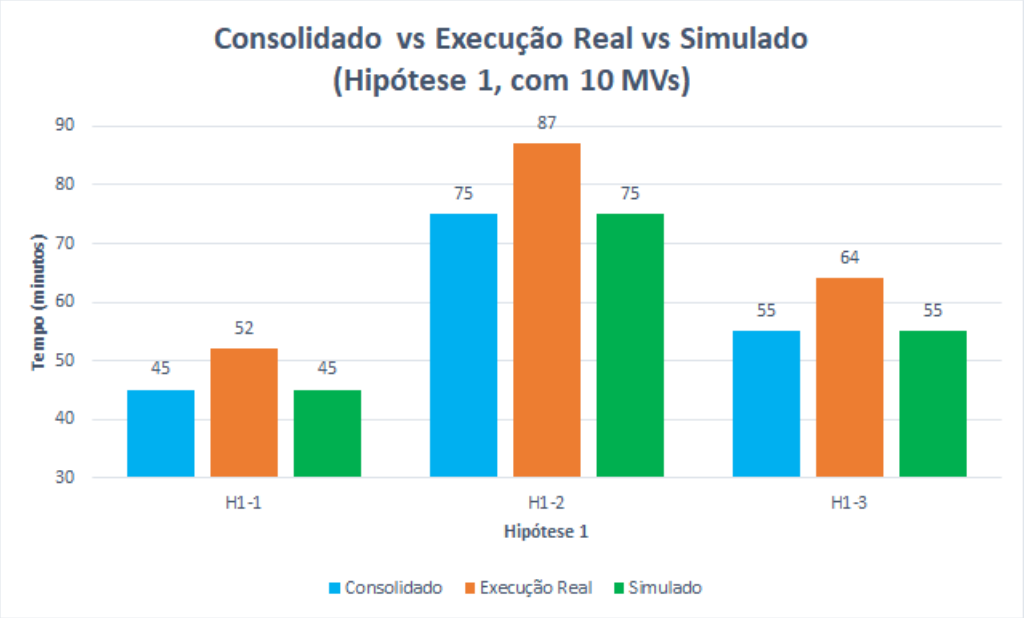

Figura 7 – Comparação da Hipótese 1.

Hipótese 2 : Dado um BoT descrito por tarefas com alto consumo de CPU, havendo uma quantidade insuficiente de recursos físicos de processamento, o tempo total de execução real será maior que o tempo de execução em uma arquitetura não contingenciada.

Foram gerados três BoTs para validar esta Hipótese. Estes BoTs são dados por:

- $H2$-1 = {[0,5,3,84],[3,6,7,90]}
- $H2$-2 = {[0,3,5,78],[2,4,5,75],[3,2,4,80],[3,3,8,60],[5,5,4,69],[6,4,2,65]}

- $H2\text{-}3$ = {[0,5,10,68],[1,4,5,73],[2,6,6,83],[2,2,3,65],[4,3,5,65]}

Novamente, percebe-se que, de acordo com a Figura 8, os resultados obtidos por meio de simulação (barras verdes) confirmam a Hipótese 2 pois ficaram dentro do esperado, na comparação com os resultados obtidos pelo escalonamento em nível aplicativo (barras azuis) e executados na infraestrutura real (barras laranjas). Uma ressalva é feita aos resultados de simulação referentes à hipótese H2-3. Neste caso, o limite de MVs provisionadas pelo *CloudSim* em função da infraestrutura é de 20 MVs. Esse aspecto ressalta o objetivo do experimento que é o de estressar a capacidade de processamento disponível. Nos demais métodos este limite não é imposto.

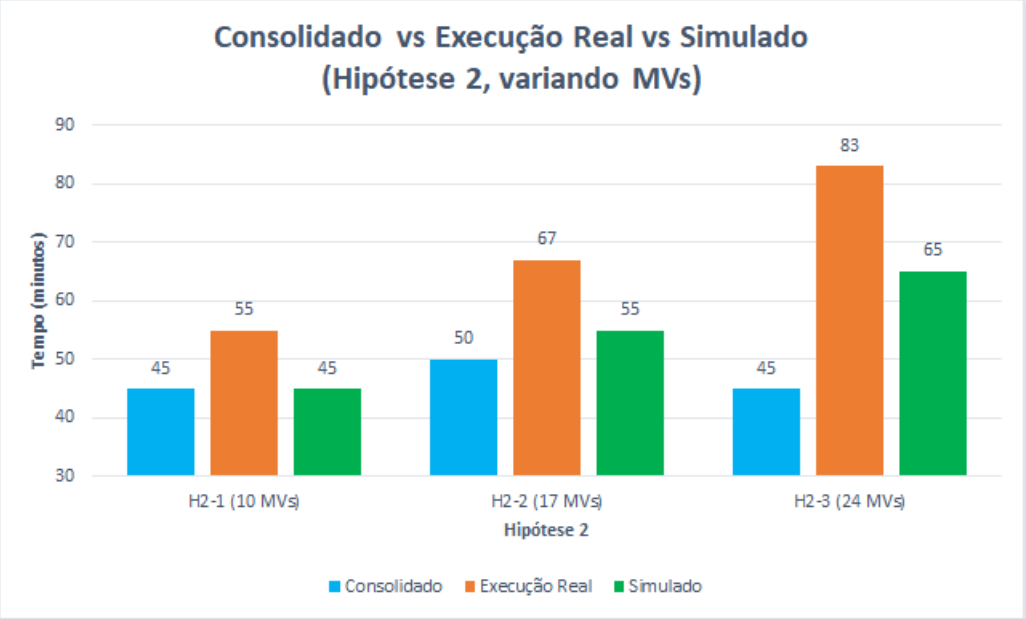

Figura 8 – Comparação da Hipótese 2.

Para termos maior certeza do comportamento correto da infraestrutura virtual frente a infraestrutura real disponível, um terceiro conjunto de testes considerou o Uso em Produção. Neste caso, um conjunto de BoTs é lançado à execução onde cada BoT tem seu tempo de chegada definido. Conforme estes BoTs vão sendo lançados, são executados paralelamente. Podemos perceber, ao analisar o gráfico apresentado na Figura 9, que a simulação (barras verdes) apresenta desempenho muito próximo ao executado pelo escalonamento em nível aplicativo (barras azuis) e sempre mais baixo que a execução real. Os BoTs utilizados estão descritos no Anexo A.

5.5 Escalabilidade de Tarefas

Na Figura 10 são apresentados os tempos de execução obtidos por simulação de diferentes BoTs sobre uma arquitetura idêntica àquela disponível para experimentação. Conforme destacado em cada gráfico desta figura, são variados o tempo de

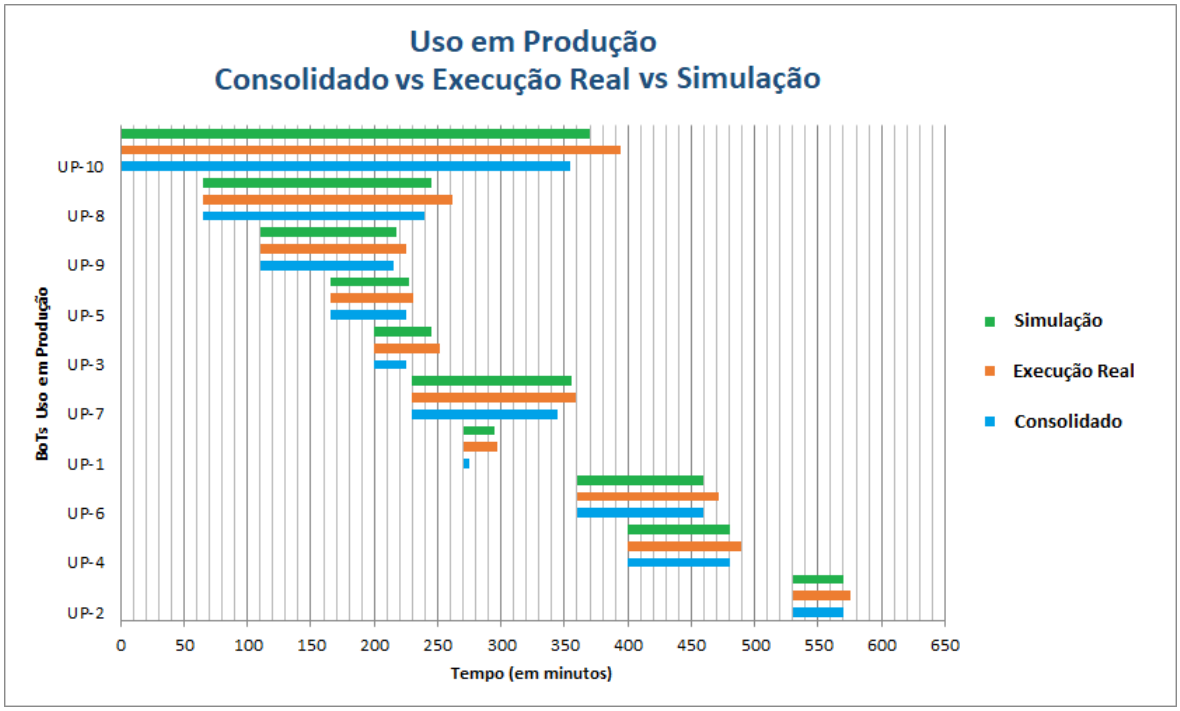

Figura 9 – Comparação dos usos em produção.

execução das tarefas individualmente e o intervalo entre submissão de duas tuplas, considerando diferentes custos computacionais das tarefas. Os BoTs gerados e utilizados estão descritos no Apêndice A. São consideradas duas classes de tempo, curto e longo, tanto para a duração das tarefas como para o intervalo de submissão entre tuplas. O uso de CPU é caracterizado em duas classes: baixo e alto. Para cada caso, foram considerados diferentes comprimentos de aplicação, com 10, 25, 50 e 100 quádruplas. Os pontos em destaque caracterizam o número de MVs utilizadas para execução da aplicação foram 1, 2, 4, 8, 16, 32, 64 e 128.

Os resultados de desempenho nos gráficos da Figura 10 mostram que o comprimento das tarefas e o intervalo entre submissão de quádruplas tem efeito direto no tempo total de execução. Também é observado que há um limite de desempenho a ser atingido, e este limite é dependente do paralelismo que pode ser obtido pelo número de MVs utilizados e do número de tarefas que podem ser executadas em paralelo.

Para fins de avaliação de desempenho, tanto o intervalo entre submissões de BoTs como a duração das tarefas será mantido constante, uma vez que foi mostrado que estes possuem uma relação direta com as variações no tempo total de execução. O aspecto a ser estudado é a granularidade da computação, considerando a relação entre o número de tarefas e o consumo total de CPU oferecido pela infraestrutura. O cenário é aquele em que existe um BoT a ser lançado e diferentes configurações de infraestrutura estão disponíveis. O usuário da infraestrutura, ao submeter o BoT pode avaliar se o custo total de execução deverá ser particionado em um número maior ou menor de tarefas.

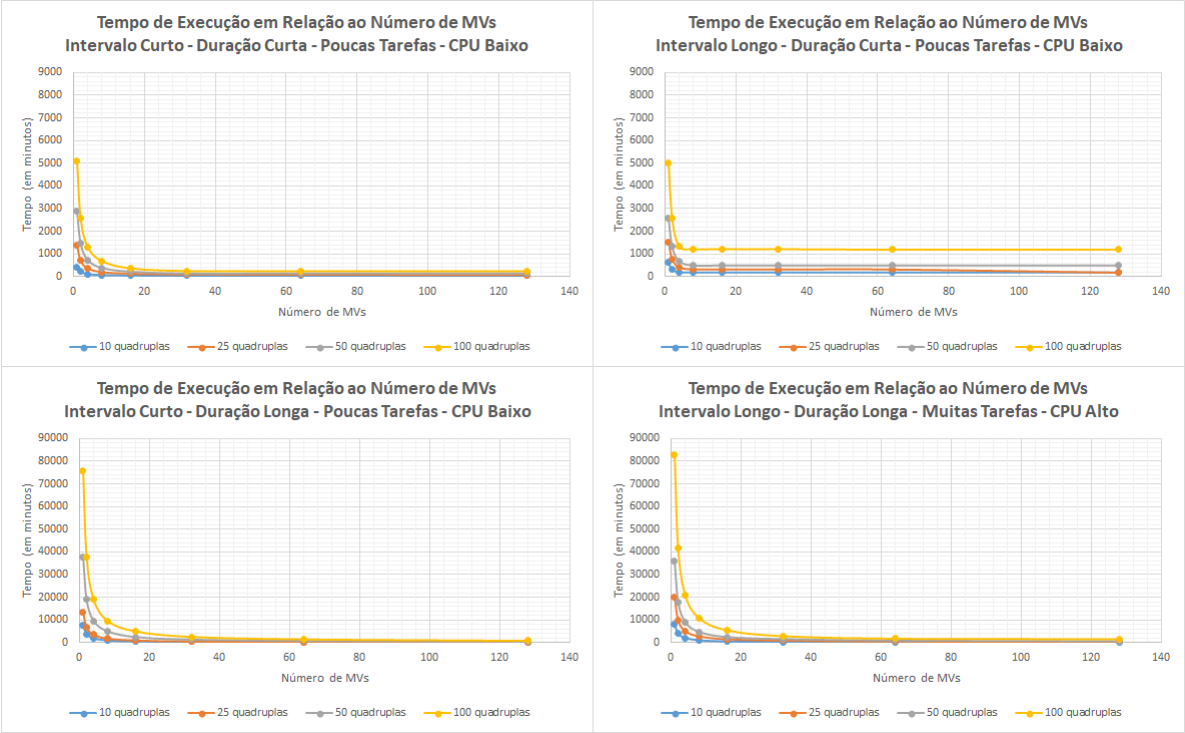

Figura 10 – Conjunto de gráficos que mostram a inviabilidade da comparação variando o intervalo de chegada e duração das tarefas do BoT.

Para efeitos de determinação dos parâmetros, o número de bogoMIPS da arquitetura utilizada serve como referencial de uso de CPU. Na arquitetura física disponível, o valor é de 5768 bogoMIPS em cada núcleo de cada nó de processamento. Isto significa que, em um (1) minuto, são executadas 5768 * 60 = 346.080 bogoMIPS por núcleo. Em 24 horas de processamento, este núcleo disponibiliza 498.355.200 bogoMIPS. Na arquitetura disponível, havendo nove (9) nós com quatro (4) núcleos, o total de processamento obtido em 24 horas é de 17.940.787.200 bogoMIPS.

Generalizando, a carga total de processamento disponível em 24 horas (ou 86.400 segundos) C^d em uma arquitetura homogênea com p nós de c *cores*, cada um oferecendo b bogoMIPS, é dada por:

$$C^d = 86.400s * b * p * c$$

Para efeitos de experimentação, C^d foi utilizado como parâmetro para determinar o custo computacional de um BoT. Um BoT é considerado leve quando descrever uma carga de processamento até 50% da carga diária disponibilizada, mediano entre 50 e 80% e pesado acima 80%.

Em relação ao número de tarefas, este valor também é definido em função das características da arquitetura utilizada para efeitos de referência. Considerando que uma arquitetura homogênea composta por p nós de c núcleos é capaz de executar $p * c$ atividades simultâneas, determina-se o tamanho de um BoT, considera-se um

BoT pequeno quando o número de tarefas por quádrupla for inferior à 50% de $p * c$, mediano quando o número de tarefas por quádrupla estiver entre 50 e 80% e longo quando acima 80%.

A Tabela 3 e Figura 11 apresentam os dados de entrada no experimento. São consideradas nove combinações entre número de tarefas e custo computacional de BoTs em três configurações de infraestrutura, com 9, 100 e 1.000 nós *quad-core*. São considerados BoTs com um número de tarefas pequenos, médios e grande, com consumo de CPU igualmente pequeno, médio ou grande conforme descritos na Tabela 3. Estes tamanhos definidos em conforme descrito anteriormente, sendo assumido os valores conforme a Tabela 3. Os custos computacionais por tarefa encontram-se identificados na Figura 11.

Tabela 3 – Parâmetros para análise de BoTs

Classe	#tarefas para #nós			Utilização CPU
	9	100	1000	
Pequeno	18	200	400	50%
Médio	25	280	2800	70%
Grande	36	400	4000	100%

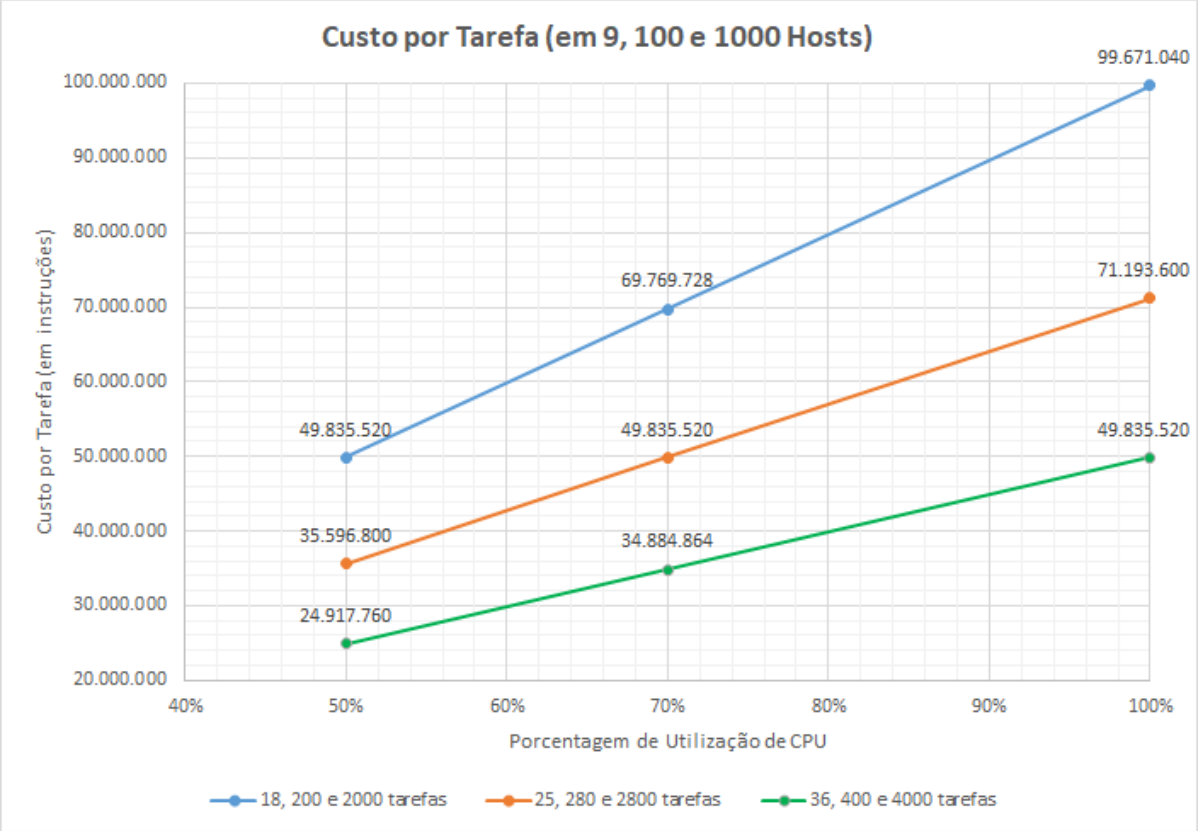

Figura 11 – Custo por tarefa em relação à utilização de CPU.

Um aspecto a ser observado é que, independente da capacidade de processamento da infraestrutura (nós e núcleos) a carga computacional atribuída a cada tarefa é constante. Isto deve-se à metodologia de determinação de carga dos BoTs, realizada em função da carga de processamento oferecida em 24 horas pela infraestrutura. O que varia é a carga total do BoT em função dos recursos disponíveis.

Os BoTs gerados conforme caracterizado na Tabela 3 e na Figura 11 foram executados sob o simulador *CloudSim*. No primeiro caso foi considerada uma infraestrutura totalmente dedicada à execução dos BoTs. Sabendo que as demandas dos BoTs correspondem à 50, 70 e 100% da capacidade oferecida pela infraestrutura, os tempos de execução obtidos refletem exatamente o comportamento esperado, tendo sido observados tempos de execução de 12h, 16h e 48min e 24h, respectivamente.

Um segundo experimento considerou a execução dos BoTs em uma infraestrutura compartilhada. Nesta situação, além do BoT em execução, a infraestrutura é submetida a uma carga externa que consome também consome seus recursos. Foram considerados dois casos, consumo externo de 50 e 75%, sendo os tempos de execução obtidos descritos na Tabela 4.

Tabela 4 – Tempo de execução dos BoTs com sobrecarga externa (9, 100 e 1.000 nós)

BoT	Carga externa		
	0%	50%	75%
Pequeno	12h	24h	48h
Médio	16h48min	33h36min	67h12min
Grande	24h	48h	96h

Pode-se perceber que o tempo necessário para a finalização da execução na infraestrutura que está com 50% de carga externa, os BoTs demoraram o dobro do tempo de quando tem a infraestrutura ociosa, bem como o tempo quadruplica quando a infraestrutura está com 75% de carga externa.

Não foram realizados experimentos repetitivos, com finalidades estatísticas, devido ao fato da execução do simulador ser determinística, ou seja, o valor da saída não é alterado se a entrada se manter. Isso inclui também a execução em outro computador. A saída continua a mesma, se a entrada se manter, mesmo em um processador melhor.

5.6 Estudo de Caso

Após a validação da modelagem da infraestrutura real no simulador, podemos partir para um caso de estudo onde possamos escalar uma infraestrutura e um BoT. Um BoT' foi gerado, nas mesmas características apresentadas em (SANTOS, 2016), para

executar em aproximadamente 2 horas, na infraestrutura física disponível, ou seja, nove (9) nós de processamento com 36 núcleos e também, na arquitetura simulada, com essas mesmas características. Esse BoT foi mapeado em BoT' para executar, também, em aproximadamente 2 horas em uma infraestrutura simulada escalada para 125 nós de processamento com 500 núcleos, no simulador. Estes BoTs também passaram pela consolidação de tarefas. O BoT foi escalado em BoT', apenas no número de tarefas, mantendo o mesmo custo computacional das tarefas. Os resultados da execução, simulação e consolidação estão sumarizadas na Tabela 5. BoT e BoT' gerados e utilizados estão no Apêndice B.

Tabela 5 – Tempos de execução de um BoT em 9 *hosts* físicos e simulados; tempo de execução deste BoT escalado em BoT' para execução em 125 *hosts*; tempo dado pelo consolidação de tarefas para BoT e BoT' (tempo em minutos)

	Execuções			Consolidações	
Infraestruturas	Execução Real	Simulação		-	
BoTs	BoT	BoT	BoT'	BoT	BoT'
Tempo de Execução	139	130	130	120	120

Podemos perceber que, realizando a consolidação de tarefas, o tempo foi de 120 minutos tanto para o BoT quanto para o BoT' escalado. A execução do BoT na infraestrutura física com 9 *hosts* e 36 núcleos, com 36 MVs, ficou acima do tempo consolidado por volta de 16%. A execução do mesmo BoT na infraestrutura virtual, também com 9 *hosts* e 36 núcleos, com 36 MVs, ficou acima do tempo consolidado em torno de 8%. O BoT' foi executado na infraestrutura simulada, com 125 *hosts* e 500 núcleos, com 500 MVs. O tempo de execução se manteve em torno de 8% acima do tempo consolidado.

A partir deste estudo, podemos concluir que o tempo de execução na infraestrutura simulada é menor que na infraestrutura real, porém, ela não leva em consideração o *overhead* do sistema da infraestrutura real, como comunicação por rede e migrações de MVs. A simulação em uma infraestrutura simulada se apresenta eficaz, pelo fato de demorar menos para executar e apresentar valores semelhantes ao simulado na infraestrutura real.

5.7 Síntese dos Resultados e Considerações sobre o Capítulo

Este capítulo mostra que os objetivos foram alcançados. Desde a modelagem da infraestrutura física disponível para a infraestrutura virtual no *CloudSim*, passando pela equiparação de tais infraestruturas até chegar no dimensionamento de tarefas e infraestrutura.

Na avaliação realizada, foi estabelecida uma relação entre o custo computacional total de um BoT e a quantidade de processamento disponibilizada pela infraestrutura em 24 horas. Desta forma foi possível dimensionar BoTs com custo computacional pequeno, médio e grande em função do processamento disponível. Também foi definido o comprimento, em relação ao número de tarefas, em função do número de atividades concorrentes disponibilizadas pela infraestrutura, igualmente definidas em pequena, média e grande.

Alterando combinações entre custo computacional e número de tarefas, foi analisado o impacto na execução, considerando diferentes relações de granularidade. Outro ponto positivo foi observar que o sistema se manteve estável durante a execução bem como seus resultados.

Diminuir a granularidade, ou seja, ter tarefas menores, faz com que o escalonador realize melhor o escalonamento, alocando melhor os recursos disponíveis. E tarefas que são consideradas grandes para infraestruturas pequenas, podem ser medianas em infraestruturas um pouco maiores e ao mesmo tempo, pequenas para infraestruturas grandes, sendo estas homogêneas entre si.

6 CONCLUSÃO

A Computação em Nuvem vem apresentando um crescimento extraordinário nos últimos anos, em questão de quantidade e variedade dos serviços oferecidos. Isso traz junto, uma determinada onipresença no cotidiano na maioria das pessoas, pois estas convertem-se em usuários que, tornam-se dependentes de certos serviços, tais como *email*, *backups online*, redes sociais, etc, necessitando apenas um acesso à *internet* e um navegador.

Outra classe de usuários buscam utilizar a Computação em Nuvem para atividades mais específicas, na maioria das vezes, procurando por alta disponibilidade de processamento. Estas atividades podem ser desde buscar uma alternativa à construção e manutenção de uma infraestrutura privada até um dimensionamento de uma possível infraestrutura, porém, virtualizando, para ter a certeza que a infraestrutura planejada não seja subdimensionada ou superdimensionada para sua demanda.

Neste trabalho foi apresentado um modelo de simulação mista, ou seja, que se utilizou de resultados executados em uma infraestrutura física limitada em número de *hosts*, processadores e núcleos de processamento e, resultados simulados em uma infraestrutura virtual modelada nas mesmas características da infraestrutura física e depois escalados no que se é limitado na mesma.

As execuções na infraestrutura física se utilizou da plataforma *OpenStack* e de BoTs e *scripts* apresentados em (SANTOS, 2016). As simulações foram realizadas com o auxílio do *framework* CloudSim. Para esta, foram desenvolvidas classes específicas para o tratamento dos BoTs e para a criação da infraestrutura, criando arquivos JSON.

Pudemos perceber que, pelos testes de comparação de tempos de execução, a infraestrutura simulada apresentou resultados bem próximos ao escalonamento em nível aplicativo e ficando sempre abaixo da execução na infraestrutura física.

Quando escalada, a infraestrutura permaneceu homogênea e o custo das tarefas continuou o mesmo para ambas. Pôde-se perceber que com a diminuição da granularidade (tarefas menores) porem com mesmo custo, foram melhor alocadas pelo escalonador. Quando a infraestrutura ficava ociosa e só esperava aquelas tarefas a

serem executadas, apresentaram certo tempo de processamento. Nestes casos, com as características citadas no Capítulo 5, um BoT pequeno, médio e grande demoram 12h, 16h48min e 24 horas, respectivamente. Quando a infraestrutura oferece 50% de carga externa, ou seja, a infraestrutura já está ocupada por outras atividades, demora o dobro do tempo, em todos os casos. Da mesma forma, quando a infraestrutura apresenta 75% de utilização externa, as mesmas tarefas demoram quatro (4) vezes mais para executar.

Também, pôde-se perceber que BoTs classificados como grandes para uma infraestrutura dita pequena, podem ser considerados médios em um infraestrutura um pouco maior e, consequentemente, julgados como pequeno numa infraestrutura dita grande.

A Computação em Nuvem é uma área da computação que sempre demandará pesquisa, pois suas tecnologias estão cada vez mais presentes no cotidiano pessoal como no de grandes empresas e, a satisfação dos clientes enquanto usam tais serviços e a otimização dos recursos de quem oferece esses serviços, sempre serão objetivos primordiais.

Na sequência deste trabalho, o objetivo será definir um modelo de cargas no qual possam ser especificados comprimentos e custos de BoTs de forma geral.

Além disso, este trabalho apresenta um vasto leque de opções para dar continuidade. Por exemplo, descrever outros tipos de infraestrutura, podendo estas serem homogêneas ou heterogêneas, a fim de verificar se as tarefas tendem a ter o mesmo comportamento apresentado neste trabalho.

Outras direções a serem tomadas seriam a de utilizar outros modelos de escalonamento, como os citados no Capítulo 2 para comparação tanto na infraestrutura real disponível quanto numa infraestrutura modelada no *CloudSim*.

Também é interessante a utilização de outros tipos de tarefas de conceito diferente de BoT, como o *Bulk Synchronous Parallel* (BSP) (VALIANT, 1990), que difere do BoT devido ao fato das tarefas terem possível comunicação umas entre as outras.

REFERÊNCIAS

ANDREWS, G. R. **Foundations of Multithreaded, Parallel, and Distributed Programming**. 1.ed. University of Arizona, USA: Pearson, 2000.

ARMBRUST et al. A View of Cloud Computing. **Commun. ACM**, New York, NY, USA, v.53, n.4, p.50–58, Apr 2010.

BARKAT, A.; SANTOS, A. Diniz dos; IKKEN, S. Open Source Solutions for Building IaaS Clouds. **Scalable Computing: Practice and Experience**, West University of Timisoara, ROMANIA, v.16, n.2, p.187–204, 2015.

BELLIFEMINE, F.; POGGI, A.; RIMASSA, G. JADE–A FIPA-Compliant Agent Framework. In: PAAM, 1999, London, ENGLAND. **Proceedings...** Practical Application Co Ltd, 1999. v.99, n.97–108, p.33.

BELOGLAZOV, A. **Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing**. 2013. Tese de Doutorado — THE UNIVERSITY OF MELBOURNE, Melbourne, AUSTRALIA.

BENEDICTIS, A. D.; RAK, M.; TURTUR, M.; VILLANO, U. A Framework for Cloud-aware Development of Bag-of-tasks Scientific Applications. **Int. J. Grid Util. Comput.**, Inderscience Publishers, Geneva, SWITZERLAND, v.7, n.2, p.130–140, Jan. 2016.

BERGER, J.; FISEK, H. F.; NORMAN, R. Z.; ZELDITCH, M. Status Characteristics and Social Interaction: An Expectation-States Approach. **Sociological theories in progress**, Boston, MA, USA, v.1, p.47–74, 1966.

BRAUN, T. D. et al. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. **Journal of Parallel and Distributed Computing**, Atlanta, GA, USA, v.61, n.6, p.810–837, 2001.

CALHEIROS, R. N. et al. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. **Software: Practice and Experience**, New York, NY, USA, v.41, n.1, p.23–50, Jan. 2011.

CASANOVA, H. Simgrid: A Toolkit for the Simulation of Application Scheduling. In: FIRST IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID, 2001, Brisbane, Queensland, AUSTRALIA. **Proceedings...** IEEE, 2001. p.430–437.

CIRNE, W. et al. Running Bag-of-Tasks Applications on Computational Grids: the MyGrid Approach. In: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, 2003. PROCEEDINGS., 2003., 2003, Kaohsiung, Taiwan. **Anais...** IEEE, 2003. p.407–416.

ER. SHIMPY, M. J. S. Different Scheduling Algorithms in Different Cloud Environment. **Applied and Environmental Microbiology**, ONLINE, v.3, p.8003–8006, Sep 2014.

GHANBARIA, S.; OTHMANA, M. A Priority based Job Scheduling Algorithm in Cloud Computing. In: PROCEDIA ENGINEERING 50, 2012, ONLINE. **Anais...** IJITECH, 2012. v.3, p.778–785.

GUTIERREZ-GARCIA, J. O.; SIM, K. M. A Family of Heuristics for Agent-based Elastic Cloud Bag-of-tasks Concurrent Scheduling. **Future Gener. Comput. Syst.**, Amsterdam, The Netherlands, The Netherlands, v.29, n.7, p.1682–1699, Sept. 2013.

IOSUP, A.; EPEMA, D. Grid Computing Workloads. **IEEE Internet Computing**, Piscataway, NJ, USA, v.15, n.2, p.19–26, Mar. 2011.

IOSUP, A.; SONMEZ, O.; ANOEP, S.; EPEMA, D. The Performance of Bags-of-tasks in Large-scale Distributed Systems. In: INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING, 17., 2008, New York, NY, USA. **Proceedings...** ACM, 2008. p.97–108. (HPDC '08).

ISSARIYAKUL, T.; HOSSAIN, E. **Introduction to Network Simulator NS2**. 1.ed. New York, NY, USA: Springer Publishing Company, Incorporated, 2008.

KAVIS, M. J. **Architecting the Cloud**: Design Decisions for Cloud Computing Service Models (SAAS, PAAS, and IAAS). Hoboken, NJ, USA: John Wiley & Sons, 2014.

KENNEDY, J. Particle Swarm Optimization. In: **Encyclopedia of machine learning**. [S.l.]: Springer, 2011. p.760–766.

KIM, W. Cloud Computing Adoption. In: JOURNAL OF WEB AND GRID SERVICES, 2011. **Anais...** InderScience Publishers, 2011. p.225–245.

KONG, W.; LEI, Y.; MA, J. Virtual Machine Resource Scheduling Algorithm for Cloud Computing Based on Auction Mechanism. **Optik - International Journal for Light and Electron Optics**, Amsterdam, NETHERLANDS, v.127, n.12, p.5099–5104, 2016.

LIBERTY. **Liberty Install Guide @ONLINE**. Acessado em 05 de Dezembro de 2016, traduzido, http://docs.openstack.org/liberty/install-guide-ubuntu/overview.html.

LOGANATHAN, S.; MUKHERJEE, S. Job Scheduling with Efficient Resource Monitoring in Cloud Datacenter. **The Scientific World Journal**, ONLINE, p.11, Oct 2015.

LUKE, E. A. **Introduction to Parallel Computing @ONLINE**. Acessado em 20 de Outubro de 2016., http://web.cse.msstate.edu/~luke/Courses/fl10/CSE4163/notes1.pdf.

MATEOS, C.; PACINI, E.; GARINO, C. G. An ACO-inspired Algorithm for Minimizing Weighted Flowtime in Cloud-based Parameter Sweep Experiments. **Adv. Eng. Softw.**, Oxford, UK, UK, v.56, p.38–50, Feb 2013.

MELL, P. M.; GRANCE, T. **SP 800-145. The NIST Definition of Cloud Computing**. Gaithersburg, MD, United States: National Institute of Standards and Technology, 2011.

MOHAMMADI, F.; JAMALI, D. S.; BEKRAVI, M. Survey on Job Scheduling algorithms in Cloud Computing. **International Journal of Emerging Trends and Technology in Computer Science (IJETTCS)**, ONLINE, v.3, n.2, p.151–154, 2014.

MONDAL, B.; DASGUPTA, K.; DUTTA, P. Load Balancing in Cloud Computing using Stochastic Hill Climbing-A Soft Computing Approach. **Procedia Technology**, Amsterdam, NETHERLANDS, v.4, n.Supplement C, p.783–789, Jun 2012. 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012.

NETTO, M. A. S.; BUYYA, R. Rescheduling co-allocation requests based on flexible advance reservations and processor remapping. In: IEEE/ACM INTERNATIONAL CONFERENCE ON GRID COMPUTING, 2008., 2008, Tsukuba, JAPAN. **Anais...** IEEE, 2008. p.144–151.

NETTO, M. A. S.; BUYYA, R. Offer-based scheduling of deadline-constrained Bag-of-Tasks applications for utility computing systems. In: IEEE INTERNATIONAL SYMPOSIUM ON PARALLEL DISTRIBUTED PROCESSING, 2009., 2009, Rome, ITALY. **Anais...** IEEE, 2009. p.1–11.

OPENSTACK. **What is OpenStack? @ONLINE**. Acessado em 05 de Dezembro de 2016, traduzido, https://www.openstack.org/software.

PACINI, E. et al. Simulation on Cloud Computing Infrastructures of Parametric Studies of Nonlinear Solids Problems. In: SECOND INTERNATIONAL CONFERENCE ON ADVANCES IN NEW TECHNOLOGIES, INTERACTIVE INTERFACES AND COMMUNICABILITY, 2012, Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2012. p.58–70. (ADNTIIC'11).

PANDEY, S.; WU, L.; GURU, S. M.; BUYYA, R. A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments. In: IEEE INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS, 2010., 2010, Perth, WA, AUSTRALIA. **Anais...** IEEE, 2010. p.400–407.

PINGALE, S. H.; MOGAL, P. V. New Approach to Schedule Bag of Tasks (BoTs) with Increase in Parallel Process. **International Journal of Innovative Research in Computer and Communication Engeneering**, ONLINE, v.3, n.7, p.6530–6535, 2015.

PREVE, N. **Computational and Data Grids**: Principles, Applications and Design: Principles, Applications and Design. Hershey, PA, USA: IGI Global, 2011.

SAMAL, P.; MISHRA, P. Analysis of Variants in Round Robin Algorithms for Load Balancing in Cloud Computing. **International Journal of computer science and Information Technologies**, ONLINE, v.4, n.3, p.416–419, 2013.

SANTOS, M. A. d. **Modelo de Escalonamento Aplicativo para Bag of Tasks em Ambientes de Nuvem Computacional**. 2016. Dissertação de Mestrado — Universidade Federal de Pelotas, Pelotas, RS, BRAZIL.

SANTOS, M. A. dos; BOIS, A. R. D.; CAVALHEIRO, G. G. H. A User Level Approach to Schedule BoT Applications on Private Clouds. In: IEEE 15TH INTERNATIONAL SYMPOSIUM ON NETWORK COMPUTING AND APPLICATIONS (NCA), 2016., 2016, Cambridge, MA, USA. **Anais...** IEEE, 2016. p.31–34.

SINDHU, S.; MUKHERJEE, S. Efficient Task Scheduling Algorithms for Cloud Computing Environment. In: HIGH PERFORMANCE ARCHITECTURE AND GRID COMPUTING: INTERNATIONAL CONFERENCE, HPAGC 2011, CHANDIGARH, INDIA, JULY 19-20, 2011. PROCEEDINGS, 2011, Berlin, Heidelberg. **Anais...** Springer Berlin Heidelberg, 2011. p.79–83.

TOHIDIRAD, Y. et al. Virtual Machine Scheduling in Cloud Computing Environment. **International Journal of Managing Public Sector Information and Communication Technologies (IJMPICT)**, ONLINE, v.6, n.4, p.1–6, 2015.

TUFFIN, B.; CHOUDHARY, P. K.; HIREL, C.; TRIVEDI, K. S. Simulation Versus Analytic-numeric Methods: Illustrative Examples. In: ND INTERNATIONAL CONFERENCE ON PERFORMANCE EVALUATION METHODOLOGIES AND TOOLS, 2., 2007, ICST, Brussels, Belgium, Belgium. **Proceedings...** ICST (Institute for Computer Sciences: Social-Informatics and Telecommunications Engineering), 2007. p.63:1–63:10. (ValueTools '07).

VALIANT, L. G. A Bridging Model for Parallel Computation. **Commun. ACM**, New York, NY, USA, v.33, n.8, p.103–111, Aug 1990.

WADHONKAR, A.; THENG, D. A survey on different scheduling algorithms in cloud computing. In: INTERNATIONAL CONFERENCE ON ADVANCES IN ELECTRICAL, ELECTRONICS, INFORMATION, COMMUNICATION AND BIO-INFORMATICS (AEEICB), 2016., 2016, Chennai, INDIA. **Anais...** IEEE, 2016. p.665–669.

WANG, W.; ZENG, G.; TANG, D.; YAO, J. Cloud-DLS: Dynamic Trusted Scheduling for Cloud Computing. **Expert Systems with Applications**, Amsterdam, NETHERLANDS, v.39, n.3, p.2321–2329, 2012.

XU, B.; ZHAO, C.; HU, E.; HU, B. Job Scheduling Algorithm Based on Berger Model in Cloud Environment. **Adv. Eng. Softw.**, Oxford, UK, UK, v.42, n.7, p.419–425, July 2011.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: State-of-the-art and Research Challenges. **Journal of Internet Services and Applications**, Berlin, GERMANY, v.1, n.1, p.7–18, May 2010.

ANEXO A BOTS UTILIZADOS NO USO EM PRODUÇÃO

Este anexo apresenta os BoTs utilizados por SANTOS em (SANTOS, 2016) no caso de Uso em Produção, apenas alterando as unidades e tipos para que se encaixem na execução no *CloudSim*.

- UP-1=
  - {[0.0,86520,1,7.0], [5.0,86520,1,10.0], [15.0,57680,6,10.0], [20.0,115360,0,7.0], [25.0,57680,0,11.0]}
- UP-2=
  - { [0.0,115360,2,24.0], [5.0,115360,2,24.0], [15.0,115360,5,29.0], [20.0,115360,1,27.0], [25.0,115360,0,19.0]}
- UP-3=
  - { [0.0,115360,2,27.0], [5.0,86520,2,22.0], [15.0,86520,5,30.0], [20.0,144200,1,32.0], [25.0,144200,0,25.0]}
- UP-4=
  - { [0.0,86520,3,44.0], [15.0,86520,3,46.0], [35.0,115360,8,47.0], [50.0,144200,1,62.0], [65.0,86520,1,59.0]}
- UP-5=
  - { [0.0,115360,1,9.0], [5.0,28840,1,13.0], [15.0,86520,6,8.0], [20.0,86520,0,9.0], [25.0,57680,0,7.0], [25.0,57680,4,11.0], [50.0,28840,3,7.0], [60.0,28840,4,7.0], [70.0,0,0,11.0], [70.0,17304,4,14.0]}
- UP-6=
  - { [0.0,115360,2,23.0], [5.0,144200,2,30.0], [15.0,115360,5,16.0], [20.0,115360,1,24.0], [25.0,115360,0,24.0], [25.0,144200,4,25.0], [50.0,144200,3,20.0], [60.0,115360,4,29.0], [70.0,173040,1,26.0], [70.0,115360,4,23.0]}

- UP-7=
  - { [0.0,86520,2,59.0], [15.0,86520,2,62.0], [35.0,86520,9,50.0], [50.0,57680,1,45.0], [65.0,115360,0,37.0], [70.0,86520,6,63.0], [100.0,86520,5,58.0], [120.0,28840,6,49.0], [140.0,86520,1,34.0], [150.0,57680,7,61.0]}
- UP-8=
  - { [0.0,115360,3,54.0], [15.0,173040,3,45.0], [35.0,86520,8,40.0], [50.0,144200,1,54.0], [65.0,144200,1,53.0], [70.0,144200,6,52.0], [100.0,115360,5,52.0], [120.0,201880,6,47.0], [140.0,86520,2,55.0], [150.0,173040,6,39.0]}
- UP-9=
  - { [0.0,86520,1,10.0], [5.0,57680,1,7.0], [15.0,86520,6,10.0], [20.0,115360,0,9.0], [25.0,0,0,7.0], [25.0,86520,4,5.0], [50.0,115360,3,10.0], [60.0,173040,4,13.0], [70.0,115360,0,10.0], [70.0,28840,4,11.0], [70.0,115360,8,9.0], [80.0,57680,1,15.0], [100.0,86520,3,10.0], [105.0,86520,2,10.0], [105.0,86520,6,6.0], [105.0,115360,3,13.0], [105.0,28840,3,9.0], [105.0,28840,1,8.0], [105.0,86520,6,13.0], [110.0,28840,0,12.0]}
- UP-10=
  - { [0.0,173040,3,38.0], [15.0,115360,3,48.0], [35.0,144200,8,42.0], [50.0,115360,1,62.0], [65.0,115360,1,45.0], [70.0,115360,6,52.0], [100.0,144200,5,81.0], [120.0,144200,6,47.0], [140.0,115360,2,51.0], [150.0,173040,6,48.0], [160.0,144200,9,47.0], [180.0,144200,2,46.0], [205.0,86520,4,36.0], [220.0,57680,4,63.0], [225.0,144200,7,26.0], [225.0,115360,5,67.0], [235.0,144200,5,54.0], [240.0,144200,2,51.0], [250.0,115360,8,43.0], [265.0,115360,2,42.0]}

APÊNDICE A BOTS GERADOS E UTILIZADOS COM VARIAÇÃO NO INTERVALO DE CHEGADA, DURAÇÃO DE TAREFAS, TAMANHO DE TAREFAS E UTILIZAÇÃO DE CPU

Os BoTs utilizados nos experimentos de simulação na infraestrutura virtual são listados neste apêndice. As quádruplas possuem o seguinte formato:

- Intervalo Curto - Duração Curta - Poucas Tarefas - CPU Baixo:
  - 10 quádruplas:
    - {[0.0,5768,5,16.0], [0.0,5768,3,3.0], [0.0,28840,2,22.0], [5.0,23072,2,1.0], [5.0,5768,2,24.0], [10.0,11536,5,24.0], [15.0,5768,5,3.0], [15.0,17304,2,16.0], [15.0,23072,4,1.0], [20.0,11536,3,19.0]}
  - 25 quádruplas:
    - {[5.0,23072,5,10.0], [10.0,23072,3,4.0], [15.0,23072,5,4.0], [15.0,28840,4,10.0], [20.0,23072,1,24.0], [20.0,11536,2,11.0], [25.0,28840,2,16.0], [25.0,11536,1,1.0], [25.0,11536,3,25.0], [30.0,23072,1,19.0], [35.0,17304,3,13.0], [40.0,5768,2,14.0], [45.0,28840,5,12.0], [45.0,5768,4,8.0], [45.0,23072,3,10.0], [45.0,5768,1,17.0], [45.0,28840,3,4.0], [45.0,23072,1,15.0], [45.0,28840,4,11.0], [50.0,11536,2,11.0], [55.0,17304,2,9.0], [60.0,11536,4,16.0], [60.0,11536,1,2.0], [60.0,23072,5,23.0], [65.0,11536,3,23.0]}
  - 50 quádruplas:
    - {[5.0,11536,4,8.0], [5.0,23072,4,14.0], [5.0,11536,4,11.0], [5.0,11536,2,2.0], [5.0,11536,5,10.0], [10.0,23072,4,16.0], [15.0,17304,3,17.0], [15.0,5768,4,15.0], [15.0,28840,2,11.0], [20.0,23072,4,12.0], [20.0,17304,2,16.0], [20.0,17304,3,25.0], [25.0,17304,2,13.0], [25.0,11536,3,15.0], [30.0,17304,2,14.0],

[30.0,23072,2,11.0], [30.0,17304,4,4.0], [35.0,11536,1,15.0],
[35.0,23072,4,17.0], [35.0,17304,5,12.0], [40.0,11536,1,21.0],
[40.0,28840,5,9.0], [40.0,17304,4,16.0], [45.0,11536,5,5.0],
[45.0,17304,4,24.0], [50.0,28840,2,5.0], [55.0,28840,4,12.0],
[55.0,11536,3,19.0], [55.0,28840,5,22.0], [60.0,28840,3,25.0],
[65.0,17304,4,6.0], [65.0,17304,4,15.0], [70.0,28840,4,8.0],
[70.0,23072,1,11.0], [75.0,5768,1,18.0], [80.0,5768,1,7.0],
[85.0,17304,5,6.0], [90.0,11536,5,21.0], [90.0,5768,5,5.0],
[90.0,17304,3,6.0], [95.0,28840,5,5.0], [95.0,23072,4,15.0],
[95.0,5768,4,17.0], [100.0,5768,3,7.0], [100.0,17304,4,13.0],
[100.0,23072,1,2.0], [105.0,23072,1,11.0], [110.0,28840,1,8.0],
[110.0,11536,3,21.0], [110.0,11536,4,8.0]}

- 100 quádruplas:
  * {[0.0,23072,5,12.0], [0.0,5768,2,12.0], [5.0,23072,4,19.0],
    [5.0,28840,4,10.0], [5.0,23072,3,4.0], [10.0,11536,3,7.0],
    [15.0,5768,4,6.0], [20.0,11536,5,17.0], [20.0,11536,3,7.0],
    [25.0,23072,3,18.0], [30.0,5768,3,20.0], [35.0,23072,4,13.0],
    [35.0,17304,1,24.0], [40.0,11536,4,20.0], [40.0,23072,1,6.0],
    [40.0,11536,2,1.0], [40.0,11536,1,12.0], [45.0,5768,5,21.0],
    [45.0,5768,2,16.0], [45.0,5768,1,15.0], [45.0,28840,1,3.0],
    [45.0,5768,4,3.0], [45.0,17304,1,18.0], [50.0,28840,5,12.0],
    [50.0,5768,1,23.0], [55.0,28840,1,16.0], [55.0,23072,4,15.0],
    [55.0,11536,3,17.0], [60.0,5768,3,10.0], [60.0,5768,2,24.0],
    [60.0,28840,3,23.0], [60.0,28840,5,21.0], [65.0,17304,2,4.0],
    [65.0,11536,1,10.0], [70.0,11536,3,4.0], [70.0,5768,1,25.0],
    [70.0,28840,3,4.0], [75.0,11536,3,10.0], [75.0,5768,5,23.0],
    [75.0,28840,1,2.0], [75.0,5768,4,20.0], [80.0,17304,2,5.0],
    [85.0,11536,2,22.0], [90.0,23072,4,8.0], [90.0,17304,2,16.0],
    [90.0,5768,4,17.0], [90.0,28840,3,13.0], [95.0,23072,3,11.0],
    [95.0,23072,5,21.0], [95.0,28840,4,23.0], [100.0,28840,4,8.0],
    [105.0,17304,1,14.0], [105.0,5768,1,24.0], [105.0,11536,5,22.0],
    [105.0,11536,1,24.0], [110.0,5768,5,6.0], [115.0,23072,2,17.0],
    [115.0,28840,2,18.0], [120.0,17304,5,25.0], [120.0,11536,5,13.0],
    [125.0,28840,3,10.0], [130.0,5768,3,18.0], [135.0,23072,3,19.0],
    [135.0,28840,4,16.0], [140.0,28840,2,16.0], [145.0,28840,1,1.0],
    [145.0,28840,5,6.0], [150.0,5768,2,1.0], [150.0,17304,3,18.0],
    [155.0,17304,2,9.0], [160.0,11536,5,15.0], [160.0,28840,2,15.0],
    [165.0,23072,2,4.0], [165.0,5768,2,5.0], [170.0,11536,2,12.0],
    [170.0,5768,4,24.0], [170.0,28840,5,3.0], [170.0,23072,4,17.0],

[170.0,17304,1,13.0], [175.0,11536,2,14.0], [175.0,11536,5,7.0], [180.0,17304,3,18.0], [185.0,17304,1,14.0], [190.0,17304,2,14.0], [190.0,23072,2,24.0], [190.0,28840,1,11.0], [190.0,11536,5,1.0], [190.0,17304,5,4.0], [190.0,5768,4,2.0], [195.0,28840,3,22.0], [200.0,11536,4,13.0], [205.0,11536,1,18.0], [205.0,23072,4,3.0], [210.0,11536,5,20.0], [210.0,28840,5,21.0], [210.0,5768,4,14.0], [210.0,28840,3,17.0], [210.0,23072,4,21.0], [210.0,5768,4,23.0], [210.0,11536,1,20.0]}

- Intervalo Longo - Duração Curta - Poucas Tarefas - CPU Baixo:
  - 10 quádruplas:
    - {[10.0,28840,2,17.0], [30.0,23072,5,4.0], [45.0,23072,1,17.0], [70.0,23072,5,11.0], [90.0,5768,2,1.0], [105.0,23072,1,14.0], [125.0,23072,4,4.0], [135.0,17304,4,19.0], [150.0,11536,5,10.0], [165.0,17304,2,14.0]}
  - 25 quádruplas:
    - {[10.0,17304,4,10.0], [25.0,28840,2,21.0], [35.0,28840,1,14.0], [40.0,28840,4,16.0], [40.0,5768,4,11.0], [40.0,28840,3,8.0], [55.0,5768,3,14.0], [80.0,23072,1,23.0], [105.0,23072,1,10.0], [130.0,28840,5,4.0], [140.0,28840,3,17.0], [145.0,28840,5,5.0], [165.0,5768,1,11.0], [165.0,28840,2,25.0], [175.0,23072,4,5.0], [180.0,23072,1,1.0], [200.0,11536,4,6.0], [215.0,23072,5,16.0], [235.0,17304,4,7.0], [235.0,17304,3,9.0], [260.0,5768,4,10.0], [275.0,28840,4,21.0], [285.0,28840,1,21.0], [290.0,11536,4,25.0], [310.0,5768,4,13.0]}
  - 50 quádruplas:
    - {[5.0,28840,4,5.0], [15.0,23072,5,10.0], [30.0,11536,2,9.0], [35.0,17304,1,6.0], [40.0,23072,1,17.0], [55.0,11536,1,18.0], [70.0,23072,5,19.0], [70.0,11536,1,1.0], [70.0,11536,2,12.0], [75.0,23072,3,25.0], [85.0,5768,5,14.0], [90.0,28840,2,7.0], [95.0,28840,2,17.0], [105.0,5768,4,25.0], [120.0,17304,1,9.0], [135.0,5768,5,3.0], [135.0,23072,3,16.0], [135.0,17304,4,23.0], [155.0,17304,1,23.0], [165.0,11536,2,23.0], [170.0,5768,4,17.0], [180.0,5768,1,5.0], [185.0,5768,3,22.0], [185.0,17304,3,15.0], [200.0,17304,4,13.0], [210.0,28840,1,3.0], [210.0,23072,3,2.0], [230.0,11536,1,21.0], [245.0,28840,2,3.0], [250.0,23072,5,21.0], [250.0,23072,1,19.0], [275.0,11536,2,10.0], [295.0,5768,3,20.0], [295.0,11536,4,20.0], [300.0,23072,4,22.0], [320.0,28840,5,16.0],

[325.0,11536,3,17.0], [335.0,28840,4,21.0], [340.0,11536,1,5.0],
[360.0,5768,2,5.0], [360.0,17304,5,25.0], [380.0,28840,5,6.0],
[390.0,17304,4,20.0], [415.0,17304,1,6.0], [415.0,5768,5,18.0],
[420.0,28840,4,3.0], [435.0,23072,2,20.0], [450.0,23072,1,1.0],
[455.0,28840,2,17.0], [470.0,23072,4,19.0]}

– 100 quádruplas:

* {[10.0,23072,1,25.0], [10.0,5768,3,22.0], [30.0,11536,4,2.0],
[30.0,5768,2,4.0], [55.0,5768,3,15.0], [80.0,28840,4,1.0],
[100.0,28840,3,6.0], [115.0,23072,4,8.0], [125.0,23072,4,13.0],
[130.0,17304,3,24.0], [140.0,28840,1,15.0], [140.0,11536,4,14.0],
[155.0,5768,4,4.0], [155.0,23072,2,6.0], [165.0,23072,4,16.0],
[175.0,11536,3,24.0], [200.0,17304,4,5.0], [200.0,28840,5,21.0],
[205.0,23072,2,6.0], [215.0,11536,1,14.0], [230.0,5768,3,13.0],
[245.0,5768,5,3.0], [250.0,5768,3,3.0], [270.0,28840,5,12.0],
[280.0,23072,2,5.0], [280.0,23072,1,18.0], [295.0,28840,1,12.0],
[305.0,11536,4,17.0], [315.0,5768,3,25.0], [320.0,17304,1,14.0],
[340.0,5768,3,12.0], [340.0,23072,3,6.0], [345.0,28840,2,15.0],
[370.0,23072,1,16.0], [385.0,11536,1,5.0], [395.0,28840,4,8.0],
[400.0,11536,5,7.0], [415.0,28840,4,6.0], [425.0,23072,2,19.0],
[425.0,11536,5,11.0], [450.0,11536,2,22.0], [450.0,23072,4,18.0],
[455.0,5768,4,20.0], [465.0,17304,3,3.0], [470.0,23072,4,7.0],
[475.0,17304,2,20.0], [495.0,23072,1,7.0], [520.0,5768,4,3.0],
[520.0,11536,1,2.0], [530.0,28840,4,14.0], [535.0,17304,3,10.0],
[540.0,5768,5,23.0], [540.0,28840,1,23.0], [555.0,23072,1,23.0],
[555.0,5768,3,7.0], [570.0,11536,4,11.0], [575.0,23072,2,10.0],
[585.0,11536,4,5.0], [585.0,11536,2,20.0], [600.0,28840,2,4.0],
[615.0,11536,2,9.0], [635.0,5768,4,14.0], [635.0,23072,4,7.0],
[655.0,5768,1,13.0], [680.0,28840,3,17.0], [705.0,11536,5,16.0],
[720.0,17304,5,16.0], [740.0,28840,1,21.0], [740.0,17304,5,14.0],
[765.0,28840,4,16.0], [775.0,17304,3,4.0], [780.0,11536,4,15.0],
[780.0,5768,5,2.0], [780.0,28840,3,14.0], [800.0,17304,3,2.0],
[800.0,11536,5,21.0], [820.0,28840,1,17.0], [840.0,23072,1,7.0],
[865.0,11536,5,4.0], [875.0,11536,5,13.0], [880.0,17304,5,3.0],
[885.0,17304,2,15.0], [895.0,11536,4,1.0], [920.0,5768,3,1.0],
[940.0,17304,4,24.0], [940.0,17304,2,10.0], [955.0,28840,4,17.0],
[970.0,28840,1,19.0], [990.0,11536,1,14.0], [1010.0,17304,5,13.0],
[1030.0,23072,1,22.0], [1055.0,28840,3,23.0], [1075.0,5768,3,20.0],
[1095.0,11536,2,12.0], [1115.0,11536,2,19.0], [1135.0,28840,1,5.0],
[1150.0,23072,2,5.0], [1160.0,28840,2,10.0], [1165.0,11536,1,21.0],

[1185.0,28840,5,5.0]}

- Intervalo Curto - Duração Longa - Poucas Tarefas - CPU Baixo:
  - 10 quádruplas:
    - {[5.0,86520,7,55.0], [5.0,80752,17,2.0], [10.0,126896,10,42.0], [10.0,63448,3,8.0], [15.0,40376,8,12.0], [15.0,23072,11,34.0], [20.0,144200,20,73.0], [25.0,23072,8,68.0], [30.0,103824,1,7.0], [35.0,144200,3,48.0]}
  - 25 quádruplas:
    - {[5.0,57680,9,83.0], [10.0,74984,13,73.0], [10.0,23072,13,98.0], [10.0,57680,1,94.0], [15.0,28840,10,43.0], [15.0,121128,16,5.0], [15.0,17304,18,43.0], [15.0,51912,3,18.0], [15.0,46144,14,30.0], [15.0,69216,4,33.0], [20.0,144200,8,94.0], [20.0,46144,7,39.0], [25.0,23072,8,52.0], [30.0,40376,1,60.0], [30.0,132664,5,62.0], [35.0,40376,8,33.0], [35.0,11536,7,3.0], [35.0,5768,4,43.0], [40.0,144200,18,97.0], [40.0,86520,1,6.0], [45.0,40376,3,38.0], [45.0,80752,13,65.0], [50.0,51912,2,73.0], [55.0,46144,20,95.0], [60.0,28840,3,46.0]}
  - 50 quádruplas:
    - {[0.0,28840,10,44.0], [5.0,98056,1,97.0], [10.0,138432,15,57.0], [10.0,144200,8,88.0], [15.0,69216,6,68.0], [15.0,51912,15,79.0], [15.0,138432,3,92.0], [20.0,92288,13,19.0], [20.0,5768,12,30.0], [20.0,80752,3,95.0], [20.0,11536,13,35.0], [20.0,92288,2,95.0], [20.0,11536,15,20.0], [25.0,51912,15,63.0], [30.0,34608,6,29.0], [35.0,17304,17,89.0], [35.0,28840,11,55.0], [40.0,103824,9,81.0], [40.0,115360,14,59.0], [45.0,115360,2,33.0], [50.0,23072,16,89.0], [55.0,80752,8,47.0], [60.0,98056,15,64.0], [60.0,138432,12,31.0], [65.0,63448,18,87.0], [65.0,34608,1,11.0], [65.0,144200,19,53.0], [70.0,80752,12,57.0], [75.0,23072,17,78.0], [75.0,34608,8,93.0], [75.0,132664,20,14.0], [80.0,109592,3,56.0], [80.0,74984,16,78.0], [85.0,51912,6,85.0], [85.0,5768,19,24.0], [90.0,23072,19,13.0], [95.0,80752,13,62.0], [100.0,63448,18,75.0], [105.0,57680,9,97.0], [105.0,74984,8,10.0], [110.0,92288,15,81.0], [110.0,11536,7,60.0], [115.0,144200,12,24.0], [115.0,109592,13,58.0], [115.0,121128,4,25.0], [120.0,11536,3,12.0], [120.0,98056,11,36.0], [125.0,57680,16,31.0], [125.0,23072,16,54.0], [125.0,40376,14,53.0]}
  - 100 quádruplas:

    * {[5.0,126896,20,85.0], [5.0,46144,4,69.0], [10.0,115360,9,18.0], [10.0,144200,13,13.0], [15.0,5768,3,68.0], [20.0,40376,14,57.0], [20.0,63448,1,6.0], [25.0,40376,2,47.0], [30.0,34608,4,57.0], [30.0,51912,4,98.0], [35.0,40376,14,74.0], [35.0,5768,9,69.0], [35.0,51912,2,49.0], [40.0,121128,7,51.0], [45.0,109592,15,74.0], [50.0,51912,2,77.0], [50.0,92288,18,71.0], [50.0,80752,15,75.0], [55.0,40376,17,40.0], [60.0,80752,3,79.0], [60.0,57680,3,80.0], [65.0,80752,14,75.0], [65.0,92288,16,76.0], [70.0,115360,9,81.0], [70.0,51912,17,12.0], [75.0,132664,5,8.0], [75.0,115360,14,46.0], [80.0,132664,2,71.0], [85.0,121128,13,4.0], [90.0,103824,17,60.0], [95.0,17304,18,69.0], [95.0,121128,2,16.0], [95.0,121128,12,27.0], [95.0,80752,5,28.0], [100.0,86520,6,98.0], [100.0,17304,4,6.0], [105.0,126896,11,73.0], [110.0,57680,11,84.0], [115.0,17304,5,22.0], [120.0,98056,1,80.0], [125.0,46144,1,71.0], [130.0,86520,3,71.0], [130.0,11536,3,41.0], [135.0,51912,11,7.0], [135.0,23072,18,74.0], [135.0,40376,15,34.0], [135.0,115360,17,52.0], [135.0,98056,13,60.0], [140.0,34608,17,83.0], [140.0,57680,18,21.0], [140.0,132664,10,66.0], [145.0,121128,9,4.0], [150.0,86520,12,21.0], [150.0,98056,6,67.0], [150.0,121128,16,28.0], [155.0,11536,6,33.0], [160.0,126896,17,1.0], [160.0,40376,5,57.0], [160.0,11536,11,41.0], [165.0,86520,7,86.0], [170.0,63448,11,87.0], [175.0,63448,9,85.0], [180.0,63448,10,83.0], [180.0,63448,16,66.0], [180.0,126896,7,7.0], [180.0,63448,15,41.0], [185.0,57680,1,51.0], [185.0,40376,13,43.0], [190.0,74984,12,34.0], [195.0,69216,8,12.0], [200.0,5768,6,14.0], [200.0,109592,19,44.0], [200.0,5768,7,58.0], [200.0,80752,10,73.0], [205.0,40376,2,67.0], [205.0,63448,14,19.0], [205.0,17304,6,88.0], [210.0,57680,5,58.0], [210.0,40376,4,33.0], [210.0,92288,11,85.0], [210.0,51912,4,4.0], [210.0,144200,19,17.0], [215.0,144200,20,92.0], [215.0,63448,3,28.0], [220.0,5768,13,90.0], [225.0,132664,9,32.0], [225.0,144200,5,94.0], [230.0,132664,10,15.0], [235.0,144200,4,64.0], [235.0,46144,20,87.0], [240.0,69216,19,62.0], [240.0,40376,18,42.0], [240.0,40376,7,22.0], [240.0,63448,12,65.0], [245.0,5768,4,18.0], [245.0,115360,9,50.0], [250.0,23072,12,41.0], [250.0,109592,3,15.0], [250.0,51912,10,61.0], [255.0,138432,12,53.0]}

- Intervalo Longo - Duração Longa - Muitas Tarefas - CPU ALTO:
  - 10 quádruplas:
    * {[0.0,40376,2,4.0], [0.0,109592,10,45.0], [15.0,115360,7,7.0], [25.0,46144,7,48.0], [30.0,34608,17,18.0], [30.0,63448,15,31.0],

[30.0,126896,17,75.0], [40.0,86520,15,96.0], [65.0,80752,6,76.0], [75.0,23072,14,13.0]}

- 25 quádruplas:
  - {[15.0,86520,3,50.0], [30.0,74984,5,37.0], [50.0,57680,18,97.0], [70.0,138432,12,67.0], [90.0,46144,13,95.0], [90.0,138432,20,73.0], [115.0,23072,1,13.0], [125.0,138432,13,19.0], [125.0,46144,6,98.0], [145.0,132664,8,92.0], [160.0,132664,12,2.0], [175.0,92288,9,86.0], [180.0,23072,3,24.0], [180.0,46144,9,67.0], [190.0,5768,11,92.0], [195.0,11536,12,95.0], [195.0,57680,17,95.0], [215.0,5768,17,45.0], [220.0,51912,19,37.0], [245.0,121128,2,12.0], [250.0,51912,17,61.0], [255.0,98056,3,19.0], [270.0,17304,20,61.0], [270.0,138432,14,63.0], [275.0,86520,13,29.0]}
- 50 quádruplas:
  - {[10.0,46144,2,1.0], [25.0,103824,16,50.0], [35.0,57680,6,27.0], [40.0,80752,19,82.0], [50.0,11536,11,76.0], [50.0,132664,18,33.0], [60.0,144200,6,55.0], [60.0,40376,12,51.0], [70.0,23072,18,55.0], [80.0,34608,4,100.0], [95.0,40376,3,16.0], [100.0,46144,3,96.0], [120.0,5768,17,71.0], [145.0,103824,15,89.0], [145.0,86520,14,86.0], [155.0,17304,12,24.0], [175.0,34608,11,41.0], [180.0,103824,8,35.0], [190.0,138432,1,72.0], [195.0,92288,14,23.0], [215.0,40376,19,69.0], [240.0,138432,12,81.0], [260.0,103824,3,89.0], [270.0,28840,1,2.0], [270.0,17304,4,33.0], [280.0,17304,12,74.0], [300.0,144200,9,60.0], [310.0,132664,14,11.0], [335.0,132664,6,54.0], [360.0,40376,9,69.0], [375.0,28840,15,36.0], [400.0,138432,17,59.0], [415.0,51912,20,32.0], [425.0,5768,8,54.0], [425.0,132664,9,44.0], [425.0,23072,17,11.0], [440.0,40376,5,86.0], [445.0,115360,7,5.0], [465.0,28840,15,90.0], [465.0,80752,11,30.0], [485.0,63448,6,63.0], [495.0,144200,11,98.0], [510.0,51912,15,80.0], [525.0,80752,7,17.0], [530.0,80752,13,5.0], [535.0,69216,3,60.0], [550.0,138432,10,93.0], [565.0,17304,18,99.0], [580.0,17304,4,45.0], [585.0,28840,8,1.0]}
- 100 quádruplas:
  - {[5.0,103824,10,87.0], [15.0,115360,19,47.0], [40.0,115360,16,41.0], [65.0,51912,9,81.0], [75.0,132664,12,49.0], [100.0,17304,3,19.0], [120.0,80752,5,55.0], [140.0,57680,6,60.0], [165.0,92288,16,94.0], [180.0,121128,16,46.0], [205.0,40376,12,11.0], [210.0,11536,10,57.0], [215.0,109592,1,12.0], [240.0,5768,7,74.0], [255.0,17304,10,57.0], [275.0,46144,18,45.0], [295.0,132664,3,9.0], [300.0,132664,18,14.0], [315.0,57680,16,72.0], [315.0,63448,9,99.0], [335.0,57680,16,71.0],

[335.0,86520,7,24.0], [350.0,69216,13,91.0], [375.0,11536,6,60.0], [400.0,126896,20,93.0], [425.0,74984,5,80.0], [445.0,92288,4,43.0], [470.0,103824,1,82.0], [485.0,34608,2,4.0], [505.0,23072,6,100.0], [505.0,46144,14,24.0], [525.0,17304,9,20.0], [550.0,126896,10,55.0], [575.0,103824,9,32.0], [595.0,121128,8,3.0], [610.0,126896,18,9.0], [610.0,63448,16,56.0], [630.0,57680,13,3.0], [635.0,69216,16,11.0], [640.0,109592,14,55.0], [655.0,57680,14,22.0], [680.0,57680,8,100.0], [685.0,28840,16,34.0], [710.0,5768,19,74.0], [720.0,51912,20,49.0], [735.0,132664,9,15.0], [745.0,98056,8,97.0], [745.0,17304,9,98.0], [750.0,57680,18,75.0], [760.0,86520,12,7.0], [770.0,92288,19,1.0], [770.0,11536,11,27.0], [785.0,46144,16,7.0], [795.0,138432,3,67.0], [800.0,132664,3,48.0], [805.0,115360,19,58.0], [820.0,11536,11,10.0], [825.0,132664,9,33.0], [830.0,11536,2,45.0], [855.0,144200,18,30.0], [880.0,74984,11,94.0], [890.0,86520,5,24.0], [895.0,34608,12,75.0], [920.0,40376,9,37.0], [930.0,63448,20,44.0], [945.0,11536,11,77.0], [965.0,51912,13,80.0], [965.0,51912,1,21.0], [980.0,86520,20,27.0], [985.0,138432,6,68.0], [1005.0,46144,11,58.0], [1005.0,121128,15,63.0], [1020.0,132664,17,61.0], [1045.0,57680,3,4.0], [1070.0,5768,11,65.0], [1075.0,57680,13,70.0], [1085.0,132664,11,16.0], [1110.0,69216,3,42.0], [1125.0,80752,4,38.0], [1130.0,103824,6,47.0], [1135.0,40376,1,79.0], [1160.0,40376,10,62.0], [1180.0,138432,4,61.0], [1190.0,98056,19,72.0], [1210.0,28840,14,13.0], [1235.0,103824,6,14.0], [1260.0,46144,20,60.0], [1270.0,57680,5,14.0], [1295.0,109592,12,28.0], [1310.0,103824,14,5.0], [1330.0,40376,11,23.0], [1345.0,69216,3,1.0], [1355.0,132664,10,1.0], [1360.0,98056,12,12.0], [1370.0,74984,10,56.0], [1390.0,86520,19,18.0], [1410.0,63448,16,44.0], [1415.0,98056,2,100.0], [1420.0,109592,20,44.0], [1430.0,28840,14,23.0]}

APÊNDICE B BOT E BOT' (BOT ESCALADO) GERADOS E UTILIZADOS NO ESTUDO DE CASO

- BoT
  - { [0.0,144200,36,100.0], [1.0,115360,10,100.0], [1.0,144200,26,100.0], [2.0,86520,15,100.0], [2.0,144200,11,100.0], [2.0,57680,19,100.0], [2.0,115360,10,100.0], [2.0,173040,13,100.0], [3.0,144200,9,100.0], [4.0,144200,10,100.0], [4.0,57680,11,100.0], [4.0,86520,12,100.0], [4.0,115360,13,100.0], [5.0,57680,10,100.0], [5.0,57680,15,100.0] }
- BoT'
  - [{0.0,144200,468,100.0], [1.0,115360,130,100.0], [1.0,144200,338,100.0], [2.0,86520,195,100.0], [2.0,144200,143,100.0], [2.0,57680,247,100.0], [2.0,115360,130,100.0], [2.0,173040,169,100.0], [3.0,144200,117,100.0], [4.0,144200,130,100.0], [4.0,57680,143,100.0], [4.0,86520,146,100.0], [4.0,115360,169,100.0], [5.0,57680,130,100.0], [5.0,57680,195,100.0]}