
UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

ANTONIO RAFAEL BRAGA

# ELASTICCLUSTER: EXPLORANDO A OCIOSIDADE DE *CLUSTERS* VIRTUAIS PARA EXECUÇÃO DE APLICAÇÕES DO TIPO SACO DE TAREFAS

FORTALEZA, CEARÁ

2012

**ANTONIO RAFAEL BRAGA**

# ELASTICCLUSTER: EXPLORANDO A OCIOSIDADE DE *CLUSTERS* VIRTUAIS PARA EXECUÇÃO DE APLICAÇÕES DO TIPO SACO DE TAREFAS

Dissertação de Mestrado apresentada à Coordenação do Curso de Pós-Graduação em Engenharia de Teleinformática da Universidade Federal do Ceará como parte dos requisitos para obtenção do grau de Mestre em Engenharia de Teleinformática

Área de concentração: Sinais e Sistemas

Orientador: Prof. Dr. Danielo G. Gomes

Co-Orientador: Prof. Dr. José M. Soares

**FORTALEZA, CEARÁ**

**2012**

A000z Braga, Antonio Rafael.

ElasticCluster: Explorando a Ociosidade de *Clusters* Virtuais para Execução de Aplicações do Tipo Saco de Tarefas / Antonio Rafael Braga. – Fortaleza, 2012.

91p.;il.

Orientador: Prof. Dr. Danielo G. Gomes

Co-Orientador: Prof. Dr. José M. Soares

Monografia (Programa de Pós Graduação em Engenharia de Teleinformática) - Universidade Federal do Ceará, Centro de Tecnologia.

1. Virtualização 2. *Clusters* Virtuais 3. Suporte a Serviços Elásticos I. Universidade Federal do Ceará, Centro de Tecnologia.

~~CDD:000.0~~

UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

CAMPUS DO PICI, CAIXA POSTAL 6007 CEP 60.738-640
FORTALEZA – CEARÁ - BRASIL
FONE (+55) 85 3366-9467 – FAX (+55) 85 3366-9468

**ANTONIO RAFAEL BRAGA**

# ELASTICCLUSTER: EXPLORANDO A OCIOSIDADE DE CLUSTERS VIRTUAIS PARA EXECUÇÃO DE APLICAÇÕES DO TIPO SACO DE TAREFAS

**Dissertação submetida à Coordenação do Programa de Pós-Graduação em Engenharia de Teleinformática, da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Teleinformática.**
**Área de concentração: Eletromagnetismo Aplicado.**

**Aprovada em 31/08/2012.**

**BANCA EXAMINADORA**

**Prof. Dr. Daniel Gonçalves Gomes (Orientador)**
**Universidade Federal do Ceará - UFC**

**Prof. Dr. José Marques Soares**
**Universidade Federal do Ceará - UFC**

**Prof. Dr. José Neuman de Souza**
**Universidade Federal do Ceará - UFC**

**Prof. Dr. Bruno Richard Schulze**
**Laboratório Nacional de Computação Científica - LNCC**

*Dedico esta dissertação à minha família: minha mãe Maria Lúcia, meu pai Antonio Braga, minhas irmãs Mirla Rafaela e Mirlane Rafael e à minha namorada Lívia Almada pela força, incentivo, apoio total, carinho e companheirismo sempre presentes.*

# AGRADECIMENTOS

A Deus por me dar perseverança em todos os momentos.

À minha família: minha mãe Maria Lúcia, meu pai Antonio Braga, minhas irmãs Mirla Rafaela, Mirlane Rafael e à minha namorada Lívia Almada pela força, incentivo, apoio total, carinho e companheirismo sempre presentes.

Ao meu orientador, Prof. Dr. Danielo Gonçalves Gomes pela dedicação, atenção, motivação, orientação, paciência e por acreditar em mim e no meu trabalho.

Ao Profº José Marques pela dedicação, atenção, motivação, paciência e por, através de suas sugestões, fazer com que este trabalho evoluísse bastante.

Aos professores José Neuman e Bruno Schulze pela participação na banca examinadora e pelas valiosas sugestões para crescimento desse trabalho.

A todos os meus colegas de laboratório e, em especial, ao Paulo Renato que teve importante participação na implementação dos algoritmos, na realização dos experimentos e nas discussões relacionadas com a avaliação do trabalho.

Agradeço aos professores e funcionários do PPGETI e do Grupo de Redes de Computadores Engenharia de Software e Sistemas (GREat) pelo apoio e por se mostrarem compromissados com a pesquisa, inovação e desenvolvimento de qualidade.

Por fim, agradeço à CAPES pelo auxílio financeiro que me permitiu dedicação integral ao mestrado e à minha pesquisa.

*"If I have seen farther than others it is because I have stood on the shoulders of gigants"*
***Issac Newton***

# RESUMO

Na computação em nuvem, a elasticidade e a capacidade de isolamento de carga permitem que seus recursos sejam aprovisionados e liberados em resposta a cargas de trabalho dinâmicas com baixo tempo de inatividade. Tais características são típicas de *clusters* hospedados em nuvem (*cluster* virtual - CV), de tal forma que estes recursos precisam ser gerenciados a fim de se garantir a minimização do desperdício de recursos nos provedores e garantir que o desempenho dos recursos não seja afetado negativamente. Este trabalho propõe uma política para adaptação dinâmica de *clusters* virtuais (CVs) a fim de reduzir o número de recursos ociosos sem comprometer o desempenho dos serviços. O algoritmo proposto baseado em heurística, realiza instanciação e desligamento de máquinas virtuais nos CVs conforme variação na demanda por recursos de aplicações do tipo saco de tarefas (Bag-of-Tasks, BoT). O algoritmo foi especificado, verificado e validado através de simulações em Redes de Petri (RdP). O desempenho da proposta é avaliado em três cenários distintos a partir das métricas: quantidade total de máquinas iniciadas, quantidade de máquinas ociosas reutilizadas, tempo total de execução da aplicação e quantidade média de *clusters* iniciados. Os resultados mostraram que a política de adaptação proposta é capaz reduzir a ociosidade e a sobrecarga de um CV e, consequentemente, melhorar o consumo de energia.

Palavras-Chave: Virtualização, *Clusters* Virtuais, Suporte a Serviços Elásticos

# ABSTRACT

In cloud computing, elasticity and capacity of load isolation allow their resources to be provisioned and released in response to dynamic workloads with reduced downtime. These characteristics are typical of clusters hosted in a cloud (virtual cluster - VC), so that these resources need to be managed in order to minimize its waste in cloud providers and ensure that resource performance is not adversely affected. This work proposes a policy for dynamic adaptation of virtual clusters (VCs) to reduce the number of idle resources without compromising their performance of resources. The proposed algorithm, based on heuristics, performs instantiation/shut-down of the virtual machines (VMs) in virtual clusters according variation in demand for resources of applications of type Bag-of-Tasks, BoT. The algorithm has been specified, verified and validated using Petri Nets formalism. The selected metrics to evaluate the proposal performance in three different scenarios are the total amount of machine started, amount of idle machines reused, total time of execution the application and average number of clusters started. The results showed that the adaptation policy proposal is able to reduce idle and the overhead of a virtual cluster and thus improve power consumption.

Keywords: Virtualization, Virtual *Clusters*, Support for Elastic Services

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE SIGLAS

| | |
|---|---|
| CV | *cluster* virtual |
| BoT | *Bag-of-Tasks* |
| MVs | Máquinas Virtuais |
| HPC | *High-Performance Computing* |
| SaaS | *Software as a Service* |
| PaaS | *Platform as a Service* |
| IaaS | *Infrastructure as a Service* |
| SOA | *Service-Oriented Architecture* |
| VMM | *Virtual Machine Manager* |
| MPI | *Message Passing Interface* |
| CAD | Computação de Alto Desempenho |
| SLA | *Service Level Agreement* |
| RPC | Redes de Petri Coloridas |
| FIFO | *First In, First Out* |
| RdP | Redes de Petri |

# LISTA DE SÍMBOLOS

| | |
|---|---|
| $c_i$ | *Cluster* c de índice i |
| $MV_{j,i}$ | Nó de índice j que compõem o *cluster* de índice i |
| $A_{k+1}$ | Novo valor do atuador do sistema |
| $A_k$ | Atual valor do atuador do sistema |
| $QMV_0$ | Atual valor da quantidade de MVs em um dado *cluster* $c_i$ |
| $QMV_{0+1}$ | Novo valor da quantidade de MVs em um dado *cluster* $c_i$ |
| $U_{máx}$ | Utilização máxima configurada para qualquer *cluster* $c_i$ |
| $MV_{máx}$ | Número máximo de nós configurado para qualquer *cluster* $c_i$ |
| $QMV_{ócio}$ | Ociosidade de um *cluster* $c_i$ em quantidade de MVs |
| $QMV_{atual}$ | Quantidade de MVs instanciadas que compõem o *cluster* $c_i$ |
| $U$ | Utilização do *cluster* $c_i$ |
| $QMV$ | Quantidade de MVs a serem inicializadas |
| $TTLMachines$ | Tempo de vida das MVs $MV_{j,i}$ |
| $TTLDoBoT$ | Tempo de vida de um BoT |
| $BoT_i$ | BoT que está sendo finalizado ou cancelado |
| $QMVD_o$ | Quantidade inicial de MVs que devem ser desligadas |
| $QMVD$ | Quantidade de máquinas que serão desligadas se existirem BoTs enfileirados |

# SUMÁRIO

# 1 INTRODUÇÃO

Na Seção 1.1, é apresentado o contexto em que a computação em nuvem tornou-se um importante paradigma para a execução de aplicações científicas em larga escala. Na Seção 1.2 são apresentadas a motivação e os problemas encontrados na computação em nuvem que impulsionaram o desenvolvimento da solução proposta. Na Seção 1.3 expõe os principais objetivos desta dissertação. O método utilizado para alcançar os objetivos é discutido na Seção 1.4. Na seção 1.5 são apresentadas as principais contribuições deste trabalho. Na Seção 1.6, é apresentada a estrutura dos capítulos desta dissertação.

## 1.1 Contextualização

A computação em nuvem tem sido utilizada com frequência pela comunidade científica, pois mostra-se como uma alternativa barata para provisão de supercomputadores, sendo uma plataforma mais escalável do que os *clusters* especializados. No entanto, as nuvens também apresentam desafios importantes em muitas áreas ligadas à computação científica (IOSUP et al., 2011). Este trabalho apresenta uma política para adaptação dinâmica de *clusters* virtuais (CVs) capaz de reduzir o número de recursos ociosos e sem comprometer o desempenho dos *clusters* virtuais. Geralmente usados para a execução de aplicações científicas na nuvem, os *clusters* virtuais (ANEDDA et al., 2010), são grupos de máquinas virtuais (MVs) projetados para executar e compartilhar uma infraestrutura confiável de computação. A implantação dos *clusters* virtuais permite especificar a alocação de recursos diferentes para diferentes grupos de usuários e aplicações (FOSTER et al., 2006).

Algumas das principais características da computação em nuvem são a virtualização (abstração de um recurso computacional), balanceamento/isolamento de carga, alta confiabilidade, alta disponibilidade, baixo custo e a elasticidade. Neste trabalho, considera-se que a elasticidade e a capacidade de isolamento de carga permitem que a nuvem aprovisione e libere seus recursos em resposta a cargas de trabalho dinâmicas com baixo tempo de inatividade (LIM; BABU; CHASE, 2010). Entretanto, esta dinâmica de alocação torna-se mais crítica diante da dificuldade dos provedores em determinar precisamente qual a demanda de recursos para execução das aplicações. Portanto, estimar a quantidade de recursos computacionais consumidos pelas aplicações é um problema clássico de QoS ainda de difícil trato (HUU et al., 2011).

Nas infraestruturas de nuvens comerciais atuais, cabe ao usuário final estimar sua própria fatia de recursos a serem consumidos, sob pena de pagar pelo não-uso (*over-provisioning*) (MARSHALL; KEAHEY; FREEMAN, 2011). A ociosidade gerada pelo *overprovisioning* traz mais prejuízos quando as cargas de trabalho (*workloads*) são submetidas em rajadas. Considere, por exemplo, as aplicações do tipo saco de tarefas (Bag-of-Tasks, BoT), que são aplicações massivamente paralelas, de computação intensiva, constituídas por uma grande quantidade de tarefas que podem ser executadas de forma independente (FRAGA; BRASILEIRO; GUERRERO, 2011). Uma aplicação deste tipo pode ser executada em vários provedores de recursos de modo concorrente, tal que seu tempo de execução seja aquele esperado pelo usuário. Este tempo de execução é calculado a partir da soma das estimativas de tempo de execução de cada tarefa do BoT e do tempo de espera por recursos. Na prática, em geral, o usuário faz a superestimação dos tempos de execução a fim de evitar que a aplicação seja abortada (NETTO; BUYYA, 2011).

## 1.2 Motivação

Para executar aplicações do tipo BoT em nuvem, faz-se o uso de *clusters* virtuais (ANEDDA et al., 2010), que são grupos de máquinas virtuais (MVs) projetados para executar e compartilhar uma infraestrutura confiável de computação. A implantação dos mesmos permite especificar a alocação de recursos diferentes para diferentes grupos de usuários e aplicações (FOSTER et al., 2006). Se, para executar um BoT, um cliente fosse capaz de requisitar a um provedor de nuvem uma quantidade de MVs suficiente para maximizar sua paralelização, isto lhe permitiria executar este BoT no menor tempo possível. Contudo, na nuvem, se gasta tempo para prover os recursos e impõe-se limites em relação à quantidade de máquinas que podem ser instanciadas (COSTA et al., 2011). Nessa perspectiva, uma política de alocação dinâmica que leve em consideração aspectos relativos aos recursos e às aplicações é proposta por (MARSHALL; KEAHEY; FREEMAN, 2010). Contudo, a política proposta pelos autores não faz a reutilização de recursos ociosos gerados devido ao *overprovisioning*, a mesma lacuna pode ser observada em outros trabalhos (CHI; QIAN; LU, 2011), (KHATUA; GHOSH; MUKHERJEE, 2010), (LIAO; HU; JIN, 2010), (LIAO; JIN; LIU, 2012).

Diante desse cenário, neste trabalho é feito o uso da capacidade ociosa da nuvem a fim de diminuir a quantidade de recursos a serem alocados para execução de uma aplicação do tipo BoT (BRAGA et al., 2012). Com isso, espera-se uma redução do número de MVs instanciadas na nuvem e, consequentemente, uma redução dos gastos financeiros decorrentes a alocação de recursos. Para atingi-lo, propõem-se algoritmos para instanciação de desligamento de MVs que usa a capacidade ociosa dos *clusters* e observa a utilização dos recursos para tomada de decisão.

## 1.3 Objetivos

### 1.3.1 Objetivo Geral

O principal objetivo deste trabalho é desenvolver uma política para alocação dinâmica de MVs em *clusters* visando à redução do número de MVs alocadas através do aproveitamento da ociosidade, sem promover sobrecargas que podem comprometer requisitos de desempenho e a disponibilidade dos recursos alocados.

### 1.3.2 Objetivos Específicos

Para desenvolver a política proposta neste trabalho, os seguintes objetivos específicos devem ser alcançados:

1. Fazer o uso efetivo dos recursos a fim de cumprir o tempo esperado para execução de uma aplicação científica sem comprometer os requisitos de desempenho. Para tal, propomos uma política baseada em uma heurística, que realiza instanciação e desligamento de máquinas virtuais de *cluster* virtuais distintos conforme variação na demanda por recursos.

2. Na política proposta, a decisão para alocação de um novo *cluster* deve ser tomada com base na utilização que as aplicações já escalonadas fazem dos recursos, a partir disso é possível capturar o grau de sobrecarga/ócio dos *clusters* e decidir por uma reutilização de recursos ou alocação de novos recursos;

3. Apresentar um algoritmo de inicialização de MVs, que inclua a política proposta, e o algoritmo de desligamento de MVs automatizam o processo de adaptação de *clusters* virtuais com a inicialização e desligamento de MVs. A quantidade de MVs envolvidas e o instante de tempo em que os processos ocorrem são determinados através de heurística;

4. Apresentar uma nova heurística para determinação da quantidade de MVs que serão inciadas ou desligadas de acordo com as carga de trabalho dinâmicas. A heurística proposta é uma adaptação da heurística de (MARSHALL; KEAHEY; FREEMAN, 2010) e minimiza a quantidade total de MVs iniciadas, o tempo total de execução das aplicações e quantidade média de *clusters* iniciados;

5. Desenvolver uma modelagem feita com Redes de Petri de um provedor de recursos virtualizados com suporte a adaptação de *clusters* virtuais. Com essa modelagem é possível realizar simulações de alocações e liberação de *clusters* de MVs de acordo com algoritmos propostos e os estudados na literatura (MARSHALL; KEAHEY; FREEMAN, 2010).

6. Com o uso da heurística, identificar os cenários em que a operação do algoritmos oferece uma grande economia de recursos de tal forma que ocorra também economia de energia, garantias de disponibilidade e desempenho (BRAGA et al., 2012).

## 1.4 Contribuições

A principal contribuição deste trabalho é uma nova política de adaptação dinâmica de *clusters* que estende o trabalho de (MARSHALL; KEAHEY; FREEMAN, 2010) para execução de BoTs concorrentes. Na proposta de (MARSHALL; KEAHEY; FREEMAN, 2010) não há uso da ociosidade, pois as MVs alocadas para um BoT são desligadas após o final do seu tempo de vida. Por outro lado, a política proposta instancia apenas uma quantidade complementar de MVs tal que haja aproveitamento da ociosidade do *cluster* para o novo BoT. Conforme descrito no capitulo 3, a política proposta realiza o desligamento de MVs após a finalização ou cancelamento de um BoT e tem como escopo de uso os provedores de nuvem. A proposta é modelada e avaliada, através de simulações, em Redes de Petri, e comparada com a heurística de (MARSHALL; KEAHEY; FREEMAN, 2010).

Outras contribuições desta dissertação:

1. Uma solução para execução de aplicações de alto desempenho submetidas em rajadas para (*clusters*) em nuvem que leva em consideração características relativas às aplicações, aos recursos que serão alocados e os já alocados, diferentemente de muitas soluções presentes na literatura, as quais consideram aspectos da aplicação ou aspectos dos recursos virtualizados;

2. Uma modelagem feita com Redes de Petri de um provedor de recursos virtualizados com suporte a adaptação de *clusters* virtuais. Com essa modelagem é possível realizar simulações de alocações e liberação de *clusters* de MVs de acordo com algoritmos propostos e os estudados na literatura (MARSHALL; KEAHEY; FREEMAN, 2010).

## 1.5 Organização do documento

No Capítulo 2 é apresentada uma visão geral dos fundamentos conceituais e desafios relevantes ao contexto da execução de aplicações paralelas de alto desempenho em nuvens, especialmente, em *clusters* virtuais. É apresentada também uma discussão sobre a elasticidade de *clusters* virtuais. Discutimos trabalhos relacionados com a problemática da gerencia de *clusters* virtuais para execução de aplicações paralelas e da elasticidade de *clusters*.

No Capítulo 3, é apresentado o problema de alocação de VMs em *clusters*, destacando suas principais variáveis e uma visão geral da solução proposta para de alocação dinâmica de VMs, com seus componentes principais e respectivas funcionalidades. Em

seguida, descrevemos a heurística e políticas utilizadas e propostas. Detalhamos a abordagem de monitoramento e adaptação de *clusters* virtuais. Por fim, mostramos como os algoritmos de instanciação e desligamento atuam nos *clusters* para diminuir a quantidade de recursos alocados sem comprometer o desempenho nem gerar sobrecargas nos *clusters*.

No Capítulo 4, é apresentada a modelagem de um provedor de recursos virtualizados com suporte a adaptação de *clusters* virtuais. Usando Redes de Petri coloridas detalhamos as páginas que compõem a modelagem, descrevendo as principais cores e funções criadas, além de discutir o funcionamento da Rede de Petri através da descrição de troca de fichas entre os lugares.

No Capítulo 5, definimos as métricas utilizadas para avaliar o desempenho dos algoritmos propostos. Em seguida, apresentamos em detalhes o *setup* da experimentação realizada destacando o *workload* utilizado.

No Capítulo 6, é feita uma síntese das principais conclusões e contribuições, apontando perspectivas de trabalhos futuros.

No Anexo 1, é apresentada uma visão geral sobre as Redes de Petri.

No Apêndice A, são encontradas as descrições das cores utilizadas da modelagem com Rede de Petri.

# 2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Este capítulo destina-se à introdução dos fundamentos conceituais sobre os temas computação em nuvem, virtualização e alocação de recursos, os quais estão diretamente relacionados a este trabalho. O objetivo deste capítulo é apresentar conceitos fundamentais e definições relacionados com o desenvolvimento deste trabalho. Na seção 2.1 a computação em nuvem é discutida, bem como suas principais características, modelo de serviço e implementações. Na seção 2.2 discute-se sobre a virtualização destacando sua aplicação para geração de *clusters* virtualizados. Na seção 2.3 faz-se uma discussão sobre a aplicabilidade da computação em nuvem para a *High-Performance Computing* (HPC) dando ênfase às aplicações do tipo saco de tarefas (*Bag-of-Tasks*, BoT). Os trabalhos relacionados são apresentados na seção 2.4 e, por fim, na seção 2.5 são apresentadas as conclusões.

## 2.1 Computação em nuvem

A computação em nuvem tem o objetivo de fornecer serviços computadorizados sob demanda com pagamento baseado no uso (*pay-per-use*) (BUYYA et al., 2009). Pretende ser global e prover serviços em larga escala, os quais atendam desde o usuário final que hospeda seus documentos pessoais na Internet, até empresas que terceirizam toda infraestrutura de TI (SOUSA et al., 2010). Além dessas, a computação em nuvens possui muitas outras características. Em função dessas muitas características, vários autores lançaram definições para a computação em nuvem. Dentre as várias definições encontradas na literatura, adotamos neste trabalho a seguinte:

> Computação em nuvem é um modelo para acesso conveniente, sob demanda, e de qualquer localização, a uma rede compartilhada de recursos de computação (isto é, redes, servidores, armazenamento, aplicativos e serviços) que possam ser prontamente disponibilizados e liberados com um esforço mínimo de gestão ou de interação com o provedor de serviços. Este modelo de nuvem é composto de cinco características essenciais, três modelos de serviço e quatro modelos de implementação (MELL; GRANCE, 2009).

### 2.1.1 Características essenciais da computação em nuvem

A elasticidade rápida de recursos, amplo acesso e a medição de serviço são características básicas para compor uma solução de computação em nuvem (SOUSA et al., 2010) e são brevemente explicadas a seguir:

- Autoatendimento sob demanda – os recursos são providos de acordo com a demanda do consumidor de modo automático;
- Amplo acesso à rede – os recursos são disponibilizados através da Internet e acessados por plataformas-cliente heterogêneas com qualquer capacidade de processamento (*e.g.* telefones celulares, *tablets*, notebooks e estações de trabalho);
- Agrupamento de recursos (*pooling*) – os recursos são agrupados a fim de atender múltiplos consumidores, com diferentes recursos físicos e virtuais atribuídos dinamicamente e de acordo com a demanda do consumidor. O consumidor não precisa conhecer a localização física exata dos recursos, podendo somente conhecer a localização em um nível mais alto de abstração, por exemplo: país, estado ou *datacenter*;
- Elasticidade rápida – os recursos podem ser provisionados e liberados, em alguns casos automaticamente, para se ajustar à demanda crescente ou decrescente. Para o consumidor, as capacidades disponíveis para provisionamento frequentemente parecem ser ilimitadas e podem ser apropriadas em qualquer quantidade e a qualquer momento. A elasticidade rápida, se comparada com outros paradigmas, permite que os recursos possam ser provisionados e liberados em um curto espaço de tempo;
- Medição do serviço – controla e aperfeiçoa automaticamente o uso dos recursos, aproveitando uma capacidade de medição em algum nível de abstração apropriado para o tipo de serviço (por exemplo, contas de armazenamento, processamento, largura de banda e usuário ativo). O uso de recursos pode ser monitorado, controlado e posto em relatórios, proporcionando transparência, tanto para o provedor quanto para o consumidor;

Como veremos no capítulo 3, este trabalho possui características de autoatendimento sob demanda, de agrupamento de recursos e de elasticidade rápida. Sendo a elasticidade rápida, possivelmente a características mais utilizadas pela proposta apresentada. Por isso, a seguir é feita uma breve discussão sobre elasticidade.

#### 2.1.1.1 Elasticidade

Elasticidade é a característica da nuvem que permite o redimensionamento dinâmico, flexível, automatizado e frequente de recursos que são oferecidos em função de uma carga dinâmica que é enviada para a plataforma de execução (KUPERBERG et al., 2011). Elasticidade pode ser considerada como um dos principais benefícios da computação em nuvem. Nessa definição, "plataforma de execução"pode ser uma MV, um conjunto de MVs ou o

*hypervisor* do sistema operacional hospedeiro. Elasticidade carrega um potencial para otimizar a produtividade e utilização do sistema, mantendo o SLA e o QoS, bem como economia de energia e custos. O termo "elasticidade"é usado frequentemente em propagandas de provedores de infraestrutura de nuvem, como na Amazon Elastic Compute Cloud (EC2). A Amazon oferece uma API de dimensionamento automatizado para seus clientes do EC2. Dessa forma, o cliente pode controlar o número de instâncias de MVs através de políticas que observam, por exemplo, a utilização média de CPU em um grupo de MVs.

A definição de elasticidade de recursos é baseada no termo mais geral, escalabilidade. De maneira geral, a investigação sobre a escalabilidade de sistemas não considerou adaptação dinâmica e também ignorou o aspecto temporal. Contudo, métricas de escalabilidade existentes observam os tempos de reação, extensão e o comportamento de atividades automatizadas durante o redimensionamento de uma MV. Métricas de escalabilidade já foram propostos por (JOGALEKAR; WOODSIDE, 2000), que avaliam a produtividade de um sistema em diferentes níveis de escala. Essas métricas propostas não podem expressar a elasticidade, uma vez que não têm o aspecto temporal de ações de escala automáticas. O ponto de vista de escalabilidade dinâmica e adaptável exige mais métricas sobre, por exemplo, o quão rápido a escala de recursos de um sistema pode ser executada.

A escalabilidade é um termo que pode ser aplicado tanto para aplicações como em plataformas de execução. Nesta seção, apresentamos uma terminologia mais precisa para escalabilidade. Escalabilidade de aplicativo é uma propriedade que significa que o aplicativo mantém suas metas de desempenho/SLA, mesmo quando há um aumenta sua carga de trabalho (até um certo limite). Essa carga de trabalho limite destaca o fato de que a escalabilidade não é infinita. Logo, acima de uma certa carga de trabalho, o aplicativo não será capaz para manter suas metas de desempenho/SLAs.

Correspondentemente, a escalabilidade da plataforma é a capacidade da plataforma de execução proporcionar e recursos adicionais conforme necessário (ou explicitamente solicitado) por uma aplicação. Um exemplo, pode ser um sistema operacional (onde o sistema de execução compreende hardware e possivelmente um *hypervisor*) ou uma loja da web (onde a plataforma de execução compreende a *middleware*, sistema operacional e hardware).

Existem duas "dimensões"de escalabilidade de plataforma, e um sistema pode ser escalável em nenhuma, uma ou ambas (SULEIMAN et al., 2012):

- Escalabilidade vertical: melhora-se a capacidade do *hardware*, incrementando individualmente os *hosts* existentes, como por exemplo, por meio da adição de mais memória física, de mais processadores ou do aumento do tamanho do dispositivo de armazenamento;
- Escalabilidade horizontal: consiste em adicionar mais *hosts* à configuração atual de tal modo que seja possível distribuir as requisições entre estes novos *hosts*. Este tipo de escalabilidade oferece maior flexibilidade (CZAJKOWSKI et al., 2005).

### 2.1.2 Modelos de serviços

A computação em nuvem é composta de três modelos de serviços, os quais definem um padrão arquitetural para soluções de computação em nuvem (SOUSA et al., 2010):

- Software como Serviço (SaaS - *Software as a Service*) – destina-se à utilização de aplicativos do provedor rodando em uma infraestrutura de nuvem. Tem como público-alvo, em geral, usuários finais;
- Plataforma como Serviço (PaaS - *Platform as a Service*) – fornece sistema operacional, linguagens de programação e ambientes de desenvolvimento para as aplicações, auxiliando a implementação de sistemas de software em uma infraestrutura de nuvem;
- Infraestrutura como serviço (IaaS - *Infrastructure as a Service*) – destina-se ao provisionamento de processamento, armazenamento, redes e outros recursos de computação onde o cliente é capaz de implementar e executar softwares arbitrários, que podem incluir sistemas operacionais e aplicativos.

Apesar dos serviços mais utilizados serem geralmente aplicações interativas do tipo SaaS, a computação em nuvem vem sendo bastante utilizada para fazer processamento paralelo em lote de grandes massas de dados, o que possibilita a análise de vários terabytes de dados de forma mais rápida, ao executar processamentos em paralelo (ARMBRUST et al., 2009). A política e os algoritmos propostos neste trabalho (vide capítulo 3) atuam em conjunto para realizar a alocação de MVs em *clusters* virtuais para atender uma demanda variável por recursos. Logo, a solução proposta nos capítulos 3 e 4 tem como escopo de uso os provedores de recursos computacionais virtualizados, ou seja, o provimento de infraestrutura como serviço.

### 2.1.3 Modelos de implementação

Quanto ao acesso e à disponibilidade, há diferentes tipos de modelos de implantação para os ambientes de computação em nuvem. A restrição ou abertura de acesso dependem do processo de negócios, do tipo de informação e do nível de visão desejado (SOUSA et al., 2010), a saber:

- Nuvem privada – A infraestrutura de nuvem é provisionada para uso exclusivo por uma única organização, podendo compreender múltiplos clientes (por exemplo, unidades acadêmicas);
- Nuvem comunitária – A infraestrutura de nuvem é provisionada para uso exclusivo por uma comunidade específica de clientes de organizações que têm preocupações comuns;

- Nuvem pública – A infraestrutura de nuvem é provisionada para uso aberto ao público em geral;
- Nuvem híbrida – A infraestrutura de nuvem é uma composição de duas ou mais infraestruturas de nuvem distintas (privada, comunitária ou pública) que permanecem como entidades únicas, mas são unidas por tecnologia padronizada ou proprietária que permita a portabilidade de dados e aplicativos.

Diante de todas essas características e dos 4 modelos de implementação é possível afirmar que para viabilizar um modelo de computação em nuvem é necessário fazer uso de várias tecnologias. Algumas das tecnologias que permitiram a implementação da computação em nuvem são recentes, outras nem tanto, tais como a virtualização. A virtualização forma a base da computação em nuvem, uma vez que fornece a capacidade de reunir recursos de computação e, dinamicamente, atribuir ou reatribuir recursos virtuais para aplicações sob demanda (ZHANG; CHENG; BOUTABA, 2010). Outras tecnologias relacionadas à computação em nuvem são:

- Sistemas de armazenamento distribuído;
- Orquestração de fluxo de serviço e fluxo de trabalho;
- Web 2.0;
- Serviços Web e arquitetura orientada a serviços (SOA).

Na próxima seção, a virtualização será discutida brevemente, a fim de destacar suas principais vantagens e suas possíveis aplicações.

## 2.2 Virtualização

A virtualização foi criada nos anos de 1960 pela IBM e resultou no desenvolvimento do primeiro modelo de máquina virtual. Surgiu na época em que os *mainframes*, computadores grandes e caros, tinham que ser compartilhados por um grande número de usuários, os quais tipicamente desejavam utilizar aplicativos diferentes. Com o desenvolvimento tecnológico e consequentemente a queda do custo do hardware, os *mainframes* foram sendo substituídos por computadores pessoais, diminuindo, assim, o interesse pela virtualização.

Quando os grandes e caros mainframes deram lugar aos, também grandes e caros, *clusters* computacionais, que são compartilhados por muitos usuários e que possuem distintas exigências, o uso da virtualização voltou a se tornar popular. Outro motivo para o "retorno"da virtualização está no grande avanço das novas tecnologias de virtualização que torna muito baixo o *overhead* resultante do uso da mesma. Com base nisso, é possível notar que a virtualização consiste na abstração das características reais dos

recursos computacionais em um ambiente capaz de atender as necessidades específicas de uma aplicação ou usuário, possibilitando a sua execução e o seu uso independente do meio em que se encontra. Contudo, a definição muito comum de virtualização é:

> A virtualização consiste em estender ou substituir um recurso existente, ou uma interface, por outro de modo a imitar um comportamento. Isso é feito através de uma camada de software responsável por transformar ações de um sistema A em ações equivalentes em um sistema B (isomorfismo) (ROSENBLUM, 2004).

O recurso que a definição se refere por vezes tem sido associado a computadores (máquinas). Nesse caso, a virtualização produz uma máquina virtual (MV). Para tal, de acordo com a definição, é necessária também a existência de um software responsável por fazer a interlocução entre o sistema estendido e o sistema virtual. No caso das máquinas virtuais esse sistema é chamado de Monitor de Máquinas Virtuais (VMM). Segundo (LAUREANO, 2006), as principais características e vantagens da virtualização de computadores são:

- Aperfeiçoamento nos testes de novos sistemas operacionais e aplicações;
- Gerenciamento, migração e replicação de computadores, aplicações ou sistemas operacionais;
- Provimento de um serviço dedicado para um cliente específico com segurança e confiabilidade;
- Redução na aquisição de hardware;
- Redução no custo de manutenção, suporte técnico e treinamento;
- Aproveitamento da ociosidade das máquinas;
- Padronização do ambiente e melhorias na segurança;
- Diminuição do tempo de recuperação em caso de desastres;
- Viabilização de melhor disponibilidade com menos redundância;
- Utilização de vários sistemas operacionais simultaneamente;
- Simulação configurações de hardware.

A virtualização permite a criação de diversas MVs sob demanda. É neste aspecto que a computação em nuvem influenciando nas técnicas de programação, permitiu que as empresas com grandes tarefas orientadas a lote pudessem obter resultados tão mais rápido quanto seus programas pudessem ser escalados (ARMBRUST et al., 2009). Por esses motivos, a virtualização tornou-se a base do modelo de computação em nuvem, em que as aplicações são encapsuladas dentro de MVs (SONNEK; CHANDRA, 2009).

Problemas específicos da utilização das MVs, tais como: alocação, migração, garantia de desempenho vêm sendo amplamente estudados. Por exemplo, em relação à perda de desempenho das MVs ocasionada pelas migrações entre servidores físicos, a solução proposta por (REGO et al., 2011) garante que a capacidade de processamento de uma MV seja mantida independente da máquina física em que ela foi alocada.

A imprevisibilidade do desempenho na nuvem é um grande problema para muitos usuários e é considerado como um dos maiores obstáculos para computação em nuvem (ARMBRUST et al., 2009). Em relação ao processo migratório, observa-se uma degradação no desempenho das aplicações executadas nas MVs, degradação esta que é avaliada em (MAGALHãES; SOARES; GOMES, 2011). Os autores apresentaram uma avaliação de desempenho das abordagens *stop-and-copy* e pré-cópia de migração de MVs. O objetivo dos autores foi compreender o impacto gerado pelas duas abordagens no desempenho do *testbed* real executado, destacando as vantagens e desvantagens de cada uma delas em diversos cenários.

A alocação de recursos computacionais tem sido estudado há décadas, o que resultou em diversas abordagens que tendem a ser especializadas para um determinado cenário de uso, como, por exemplo, a alocação de MVs. No cenário de alocação de MVs, o problema se estende ao de como prover recursos computacionais compartilhados de maneira eficiente (SOTOMAYOR, 2010). A criação de recursos compartilhados através da virtualização na computação em nuvem também ganhou destaque na área de sistemas distribuídos, permitindo que os ambientes de Grades e *Clusters* fossem virtualizados. Na próxima subseção será apresentado um resumo dos benefícios que a virtualização trouxe, especificamente, no ambiente de computação em *cluster* que é o foco desde trabalho.

### 2.2.1 *Clusters* Virtuais

A computação em *cluster* é um tipo de sistema de processamento distribuído ou paralelo, o qual consiste de um conjunto de computadores interligados trabalhando juntos (nós) como se fosse um único recurso de computação integrada (BUYYA et al., 2009). Em um ambiente de *cluster* computacional a virtualização se torna uma solução para o gerenciamento de recursos compartilhados entre os diversos usuários, permitindo o compartilhamento dos nós físicos. Além disso, através da consolidação de diversos serviços em um mesmo servidor, é possível reduzir os gastos de energia do *cluster*. O compartilhamento de recursos em *clusters*, associado à crescente disponibilidade de poder computacional e à habilidade das MVs de garantir o isolamento entre as diversas instâncias, possibilitou o surgimento de *clusters* virtualizados ou *clusters* virtuais (AMORIM; WHATELY; DUTRA, 2009). Nessa perspectiva (FOSTER et al., 2006) definem *cluster* virtuais como:

> Grupos de máquinas virtuais projetados para executar aplicações paralelas e compartilhar uma infraestrutura confiável de computação. Sua implantação permite especificar a alocação de recursos diferentes para diferentes grupos de usuários e aplicações.

Com base na definição acima, é possível formalizar o conceito de *cluster* virtual como sendo o conjunto de nós (MVs) obtido a partir da demanda de recursos requisitados pelo usuário. Logo, cada *cluster* virtual ($c_i$) é um somatório de $n$ máquinas virtuais (PRUEKSAAROON; VARAVITHYA, 2012):

$$c_i = \sum_{j=0}^{n} MV_{j,i}, \tag{2.1}$$

em que $i$ é o índice de um *cluster* $c_i$, $j$ é o índice das MVs ($MV_{j,i}$) do *cluster* $c_i$, com $j \in \{0, n\}$, $n$ é quantidade de MVs do *cluster* e $MV_{0,i}$ é o nó central do *cluster* $c_i$. Considerando que um provedor pode alocar até $k$ *clusters*, então os $k$ *clusters* alocados constituem um conjunto $C$ de diferentes *clusters* virtuais, em que cada *cluster* $c_i \in C$, com $i \in \{0, k\}$. Na figura 1 é possível observar uma ilustração com dois *cluster* representados com a notação apresentada. O *cluster* $c_0$ possui 8 nós de trabalho e o nó central $MV_{0,0}$. O *cluster* $c_1$ possui 6 nós de trabalho e o nó central $MV_{0,1}$. Note que na ilustração do *cluster* $c_1$ há uma região não preenchida, o que indica que o *cluster* pode admitir mais MVs através da elasticidade.

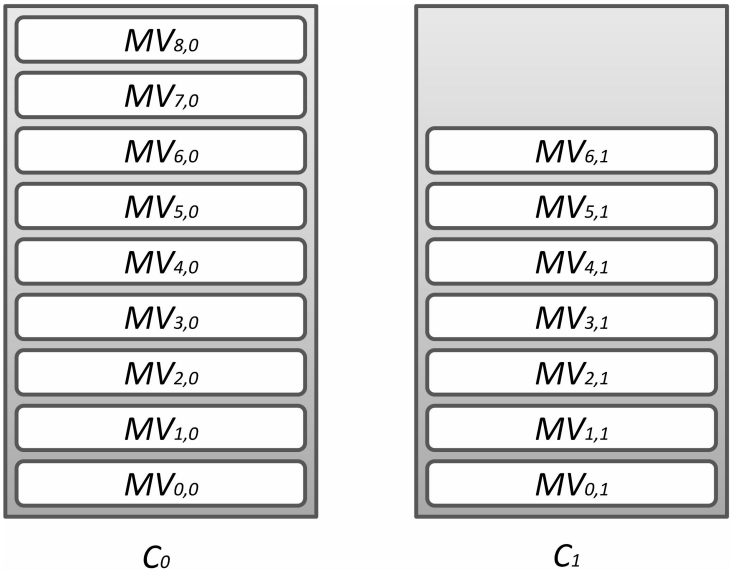


Figura 1: Exemplos de *clusters* virtuais de acordo com a notação utilizada.

Uma das principais características da computação em nuvem é a elasticidade que oferece serviços e aplicações à medida que são necessários, assim como no caso dos *clusters* virtuais, que proveem serviços que são utilizados à medida que são requeridos. Logo, a distribuição desses serviços entre os usuários pode ser rapidamente alterada, o que exige um suporte para tal dinamismo. A virtualização é o componente responsável pela característica dinâmica dos *clusters* virtuais, ou seja, ela permite que os ambientes virtuais de cada usuário possam ser ampliados ou reduzidos dinamicamente de maneira a atender à demanda (CHIRIGATI, 2009).

Em função de limites impostos pelos recursos físicos que abrigam os recursos vir-

tualizados, dos níveis de desempenho desejados, dos índices de qualidade almejados, do recursos financeiros alocados ou até mesmo dos limites energéticos estipulados, o crescimento dos recursos virtualizados via elasticidade são limitados. Contudo, para o escopo desse trabalho serão considerados apenas aspectos relacionados com o desempenho que limitam o crescimento de um *cluster* virtualizado. Nessa perspectiva, utilizamos nesse trabalho dois parâmetros responsáveis por quantificar os limites de crescimento em relação a quantidade de MVs e utilização de um *cluster*: i) A quantidade de MVs, visto que o desempenho de um *cluster* é uma função do número de nós de trabalho que compõem o *cluster* (MONTERO; MORENO-VOZMEDIANO; LLORENTE, 2011). ii) A utilização do *cluster*, a má gerencia da utilização pode gerar instabilidade ou eventuais falhas causadas por sobrecarga no nó coordenador (WERSTEIN; SITU; HUANG, 2006). Vale destacar, ainda, que a baixa utilização média dos recursos é uma das principais causas da ineficiência na alocação de recursos em *clusters* (LIAO; JIN; LIU, 2012).

## 2.3 Computação em Nuvem para HPC

O uso de computação intensiva em apoio à pesquisa e ao processamento de grande quantidade de dados científicos exige utilização compartilhada dos recursos. A infraestrutura para o tratamento desse tipo de informação e as ferramentas para auxiliarem nessa tarefa devem possibilitar que seus usuários (*e-scientists*) as utilizem de forma transparente. A otimização do uso dos recursos em grandes centros de HPC e sua integração com a infraestrutura de *e-science* apresentam desafios, destacando-se a necessidade de algoritmos mais eficientes para o processamento da informação e para o uso racional dos recursos (RIEDEL et al., 2007).

Em (MERGEN et al., 2006), argumenta-se que o uso da virtualização em HPC possibilita tanto a configuração dedicada dos sistemas quanto a manutenção de compatibilidade com sistemas legados, ou seja, a coexistência nos mesmos recursos. A extensão do número de máquinas virtuais permite, por exemplo, que o VMM apresente, para um *cluster* virtual, a ideia da existência de inúmeros nós virtuais em poucos nós reais, permitindo que um teste em escala real de aplicações, tipo *Message Passing Interface* (MPI), possa ser feito mesmo na presença de poucos recursos, ainda que estes estejam sendo utilizados por outras aplicações.

*E-science* envolve a construção de modelos matemáticos para resolver problemas científicos, sociais e de engenharia. Esses modelos geralmente requerem um grande número de recursos de computação para realizar experimentos de larga escala ou para reduzir a complexidade computacional em um prazo razoável. Estas necessidades foram inicialmente tratadas com infra-estruturas dedicadas à computação de alto desempenho como *clusters* ou com um *pool* de máquinas em rede no mesmo departamento, gerenciados por algum softwares como o Condor (VECCHIOLA; PANDEY; BUYYA, 2009).

Nesse cenário, a computação em nuvem surge como uma plataforma de computação alternativa para suprir a crescente demanda dos pesquisadores por recursos. Em-

bora os limites atualmente impostos pelos provedores de IaaS não impeçam que a maioria dos clientes enxerguem o serviço provido como uma fonte infinita de recursos, este não é o caso para algumas aplicações, as quais requerem a instanciação de um sistema com, geralmente, centenas de MVs. O projeto Belle II Monte Carlo (SEVIOR; FIFIELD; KATAYAMA, 2010), por exemplo, requer de 20.000 a 120.000 MVs para o processamento em tempo aceitável dos dados produzidos em três meses de experimentos. Ou seja, nesse projeto há uma altíssima demanda por recursos de forma esporádica. Esse padrão de consumo é muito comum entre os usuários que executam aplicações do tipo saco de tarefas (Bag-of-Tasks, BoT) (COSTA et al., 2011), tópico detalhado na próxima seção.

### 2.3.1 Aplicações do Tipo Saco de Tarefas (Bag-of-Tasks)

As aplicações do tipo saco de tarefas (Bag-of-Tasks, BoT), são aplicações massivamente paralelas, de computação intensiva, constituídas por uma grande quantidade de tarefas que podem ser executadas de forma independente (FRAGA; BRASILEIRO; GUERRERO, 2011). Uma aplicação deste tipo pode ser executada em vários provedores de recursos de modo concorrente, tal que seu tempo de execução seja aquele esperado pelo usuário. Este tempo de execução é calculado a partir da soma das estimativas de tempo de execução de cada tarefa do BoT e do tempo de espera por recursos. Na prática, em geral, o usuário faz a superestimação dos tempos de execução a fim de evitar que a aplicação seja abortada (NETTO; BUYYA, 2011).

Desde a década de 90, grades de *desktops* têm sido amplamente utilizadas para a execução de aplicações BoT. Em particular, as grades se consolidaram como as plataformas mais populares para a execução de aplicações de *e-science*. Tais infraestruturas se baseiam no aproveitamento oportunista de recursos, utilizando ciclos ociosos de máquinas *desktops* não dedicadas. A demanda de recursos computacionais gerada pelas aplicações do tipo BoT tem tornado muito popular o uso da computação em nuvem para execução de aplicações científicas (COSTA et al., 2011). Nessa perspectiva, para executar aplicações do tipo BoT faz-se o uso de *clusters* virtuais (ANEDDA et al., 2010). Por outro lado, a demanda por recursos pelos BoTs ocorre em rajadas (IOSUP; EPEMA, 2011). Uma rajada é caracterizada por breves períodos de alta demanda por recursos seguidos por longos períodos de baixa demanda. As implicações que essas diferenças têm no desempenho de *clusters* virtuais é uma das questões investigadas neste trabalho.

A demanda em rajadas faz com que os *clusters* virtuais oscilem entre períodos de alta e de baixa contenção por recursos. Quando há uma rajada de demanda, a quantidade de tarefas para ser executada é maior que a quantidade de recursos disponíveis no *cluster*, caracterizando um cenário de alta contenção por recursos. Por outro lado, após uma rajada de demanda o *cluster* opera em baixa contenção, pois a quantidade de tarefas para serem executadas é menor que a quantidade de recursos disponíveis. Consequentemente, nos períodos de baixa contenção, existem recursos ociosos no *cluster*, i.e., com baixa utilização média. Logo, manter os recursos ociosos não é uma prática eficiente em termos de desempenho, sendo a reutilização da ociosidade importante, nessa perspectiva (BRAGA

et al., 2012).

## 2.4 Trabalhos Relacionados

Para a concepção da proposta desta dissertação, foram estudados alguns trabalhos recentes da literatura que abordam temas relacionados com o conceito de *cluster* virtual e das aplicações da HPC, especificamente, as do tipo *bag-of-tasks*.

Serão apresentados, a seguir, soluções relacionadas a esses temas, descrevendo suas principais características. Esses trabalhos foram organizados em dois tópicos: o uso de recursos virtualizados pela HPC e adaptação de recursos virtualizados via elasticidade.

**O uso de recursos virtualizados pela HPC**

Segundo (KHATUA; GHOSH; MUKHERJEE, 2010), na pesquisa de Computação de Alto Desempenho (CAD), a utilização ótima dos recursos tem sido sempre considerada como um dos principais desafios. Certos problemas de utilização de recursos surgem mesmo em ambientes simples, como estações de trabalho individuais, eles são agravados em ambientes complexos, especialmente os dinâmicos e heterogêneos. Algumas soluções foram propostas na literatura para ambientes tradicionais, como *clusters*. Estas soluções são baseadas, principalmente, em técnicas de balanceamento dinâmico de carga. No entanto, atualmente, têm sido realizados muitos trabalhos para ambientes virtuais a este respeito.

Em (GAVRILOVSKA et al., 2009), é estudada a importância de melhorar as soluções de virtualização para suportar as arquiteturas com múltiplos núcleos. Nessa perspectiva, (NAZIR; LIU; SORENSEN, 2009) propõem um meta-escalonador que atua controlando completamente todos os *clusters* disponíveis. O *framework* proposto pelos autores permite selecionar recursos computacionais apropriados de acordo com a em escala global e criar *clusters* virtuais, realizando escalonamento local das tarefas e priorizando a redução do tempo de resposta. Contudo, o *framework* decide criar um novo *cluster* a partir das informações relativas à quantidade de núcleos por MV e em relação a utilização de I/O.

Em (EVANGELINOS; HILL, 2008), a fim de estudar o desempenho das operações de I/O em uma infraestrutura de nuvem, os autores realizaram vários experimentos com o pacote NAS *Parallel Benchmarks*. Destaque para a influência da largura de banda no sucesso das aplicações HPC. Os autores utilizaram os *clusters* virtuais como alternativa de execução ou para fazer uma comparação de desempenho. Além disso, eles constataram que o uso de *clusters* sob demanda, associados à customização com foco em cenário específico é vantajosa mesmo com perda de desempenho.

Em (XU; BAI; LUO, 2009), discute-se o desempenho das máquinas virtuais na execução de aplicações paralelas. Os autores investigam o desempenho de três paradigmas de programação paralela típica, incluindo OpenMP, MPI e um híbrido de OpenMP e MPI no sistema de virtualização Xen. Os autores mostraram que o desempenho do pro-

grama paralelo tradicional em VMs Xen está perto do desempenho no ambiente nativo não-virtualizados, se há pouca comunicação ou sincronização entre *threads* ou os processos. O desempenho da computação em nuvem para aplicações comerciais é estudado em (STANTCHEV, 2009).

Em (BARKER; SHENOY, 2010), é feito um estudo comparativo entre a plataforma virtualizada Amazon's EC2 e uma estrutura local real para aplicações multimídia, quantificando métricas a respeito do consumo de processamento, armazenamento e rede. Nos experimentos, os autores utilizam *microbenchmarks*, que são pequenas aplicações que testam intensivamente o sistema alvo. Entretanto, os autores não fazem considerações a respeito de aplicações paralelas. Em (MELLO et al., 2010), são utilizados *clusters* virtuais para determinar a perda de desempenho para aplicações paralelas em ambientes virtualizados. Para tal, são utilizados *benchmarks* como carga de trabalho para estudar a perda de desempenho na taxa de comunicação e do processamento. Os autores sugerem que o virtualizador e a aplicação que será executada devem ser submetidos a uma análise criteriosa. Sugerem ainda que até mesmo o sistema operacional deve ser considerado ao se escolher um *cluster*.

Além de trabalhos citados, algumas outras ferramentas têm sido propostas para melhorar a utilização dos ambientes virtualizados para a HPC. De modo geral, essas ferramentas apresentam ambientes de gerenciamento de máquinas virtuais ou *clusters* virtuais, oferecendo operações como criação, destruição e migração desses componentes. É possível ver exemplos em (XU et al., 2009), (ONG et al., 2009) e (MANCINI; RAK; VILLANO, 2009), os quais propõem escalonamento dos *jobs* entre os *clusters*, sem levar em conta as características específicas das aplicações ou da nuvem em que os recursos virtualizados serão alocados.

Percebe-se, nos trabalhos relacionados, que a computação em nuvem ainda possui diversas desafios para executar aplicações da HPC de modo eficiente. Os problemas relacionados com a comunicação entre tarefas são os mais recorrentes. Neste cenário, as aplicações do tipo BoT possuem uma vantagem pois a comunicação entre as tarefas é praticamente nula. Visto que essas aplicações podem ser executadas em diversos recursos de maneira assíncrona, os BoTs podem utilizar a computação em nuvem, uma vez que as mesmas oferecem bastante flexibilidade para alocação de recursos através da elasticidade. Nessa perspectiva, os recursos virtualizados podem ser providos por demanda de acordo com determinadas previsões de consumo ("orçamentos") (OPRESCU; KIELMANN; LEAHU, 2011), (OPRESCU; KIELMANN, 2010). Neste trabalho, a determinação do "orçamento"é feita a partir da características a do BoT submetido e da nuvem em que o *cluster* será executado.

**Adaptação de recursos virtualizados via elasticidade**

Baseados no ganho de poder de processamento devido ao acréscimo do número de MVs em infraestruturas de nuvens locais versus infraestruturas de nuvens remotas, (MONTERO; MORENO-VOZMEDIANO; LLORENTE, 2011) propuseram um modelo analítico para cálculo da vazão de processamento (*jobs*/s) em função da adição de nós, aferindo

inclusive o limiar superior para o total de nós. Contudo, esse modelo considera apenas o número de nós do *cluster*. Na sua proposta, além do número de nós, é considerada a utilização do *cluster*, o tempo máximo de execução do *workload*, a fila de *workloads* e o *overhead* da nuvem. Outros trabalhos apresentam modelos para aperfeiçoar o consumo de banda em nuvens em relação à elasticidade (HUU et al., 2011) e para melhoria do processo de publicação de *clusters* (YAMASAKI; MARUYAMA; MATSUOKA, 2007).

(LIM; BABU; CHASE, 2010) utilizaram a teoria clássica de controle para definir políticas de controle para o monitoramento de recursos em nuvem levando em consideração a taxa de utilização da CPU e largura de banda. Os autores também estudaram o impacto da reconfiguração do *cluster* versus o cumprimento do *Service Level Agreement* (SLA) e propuseram também uma política para balanceamento de carga. Contudo, suas políticas propostas não levam em consideração as filas de *workloads* e o *overhead* da nuvem, os quais são tratados na política proposta.

Alguns trabalhos tratam a escalabilidade de *clusters* em nuvem que executam aplicações Web. Tais aplicações fazem uso de servidores específicos, de modo que, há camadas de *clusters* de servidores, por exemplo: um *cluster* para servir às requisições Web, um *cluster* para execução da lógica de negócios da aplicação e outro para o banco de dados (CHI; QIAN; LU, 2011) (Simon et. al., 2011). Nessas soluções é feito o emprego de vários algoritmos que atuam especificamente em cada camada.

Uma metodologia bastante adotada para a adaptação dos clusters virtuais à carga de trabalho é o balanceamento de carga. Nesta estratégia, vale destacar a migração de nós. Em (MANETTI et al., 2010) é proposto o NEPTUNE-IaaS, um software capaz de gerenciar todo o ciclo de vida de um *cluster* virtual. No ciclo de vida proposto, o estado "RUNNING"pode ser substituído apenas pelos estados "PAUSED"ou "MODIFIED"durante sua atuação. A migração de nós ocorre entre *clusters* remotos para realizar o balanceamento de carga ou a recuperação de falhas. Os autores propõem ainda uma ferramenta que define dinamicamente a topologia da rede virtual baseada na heurística de Lin-Kernighan.

Uma abordagem para migração de MVs baseada no método Fuzzy TOPSIS é proposta por (JING S., 2011). O método é utilizado para detecção de pontos de possíveis falhas. Os autores propõem um modelo de decisão que é focado na utilização de recursos de processamento, armazenamento e I/O. Usando ainda a migração de MVs há trabalhos que estudam o comportamento da migração de um *cluster* completo, como em (ANEDDA et al., 2010). A abordagem discutida pelos autores tenta minimizar a quantidade de páginas de estados da VM em relação à memória e o disco que precisam ser transferidas. Os autores concentraram-se no estudo sobre o equilíbrio entre a compressão/descompressão do *cluster* e no transporte de dados.

Uma proposta de elasticidade em *clusters* virtuais para a HPC é encontrada no trabalho de (MARSHALL; KEAHEY; FREEMAN, 2010). Os autores apresentam um modelo (ElasticSite) que contempla a monitoração desde a fila de submissão de *jobs* até o encaminhamento dos resultados da execução sem considerar, no entanto, métricas intrínsecas ao *cluster* (e.g., taxa de utilização). O ElasticSite apresenta a política de alocação de

recursos *bursts* (rajadas). Nessa política, os autores consideram apenas o tempo esperado e informado pelo usuário para execução dos *workloads* (*walltime*) e o tempo médio para migrar, instanciar e desligar um conjunto de MV (*wastetime*, i.e., *overhead* da nuvem). Os autores não levam em consideração o estado atual do Adaptação de recursos virtualizados via elasticidade, o que pode ocasionar grande desperdício de recursos quando executam aplicações BoT submetidas em rajadas.

Estendendo a política *burts* de (MARSHALL; KEAHEY; FREEMAN, 2010), neste trabalho é apresentada uma nova política (ElasticCluster) em que são consideradas também a quantidade de MV em uso, taxa de utilização do *cluster* e o *overhead* da nuvem. A tabela 1 resume as principais diferenças entre os trabalhos correlatos brevemente discutidos nesta seção e a política de adaptação proposta. As lacunas relativas ao número de MV alocadas, utilização do *cluster*, *walltime*, fila de *workloads* e *overhead* da nuvem são todas preenchidas pela política proposta.

Tabela 1: Comparação das principais características das soluções.

| **Critérios** / **Trabalhos** | **# MV Alocadas** | **Utilização do *Cluster*** | ***walltime*** | **Fila de *workloads*** | ***Overhead* da Nuvem** |
|---|---|---|---|---|---|
| Montero *et al.* (2011) | Sim | Não | Não | Não | Não |
| Lim *et al.* (2010) | Sim | Sim | Sim | Não | Não |
| Marshall *et al.* (2010) | Não | Não | Sim | Sim | Sim |
| **ElasticCluster** | **Sim** | **Sim** | **Sim** | **Sim** | **Sim** |

## 2.5 Conclusões

O grande crescimento da computação em nuvem abriu um grande leque de possibilidade em relação ao fornecimento de recursos, de aplicações e *softwares*. Todo esse crescimento é, em grande parte, devido ao uso da virtualização que, recentemente, voltou a ser utilizada com mais intensidade graças aos avanços obtidos nas pesquisas em torno dos VMMs.

Umas das áreas da computação que se beneficiou como crescimento da computação em nuvem foi a computação da alto desempenho. Em especial a computação em *cluster*, que conta com os *clusters* virtualizados, objeto deste trabalho. Discutimos que esse *clusters* possuem um desempenho semelhante ao desempenho de *clusters* físicos e que ainda existem muitos desafios para tornar a computação em nuvem totalmente apta para a execução de aplicações paralelas, como por exemplo, melhoria da comunicação em máquinas virtuais e garantia desempenho de recursos virtuais.

Diante desse cenário, na computação em nuvem para a HPC, vimos neste capítulo que as aplicações do tipo BoT são indicadas para uso das nuvens, visto que as mesmas possuem baixa taxa de comunicação entre as suas tarefas e podem ser executadas em

diversos provedores de maneira assíncrona. Além disso, em função do típico padrão de submissão dos BoTs, as rajadas, percebeu-se a necessidade de realizar uma reutilização dos recursos ociosos gerados nos períodos entre a chegada de dois BoTs consecutivos.

Discutimos trabalhos recentes que propuseram soluções para execução de diversos tipos de aplicações da HPC a fim de minimizar a perda de desempenho em nível de *clusters* virtuais. Foi discutido também um conjunto de trabalhos que tratam do problema da escalabilidade de recursos virtuais através a elasticidade. Com base nesses trabalhos, no capítulo 3, será caracterizado o problema tratado nesse trabalho, destacando seus principais parâmetros, bem como a solução apresentada, composta por uma política de alocação e algoritmos responsáveis pela alocação e liberação de recursos.

# 3 CARACTERIZAÇÃO DO PROBLEMA E SOLUÇÃO PROPOSTA

A baixa utilização média dos recursos é uma das principais causas da ineficiência na alocação de recursos em *clusters* (LIAO; JIN; LIU, 2012). Um dos principais fatores para essa baixa utilização decorre da intenção dos provedores de recursos de garantir um bom desempenho em períodos de pico de demanda, então a capacidade de processamento é provisionada em excesso. Em vista disso, propomos nesta dissertação um algoritmo de alocação de MVs em *clusters* para que tais ambientes possam lidar com a natureza dinâmica da carga a fim de evitar a baixa utilização dos recursos. Dessa forma, na alocação de recursos para uma determina carga, as MVs ociosas de um *cluster* (o de maior disponibilidade) são reutilizadas, diminuindo o número total de máquinas iniciadas no *cluster*. Além disso, através da limitação do número máximo de MVs por *cluster* e da utilização máxima por *cluster* é possível também controlar a disponibilidade dos *clusters* a fim de evitar sobrecargas que comprometem as expectativas de QoS dos usuários, promovendo até mesmo a indisponibilidade dos recursos (BOBROFF; KOCHUT; BEATY, 2007) (BRAGA et al., 2012).

Este capítulo está organizado da seguinte maneira: a seção 3.1 apresenta uma caracterização formal do problema, com seus principais componentes e respectivas funcionalidades. A seção 3.2 descreve a solução proposta, apresentando as políticas e heurísticas da literatura utilizadas, bem como a política e a heurística propostas para determinar a quantidade de MVs e o tempo em que essas MVs devem ficar disponíveis para um determinado BoT. Ainda na seção 3.2, são apresentados os algoritmos de instanciação e desligamento de MVs propostos.

## 3.1 Caracterização do Problema

A necessidade por recursos computacionais tornou-se ao longos dos anos um requisito fundamental na ciência e na indústria. Em muitos casos essa necessidade é transiente, ou seja, um usuário utiliza os recursos computacionais apenas durante a execução de um determinada aplicação. Por exemplo, um cientista pode requisitar um grande número de computadores em *cluster* para executar uma simulação por algumas horas, liberando o uso computadores após a execução. Um professor universitário pode querer um *cluster* de computadores disponíveis para os estudantes apenas durante as aulas de laboratório de

uma disciplina, às vezes muito específicas durante a semana, e com uma configuração de software específico. Esta utilização transiente dos recursos apresenta o problema de *fornecimento eficiente de recursos compartilhados* (SOTOMAYOR; KEAHEY; FOSTER, 2008).

Devido a essa utilização transiente e à demanda variável por recursos, pode ser necessário complementar a infraestrutura já existente com recursos adicionais durante períodos imprevistos de aumento do tráfego, ou seja, esses recursos devem ser disponibilizados rapidamente, com pouco tempo de espera. Isso é possível com o uso da elasticidade no modelo de computação em nuvem, a qual permite o uso de recursos virtualizados, como os *clusters* virtuais. Apesar da quantidade de MVs a serem alocadas em um *cluster* virtual ser determinada principalmente pelo usuário que submete a aplicação, trabalhos recentes têm mostrado que, através de heurísticas, é possível determinar automaticamente essa quantidade tomando como base características da aplicação e do provedor de recursos virtualizados, de modo que a quantidade de MVs alocadas pode ser descrita pela clássica relação linear da teoria de controle (LIAO; HU; JIN, 2010) (LIM; BABU; CHASE, 2010):

$$A_{k+1} = A_k + K_i * (S_{ref} - S_k), \tag{3.1}$$

em que $A_{k+1}$ e $A_k$ são, respectivamente, o novo e o atual valor do atuador do sistema. $K_i$ é uma constante definida de acordo com o sistema. $S_{ref}$ e $S_k$ são, respectivamente, o valor desejado e o valor atual do sensor do sistema. Dessa forma, a saída desejada do sistema é chamada de *referência*. Quando uma ou mais variáveis de saída necessitam seguir uma certa *referência* ao longo do tempo, um atuador manipula as entradas do sistema para obter o efeito desejado nas suas saídas (figura 2) (PAREKH et al., 2002).

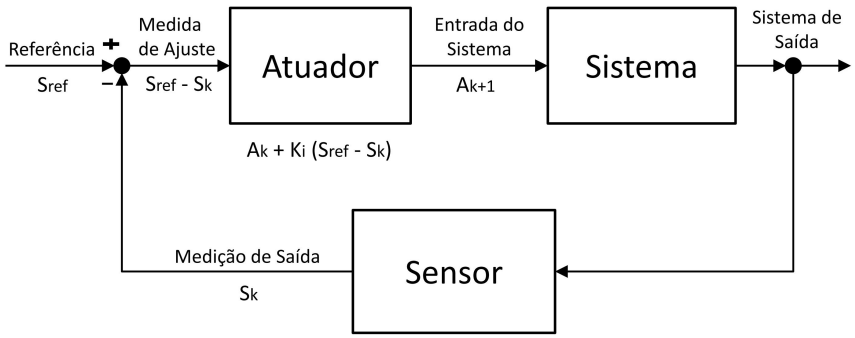


Figura 2: Ciclo que controla o comportamento dinâmico da demanda do sistema.

Neste caso, a realimentação é negativa, pois o valor medido da saída do sensor é subtraído do valor de referência almejado, resultando em uma medição de ajuste, que é aplicada pelo atuador à entrada do sistema, visando uma atuação corretiva. No contexto deste trabalho, o sistema considerado é um sistema de computação em nuvem que provê *clusters* virtuais usados na execução de aplicações de alto desempenho (BoT). Nessa perspectiva, define-se a variável do atuador como a quantidade de MVs do *cluster* ($QMV$) e a variável do sensor como o tempo esperado e atribuído pelo usuário para execução de um BoT (*walltime*), assim como apresentado pela seguinte equação 3.2,

$$QMV_{0+1} = QMV_0 + K_i * (walltime_{ref} - walltime_0), \tag{3.2}$$

em que $QMV_0$ e $QMV_{0+1}$ são, respectivamente, o atual e o novo valor da quantidade de MVs em um dado *cluster* $c_i$, $K_i$ é uma constante definida heuristicamente. O $walltime_{ref}$ é o tempo total esperado e atribuído pelos usuários para execução dos BoTs enfileirados e $walltime_0$ é o tempo estimado para a execução de um BoT com a configuração atual, ambos armazenados na fila de submissão de *jobs*. Assim, considerando um conjunto C de diferentes *clusters*, cada *cluster* $c_i \in$ C possui uma utilização máxima $U_{máx}$ e uma quantidade máxima de MVs $MV_{máx}$ pré-configurados no atuador.

No capítulo anterior, foram discutidos trabalhos recentes que propuseram diferentes soluções para o fornecimento e o gerenciamento de recursos virtualizados a fim de diminuir a quantidade de MVs alocadas e/ou minimizar a perda de desempenho a nível de *clusters* virtuais. Com base nesses trabalhos, caracterizamos nosso problema como o de *alocação dinâmica e eficiente de MVs em clusters virtuais compartilhados*. O objetivo da nossa solução para este problema é reduzir a quantidade total de MVs alocadas através da utilização dos recursos ociosos. Uma vez que a quantidade de MVs alocadas representa o custo operacional para provedor de *clusters*, a solução visa também diminuir a quantidade de *clusters* alocados, diminuindo a quantidade de MVs alocadas por *cluster* e aumentando a disponibilidade dos *clusters*. Para tal apresentamos uma nova política de alocação de MVs, incluída no algoritmo de instanciação de MVs em *clusters* que também é proposto.

## 3.2 Descrição da Solução Proposta

A fim de responder à demanda variável, os recursos já providos podem ser estendidos de várias formas. Uma opção poderia ser modificar as aplicações individualmente. Dessa forma, elas seriam capazes de adquirir, alocar e utilizar recursos adicionais quando precisassem. Por exemplo, um escalonador de *jobs* (e.g., Torque), instalado no nó central, pode ser modificado para iniciar um conjunto de MVs em uma nuvem e encaminhar *jobs* para essas MVs. No entanto, uma desvantagem dessa abordagem é que cada aplicação ou parte do escalonador deve sofrer modificações, resultando em aplicações com funcionalidade redundante. Uma segunda e mais extensível opção seria construir uma camada adicional, separada das aplicações, capaz de mensurar a partir da carga de trabalho os recursos adicionais necessários, obtendo ou liberando MVs de acordo com a necessidade. Esta segunda opção permite que qualquer aplicação utilize recursos extensíveis, criando assim recursos elásticos, inclusive *clusters* (MARSHALL; KEAHEY; FREEMAN, 2010).

A solução proposta nesse trabalho compõe uma camada adicional de *software*, usada para criar *clusters* elásticos que dinamicamente adaptam-se de acordo com a demanda das cargas de trabalho. Para que esse *software* realize adaptações eficientes, aspectos técnicos, logísticos e até econômicos das nuvens utilizadas podem ser considerados. Nós focamos especificamente nos aspectos técnicos e logísticos com vistas no ganho de

desempenho com a extensão dos recursos de maneira a reduzir a quantidade total de MVs iniciadas e, consequentemente, trazendo ganhos do ponto de vista econômico. Nessa perspectiva, as MVs adicionais são dinamicamente incluídas ou liberadas em um *cluster* com base na mudanças na fila de submissão de *jobs* e no atual estado dos *clusters* virtualizados disponíveis. Dessa forma, a operação de adaptação do *cluster* fica transparente para o usuário. O usuário apenas envia seus *jobs* para a fila de submissão dos recursos elásticos e aguarda suas a execuções.

Logo, os *clusters* são adaptados a partir do monitoramento da fila de submissão de *jobs* e da sua disponibilidade. Para tal, é proposto um sensor de fila que examina os *workloads* que chegam e mantém informações sobre a fila de *workloads*, incluindo o número total de *jobs* na fila, o número total de *jobs* em execução e o tempo total previsto para execução dos *jobs* enfileirados. O sensor de fila também coleta informações específicas dos *jobs*, como por exemplo o tempo previsto para execução de cada *job*, o *walltime*. A utilização dos *clusters* também é monitorada, a fim de determinar a disponibilidade dos mesmos para a adaptação sob os aspectos técnicos e logísticos.

A solução proposta usa uma política para determinar quando e quantas MVs adicionais deve ser iniciadas ou terminadas. A política baseia-se nas informações do sensor de fila, da capacidade da nuvem em prover recursos e no estado atual de utilização dos *clusters* virtuais da nuvem. A capacidade da nuvem em prover recursos é mapeada através do parâmetro *wastetime* que refere-se à soma dos tempos médios para iniciar, desligar e transferir uma determinada MV em uma nuvem específica. O estado atual de utilização dos *clusters* determina a ociosidade do sistema, permitindo que o reaproveitamento de recursos possa ser dimensionado. Com isso, é possível não só maximizar a oferta de recursos como também garantir que os recursos alocados não estão sendo desperdiçados, ou seja, que exista pouca ociosidade. Nas subseções seguintes, serão apresentadas a política proposta e os algoritmos de instanciação e desligamento de MVs.

### 3.2.1 Política e Heurística Utilizadas

Os parâmetros utilizados pela política de adaptação proposta são características relativas aos *workloads* submetidos (BoTs), ao estado da utilização do *cluster* e ao *overhead* da nuvem antes da adaptação. Em relação aos BoTs, é utilizado como parâmetro o *walltime*, que é o tempo esperado e atribuído pelo usuário para execução de um BoT dado em minutos. Para um conjunto de máquinas virtuais que se pretende disponibilizar ($QMV_0$), calcula-se o *overhead* da nuvem, utilizando o parâmetro *wastetime*, que é a soma dos tempos médios para migração, instanciação e desligamento do conjunto $QMV_0$ dado em minutos. Dessa forma, para calcular o *wastetime* é necessário calcular *a priori* o tempo médio para migração, instanciação e desligamento de uma máquina ($\overline{WasteTimePorMáquinas}$) na nuvem em questão, como apresentado na equação 3.3.

$$wastetime = QMV_0 * \overline{WasteTimePorMáquinas}. \tag{3.3}$$

Para BoTs submetidos em rajada, assume-se que o número de máquinas virtuais a serem instanciadas ($QMV_0$) é diretamente proporcional ao *walltime* e inversamente proporcional ao dobro do *wastetime* da nuvem em questão, como proposto pela política *burts* de (MARSHALL; KEAHEY; FREEMAN, 2010) e expresso na equação 3.4.

$$QMV_0 \sim \frac{walltime}{2 * wastetime}. \tag{3.4}$$

Logo, a equação que determina a quantidade de máquinas que serão disponibilizadas, segundo (MARSHALL; KEAHEY; FREEMAN, 2010), para um BoT pode ser calculada pela equação 3.5.

$$QMV_0 = (\frac{walltime}{2 * \overline{WasteTimePorMáquinas}})^{\frac{1}{2}}. \tag{3.5}$$

Visando capturar informações de ociosidade de recursos, caso ocorra, o *Elastic-Cluster* agrega o uso da métrica taxa de utilização do *cluster* em funcionamento. A ociosidade pode ocorrer devido a uma superestimação dos tempos de execução feita pelo usuário ou porque *jobs* pertencentes a BoTs diferentes, em execução concorrente na mesma máquina, podem iniciar e encerrar em tempos distintos. Tal ociosidade é quantificada, embora fragmentada (pois está presente em várias MVs), através da seguinte equação,

$$QMV_{ócio} = QMV_{atual} * (1 - U), \tag{3.6}$$

em que $QMV_{ócio}$ representa a ociosidade de um *cluster* $c_i$ em quantidade de máquinas virtuais, $QMV_{atual}$ é a quantidade de máquinas virtuais instanciadas que compõem o *cluster* $c_i$ e $U$ é a utilização do *cluster* $c_i$, sendo $0 \leq U \leq 1$. Dessa forma, a quantidade de máquinas virtuais ($QMV$) a serem inicializadas para o próximo BoT pode ser estimada pela equação 3.7

$$QMV = QMV_0 - QMV_{ócio}, \tag{3.7}$$

Caso $QMV_0 \leq QMV_{ócio}$, nenhuma máquina deverá ser criada, ocorrendo apenas uso de máquinas já instanciadas. Caso contrário, i.e., se $QMV_0 > QMV_{ócio}$, então uma quantidade MV de máquinas deverão ser adicionadas ao *cluster*. Contudo, não é possível incluir um número ilimitado de máquinas em um *cluster* sem gerar instabilidade ou eventuais falhas causadas por sobrecarga no nó coordenador (WERSTEIN; SITU; HUANG, 2006). Além disso, os provedores de nuvem estabelecem uma quantidade máxima de MVs que um usuário pode instanciar. Para definir limites de crescimento de um dado *cluster* $c_k$, foram definidos a taxa máxima de utilização ($U_{máx,c}$) e um número máximo de máquinas virtuais ($MV_{máx,c}$) que ele pode abrigar. Assim, garante-se que um novo *cluster* $c_k$ só será criado quando todos os ($k$-1) *clusters* (conjunto $C$) existentes atingirem seus respectivos limites $MV_{máx}$ e $U_{máx}$.

### 3.2.2 Algoritmo de Instanciação de MVs em *clusters* virtuais

A política de adaptação que usa ociosidade de *clusters* é apresentada no Algoritmo 1. Nas linhas 1, 2 e 3, é calculada a quantidade de MVs que serão disponibilizadas para execução do BoT de acordo com a política *bursts* de (MARSHALL; KEAHEY; FREEMAN, 2010). Em seguida (linhas 4-10) é selecionado o melhor *cluster* para adaptação e/ou reuso, o *cluster* adaptável. Tal *cluster* é aquele que possui menor taxa de utilização e maior quantidade de máquinas, ou seja, maior disponibilidade. Para isso, é feita uma varredura em todos os *clusters* (linha 5) e, para cada *cluster*, é calculada uma nota composta pelos valores normalizados de utilização e de quantidade de máquinas (linha 7). A normalização da utilização é feita com uma função decrescente com domínio e contradomínio no intervalo [0,1] e para a quantidade de máquinas com uma função crescente, sendo o domínio os números reais e o contradomínio o intervalo [0,1]. O *cluster* com maior nota é obtido na linha 10.

**Algoritmo 1** – **Instanciação de máquinas virtuais**

**Require:** *BoT*, *clusters(lista com todos os clusters existentes)*, $MV_{máx}$ e $U_{máx}$ ;
**Ensure:** *cluster* com BoT em execução;

01: *walltime* ← tempo esperado de execução do BoT
02: *WasteTimePorMáquina* ← tempo médio para migrar, instanciar e desligar uma MV
03: $QMV_0 \leftarrow \left(\frac{walltime}{2 * WasteTimePorMáquina}\right)^{1/2}$
04: notas = [];
05: **for** índice = 1 **to** (*size clusters*) **do**
06: *cluster*Temp = *clusters*[índice-1];
07: nota c = calculaNota*Cluster*(*cluster*Temp);
08: notas = notas.add(nota, *cluster*Temp);
09: **end_for**
10: *cluster* ← *cluster*ComMaiorNota(notas);
11: **if** (*cluster* ≠ **nulo**) **then**
12: $QMV_{atual}$ ← atual número de máquinas do *cluster*
13: $U$ ← taxa de utilização do *cluster*
14: $QMV_{ócio} \leftarrow QMV_{atual} * (1 - U)$
15: $QMV = QMV_0 - QMV_{ócio}$;
16: **if** ($QMV > 0$) **then**
17: **if** ($U \leq U_{máx}$ && $QMV \leq (MV_{máx} - QMV_{atual})$) **then**
18: *cluster* ← alocaNovasMáquinas(*QMV, cluster*);
19: ajustaTTLMáquinas (*cluster*, *walltime*);
20: **else**
21: *cluster* ← criaNovo*Cluster*($QMV_0$);
22: **end_if**
23: **else if** (*cluster* == **nulo**)
24: *cluster* ← criaNovo*Cluster*($QMV_0$);
25: **end_if**
26: executaBoT(BoT, *cluster*);

Política de Adaptação

Se o *cluster* obtido for nulo, então um novo *cluster* é criado (linha 24). Caso contrário (linha 11), obtém-se o número de máquinas que atualmente compõem o *cluster*, sem considerar o nó coordenador (linha 12). Em seguida, é obtida a utilização do *cluster* (linha 13) para o cálculo da quantidade de máquinas ociosas (linha 14) e a quantidade de novas máquinas necessárias ao BoT (linhas 15), descontando-se o espaço ocioso representado em quantidade de máquinas. A política de adaptação proposta é apresentada nas linhas 16-21 do Algoritmo 1. Na linha 16 ocorre a verificação da real necessidade de adaptação do *cluster* em função da utilização máxima desejável para o *cluster* ($U_{máx}$) e do número máximo de máquinas que o *cluster* pode ter ($MV_{máx}$). Caso a utilização do *cluster* esteja abaixo de $U_{máx}$ e $QMV \leq (MV_{máx} - QMV_{atual})$ serão instanciadas $QMV$ máquinas no *cluster* adaptável (linha 18) e, em seguida, ocorre o ajuste do tempo de vida das máquinas ($TTLMachines$) de acordo com o *walltime* do novo BoT que irá executar no *cluster* adaptado (linha 19); caso contrário cria-se um novo *cluster* com $QMV_0$ máquinas (linha 21). E, por fim, ocorre a execução do BoT (linha 26). O ajuste do $TTLMachines$ garante que as máquinas reutilizadas não serão desligadas antes que o tempo de vida de cada BoT alocado ($TTLDoBoT$) seja decorrido.

O $TTLDoBoT$ é calculado em função do *walltime* e do $QMV_0$ do BoT com o acréscimo de 1 minuto, conforme a equação 3.8. Além disso, o $TTLMachines$ reutilizadas por um novo BoT é ajustado para o $TTLDoBoT$ caso $TTLMachines < TTLDoBoT$, caso contrário $TTLMachines$ não muda. Isto garante a disponibilidade das máquinas para os novos BoTs sem o *overhead* da instanciação de novas MVs. Mesmo com o ajuste do tempo de vida das máquinas pré-existentes, cada BoT em execução só poderá utilizar as máquinas alocadas para si durante $TTLDoBoT$ unidades de tempo, havendo cancelamento da execução e envio de relatório da execução para o usuário, caso o $TTLDoBoT$ tenha decorrido e a execução não finalizada. As máquinas do *cluster* são desligadas quando não há BoTs usando o *cluster* e quando não há BoTs enfileirados aguardando execução. Mais detalhes podem ser vistos no algoritmo de desligamento das MVs.

$$TTLDoBoT = (\frac{walltime}{QMV_0}) + 60. \tag{3.8}$$

A política proposta tem como principais parâmetros o $MV_{máx}$ e $U_{máx}$. A partir dos valores configurados para os mesmos e do estado do *cluster*, a adaptação pode ocorre a fim de reusar recursos ociosos. Dessa forma, a política proposta permite que o administrador dos recursos virtuais estabeleça limites superiores para cada parâmetro, evitando sobrecarga ou instabilidade no nó coordenador dos *clusters*. Assim, garante-se que um novo *cluster* só será criado quando todos os *clusters* existentes atingirem seus respectivos limites $MV_{máx}$ e $U_{máx}$. Dentre os trabalhos relacionados, apenas (MONTERO; MORENO-VOZMEDIANO; LLORENTE, 2011) apresenta o parâmetro $MV_{máx}$, contudo, eles consideram apenas esse parâmetro. Na proposta apresentada nesta dissertação, além do número de nós, é considerada a utilização do *cluster*, o tempo previsto de execução do *workload* enfileirados e o *overhead* da nuvem. A título comparativo, no trabalho de (LIM; BABU; CHASE, 2010) apenas um parâmetro relacionado com a utilização foi observado,

que no caso deles, refere-se apenas utilização de CPU e não de todo o sistema como é feito pelo *ElasticCluster*.

É importante observar ainda que o provedor de recursos virtualizados possui um limite em relação a quantidade total de MVs que podem ser alocadas simultaneamente. Este limite está ligado aos recursos físicos que suportam a criação dos recursos virtuais. (LIAO; JIN; LIU, 2012) propõem estratégias para maximizar o uso dos recursos físicos durante a alocação de recursos virtualizados a fim de diminuir a quantidade de máquinas físicas ligadas e de aumentar a quantidade de recursos virtuais oferecidos. De qualquer maneira, os recursos físicos são limitados e, consequentemente, os recursos virtuais também são. Nessa perpectiva, é proposto também o monitoramento da quantidade total de MVs alocadas em todos os *clusters* a fim de não colocar em execução BoTs que precisem de uma quantidade de MVs maior que a quantidade máxima suportada pelo provedor. Dessa forma, quando o novo BoT é submetido ao *ElasticCluster*, a quantidade de MVs para ele é calculada e, se houver recursos suficientes, o BoT é colocado em execução. Caso contrário, o *ElasticCluster* aguarda a liberação de recursos para executar o BoT. A liberação ocorre com a finalização de BoT que já estavam em execução, conforme o algoritmo de desligamento de MVs.

### 3.2.3 Algoritmo de Desligamento de MVs em *clusters* virtuais

Após decorrido o $TTLDoBoT$ de um BoT, as MVs utilizadas por ele em sua execução devem ser desativadas. Contudo, essas máquinas podem estar sendo compartilhadas com a execução um outro BoT; logo, não podem ser desligadas imediatamente e nem na mesma quantidade de MVs que foram disponibilizadas inicialmente para o BoT que está sendo finalizado. É necessário calcular a quantidade de MVs que podem ser desligadas. Essa quantidade é definida a partir da utilização que o BoT finalizado estabelecia no seu *cluster*, visto que o compartilhamento de MVs entre BoTs se dá exatamente em função da utilização que os BoTs fazem dos *clusters*. Ou seja, são desativadas as MVs que eram efetivamente utilizadas pelo BoT, já que aquelas não utilizadas são reaproveitadas por outro BoT. Logo, o desligamento das MVs compartilhadas fica sob responsabilidade dos BoTs remanescentes de acordo com suas respectivas utilizações. Outro cuidado importante antes da desativação das MVs refere-se à liberação das máquinas que serão desligadas pois, se as mesmas possuírem outros BoTs em execução, as páginas de memória da execução desses BoTs devem ser migradas para as MVs que não serão desativadas.

É importante observar também a existência de BoTs enfileirados aguardando a liberação de recursos para sua execução, visto que e número de recursos virtuais do provedor é limitado. Nesse caso, se houver BoT(s) enfileirado(s) é verificada a quantidade de MV(s) requisitada em relação à quantidade que será desligada, sendo feito o desligamento apenas da quantidade de MVs que não será utilizada pelo BoT enfileirado. Se todas as MVs indicadas para o desligamento serão utilizadas, então, nada é feito. De maneira oposta, todas as MVs são desligadas caso não exista nenhum BoT enfileirado. Note que, as MVs liberadas ficam disponíveis mesmo que essa quantidade não seja suficiente para

colocar um BoT enfileirado em execução, dessa forma, o *ElasticCluster* aguarda novas liberações para obter a quantidade complementar de MVs necessárias para colocar o BoT enfileirado em execução. Assim, antes de desligar MVs alocadas, o algoritmo proposto faz duas verificações. A primeira é com relação a existência de outros BoTs nas MVs e a segunda, trata-se da verificação da existência de outros BoTs na fila de submissão do BoTs.

No Algoritmo 2 é possível observar todas as verificações e ações realizadas para o desligamento de MVs. O Algoritmo recebe 3 entradas. A primeira é o *queueOfBoTs* que é a fila de BoTs que esperam por recursos para serem executados ou simplesmente esperam o decorrer do *delay* pré-configurado para sua inicialização, por exemplo, no caso de BoTs submetidos em rajadas. A segunda é o $BoT_i$ que é o BoT que está sendo finalizado ou cancelado. A terceira é o *cluster* que executava o $BoT_i$. Na linha 1, é armazenado na variável $QMV_{atual}$ o número total de MVs que serão utilizadas na alocação de recursos de todos os BoTs que estão enfileirados. Nas linhas 2, 3 e 4 é calculada a quantidade de máquinas que podem ser desligadas. Este cálculo é baseado na utilização que o $BoT_i$ estabelecia no *cluster*. Dessa forma, na linha 2, obtém-se o número de MVs que eram utilizadas pelo $BoT_i$ no *cluster*. Na linha 3, a utilização que o $BoT_i$ estabelecia no *cluster* e, na linha 4, é feita o calculo da quantidade inicial de MVs que devem ser desligadas, sendo a mesma armazenada na variável $QMVD_o$. Contudo, é preciso verificar primeiro se tais MVs estão sendo compartilhadas para a execução de outro BoT, linha 5. Em caso positivo, linha 6, é realizada a migração das páginas de memória das MVs que serão desligadas para as outras máquinas que continuarão funcionando. É importante notar que, mesmo após a desativação das MVs, cada BoT em execução só poderá utilizar a quantidade de máquinas alocadas para si.

**Algoritmo 2** – Desligamento de MVs

**Require:** *queueOfBoTs*, $BoT_i$, *cluster* (que executava o $BoT_i$);
**Ensure:** *cluster* atualizado;

1: $QMV_{atual}$ ← número total de MVs requisitadas pelos BoTs enfileirados;
2: $QMV_o$ ← número de MVs do *cluster* que executava o $BoT_i$;
3: $U_{BoTi}$ ← taxa de utilização que o $BoT_i$ estabelecia no *cluster*;
4: $QMVD_o \leftarrow QMV_o * U_{BoTi}$;
5: **if** (*cluster*ExecutaOutroBoT()) **then**
6: migraPáginasMemóriaOutrosBoTs();
7: **if** (*length queueOfBoTs* != 0) **then**
8: $QMVD \leftarrow QMVD_o - QMV_{atual}$;
9: **if** ($QMVD > 0$) **then**
10: desligaMVs($QMVD$);
11: **else**
12: desligaMVs($QMVD_o$);

A verificação da fila de submissão de BoTs ocorre na linha 7. Se não houver *clusters* enfileirados (linha 11), então as $QMVD_o$ são desligadas (linha 12). Contudo, se

existirem BoTs aguardando para serem executados, então a quantidade de máquinas que serão desligadas ($QMVD$) será a quantidade inicial de MVs a serem desligadas ($QMVD_o$) descontada a quantidade da máquinas que já estão sendo requisitadas ($QMV_{atual}$) (linha 8). Note que, se $QMV_{atual} \geq QMVD_o$, então, nenhuma máquina é desligada. Mas, se $QMV_{atual} < QMVD_o$, ou seja $QMVD > 0$, então $QMVD$ máquinas são desligadas (linhas 10).

## 3.3 Conclusões

Neste capítulo foi apresentada a proposta de uma política de alocação dinâmica de MVs em *cluster* virtual. Esta política tem como objetivo diminuir a quantidade total de MVs iniciadas no provedor para atender a cargas de trabalho dinâmicas. A política utiliza uma heurística para determinar a quantidade de MVs que serão disponibilizadas para uma determinada carga (BoT) e propõem dois parâmetros ($MV_{máx}$ e $U_{máx}$) que são monitorados a fim de evitar sobrecargas nos *clusters* que podem comprometer o desempenho dos mesmos.

A solução proposta é composta por dois algoritmos: o algoritmo de instanciação de MVs e o algoritmo de desligamento de MVs. O algoritmo de instanciação contém a política de alocação proposta que faz uso dos recursos ociosos para instanciar uma quantidade menor de MVs. O algoritmo de desligamento realiza duas verificações: uma se as máquinas que serão desligadas não estão sendo compartilhadas com a execução de um outro BoT e outra se a fila de submissão de BoTs não está vazia. Caso as máquinas estejam sendo compartilhadas, há uma migração das páginas de memória do BoT remanescente das máquinas que serão desligadas para as máquinas que continuam executando. A fila de submissão de BoTs é verificada para calcular a quantidade de MVs que serão utilizadas nas próximas instanciações. Assim, as máquinas que podem ser reaproveitadas nas próximas alocações não são desligadas. No próximo capítulo, será apresentada a modelagem realizada com Redes de Petri coloridas de um provedor de recursos virtualizados com suporte a adaptação de *clusters* virtuais.

# 4 MODELAGEM DE UM PROVEDOR DE RECURSOS VIRTUALIZADOS COM SUPORTE A ADAPTAÇÃO DE *CLUSTERS* VIRTUAIS

## 4.1 Apresentação

As Redes de Petri (RPC) suportam hierarquização de subredes e restrições de tempo. Facilitam a construção e validação de modelos de sistemas concorrentes. Permite também que sistemas sejam simulados levando em consideração os aspectos relacionados com a escalabilidade e desempenho do sistema. Na modelagem realizada, utilizou-se as restrições de tempo para modelar a chegada de BoTs concorrentes e para modelar o tempo de execução dos BoTs. O modelo criado foi hierarquizado com subredes a fim de realizar uma melhor apresentação do modelo. Uma subrede de mais alto nível possui uma visão geral em relação a organização, fluxo de simulação e dependência entre as subredes. As Redes de Petri podem ser criadas através de ferramentas específicas de simulação como o CPN Tools [1]. Os algoritmos foram implementados com a linguagem Standard ML [2] que é suportada pelo CPN Tools. Maiores detalhes sobre as RPCs podem encontrados no Apêndice A. Maiores detalhes sobre a RPC desenvolvida podem ser encontrados no Anexo A, que possui as definições das cores criadas, e no arquivo do CPN Tools, no qual o modelo foi desenvolvido [3].

As RPCs hierárquicas permitem a construção de modelos mais complexos, os quais podem ser divididos em subredes. No CPN Tools, a hierarquização ocorre através da substituição de uma transição por uma subrede. Em uma rede de nível hierárquico mais alto, cada subrede é representada por uma transição de substituições, cuja representação gráfica é feita por um retângulo de borda dupla. A representação de lugares comuns entre subredes diferentes é feita através dos lugares de fusão. A hierarquização e os lugares de fusão permitem a construção de modelos mais fáceis de visualizar e a associação entre duas ou mais páginas de uma rede. A rede apresentada na figura 3 apresenta a visão de maior nível de abstração do modelo desenvolvido para a simulação do provedor. Nesta rede, está representado o fluxo principal que contém as fases que compõem o ciclo de vida de um *workload* executado em um ambiente virtualizado. O algoritmo *ElasticSite* (MARSHALL;

[1] http://www.cpntools.org
[2] http://www.standardml.org/
[3] http://www.great.ufc.br/downloads/elastic_cluster.zip

KEAHEY; FREEMAN, 2010) e o algoritmo *ElasticCluster*, proposto neste trabalho, são usados na transição *AllocateMachines* de acordo com a proposta selecionada para realizar a alocação de VMs.

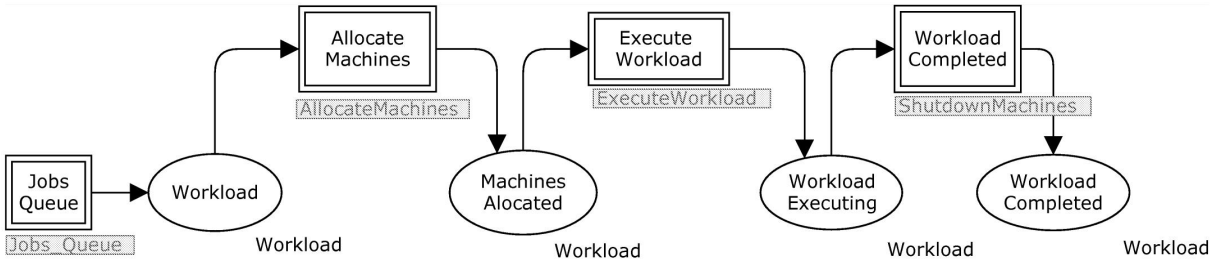


Figura 3: Página com a visão de maior nível de abstração do modelo do provedor

A figura 3 apresenta a estrutura hierárquica do modelo e todas as páginas que o compõem. O modelo possui um total de quatro páginas. A página "*Jobs Queue*"apresenta o modelo de chegada de *jobs* no sistema, o enfileiramento dos mesmos. Essa página está conectada com o lugar *Workload* que recebe e contém fichas da cor *Workload*. A cor *Workload* encapsula as fichas da cor *Jobs* e contém outros atributos específicos dos BoTs que são admitidos para execução. Esses atributos são preenchidos à medida que a ficha *Workload* é repassada para os outros lugares da rede. A página de segundo nível hierárquico "*Alocate Machines*"realiza as ações de alocação de MVs de acordo com a proposta selecionada. A transição está conectada ao lugar *Machines Alocated* que também recebe e contém fichas da cor *Workload*. A página "*Execute Workload*"modela as ações responsáveis pela criação ou adaptação de *clusters* e pelo gerenciamento da execução do BoT, repassando para o lugar *Workload Executing* o *Workload* que está sendo executado. A página "*Workload Completed*"contém os eventos associados à finalização da execução do BoT. Nessa página também é feito o controle do desligamento das MVs. O lugar *Workload Completed* contém os *workloads* finalizados ou cancelados em função do estouro do tempo esperado para sua execução.

### 4.1.1 Modelo de Carga de Trabalho e de Enfileiramento de Cargas de Trabalho

A subrede *Jobs Queue* apresentada na figura 4 modela a chegada de *jobs* no sistema e o enfileiramento dos mesmos. Essa subrede refere-se ao sensor de fila de submissão. Na figura 4 é apresentada uma carga de trabalho específica de exemplo. As restrições de tempo foram utilizadas nas fichas da cor *Job* para representar o atraso entre as chegadas dos BoTs que são submetidos em rajada. Por exemplo, no lugar BoT2_WL2 chegam 150 fichas da cor *Job* e somente após 1200 segundos (20 minutos) as mesmas estarão disponíveis para uso pela transição BoT2_WL2. A cor *Job* é uma tupla composta pelo: identificador do *job* (*idJob*), identificador do usuário que submete o *job* (usuário) e o tempo estimado pelo usuário para execução do *job* (*wallTime*). A seguir o trecho de código que cria a cor *Job*:

*colset Job = record idJob:INT * usuário:Usuário * wallTime:INT timed;*

O modificador *timed* que aparece no final da declaração da cor *Job* indica que nessa cor podem ser aplicadas restrições de tempo. Os operadores @, @+ e @@+ são usados para adicionar temporização às cores (fichas). A carga de trabalho apresentada é composta por três BoTs submetidos a cada 1200 segundos (20 minutos) e com 400, 150 e 600 *jobs* em cada BoT. Note que a quantidade de *jobs* em cada BoT é passada como parâmetro para a função *setQtdJobs(qtdJobs)*. O tempo de duração dos *jobs* de um BoT é passado como parâmetro para a função *newJobs(tempoDuracaoJobs)* que foi atribuída aos arcos que chegam ao lugar *BoTs* que armazena fichas do tipo *Jobs* que é uma lista de fichas do tipo *Job*. Note ainda que no arco que chega à transição BoT1_WL2 foi colocada a função que atribui o tempos de duração variados aos *jobs*. A função *newJobsRange()* gera valores aleatórios para o tempo de duração dentro uma faixa de valores possíveis pré-configurada. A figura 5 apresenta uma generalização da modelagem apresentada na figura 4 para a carga de trabalho.

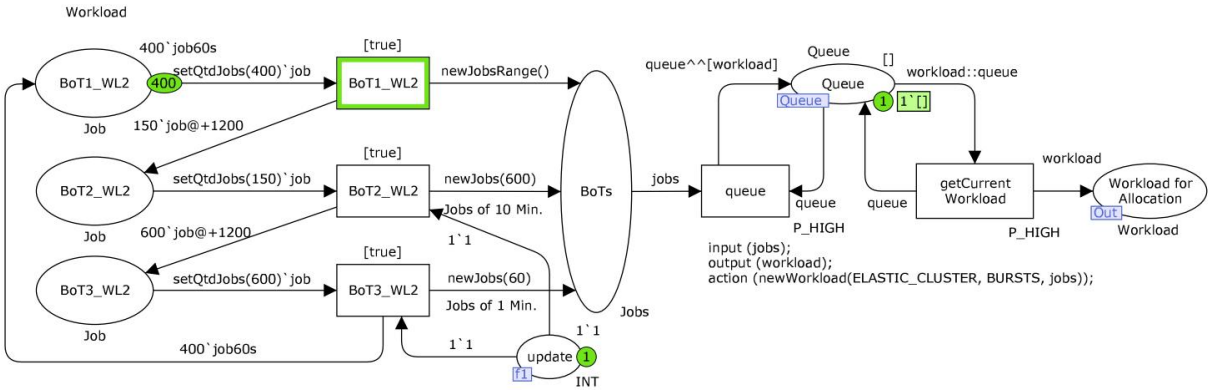


Figura 4: Subrede *Jobs Queue*.

Na rede da figura 5, o lugar *Jobs* possui uma marcação inicial com três fichas da cor *Jobs*. As fichas são criadas pela função *newJobs(qtdJobs:INT, tmpInicial:INT, tmpFinal:INT)*. No parâmetro *qtdJobs* é indicada a quantidade de *jobs* que terá o BoT. No segundo e no terceiro parâmetros é indicado o(s) *(walltime)* dos *jobs*. Para indicar um valor único de *walltime* para todos os *jobs* basta preencher o segundo parâmetro com o valor desejado e no terceiro parâmetro incluir o valor 0. É possível também como os dois parâmetros indicar uma faixa de valores possíveis para o *walltime* dos *jobs*. Para tal, basta indicar o limite inferior da faixa no segundo parâmetro e o limite superior da faixa no terceiro parâmetro. Os valores são indicados em segundos. Ainda nessa marcação inicial é indicado o tempo de *delay* de cada ficha após o operador @.

Foi criada também a cor *Queue*, uma lista de fichas da cor *Workload* ordenada com a política de prioridade FIFO a partir de comandos da linguagem Standard ML. Cada BoT (ficha da cor *Jobs*) que está no lugar *BoTs* é colocado no lugar *Queue* através do disparo da transição *queue* que executa a função *newWorkload(typeProposal:TypeProposal, typeWorkload:TypeWorkload, jobs:Jobs)*. Essa função encapsula os *jobs* que chegam em uma ficha da cor *Workload*. Na ficha *Workload* agrega-se outras propriedades para melhor

descrever o BoT e definir como será a alocação de recursos. O primeiro parâmetro dessa função indica qual proposta será utilizada (*ELASTIC_CLUSTER* ou *ELASTIC_SITE*), o segundo indica o padrão de submissão dos BoTs (*BURSTS*) e o terceiro parâmetro é a lista de *jobs* que compõem o BoT. Confira abaixo como a cor *Workload* foi definida. Trata-se de uma tupla com 5 atributos.

*colset Workload = record idWorkload:INT * typeProposal:TypeProposal * typeWorkload:TypeWorkload * idsJobs:idsJobs * workloadLoad: STRING * wallTimeTotal:INT timed;*

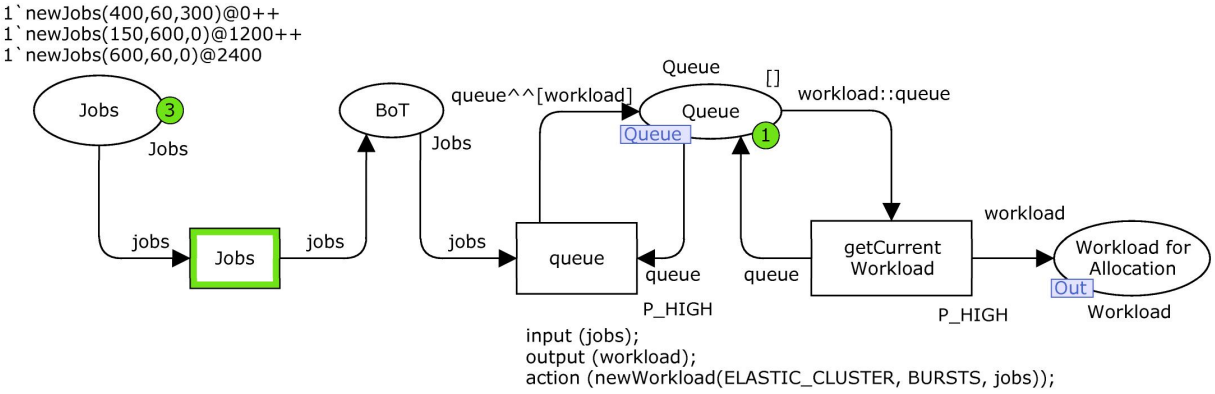


Figura 5: Subrede *Jobs Queue* generalizada.

Para cada *workload* é criado um identificador (*idWorkload*). O atributo *idsJobs* corresponde de uma lista de inteiros com os identificadores dos *jobs*. O tempo total previsto para execução do BoT é armazenado no atributo *wallTimeTotal*. Este tempo é calculado através da soma de todos os atributos *wallTime* definido para cada *job* do BoT. O atributo *workloadLoad* armazena em uma a utilização que o *workload* estabelecerá no *cluster* de MVs que será alocado para sua execução. Para simular a demanda de recursos para execução das aplicações no tempo esperado pelo usuário, definiu-se que o consumo que cada BoT pode fazer do *cluster* pode assumir valores entre 0,05 e 0,95, representando taxas de consumo entre 5% e 95%. Nessa perceptiva, o valor do *workloadLoad* é estabelecido randomicamente a partir dos valores possíveis daquela faixa.

Nesta subrede, utiliza-se também o lugar de fusão *Queue*, com isso, na subrede responsável pelo desligamento das MVs (*ShutdownMachines*) é possível incluir o lugar *Queue*, possibilitando a verificação da existência de BoTs enfileirados antes do desligamento das MVs. A transição *getCurrentWorkload* obtém o primeiro *workload* que chegou e disponibiliza-o através do lugar *Workload for Allocation* para uso pela subrede que é responsável por alocar/adaptar os recursos para execução do BoT.

### 4.1.2 Modelo de Alocação de Máquinas Virtuais

A subrede *AllocateMachines* (figura 6) é utilizada para modelar e simular os eventos relativos à alocação de MVs nos dois algoritmos, definindo a quantidade de MVs a

serem iniciadas e/ou reusadas em um *cluster* já existente ou se um novo *cluster* deverá ser criado. O BoT encapsulado na ficha *Workload* chega no lugar *Workload for Execution* (figura 6) e habilita a transição *AllocateMachines*. A depender do parâmetro *typeProposal* da ficha *workload*, a função *checkProposal(workload:Workload, ac:AdaptableCluster, mvs:NumbersOfMachines)* contida no segmento de código da transição *Allocate Machines* seleciona o algoritmo que deverá ser executado para realizar a alocação de VMs.

O parâmetro *typeProposal* define qual estratégia será executada (*ElasticCluster* ou *ElasticSite*). As funções *newMachinesAlocated_EC* e *newMachinesAlocated_ES* executam, respectivamente, os algoritmos das propostas *ElasticCluster* e *ElasticSite*. Abaixo, apresentamos as assinaturas completas das duas funções.

*function newMachinesAlocated_EC(workload:Workload, ac:AdaptableCluster, mvs:NumbersOfMachines)*

*function newMachinesAlocated_ES(workload:Workload)*

A função *newMachinesAlocated_EC* recebe como parâmetro: o *workload*, uma ficha da cor *AdaptableCluster* e uma ficha da cor *NumbersOfMachines*. As duas últimas são obtidas através dos lugares de entrada *AdaptableCluster* e *NumbersOfMachines*, respectivamente. A função *newMachinesAlocated_EC* recebe apenas o *workload*.

A ficha da cor *Adaptable Cluster* representada pela variável *ac*, representa o *cluster* com maior disponibilidade para adaptação, conforme discutido no capítulo anterior. Além disso, a ficha *ac* possui os valores atualizados do número MVs e da taxa de utilização do *cluster* adaptável. O lugar onde a ficha *ac* fica armazenada (*Adaptable Cluster*) é um lugar de fusão, portanto o mesmo pode ser atualizado em outras subredes. Na subrede de execução do BoT (*Execute Workload*), a ficha *ac* é atualizada logo após o BoT ser colocado em execução. Nesse instante, a utilização que o BoT vai estabelecer no *cluster* já foi definida randomicamente e, portanto, o *cluster* adaptável pode ser redefinido levando em consideração esse valor de utilização a as outras utilizações dos outros *clusters*, além das quantidades de MVs de cada *cluster*. Na subrede *ShutdownMachines* o *cluster* adaptável é novamente atualizado depois que o BoT tem sua execução finalizada ou cancelada.

A ficha da cor *NumbersOfMachines* faz o controle das quantidades de MVs que o provedor contém. Com essa ficha, o *ElasticCluster* levar em consideração as quantidades de MVs e seus estados no provedor para realizar as novas alocações. Esta ficha contém três informações: as quantidades de MVs inativas, de MVs em execução e de MVs reutilizadas. Esses valores são mantidos também nos lugares de fusão: *InactiveMachines, RunningMachines e ReusedMachines*, respectivamente, para que possam ser monitorados durante simulações do CPN através dos *Monitors*. Esses lugares foram incluídos também na subrede *ExecuteWorkload* apenas para apresentar as quantidades de MVs na etapa de execução do BoT. Já na subrede *Shutdown Machines* os lugares são usados para atualização das quantidades de MVs e seus estados após o desligamento de MVs. Esse valores serão definidos como métricas de interesse na avaliação da proposta.

O disparo da transição *AllocateMachines* está condicionado também à retirada das fichas dos lugares *InactiveMachines, RunningMachines e ReusedMachines*. A retirada das fichas ocorre para que os lugares possam receber as fichas com os valores atualizados após a execução da função *newMachinesAlocated_EC* ou da função *newMachinesAlocated_ES*. As retiradas das fichas é feita pelas funções *getOffVMs(mvs)*, *getRunningVMs(mvs)* e *getReusedVMs(mvs)*. O parâmetro *mvs* indica a quantidade de fichas que deverão ser retiradas de acordo com as quantidades atuais de MVs que estão em cada lugar.

A função *newMachinesAlocated_ES* recebe como parâmetro apenas o *workload*. Logo, apenas atributos intrínsecos do *workload* são considerados pelo *ElasticSite*. Os parâmetros que representam o estado atual do provedor, tais como: quantidades de MVs (*NumbersOfMachines*) e utilização dos *clusters* (*Adaptable Cluster*) não são utilizados. Para efeito de avaliação, o modelo desenvolvido para a proposta *ElasticSite* também faz a atualização das quantidades de MVs após a execução dos algoritmos da mesma.

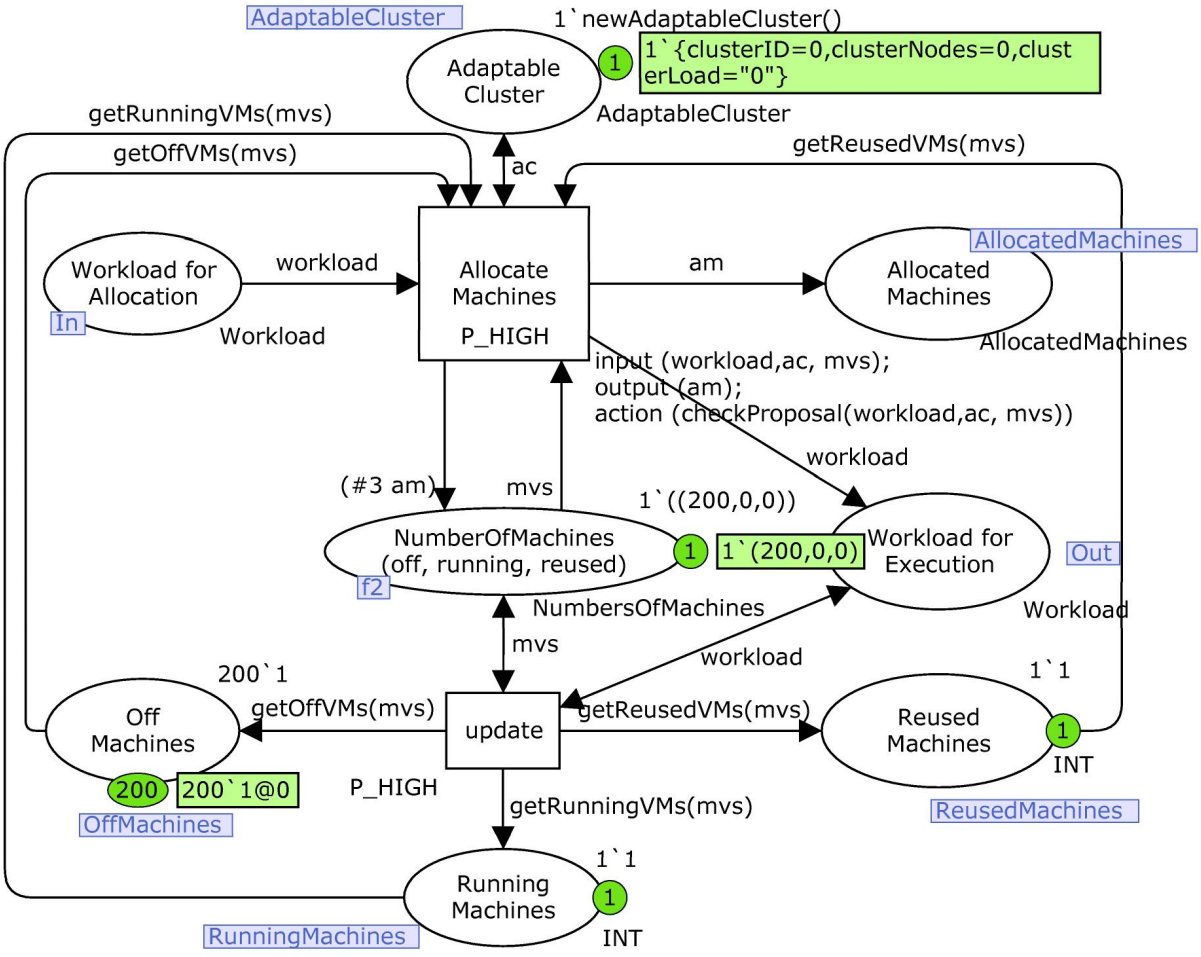


Figura 6: Subrede *AllocateMachines*.

Após o disparo da transição *AllocateMachines* e da execução da funções que o algoritmo de alocação, o lugar *Workload for Execution* recebe a ficha *workload* que será processada pela próxima subrede e o lugar *Adaptable Cluster* recebe a mesma ficha que enviou quando a transição *AllocateMachines* foi disparada. Neste momento, o *cluster*

adaptável é utilizado apenas pela função *newMachinesAlocated_EC* com o intuito de definir se ocorrerá a alocação de um novo *cluster* ou se o *cluster* adaptável será adaptado para a execução do novo *workload*. Nenhuma atualização é feita no *cluster* adaptável. Essa atualização é realizada apenas na próxima subrede.

Para indicar se um novo *cluster* será criado ou não, o modelo utiliza a ficha da cor *AllocatedMachines* que é passada para o lugar *Allocated Machines*. A cor *Allocated-Machines* é composta por três fichas, duas fichas da cor *Machines* e uma ficha da cor *NumbersOfMachines*. A ficha *NumbersOfMachines* possui as quantidades atualizadas de MVs após a alocação. A cor *Machines* é um conjunto de fichas da cor *Machine*. Uma ficha da cor *Machine* representa uma MV no modelo. Cada *Machine* contém um identificador (*idMachine*) e uma lista com os IDs dos *workloads* que executam na MV (*idsWorkloads*).

Uma das fichas *Machines* (*MVsNewCluster*) contém as MVs que vão compor um novo *cluster*, a outra (*MVsAdaptableCluster*) contém as MVs que serão incluídas no *cluster* adaptável. Se a ficha *MVsNewCluster* não estiver vazia, então haverá a instanciação de um *cluster* sem reaproveitamento do *cluster* adaptável. Caso contrário, o *cluster* adaptável será utilizado para executar o BoT. Se a ficha *MVsAdaptableCluster* estiver vazia, então apenas as MVs ociosas são suficientes para executar o BoT. Caso contrário (se o segundo elemento estiver preenchido), então ocorrerá uma adaptação do *cluster* adaptável.

### 4.1.3 Modelo Alocação/Adaptação de *Clusters* Virtuais e de Execução de BoT

A subrede *ExecuteWorkload* (figura 7) é responsável pela alocação do *cluster* e pelo gerenciamento da execução do BoT. Além disso, nesta subrede é feita a atualização do *cluster* adaptável. Para realizar essas atividades, a transição *cluster* deve ser disparada e, para isso, deve receber fichas das cores: *Workload*, *AllocatedMachines*, *AdaptableCluster* e *Clusters*. Novamente, a depender do parâmetro *typeProposal* da ficha *workload*, uma proposta é selecionada. Desta vez a seleção é feita pela função *newCluster(workload:Workload, am:AllocatedMachines, clustersIn:Clusters)* contida no segmento de código da transição *cluster*.

À partir da ficha *AllocatedMachines*, a função *newClusterEC* determina como será a alocação do *cluster* no *ElasticCluster* conforme a regra apresentada anteriormente, havendo a possibilidade de reutilização do *cluster* adaptável ou de criação de um novo *cluster*. As MVs que chegam por essa ficha já possuem os IDs dos *workload* que executam as mesmas executam. Após a criação, o novo *cluster* sera concatenado com a lista de *clusters* (*clustersIn*) que contém todos os *clusters* já alocados pelo provedor. A última etapa da alocação consiste na identificação do *cluster* adaptável. A função recursiva *findAdaptableCluster* recebe a lista *clustesIn*, calcula a nota para cada *cluser* e ordena a lista em ordem decrescente pela nota, selecionando o primeiro elemento da lista para ser o próximo *cluster* adaptável. O *cluster* selecionado é colocado no lugar *AdaptableCluster* através da variável *acOut*. A lista ordenada é colocada no lugar *cluster* através da variável

*clustersOut*.

Se a ficha *MVsNewCluster* estiver vazia, então a função *newClusterEC* reutiliza o *cluster* adaptável. Verificando em seguida se a ficha *MVsAdaptableCluster* está preenchida. Neste caso, as MVs de *MVsAdaptableCluster* devem incluídas no *cluster* adaptável. Após a inclusão das novas MVs, o *cluster* modificado recebe o *workload* em todas as suas MVs. Caso contrário, apenas MVs que já compõem o *cluster* adaptável recebem o *workload*. Após a submissão do *workload* para as MVs, a lista de *clusters* é atualizada. Para tal, retira-se o *cluster* adaptável reutilizado da lista para que sua versão atualizada seja incluída. Depois da atualização da lista de *clusters* a função *findAdaptableCluster* é chamada para determinar o novo *cluster* adaptável.

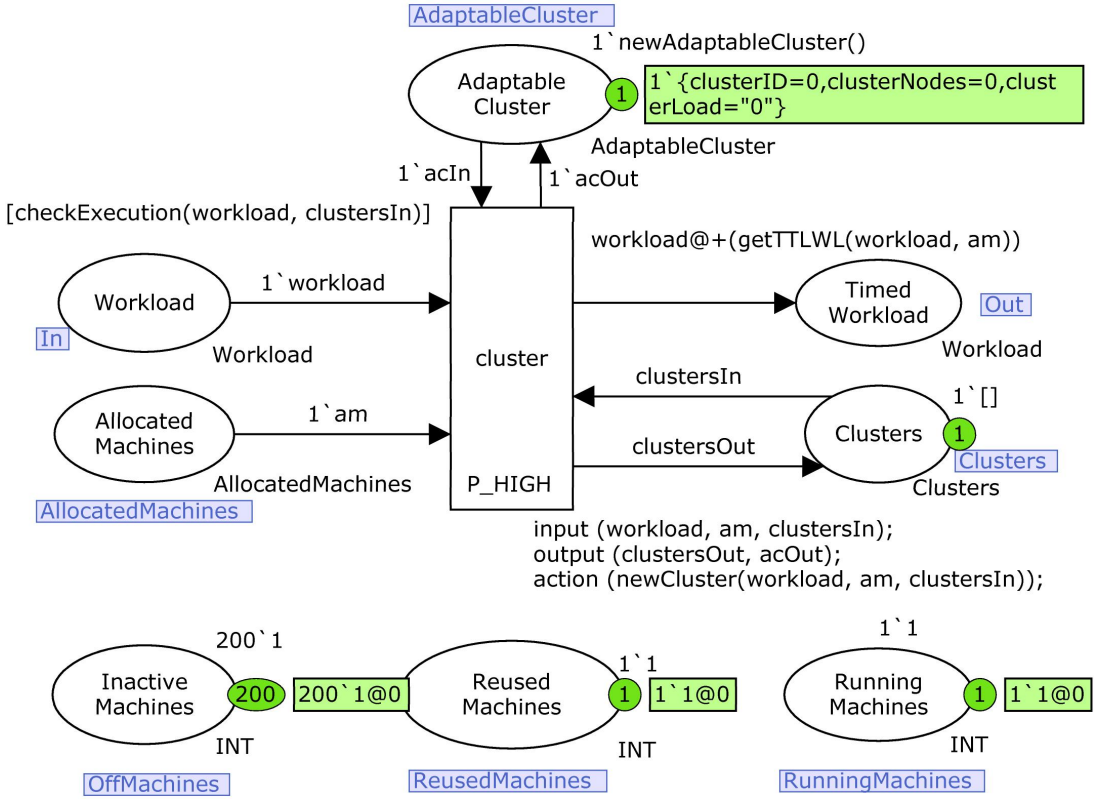


Figura 7: Subrede *Allocate Cluster/Execute Workload*.

A função *newClusterES* determina como acontece a alocação do *cluster* no *ElasticSite*. Em uma rajada de BoTs, o *ElasticSite* realiza a alocação de um *cluster* para cada BoT. De tal maneira que, dois *clusters* não ficam em execução simultaneamente. Ou seja, apenas após a desativação dos *clusters* já alocados um novo *cluster* é alocado. Dessa forma, os BoTs que ainda não foram submetidos para execução ficam enfileirados até que os recursos sejam liberados. Após a liberação dos recursos, um novo *cluster* é criado de acordo com a política já apresentada. A função *checkExecution(workload:Workload, clustersIn:Clusters)* verifica se o vetor *clustersIn* está vazio para que o *ElasticSite* possa fazer sua alocação. Por não fazer reaproveitamento dos recursos ociosos, a função *newClusterEC* recebe apenas o *workload* e as MVs que vão compor o novo *cluster*. Portanto, o *cluster* adaptável não é utilizado e o vetor *Clusters* tem apenas um elemento durante

toda a execução do *ElasticSite*.

O gerenciamento do tempo de execução do BoT é feito no arco que liga a transição *cluster* ao lugar *TimedWorkload* através uso das fichas temporizadas para fazer a cronometragem do TTLDoBoT. A função *getTTLWL(workload:Workload, am:AllocatedMachines)* calcula TTLDoBoT conforme a equação 6. Dessa forma, as fichas incluídas no lugar *TimedWorkload* só ficarão disponíveis após decorrido o TTLDoBoT para a próxima subrede. Os lugares *Clusters* e *TimedWorkload* são lugares de entrada da transição *Workload Completed or Canceled* da subrede *ShutdownMachines*.

### 4.1.4 Modelo de Desligamento de Máquinas Virtuais

Na subrede *Shutdown Machines* (figura 8) à medida que as fichas do lugar *TimedWorkload* vão sendo habilitadas, as MVs usadas pelo BoT liberado são retiradas do *cluster* que às contém (desativadas). Contudo, antes de desativar as MVs, duas verificações são feitas, primeiro, verifica-se a existência de BoTs na fila e, em seguida, é verificado, em cada MV, se há outro BoT sendo executado. Se qualquer umas das verificações retornar verdadeiro então as MVs não são desligadas, ficando o outro BoT que usa a MV ou do próximo BoT que será executado responsáveis pela desativação da MV. Para realização das verificações, a subrede faz uso do lugar de fusão *Queue* que também está presente na subrede *Jobs Queue*. A ficha *queue* recebida pela transição *Workload Complete or Canceled* é retornada para o lugar *Queue* após a verificação pelo arco duplo liga uma a transição ao lugar.

O *cluster* adaptável é atualizado após o processo de desligamento. Durante o disparo da transição *Workload Complete or Canceled* a ficha do *cluster* adaptável é retirada do lugar *AdaptableCluster* para ser atualizada com uma nova ficha, mas não é utilizada na verificação inicial assim como a ficha do lugar *NumbersOfMachines*. Note que nesses lugares é mantida apenas uma ficha. O novo *cluster* adaptável é obtido pela função *findAdaptableCluster* que recebe a lista de *clusters* atualizada sem o *cluster* (conjunto de MVs) que executava o *workload*.

A ficha temporizada que contém o *workload* ao ser habilitada inicia o processo de desligamento com as verificações através da função *shutdownVMs(workload:Workload, queue:Queue, clustersIn:Clusters, mvs:NumbersOfMachines*. Primeiramente, essa função, identifica o *cluster* que era utilizado pelo *workload* na lista *clustersIn*. Após a identificação do *cluster*, a função identifica as MVs que estavam sendo utilizadas pelo *workload* e, em seguida, para cada MV é verificado a existência de outros *workloads*, sendo desligadas apenas as MVs que só possuem o *workload* liberado. Aquelas MVs que estão sendo compartilhas com outros *workloads* terão seu desligamento definido pelo último *workload* que for liberado que executa na MV. A atualização da utilização do *cluster* acontece de tal forma que a nova utilização será igual à utilização que o *cluster* tinha antes de começar a executar a *workload* liberado. Dessa forma, pretende-se isolar a execução de cada *workload* e ao mesmo tempo realizar o compartilhamento de recursos com o reaproveitamento de recursos ociosos.

Após o desligamento das MVs, o *cluster* atualizado é reinserido na lista de *clusters* que será passado para a função *findAdaptableCluster* e recolocado no lugar *clusters* da subrede *Execute Workload*. No caso de uma simulação do *ElasticSite*, note que apenas 1 (um) *cluster* será mantido na lista de *clusters*. Nesse único *cluster* há apenas um *workload*, sendo permitida a alocação de um novo *cluster* apenas após a desativação do único *cluster* alocado. Além disso, para ambas as propostas, depois do desligamento as fichas dos lugares: *InactiveMachines, ReusedMachines e RunningMachines e AdaptableCluster* são atualizadas, vale ressaltar que as fichas desses lugares são retiradas no ato do disparo da transição *Workload Completed or Canceled*. Destacando também que esses valores são mantidos em lugares com o intuito de utiliza-los nos *monitors* (funcionalidade do CPN Tools), assim como, o tempo de execução do *workload* que é guardado no lugar *Execution Time*. Nesse lugar é incluído o instante de tempo do relógio global da rede fornecido pela função já implementada pelo CPN Tools *intTime()*.

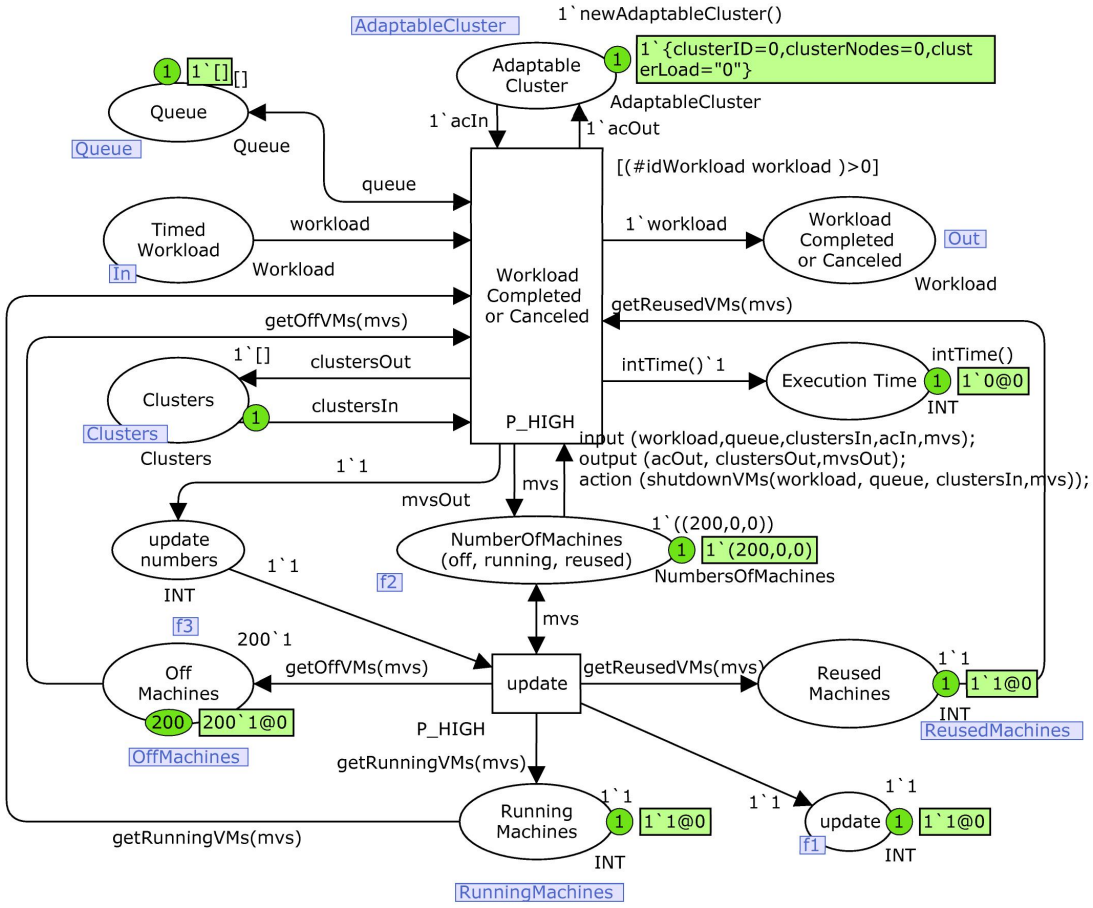


Figura 8: Subrede *Shutdown Machines*.

## 4.2 Análise e Validação do Modelo

As técnicas de análise e validação para RPCs são, em geral, adaptações de técnicas utilizadas para Redes Petri simples. Desta forma, para análise e validação de uma RPC,

pode-se empregar simulação, grafo de ocorrência e cálculo de invariantes. A simulação consiste em executar a rede, interativa e automaticamente, e avaliar o comportamento do sistema modelado, possibilitando visualizar o resultado de uma sequência de eventos. A simulação automática também permite investigar questões sobre o desempenho do sistema, tais como a identificação de gargalos (JENSEN; KRISTENSEN, 2009).

Neste trabalho é utilizada a técnica de simulação para a análise e validação da RPC proposta. Devido à complexidade da rede, não foi gerado o grafo de ocorrência e não foram formalmente avaliadas outras características da rede, como, por exemplo, a reversibilidade, visto que a análise buscava avaliar o comportamento da rede a fim de simular a modelagem de entradas específicas(BoTs), a alocação, a desalocação e a adaptação de recursos virtuais, especificamente *clusters* virtuais, de acordo com a política proposta e a política *bursts* de (MARSHALL; KEAHEY; FREEMAN, 2010).

Todas as configurações pertinentes à simulação, tais como as entradas (caracterização dos *workloads*), os cenários de simulação, os resultados e as análises dos resultados são apresentados no próximo capítulo. No mesmo, é apresentado também a metodologia para realizar as simulações.

## 4.3 Conclusão

A partir dos algoritmos propostos e descritos no capítulo anterior, foi realizada uma modelagem visando especificar o comportamento da solução proposta em um ambiente de computação em nuvem destacando a existência dos *clusters* virtuais no ambiente. Com a modelagem de um provedor foi possível realizar a validação dos algoritmos buscando definir os melhores cenários de execução da solução através da execução de simulações. O modelo descreve também o *ElasticSite* a fim de comparar as duas propostas. O provedor modelado suporta todo o ciclo de vida uma aplicação do tipo BoT, bem como o ciclo de vida dos recursos virtualizados com suporte à adaptação de *clusters* virtuais.

A modelagem desse ambiente foi realizada com Redes de Petri Coloridas (MURATA, 1989) levando em consideração aspectos de desempenho, escalabilidade e as principais características e elementos relacionados à utilização de *clusters* em nuvem: carga de trabalho, enfileiramento de carga de trabalho, alocação/adaptação de *clusters* em nuvem, execução de carga de trabalho e liberação de recursos. Para construir o modelo com Redes de Petri Coloridas foi utilizado o CPN Tools, uma ferramenta de modelagem, análise e simulação de RPC. A ferramenta foi desenvolvida na Universidade de Aarhus, Dinamarca, e é distribuída livremente para organizações não comerciais. Maiores detalhes podem ser vistos em Jensen e Kristensen (JENSEN; KRISTENSEN, 2009).

# 5 SIMULAÇÕES E RESULTADOS

Neste capítulo são apresentados os experimentos realizados com simulações (seção 5.3) usando a RPC descrita no capítulo 4 para avaliar o desempenho do *ElasticCluster* em relação ao *ElasticSite* e para identificar os cenários em que a execução do algoritmos propostos apresenta melhor desempenho. Todos os objetivos buscados com essas simulações são apresentados na seção 5.1. A Seção 5.2 apresenta em detalhes o *workload* utilizado na experimentação.

## 5.1 Objetivos das Simulações

Com as simulações pretende-se atingir os seguintes objetivos:

1. Comparar a política proposta com a política *bursts* do *ElasticSite*;
2. Determinar a quantidade total de máquinas iniciadas pela política proposta;
3. Quantificar a ociosidade reutilizada nos *clusters* pela política proposta;
4. Determinar o tempo total de execução dos BoTs de cada *workload* pela política proposta;
5. Analisar os resultados obtidos para cada cenário de simulação definido.

## 5.2 Caracterização dos *Workloads*

Nos experimentos realizados foi utilizado o mesmo *workload* dos experimentos de (MARSHALL; KEAHEY; FREEMAN, 2010), que é baseado em rajadas de BoTs (*workload* 1). Foi utilizado também outro *workload* definido, especificamente, para estender a avaliação do ElasticCluster (*workload* 2). O somatório dos tempos de todas as tarefas de cada BoT dos *workloads* define o *walltime* do *workload*. O *wastetime* médio por máquina ($\overline{WasteTimePorMáquinas}$) foi definido como 19 segundos, o mesmo utilizado nos experimentos de (MARSHALL; KEAHEY; FREEMAN, 2010). Para calcular o parâmetro $QMV_0$ (equação (3.5)) de cada BoT utiliza-se apenas o *walltime* e o $\overline{WasteTimePorMáquinas}$. Esse valores também são apresentados durante a caracterização de cada *workload*.

No teste de validação da proposta, utiliza-se o *workload* 1. Em relação à escalabilidade, o teste utiliza o segundo *workload*, o *workload* 2. No *workload* 2 o tempo de submissão entre os BoTs é menor para que o compartilhamento de recursos seja maior a fim de observar o comportamento do algoritmo de desligamento de MVs.

A taxa de utilização do *cluster* ($U$) é calculada a partir do consumo que o(s) BoT(s) do *workload* faz(em) do *cluster*. Para simular a demanda de recursos para execução das aplicações no tempo esperado pelo usuário, definiu-se que o consumo que cada BoT pode fazer do *cluster* pode assumir valores entre 0,05 e 0,95, representando taxas de consumo entre 5% e 95%. Nessa perspectiva, para cada tarefa do *workload* apenas é feita uma contagem do tempo esperado para sua execução. Na próxima seção descreveremos o *workload* utilizado para avaliação da proposta e nos testes de escalabilidade.

### 5.2.1 *Workload* 1

Uma ilustração do *workload* pode ser vista na figura 9. O *workload* foi utilizado para demostrar a escalabilidade do *ElasticSite*, os autores submeteram o *workload* para um *cluster* formado por MVs da Amazon EC2, permitindo assim que uma quantidade grande de máquinas pudesse ser instanciada. Visto que na nuvem da Amazon, cada usuário pode iniciar até 100 instâncias do tipo "*Spot*"ou, mediante solicitação, obter um número ainda maior de MVs.

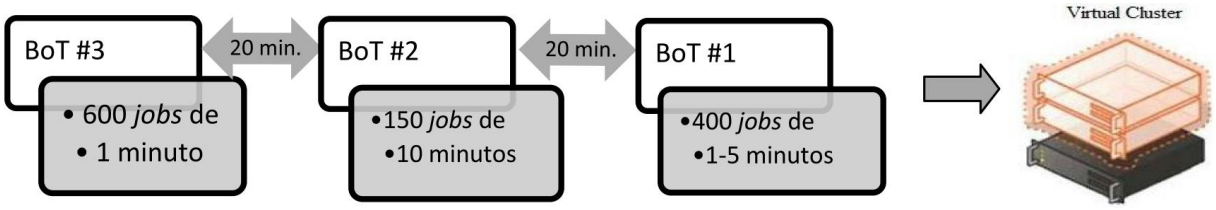


Figura 9: Ilustração do *workload*.

Outras características do *workload* podem ser vistas na tabela 2. Note que o primeiro BoT possui uma variação no tempo de execução das tarefas, provocando uma variação no *walltime* do *workload*.

Tabela 2: Resumo das características do *workload* 1.

| Parâmetro / BoTs | *Walltime* por BoT (min). | $QMV_0$ por BoT | *Walltime* Total (min). |
|---|---|---|---|
| BoT #1 | [400,2000] | [30,68] | [2500, 4100] |
| BoT #2 | 1500 | 59 | |
| BoT #3 | 600 | 37 | |

### 5.2.2 *Workload* 2

O *workload* 2 é 10 vezes maior que o *workload* 1, pois possui um quantidade 10 vezes maior de *jobs*. Outra diferença refere-se ao tempo de submissão entre os BoTs. Enquanto no *workload* 1 o tempo de entre BoTs é de 20 minutos, no *workload* 2 o tempo é de apenas 5 minutos. Com isso espera-se que os recursos alocados experimentem um maior compartilhamento entre os BoTs. Assim, quando o algoritmo de desligamento for executado haverá maior probabilidade de ocorrência de BoTs enfileirados aguardado pela liberação de recursos. Uma ilustração do *workload* 2 pode ser vista abaixo (figura 10).

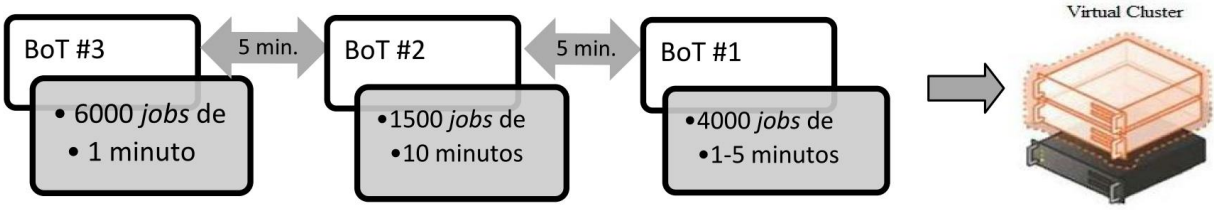


Figura 10: Ilustração do *workload* 2.

Outras características do *workload* 2 podem ser vistas na tabela 3. Note que o primeiro BoT possui uma variação no tempo execução das tarefas, provocando uma variação no *walltime* do *workload*. Em função dessa variação o $QMV_0$ também varia. Na terceira coluna é apresentado, a título de visualização, o *walltime* do *workload*. Contudo, lembrando que para calcular o $QMV_0$ e definir a quantidade da MVs a serem inicializadas para um BoT utiliza-se o *walltime* de cada BoT.

Tabela 3: Resumo das características do *workload* 2.

| Parâmetro / BoTs | ***Walltime* por BoT (min).** | **$QMV_0$ por BoT** | ***Walltime* Total (min).** |
|---|---|---|---|
| BoT #1 | [4.000, 20.000] | [96, 215] | [25.000, 41.000] |
| BoT #2 | 15.000 | 186 | |
| BoT #3 | 6.000 | 117 | |

### 5.2.3 Seleção de Métricas

O desempenho é quantificado a partir de quatro métricas relativas à execução de um *workload*: (1) quantidade total de máquinas iniciadas, refere-se a soma das quantidades de MVs utilizadas para execução de cada BoT, contabilizando inclusive as máquinas reutilizadas; (2) quantidade de máquinas ociosas reutilizadas, refere-se a quantidade de máquinas ociosas que, oportunisticamente, são utilizadas e que não serão iniciadas para um novo BoT. Tal ociosidade é fragmentada, pois está presente em várias MVs; (3)

tempo total de execução da aplicação, refere-se ao tempo total gasto para executar o BoT, considerando, inclusive, o tempo para alocação de recursos; (4) quantidade de *clusters* iniciados, trata-se da quantidade média de *clusters* iniciados para a execução de um *workload* composto por BoTs. Esta métrica é apresentada em termos percentuais.

## 5.3 Resultados Obtidos com o *Workload* 1

Para realização dos teste foram definidos 3 cenários de simulação. No primeiro cenário, os parâmetros $MV_{máx}$ e $U_{máx}$ são configurados com valores baixos. No segundo cenário, os parâmetros $MV_{máx}$ e $U_{máx}$ são configurados com valores altos e no terceiro cenário, é feita uma varredura em todos nos valores dos parâmetros. Para tal, define-se faixas de valores possíveis, onde os valores de $MV_{máx}$ dependem do $QMV_0$ dos BoTs do *workload* e os valores de $U_{máx}$ variam dentro dessa faixa: 0,05 < U < 0,95.

### 5.3.1 Teste de Validação

A validação dos algoritmos propostos foi feita através de uma comparação entre e *ElaticSite* e o *ElasticCluster*. Para tal, foi feita uma análise de desempenho nas duas propostas. Nesse teste foram executados os cenários 1 e 2.

#### 5.3.1.1 Cenário 1 - $MV_{máx}$ e $U_{máx}$ baixos

Para comparar o *ElasticSite* com *ElasticCluster*, os parâmetros $MV_{máx}$ e $U_{máx}$ foram configurados no *ElasticCluster* de tal forma que o desperdício de recursos seja grande. Logo, partindo da hipótese de que se gera um maior desperdício de recursos ao instanciar um *cluster* para cada BoT (assim como no *ElasticSite*) e instanciar um novo *cluster* quando todos os *clusters* existentes estiverem com uma baixa utilização, os parâmetros foram configurados com o $MV_{máx}$ igual a 70 (já que a máximo $QMV_0$ para um BoT do *workload* utilizado é igual a 68) e o $U_{máx}$ igual a 0,35. Foram realizados 270 submissões do *workload* (3 BoTs x 90 possíveis taxas de utilização que o BoT pode estabelecer no *cluster*), 30 repetições e intervalo de confiança igual a 95%.

Os resultados obtidos podem ser observados na figura 11. O número total médio de máquinas iniciadas (figura 11a) cai de 151 (*ElasticSite*) para 120 (*ElasticCluster*), i.e., uma diminuição de 20,5%. Isto ocorre em função da reutilização da ociosidade do *cluster* entre as rajadas de BoT, evitando a criação de novas MVs e de novos *clusters*. Neste cenário, a configuração dos parâmetros teve objetivo de experimentar um grande desperdício de recursos pelo *ElasticCluster*. O número médio de máquinas ociosas reutilizadas pelo *ElasticCluster* foi igual a 35. Na figura 11b é possível observar o tempo de execução do *workload* em cada proposta. Note que ocorre uma diminuição no tempo de execução de 60 para 55 minutos devido ao reuso das máquinas e, consequentemente, menor tempo de espera por recurso. Logo, o tempo que seria gasto para iniciar novas

máquinas é utilizado para execução dos BoTs.

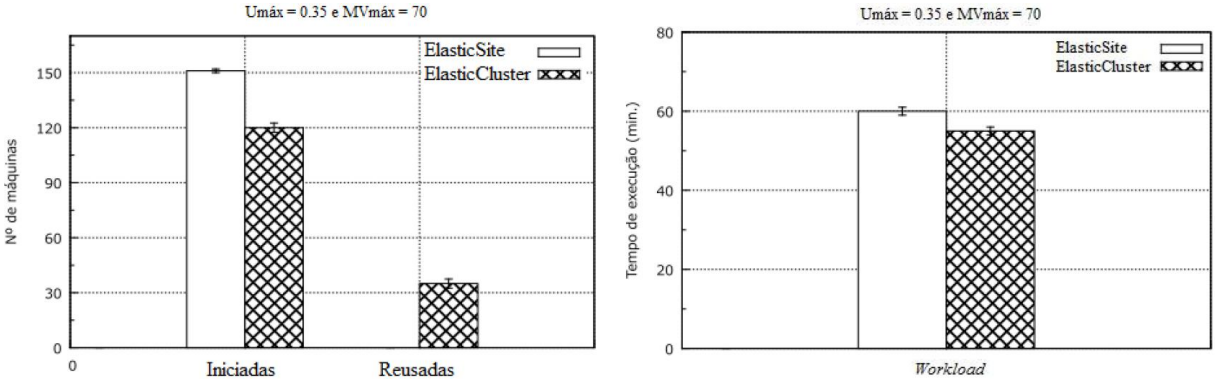


Figura 11: Execução do *workload 1* no *ElasticSite* e no *ElasticCluster* com $U_{máx}$ e $MV_{máx}$ baixos (a) Número médio de máquinas instanciadas e reusadas (b) Tempo de execução dos BoTs.

Vale destacar ainda que, devido ao baixo $MV_{máx}$ configurado, ocorreu a criação de um novo *cluster* em 82% das 270 submissões dos BoTs da simulação realizada. Mesmo quando a ociosidade era grande, o baixo $MV_{máx}$ impedia a inclusão de máquinas no *cluster*. Portanto, ocorreu a instanciação de um novo *cluster*, e, consequentemente, o acréscimo do *overhead* da instanciação de $MV_0$ novas máquinas, além do não aproveitamento da ociosidade nos *clusters* já instanciados. Logo, o fato do $U_{máx}$ ter sido configurado para 0,35 implica que, para qualquer utilização maior ou igual a 35%, o *ElasticCluster* preferia a criação de um novo *cluster*. Ou seja, na maioria dos casos o *ElasticCluster* optava por adicionar *clusters*. Consequentemente, necessitando de um maior tempo para alocar recursos, o que impactou diretamente no tempo total de execução do *workload*.

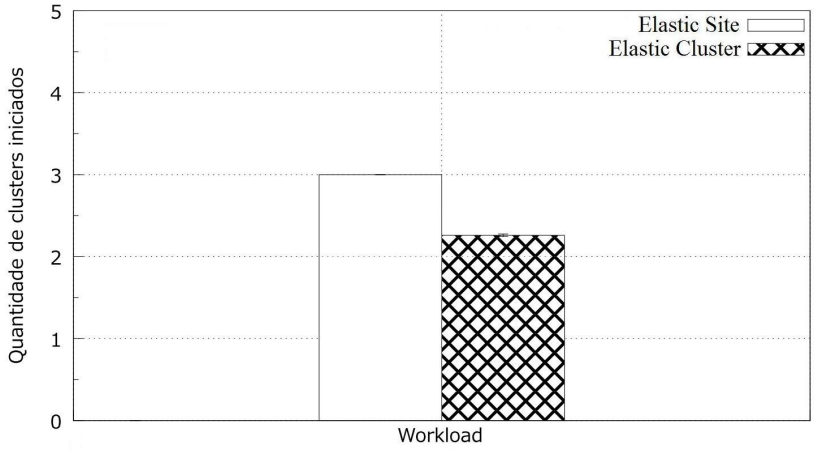


Figura 12: Quantidade média de *Clusters* iniciados para a execução do *workload* com $MV_{máx}$ e $U_{máx}$ baixos

Na figura 12 é possível observar a quantidade média de *clusters* criados para a execução do *workload* em cada proposta. O *ElasticCluster* iniciou em média 2,26 *clusters*

e o *ElasticSite* iniciou, em média, 3 *clusters*, pois realiza a instanciação de um novo *cluster* para cada BoT.

#### 5.3.1.2 Cenário 2 - $MV_{máx}$ e $U_{máx}$ altos

No segundo cenário avaliado, os parâmetros $MV_{máx}$ e $U_{máx}$ foram configurados com valores altos. A princípio, com valores altos o desperdício de recursos deve ser pequeno. Dessa forma, partindo da hipótese de que se gera um menor desperdício de recursos ao instanciar apenas um *cluster* para todos os BoTs, o *ElasticCluster* é configurado com o $MV_{máx}$ igual a 180 (já que o somatório dos máximos valores de $QMV_0$ de todos os BoTs do *workload 1* é igual a 164) e o $U_{máx}$ igual a 0,95. Foram realizados 270 submissões (3 BoTs x 90 possíveis taxas de utilização que o BoT pode estabelecer no *cluster*), 30 repetições e intervalo de confiança igual a 95%.

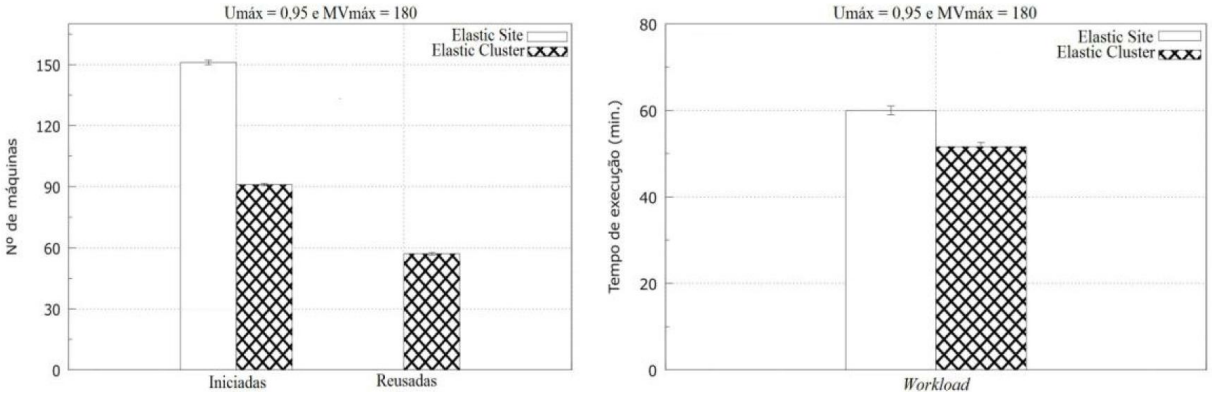


Figura 13: Execução do *workload 1* no *ElasticSite* e no *ElasticCluster* com $U_{máx}$ e $MV_{máx}$ altos (a) Número médio de máquinas instanciadas e reusadas (b) Tempo de execução.

Os resultados obtidos podem ser observados na figura 13. O número total médio de máquinas iniciadas cai de 151 (*ElasticSite*) para apenas 90 (*ElasticCluster*), i.e uma diminuição de 40,4%. Isto ocorre mais uma vez devido à reutilização da ociosidade do *cluster* entre as rajadas de BoT, evitando a criação de novas MVs e de novos *clusters*. O número médio de máquinas ociosas reutilizadas pelo *ElasticCluster* foi igual a 58. Portanto, mais máquinas reutilizadas do que quando os parâmetros $U_{máx}$ e $MV_{máx}$ são configurados com valores baixos, o que confirma a diminuição mais acentuada do número de MVs iniciadas. O aumento no número de MVs reusadas diminui também o tempo total de execução do *workload*. Na figura 13b é possível observar o tempo de execução do *workload 1* em cada proposta. Para o *ElasticCluster* ocorre uma diminuição no tempo de execução em relação ao cenário 1. Agora o tempo de execução foi de apenas 51,6 minutos (3091 segundos) em média. Como no *ElasticSite* não há reuso de MVs, o mesmo mais uma vez executa o *workload* em um tempo maior (60 minutos).

Devido ao alto $MV_{máx}$ configurado, em todas as simulações ocorreu a criação de apenas 1 *cluster* para cada submissão do *workload*. O fato do $U_{máx}$ ter sido configurado para 0,95 implica que, para qualquer utilização maior ou igual a 95%, o *ElasticCluster* optasse pela criação de um novo *cluster*. Como o consumo que os BoTs podem fazer

dos *clusters* varia de 5% a 95%, então, apenas para uma utilização igual a 95% poderia ocorrer a criação de um novo *cluster*. Nesse cenário a quantidade de MVs reutilizadas foi máxima, visto que não havia "limite"para o crescimento do *cluster*, então todas as máquinas ociosas eram sempre reutilizadas em todas as submissões de BoTs.

Na figura 14, observa-se a quantidade média de *clusters* criados no cenário 2. O *ElasticCluster* criou em média apenas um *cluster*, enquanto *ElasticSite* criou 3 *clusters* em média para execução do *workload* em cada uma das 270 submissões do *workload* na simulação realizada.

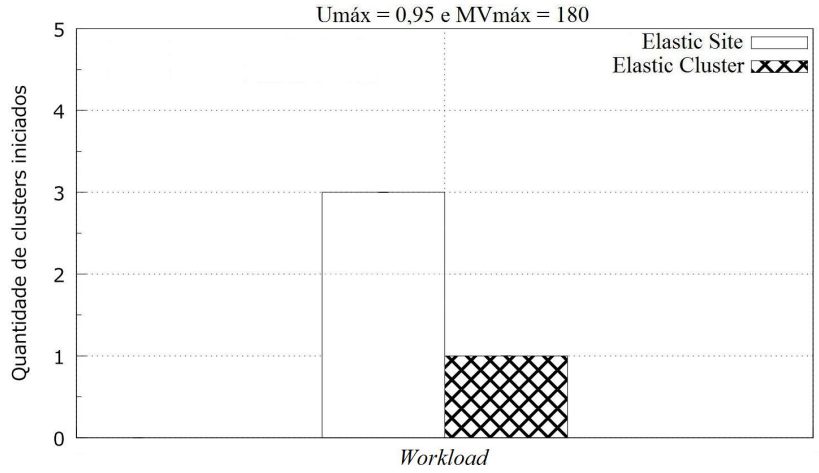


Figura 14: Quantidade média de *Clusters* iniciados para a execução do *Workload* 1 com $MV_{máx}$ e $U_{máx}$ altos.

Nos cenários simulados, o *ElasticCluster* apresentou-se melhor do que o *ElasticSite* em todas as métricas avaliadas. Contudo, com os testes de validação não foi possível determinar o melhor cenário de operação do *ElasticCluster*, visto que o parâmetro $U_{máx}$ pode assumir vários valores e o valor de $MV_{máx}$ varia em função do *workload* submetido. Além disso, para valores altos de $U_{máx}$ e $MV_{máx}$ observa-se uma grande instabilidade nos *clusters* (YAN et al., 2010). Assim, a fim determinar uma configuração que diminua ao máximo o desperdício de recursos na execução do *workload* utilizado, outras simulações são apresentadas na próxima seção. Para tal, foi feita a variação dos valores de $U_{máx}$ e $MV_{máx}$ dentro das faixas de valores pré-estabelecidas de acordo com o *workload* a fim de determinar os melhores valores para configuração dos algoritmos propostos.

### 5.3.2 Varredura dos valores de $MV_{máx}$ e $U_{máx}$

O teste de varredura dos valores de $MV_{máx}$ e $U_{máx}$ foi feito para determinar o melhor cenário de operação do *ElasticCluster*. Nesse teste, foi executado o cenário 3.

#### 5.3.2.1 Cenário 3 - Varredura dos valores de $MV_{máx}$ e $U_{máx}$

A escolha dos valores para o fator $U_{máx}$ foi realizada a fim de dividir a faixa que compreende todos os valores possíveis de U, i. e., 0,05 < U < 0,95, em alguns valores que revelassem de que forma o fator $U_{máx}$ altera a quantidade de máquinas iniciadas e reusadas, determinando qual o $U_{máx}$ proporciona menor desperdício de recursos. Tais valores são: 0,35; 0,40; 0,45; 0,50; 0,55; 0,65; 0,75; 0,85 e 0,95. Os valores de $MV_{máx}$ foram definidos com base na quantidade de MVs a serem disponibilizadas para cada BoT. Assim como no experimento anterior, é utilizado o $MV_{máx} = 70$ e à partir deste são definidos os outros dois valores: $MV_{máx} = 140$ (o dobro) e $MV_{máx} = 180$ ( $> \Sigma(QMV_0)$).

Assim como na seção anterior, foram realizadas 30 replicações com 270 submissões do *workload* cada uma. Cada replicação possui intervalo de confiança igual a 95% para todas as combinações possíveis entre os valores de $U_{máx}$ e $MV_{máx}$. A partir disso foi feita uma análise de cada combinação dos valores de $U_{máx}$ e $MV_{máx}$ em relação a reutilização de recursos ociosos.

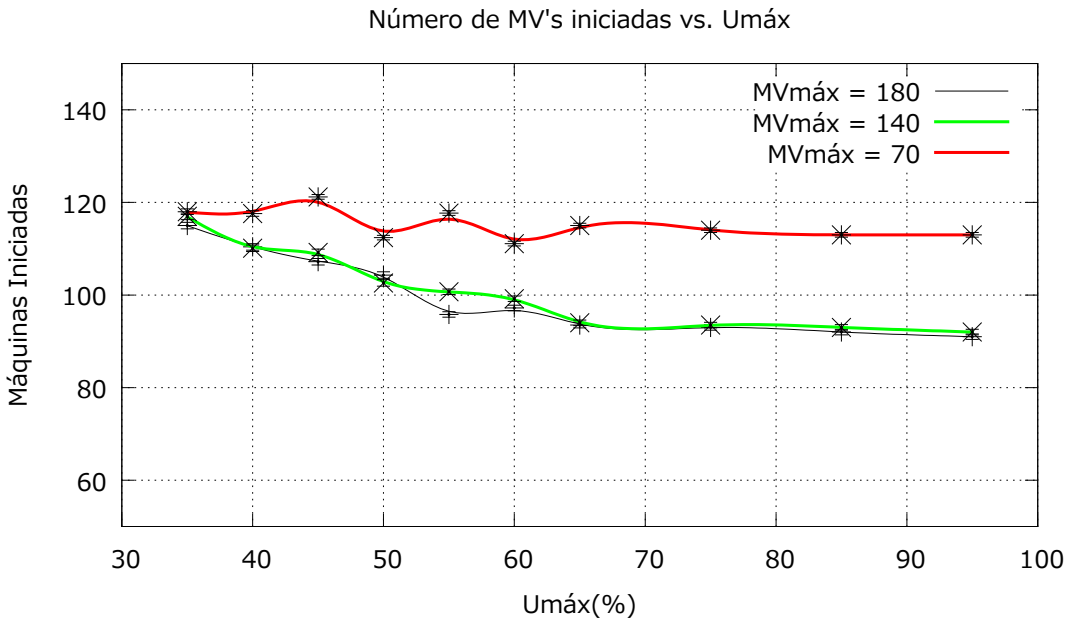


Figura 15: Número de MVs iniciadas vs. $U_{máx}$.

De acordo com a figura 15, para $MV_{máx} = 70$, o número de máquinas iniciadas é alto independente do valor de $U_{máx}$. Nessas condições, muitos novos *clusters* são criados devido ao baixo $MV_{máx}$, que possui valor próximo aos valores de $MV_0$ dos BoTs. Foram criados novos *clusters* em 73% das submissões dos BoTs. Consequentemente, a reutilização de recursos ociosos é baixa, conforme ilustrado na figura 16. Um número menor que máquinas iniciadas pode ser observado para o $MV_{máx} = 140$ e $MV_{máx} = 180$, visto que o número de novos *clusters* diminui. Para esses valores ocorre uma maior incidência de adaptações. Foram criados novos *clusters* em 50% das submissões dos BoTs. Logo, em média, são criados dois *clusters* para executar o *workload* para o $MV_{máx} = 140$ e $MV_{máx} = 180$.

Vale destacar ainda que, o número máquinas iniciadas é similar para os maiores valores de $MV_{máx}$, não havendo necessidade, portanto, de se configurar um alto valor para $MV_{máx}$. Dessa forma, a quantidade de recursos ociosos tende a diminuir. O $MV_{máx} = 140$ tem valor próximo ao somatório dos $QMV_0$ de todos dos BoTs ($\Sigma(QMV_0)$) calculados pela heurística de (MARSHALL; KEAHEY; FREEMAN, 2010) que é igual a 145. Além disso, de acordo com as figuras 15 e 16, é possível observar, que não há necessidade de manter os *clusters* com alta utilização. Ou seja, mantendo os *clusters* com uma utilização média de 65% e configurando o $MV_{máx}$ para um valor próximo do $\Sigma(QMV_0)$ resulta na instanciação de uma baixa quantidade de máquinas e, consequentemente, em um baixo desperdício de recursos sem comprometer a disponibilidade do *cluster* e sem comprometer o tempo de execução do BoT. Na figura 15, é possível notar ainda que, para o cenário em que $U_{máx} = 0.95$ e $MV_{máx} = 180$, o número de MVs iniciadas é de 91 em média. Como o *ElasticSite* inicia 151 MVs em média, então com o *ElasticCluster* a economia no número de MVs iniciadas pode chegar a até 40%.

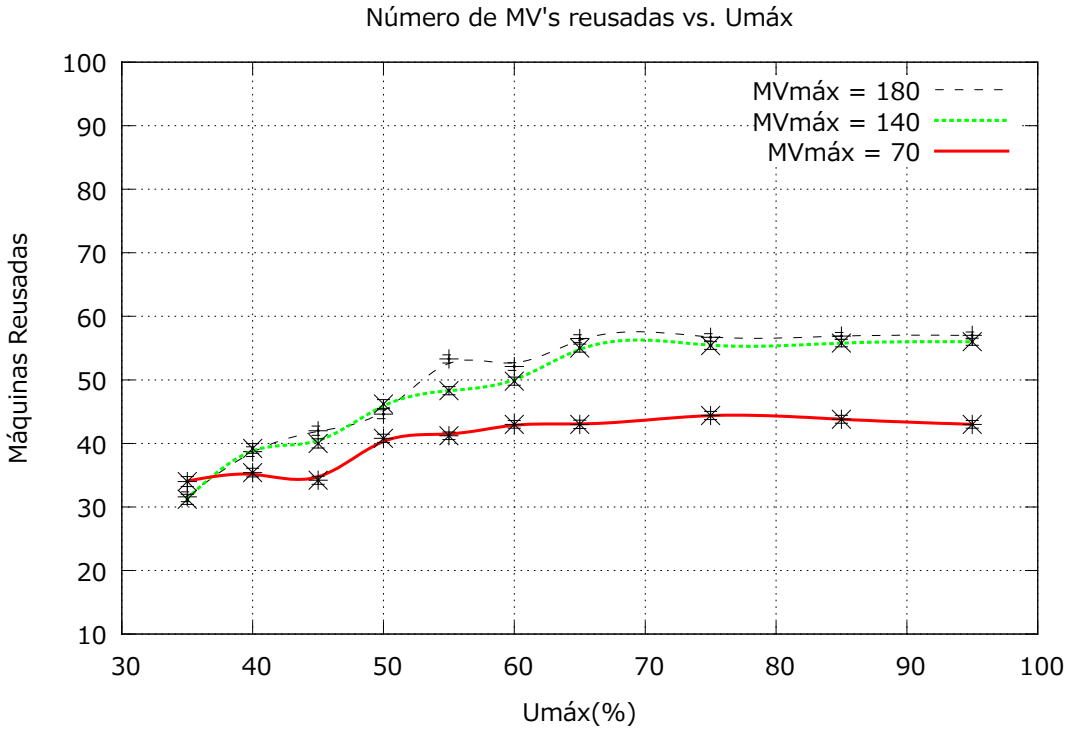


Figura 16: Número de MVs reusadas vs. $U_{máx}$.

Em relação ao número de máquinas reusadas, é possível notar na figura 16 que, para os maiores valores de $MV_{máx}$ o reuso é maior. Contudo, esse reuso não cresce linearmente com o $U_{máx}$. À partir do $U_{máx}$ igual a 65%, a quantidade de MVs reutilizadas se estabiliza em torno de 56 máquinas. Dessa forma, é possível obter um bom nível de reaproveitamento de recursos, mantendo os *clusters* com uma taxa de utilização mediana, gerando, portanto, menos desperdício de energia (MUKHERJEE et al., 2007).

Na figura 17, possível observar a curva que mostra a variação tempo execução do *workload* em função do $U_{máx}$. Para $MV_{máx}$ igual a 70 observa-se que o tempo de execução

sofre uma pequena variação em torno de um tempo igual 3.300 segundos. Isso ocorre por que, para praticamente cada BoT do *workload* é criado um novo *cluster*, provocando um pequeno reuso de MVs (conforme figura 16). De tal forma, que o *overhead* da criação de novas MVs aumenta, prejudicando o tempo total de execução. Para $MV_{máx}$ igual a 140, o tempo de execução diminui para até 3.127 segundos. Nesse caso, é possível observar que para as utilizações maiores que 60%, o tempo de execução permanece praticamente constante. Isso se dá devido, novamente, à variação do número de MVs reutilizadas por taxa de utilização, confirme figura 16. Comportamento similar é observado para $MV_{máx}$ igual a 180. Contudo, para esse último valor de $MV_{máx}$ o tempo de execução é ainda menor, visto que para cada uma das 270 submissões de BoTs realizadas apenas um *cluster* era criado, aumentando ao máximo o número de MVs reusadas e diminuindo a máximo o número de *clusters* iniciados.

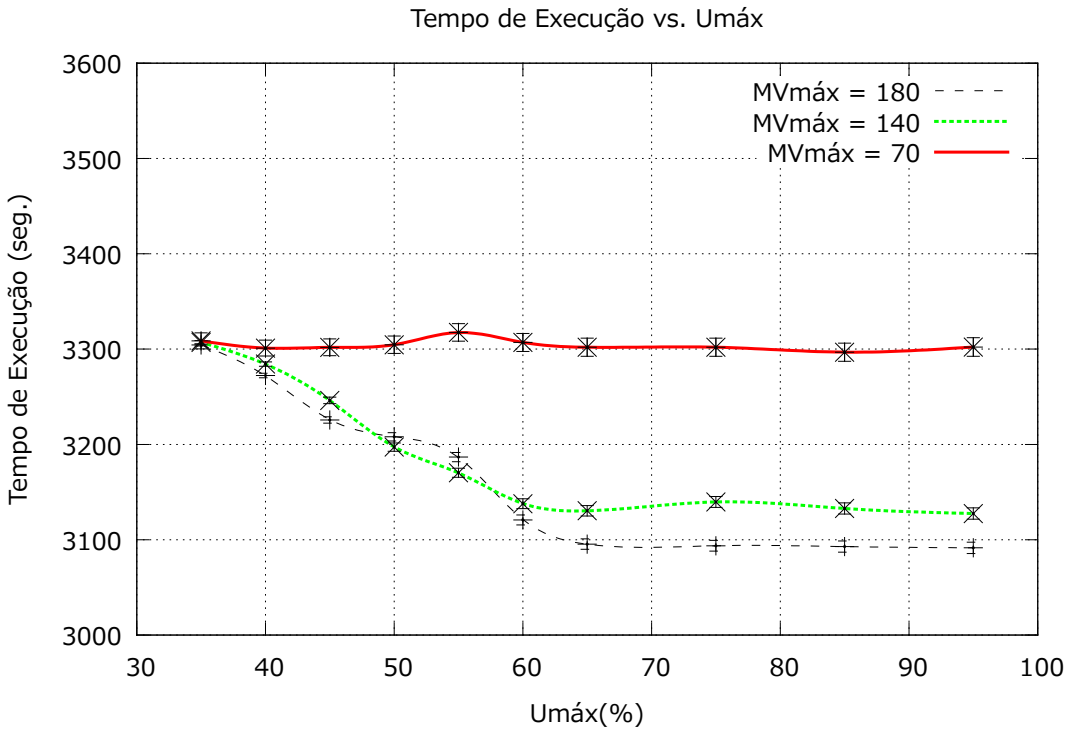


Figura 17: Tempo de Execução vs. $U_{máx}$.

Na figura 18 é possível observar o percentual de submissões em que se iniciou apenas 1 *cluster*. Note que cada percentual está associado a um par de possibilidades de valores dos parâmetros $MV_{máx}$ e $U_{máx}$. Conforme já deduzido, essa métrica possui melhores resultados para altos valores de $MV_{máx}$ e de $U_{máx}$. Para o *workload* submetido, o melhor dos casos (um *cluster* iniciado) ocorre quando $MV_{máx}$ é igual a 180, ou seja, $MV_{máx}$ é $> \Sigma(QMV_0)$. Nesse caso, em 100% das submissões do *workload* ocorre a criação de apenas um 1 *cluster*. Dessa forma, é possível obter o máximo nível de reaproveitamento de recursos, mantendo os *clusters* com uma taxa de utilização máxima. Contudo, gera-se um maior desperdício de energia (MUKHERJEE et al., 2007) (LIAO; JIN; LIU, 2012) e uma maior instabilidade para o *cluster* (BOBROFF; KOCHUT; BEATY, 2007) (YAN et al., 2010).

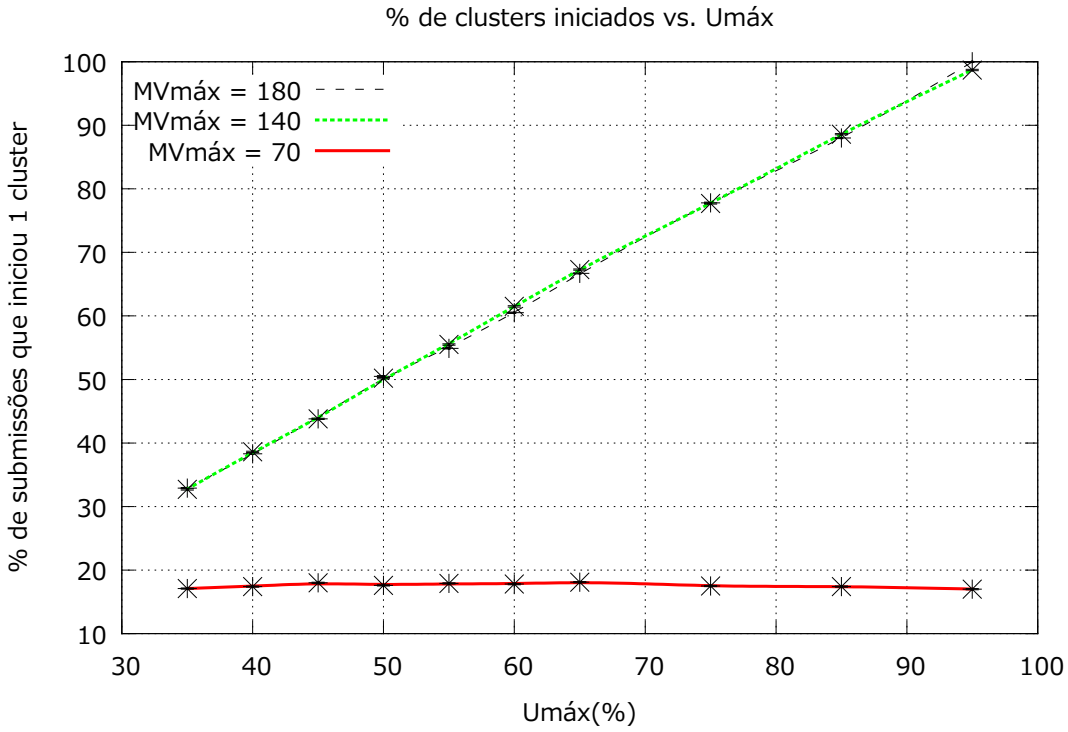


Figura 18: Número de *Clusters* Iniciados vs. $U_{máx}$.

Do exposto, conclui-se que o melhor cenário de operação dos algoritmos proposto ocorre quando o $U_{máx}$ é configurado com valores medianos, em torno de 65% e o $MV_{máx}$ é configurado com um valor próximo do dobro da menor quantidade de MVs calculada para um BoT de um *workload* de acordo com a heurística de (MARSHALL; KEAHEY; FREEMAN, 2010). Dessa forma, para um $U_{máx}$ de 65% é possível garantir que em, aproximadamente, 70% das submissões ocorrerá a criação de apenas um *cluster*. Com isso, obtém-se maior economia de energia, uma maior estabilidade do *cluster*, baixo tempo de execução do *workload* e maior aproveitamento dos recursos, haja vista que tem-se também um baixo número de MVs iniciadas e um alto número de MVs reutilizadas.

## 5.4 Resultados Obtidos com o *Workload* 2

O teste com o segundo *workload* foi realizado com o terceiro cenário utilizado no teste com o workload 1. Cenário esse em que é feita uma varredura os valores dos parâmetros. Neste cenário, definiu-se valores para $MV_{máx}$ e $U_{máx}$, onde os valores de $MV_{máx}$ dependem do $QMV_0$ dos BoTs do *workload* e os valores de $U_{máx}$ variam dentro dessa faixa: 0,05 < U < 0,95.

### 5.4.1 Teste de Escalabilidade

O segundo *workload* foi definido com uma quantidade de *jobs* 10 vezes maior que a quantidade do *workload* 1 afim de observar o comportamento do ElasticCluster diante de cargas de trabalho maiores.

#### 5.4.1.1 Cenário 3 - Varredura dos valores de $MV_{máx}$ e $U_{máx}$ no *workload* 2

Os valores possíveis de $MV_{máx}$ foram definidos de acordo com o somatório os valores de $QMV_0$ de cada BoT do *workload* 2 apresentado na tabela 3, dessa forma o primeiro valor escolhido para $MV_{máx}$ foi 180, que é maior que o valor médio de $QMV_0$ que, por possuir variação do valor do *walltime* de cada BoT, também possui valor variado de $QMV_0$. À partir do primeiro valor são definidos os outros dois valores: $MV_{máx} = 360$ (o dobro) e $MV_{máx} = 540$ (o triplo e $> \Sigma(QMV_0)$). Os valores de $U_{máx}$ continuam sendo: 0,35; 0,40; 0,45; 0,50; 0,55; 0,65; 0,75; 0,85 e 0,95.

Assim como nos experimentos anteriores, foram realizadas 30 replicações com 270 submissões do *workload* cada uma. Cada replicação possui intervalo de confiança igual a 95% para todas as combinações possíveis entre os valores de $U_{máx}$ e $MV_{máx}$. A partir disso foi feita uma análise de cada combinação dos valores de $U_{máx}$ e $MV_{máx}$ em relação a reutilização de recursos ociosos.

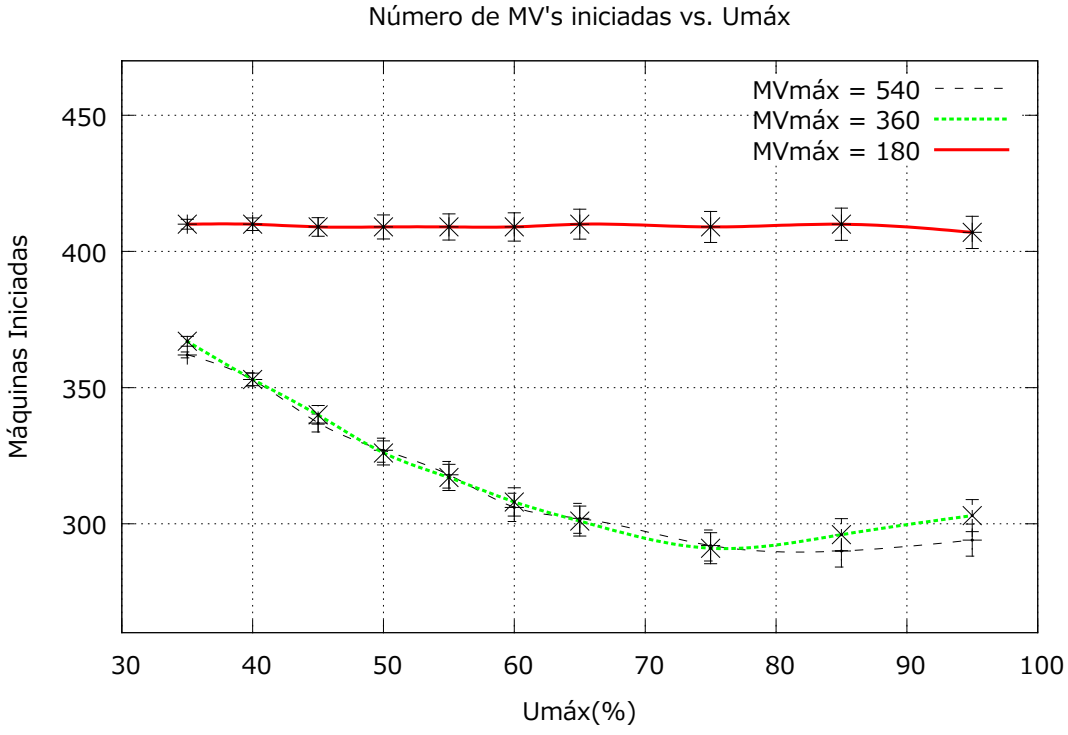


Figura 19: Número de MVs iniciadas vs. $U_{máx}$.

De acordo com a figura 19, para $MV_{máx} = 180$, assim como no *workload* 1, o número de máquinas iniciadas é alto independente do valor de $U_{máx}$. Nessas condições,

muitos novos *clusters* são criados devido ao baixo $MV_{máx}$, que possui valor próximo aos valores de $MV_0$ dos BoTs. Foram criados novos *clusters* em 85% das submissões dos BoTs. Consequentemente, a reutilização de recursos ociosos é baixa, conforme ilustrado na figura 20. Um número menor que máquinas iniciadas pode ser observado para o $MV_{máx}$ = 360 e $MV_{máx}$ = 540, visto que o número de novos *clusters* diminui. Para esses valores ocorre uma maior incidência de adaptações. Foram criados novos *clusters* em 50% das submissões dos BoTs. Logo, em média, são criados dois *clusters* para executar o *workload* 2 para o $MV_{máx}$ = 360 e $MV_{máx}$ = 540. Portanto, o mesmo comportamento observado no *workload* 1 em relação à quantidade máquinas iniciadas também foi observado no *workload* 2.

Vale destacar ainda que, o número máquinas iniciadas é similar para os maiores valores de $MV_{máx}$, não havendo necessidade, portanto, de se configurar um alto valor para $MV_{máx}$. Dessa forma, a quantidade de recursos ociosos tende a diminuir. Além disso, de acordo com as figuras 19 e 20, é possível observar, que não há necessidade de manter os *clusters* com alta utilização. Ou seja, mantendo os *clusters* com uma utilização média de 65% e configurando o $MV_{máx}$ para um valor próximo do $\Sigma(QMV_0)$ resulta na instanciação de uma baixa quantidade de máquinas e, consequentemente, em um baixo desperdício de recursos sem comprometer a disponibilidade dos *clusters* e sem comprometer o tempo de execução dos BoTs do *workload* 2.

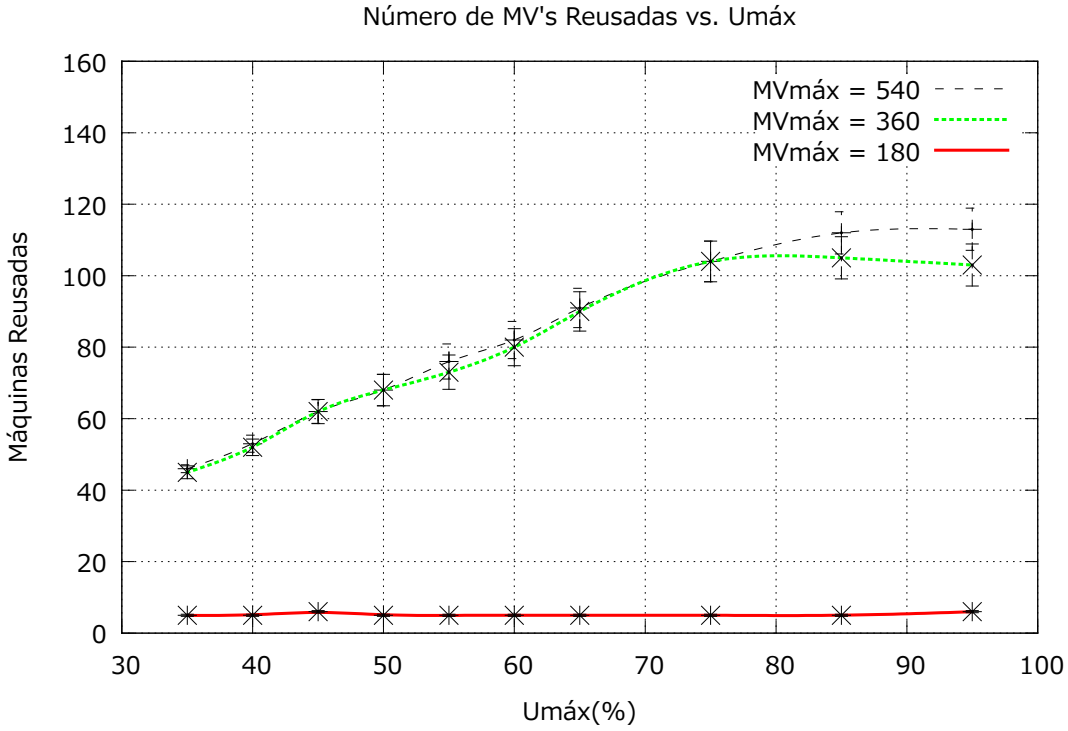


Figura 20: Número de MVs reusadas vs. $U_{máx}$.

Em relação ao número de máquinas reusadas, é possível notar na figura 20 que, para os maiores valores de $MV_{máx}$ o reuso é maior. Contudo, esse reuso não cresce linearmente com o $U_{máx}$. À partir do $U_{máx}$ igual a 65%, a quantidade de MVs reutilizadas

se estabiliza em torno de 110 máquinas. Dessa forma, é possível obter um bom nível de reaproveitamento de recursos, mantendo os *clusters* com uma taxa de utilização mediana, gerando, portanto, menos desperdício de energia (MUKHERJEE et al., 2007).

Na figura 21, possível observar a curva que mostra a variação tempo execução do *workload* 2 em função do $U_{máx}$. Para $MV_{máx}$ igual a 180 observa-se que o tempo de execução sofre uma pequena variação em torno de um tempo igual 16.300 segundos. Isso ocorre por que, para praticamente cada BoT do *workload* é criado um novo *cluster*, provocando um pequeno reuso de MVs (conforme figura 20). De tal forma, que o *overhead* da criação de novas MVs aumenta, prejudicando o tempo total de execução. Para $MV_{máx}$ igual a 360, o tempo de execução diminui para até 10.100 segundos. Nesse caso, é possível observar que para as utilizações maiores que 60%, o tempo de execução permanece praticamente constante. Isso se dá devido, novamente, à variação do número de MVs reutilizadas por taxa de utilização, confirme figura 20. Comportamento similar é observado para $MV_{máx}$ igual a 540. Contudo, para esse último valor de $MV_{máx}$ o tempo de execução é ainda menor, visto que para cada uma das 270 submissões de BoTs realizadas apenas um *cluster* era criado, aumentando ao máximo o número de MVs reusadas e diminuindo a máximo o número de *clusters* iniciados. Como isso é possível notar que em relação ao tempo de execução o comportamento observado para o *workload* 1 também é observado no *workload* 2.

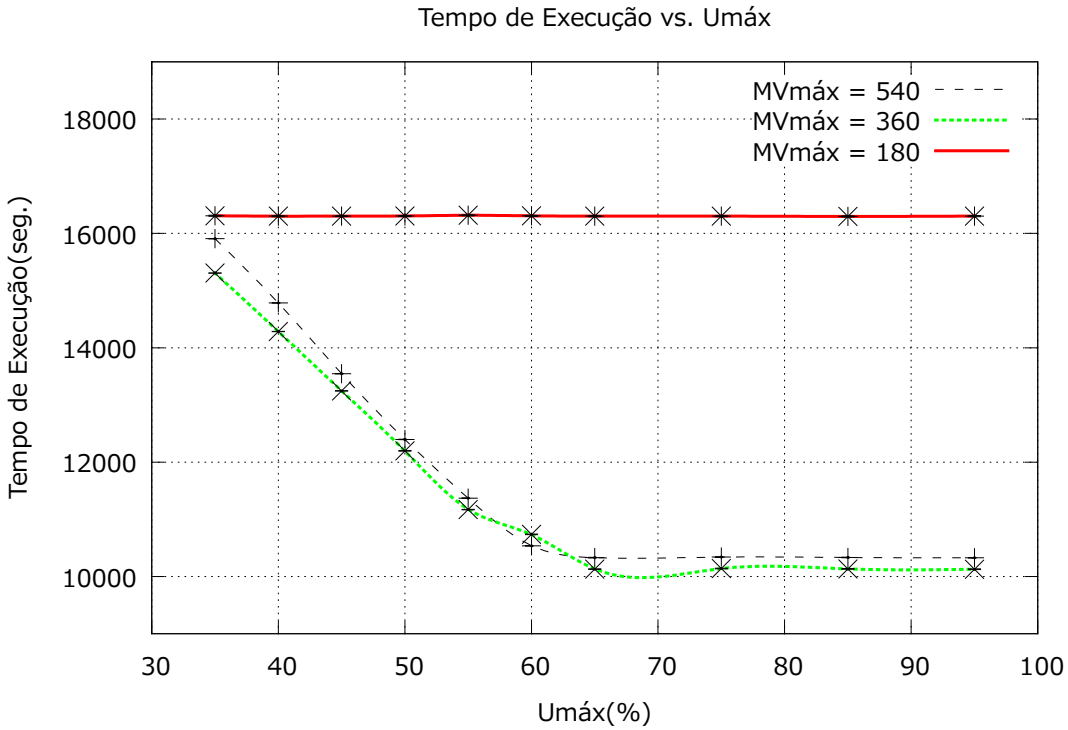


Figura 21: Tempo de Execução vs. $U_{máx}$.

Na figura 22 é possível observar o percentual de submissões em que se iniciou apenas 1 *cluster*. Note que cada percentual está associado a um par de possibilidades de valores dos parâmetros $MV_{máx}$ e $U_{máx}$. Conforme já deduzido, essa métrica possui

melhores resultados para altos valores de $MV_{máx}$ e de $U_{máx}$. Para o *workload* submetido, o melhor dos casos (um *cluster* iniciado) ocorre quando $MV_{máx}$ é igual a 540, ou seja, $MV_{máx}$ é $> \Sigma(QMV_0)$. Nesse caso, em 100% das submissões do *workload* ocorre a criação de apenas um 1 *cluster*. Dessa forma, é possível obter o máximo nível de reaproveitamento de recursos, mantendo os *clusters* com uma taxa de utilização máxima. Contudo, gera-se um maior desperdício de energia (MUKHERJEE et al., 2007) (LIAO; JIN; LIU, 2012) e uma maior instabilidade para o *cluster* (BOBROFF; KOCHUT; BEATY, 2007) (YAN et al., 2010). Para essa métrica em especial, foi possível observar comportamento diferente da proposta em relação ao *workload* 1. Note que para $MV_{máx} = 360$ o percentual de submissões que iniciou apenas 1 *clusters* para de crescer em torno de 65%. Logo, para *workloads* maiores e, portanto, com maiores uma quantidade maior de amostras revelou-se que a mesmo comportamento das curvas das outras métricas em que mantendo-se os *clusters* com uma utilização média de 65% e configurando o $MV_{máx}$ para um valor próximo do $\Sigma(QMV_0)$ resulta na instanciação de uma baixa quantidade de máquinas e de *clusters*.

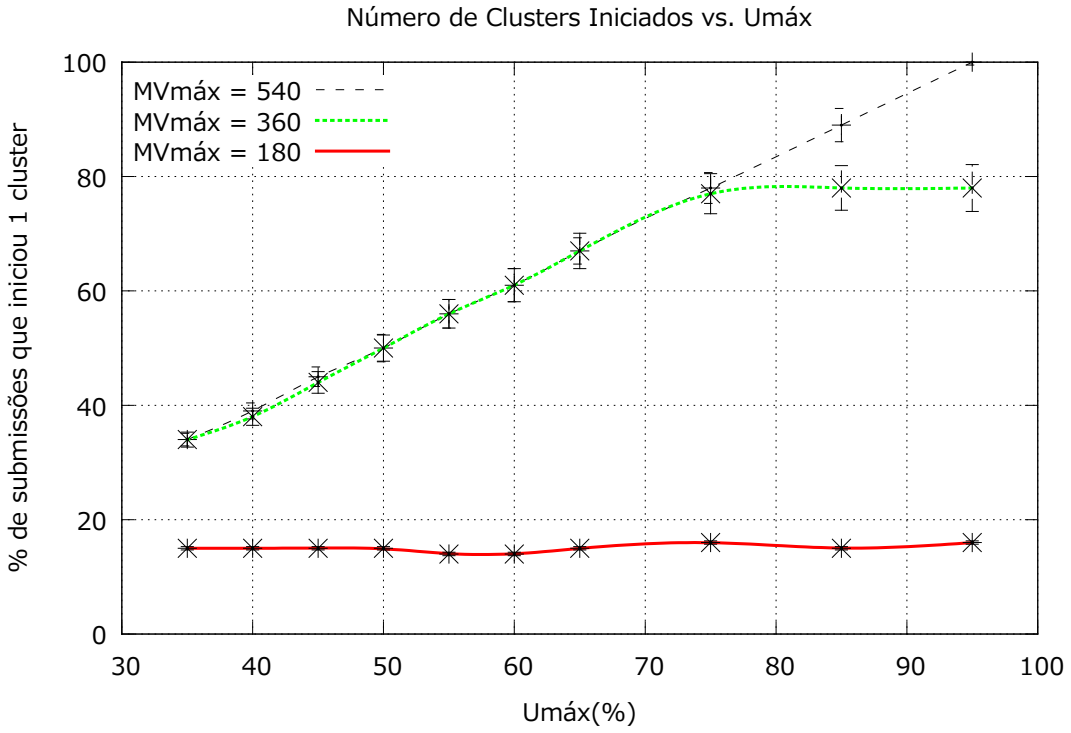


Figura 22: Número de *Clusters* Iniciados vs. $U_{máx}$.

Do exposto, conclui-se que, assim como no *workload* 1, o melhor cenário de operação dos algoritmos proposto para o *workload* 2 também ocorre quando o $U_{máx}$ é configurado com valores medianos, em torno de 65% e o $MV_{máx}$ é configurado com um valor próximo do dobro da menor quantidade de MVs calculada para um BoT de um *workload* de acordo com a heurística de (MARSHALL; KEAHEY; FREEMAN, 2010). Dessa forma, para um $U_{máx}$ de 65% é possível garantir que, em aproximadamente, 70% das submissões, ocorrerá a criação de apenas um *cluster*. Com isso, obtém-se maior economia de energia, uma maior estabilidade do *cluster*, baixo tempo de execução do *workload* e

maior aproveitamento dos recursos, haja vista que tem-se também um baixo número de MVs iniciadas e um alto número de MVs reutilizadas.

## 5.5 Conclusões

A proposta foi detalhadamente verificada e validada por simulações com o provedor de recursos virtualizados desenvolvido com Redes de Petri coloridas. O desempenho da proposta é avaliado em 3 (três) cenários distintos à partir de 4 (quatro) métricas: quantidade total de máquinas iniciadas, quantidade de máquinas ociosas reutilizadas, tempo total de execução da aplicação e quantidade de *clusters* iniciados.

Os testes de validação (seção 5.3.1) mostraram de forma geral que o algoritmo de instanciação de MVs iniciou uma quantidade menor de MVs se comparado com o *ElasticSite*. O teste de validação 1 (seção 5.3.1.1) mostrou que mesmo quando o *ElasticCluster* é configurado para trabalhar com um grande desperdício de recursos, ainda assim foi capaz de reduzir a quantidade de MVs iniciadas em torno de 20% em ralação ao *ElasticSite*. O teste de validação 2 (seção 5.3.1.2) mostrou que algoritmo de instanciação de MVs pode oferecer um ganho ainda maior na economia de recursos, podendo chegar até a mais de 40% para o número de MVs iniciadas. Além disso, notamos que a quantidade de MVs reutilizadas é máxima e o tempo de execução e a quantidade de *clusters* iniciados são mínimos, mas esse cenário pode causar instabilidade nos *clusters* e grande consumo de energia.

Os resultados do teste de varredura dos valores do parâmetros $MV_{máx}$ e de $U_{máx}$ (seções 5.3.2) mostraram que: (1) o melhor cenário de operação dos algoritmos propostos ocorre quando o $U_{máx}$ é configurado com valores medianos, em torno de 65% e o $MV_{máx}$ é configurado com um valor próximo do dobro da menor quantidade de MVs calculada para um BoT de um *workload* de acordo com a heurística de (MARSHALL; KEAHEY; FREEMAN, 2010); (2) para um $U_{máx}$ de 65% é possível garantir que em, aproximadamente, 70% das submissões ocorrerá a criação de apenas um *cluster*; (3) para um $U_{máx}$ igual a 65% obtém-se maior economia de energia, uma maior estabilidade do *cluster*, baixo tempo de execução do *workload* e maior aproveitamento dos recursos, haja vista que tem-se também um baixo número de MVs iniciadas e um alto número de MVs reutilizadas.

# 6 CONSIDERAÇÕES FINAIS

Nesta dissertação, propomos, modelamos e simulamos uma política para alocação e desalocação dinâmica de MVs em *clusters* capaz de realizar uso efetivo dos recursos computacionais, além de reduzir o consumo de energia sem comprometer os requisitos de desempenho do sistema. Na Seção 6.1, são apresentadas as conclusões obtidas a partir dos resultados apresentados no capítulo anterior e a Seção 6.2 lista as perspectivas de trabalhos futuros que podem ser desenvolvidos a partir desta dissertação.

## 6.1 Conclusões

O principal objetivo deste trabalho foi desenvolver uma política para alocação dinâmica de MVs em *clusters* visando a redução do número de MVs alocadas, sem promover sobrecargas que podem comprometer o desempenho e a disponibilidade dos recursos alocados.

Para que esse objetivo fosse alcançado, propusemos, modelamos e simulamos dois algoritmos: algoritmo de instanciação e algoritmo de desligamento. O algoritmo de instanciação de MVs contém a política proposta e também é responsável por: (1) calcular a quantidade de máquinas para execução do *workload* submetido em função do *walltime* do BoT e do *wastetime* da nuvem em que os recursos serão alocados e (2) definir a necessidade da instanciação de novas MVs ou reutilização de MVs já iniciadas. Portanto, esses dois algoritmos atuam em conjunto levando em consideração os recursos já alocados e desempenho da nuvem. Assim, o primeiro objetivo específico desta dissertação foi alcançado.

A decisão por alocar novos recursos é tomada pela política em função também do número máximo de MVs ($MV_{máx}$) e da utilização máxima ($U_{máx}$) que um *clusters* pode ter. A política proposta reutiliza a ociosidade gerada no *cluster* devido à demanda variada das aplicações submetidas e do padrão de submissão dos BoTs, geralmente, em rajadas. O algoritmo de desligamento é responsável por desativar as MVs após a execução ou cancelamento do BoT submetido, verificando se as MVs estão sendo compartilhadas com a execução de um outro BoT e a existência de BoTs enfileirados aguardando escalonamento. Por consequência, o segundo objetivo específico foi atingido.

O algoritmo de inicialização de MVs e o algoritmo de desligamento de MVs automatizam o processo de adaptação de *clusters* virtuais com a inicialização e desligamento

de MVs de acordo com cargas de trabalho dinâmicas. A quantidade de MVs envolvidas e o instante de tempo em que os processos ocorrem são determinados através de heurística. Dessa forma, o terceiro objetivo específico foi obtido.

Além disso, o algoritmo de instanciação estende a heurística de alocação proposta no trabalho de (MARSHALL; KEAHEY; FREEMAN, 2010) para definir a quantidade de MVS que serão disponibilizadas de modo a minimizar a quantidade total de MVs iniciadas, o tempo total de execução das aplicações e quantidade média de *clusters* iniciados. Logo, o quarto objetivo específico foi obtido.

É apresentada uma modelagem feita com Redes de Petri de um provedor de recursos virtualizados com suporte a adaptação de *clusters* virtuais. Com a essa modelagem é possível realizar simulações de alocações e liberação de *clusters* de MVs de acordo com algoritmos propostos e os estudados na literatura (MARSHALL; KEAHEY; FREEMAN, 2010). Portanto, os objetivos específicos previstos nesta dissertação foram totalmente atingidos.

A proposta foi validada e avaliada através de simulações com redes de Petri. O modelo foi inicialmente validado com a reprodução da política *bursts* e dos experimentos encontrados em (MARSHALL; KEAHEY; FREEMAN, 2010), encontrando-se os mesmos resultados apresentados na referida publicação. A partir do modelo da política *bursts*, foram introduzidas as modificações propostas para o ElasticCluster, utilizando-se restrições de tempo para representar a chegada e o tempo de execução dos BoTs. Assim, foi possível a comparação entre as políticas, mostrando as vantagens do reuso de recursos ociosos em *clusters* ativos.

Com a reutilização da ociosidade, o ElasticCluster apresentou uma economia de até 40% na quantidade de MVs iniciadas se comparada com a política de (MARSHALL; KEAHEY; FREEMAN, 2010). Além disso, foi analisado o número de MVs reusadas versus a variação de taxa de utilização do *cluster*. Foi possível constatar que o reuso é maior para altos valores de $MV_{máx}$, mas que para $U_{máx}$ o reuso não sofre grandes alterações para taxas acima de 65%. Com isso, concluí-se que é possível obter bons níveis de reutilização de recursos em *clusters* virtuais com carga de trabalho concorrente ao se manter os *clusters* com uma taxa de utilização em torno de 65%, gerando, portanto, economia de energia.

## 6.2 Produção Científica

O trabalho descrito nesta dissertação foi publicado e apresentado no *Workshop em Clouds e Aplicações* de 2012. A seguir, a descrição publicação.

BRAGA, A. R.; SILVA, P. R. X.; SOARES J. M.; GOMES, D. G. Ócio produtivo: Explorando a ociosidade de *clusters* virtuais para execução de aplicações do tipo saco de tarefas. In: SBRC 2012 - WCGA (X Workshop em Clouds e Aplicações). Ouro Preto, Minas Gerais, Brasil, 2012. p. 29–41.

## 6.3 Trabalhos Futuros

No decorrer desta dissertação, foram identificadas algumas questões que ainda necessitam ser aprofundadas e solucionadas. A seguir são listadas algumas limitações do trabalho atual e trabalhos futuros decorrentes desta pesquisa:

1. Executar os experimentos em ambiente real de computação em nuvem;
2. Visto que a política proposta possui uso limitado para *workloads* com padrão de submissão do tipo rajadas (*bursts*), pretende-se realizar a investigação de outros padrões de submissão de *jobs*, como por exemplo: o por demanda e o de fluxo contínuo. Isto pode levar ao desenvolvimento se novas políticas;
3. Antes de iniciar os testes em ambiente real, é desejável realizar testes da política em simuladores de propósito específico, por exemplo, o Haizea (SOTOMAYOR; KEAHEY; FOSTER, 2008), o CloudSim (CALHEIROS et al., 2011) ou o CloudReport (Sá; SOARES; GOMES, 2011);
4. A fim de quantificar o ganho energético que é possível obter com a proposta, planeja-se, durante a execução em ambiente real, medir valores consumidos de energia elétrica e comparar com outras abordagens que tratem do problema da alocação de recursos virtualizados com a economia de energia;
5. Estender a política proposta para utilização do SLA definido entre usuário e o provedor de IaaS;
6. Somados à utilização dos recursos, pretende-se realizar tratamento de outros requisitos computacionais específicos, tais como, utilizações de: memória, CPU, rede, I/O de disco;
7. Realizar outros experimentos com diferentes cargas de trabalho a fim de determinar quão generalizada está a solução em relação a cargas de trabalho;
8. Atenta-se, ainda, para a possibilidade de desenvolvimento de um mecanismo de balanceamento de carga que, em conjunto com a política de adaptação de *clusters*, possa minimizar o efeito da concorrência entre usuários;
9. Por fim, um modelo de QoS também pode ser aplicado, estabelecendo, para tanto, perfis de usuário. Dependendo desse perfil, o sistema reservaria recursos computacionais de forma a possibilitar utilizá-los de acordo com a data/hora reservada e a capacidade especificada, assim como no Haizea (SOTOMAYOR; KEAHEY; FOSTER, 2008).

# REFERÊNCIAS

AMORIM, C. L. de; WHATELY, L. L. A.; DUTRA, D. L. *Virtualização: modelos, técnicas e exemplos de uso na construção de serviços web*. [S.l.], 2009. Disponível em: <http://www.cos.ufrj.br/uploadfiles/1264798740.pdf>.

ANEDDA, P. et al. Suspending, migrating and resuming hpc virtual clusters. *Future Gener. Comput. Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 26, n. 8, p. 1063–1072, out. 2010. ISSN 0167-739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2010.05.007>.

ARMBRUST, M. et al. *Above the Clouds: A Berkeley View of Cloud Computing*. [S.l.], 2009.

BARKER, S. K.; SHENOY, P. Empirical evaluation of latency-sensitive application performance in the cloud. In: *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*. New York, NY, USA: ACM, 2010. (MMSys '10), p. 35–46. ISBN 978-1-60558-914-5. Disponível em: <http://doi.acm.org/10.1145/1730836.1730842>.

BOBROFF, N.; KOCHUT, A.; BEATY, K. Dynamic Placement of Virtual Machines for Managing SLA Violations. *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, IEEE, p. 119–128, maio 2007. Disponível em: <http://dx.doi.org/10.1109/INM.2007.374776>.

BRAGA, A. et al. Ócio produtivo: Explorando a ociosidade de clusters virtuais para execução de aplicações do tipo saco de tarefas. In: *SBRC 2012 - WCGA (X Workshop em Clouds e Aplicações)*. Ouro Preto, Minas Gerais, Brasil: [s.n.], 2012. p. 29–41.

BUYYA, R. et al. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 25, n. 6, p. 599–616, jun 2009. ISSN 0167-739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2008.12.001>.

CALHEIROS, R. N. et al. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, John Wiley & Sons, Inc., New York, NY, USA, v. 41, n. 1, p. 23–50, jan. 2011. ISSN 0038-0644. Disponível em: <http://dx.doi.org/10.1002/spe.995>.

CHI, R.; QIAN, Z.; LU, S. A heuristic approach for scalability of multi-tiers web application in clouds. In: *Proceedings of the 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. Washington, DC, USA: IEEE Computer Society, 2011. (IMIS'11), p. 28–35. ISBN 978-0-7695-4372-7. Disponível em: <http://dx.doi.org/10.1109/IMIS.2011.80>.

CHIRIGATI, F. S. *Computação nas Nuvens*. [S.l.], 2009. Disponível em: <http://www.gta.ufrj.br/ensino/eel879/trabalhos_v1_2009_2/seabra/index.html>.

COSTA, R. et al. Sobre a amplitude da elasticidade dos provedores atuais de computação na nuvem. In: *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Campo Grande, Mato Grasso do Sul, Brasil: [s.n.], 2011. p. 221–234. Disponível em: <http://sbrc2011.facom.ufms.br/files/main/ST05_2.pdf>.

CZAJKOWSKI, G. et al. Resource management for clusters of virtual machines. In: *Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2005. (CCGRID '05), p. 382–389. ISBN 0-7803-9074-1. Disponível em: <http://dl.acm.org/citation.cfm?id=1169222.1169492>.

EVANGELINOS, C.; HILL, C. N. Cloud Computing for parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2. In: CLOUD COMPUTING AND ITS APPLICATIONS. 2008. Disponível em: <http://www.cca08.org/speakers/evangelinos.php>.

FOSTER, I. et al. Virtual clusters for grid communities. In: *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2006. (CCGRID '06), p. 513–520. ISBN 0-7695-2585-7. Disponível em: <http://dx.doi.org/10.1109/CCGRID.2006.108>.

FRAGA, E.; BRASILEIRO, F.; GUERRERO, D. S. Estimando o valor de uma grade entre pares para a execução de aplicações do tipo saco de tarefas. In: SOCIEDADE BRASILEIRA DE COMPUTAçãO. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. Anais do Workshop em Clouds, Grids e Aplicações*. Campo Grande, Mato Grosso do Sul, Brasil, 2011. p. 1–14.

GAVRILOVSKA, A. et al. High-performance hypervisor architectures: Virtualization in hpc systems. In: *HPCVirt 2007: 1st Workshop on System-level Virtualization for High Performance Computing*. [S.l.: s.n.], 2009.

HUU, T. T. et al. Joint elastic cloud and virtual network framework for application performance-cost optimization. *J. Grid Comput.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 9, n. 1, p. 27–47, mar. 2011. ISSN 1570-7873. Disponível em: <http://dx.doi.org/10.1007/s10723-010-9168-6>.

IOSUP, A.; EPEMA, D. Grid computing workloads. *Internet Computing, IEEE*, v. 15, n. 2, p. 19–26, march-april 2011. Disponível em: <http://dx.doi.org/10.1109/MIC.2010.130>.

IOSUP, A. et al. Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Trans. Parallel Distrib. Syst.*, IEEE Press, Piscataway, NJ, USA, v. 22, n. 6, p. 931–945, jun. 2011. ISSN 1045-9219. Disponível em: <http://dx.doi.org/10.1109/TPDS.2011.66>.

JENSEN, K.; KRISTENSEN, L. M. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. 1. ed. Springer, 2009. Hardcover. ISBN 9783642002830. Disponível em: <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/3642002838>.

JING S., S. K. A novel model for load balancing in cloud data center. *Journal of Convergence Information Technology*, v. 6, n. 4, p. 171–179, 2011. Cited By (since 1996) 3. Disponível em: <http://dx.doi.org/10.4156/jcit.vol6.issue4.20>.

JOGALEKAR, P.; WOODSIDE, M. Evaluating the scalability of distributed systems. *IEEE Trans. Parallel Distrib. Syst.*, IEEE Press, Piscataway, NJ, USA, v. 11, n. 6, p. 589–603, jun. 2000. ISSN 1045-9219. Disponível em: <http://dx.doi.org/10.1109/71.862209>.

KHATUA, S.; GHOSH, A.; MUKHERJEE, N. Optimizing the utilization of virtual resources in cloud environment. In: *Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2010 IEEE International Conference on*. [s.n.], 2010. p. 82 –87. ISSN 1944-9429. Disponível em: <http://dx.doi.org/10.1109/VECIMS.2010.5609349>.

KUPERBERG, M. et al. *Defining and Quantifying Elasticity of Resources in Cloud Computing and Scalable Platforms*. [S.l.], 2011. Disponível em: <http://digbib.ubka.uni-karlsruhe.de/volltexte/documents/1978075>.

LAUREANO, M. A. P. Máquinas virtuais e emuladoras: conceitos, técnica e aplicações. In: . São Paulo: Editora: Novatec, 2006. ISBN 9780137142972.

LIAO, X.; HU, L.; JIN, H. Energy optimization schemes in cluster with virtual machines. *Cluster Computing*, Kluwer Academic Publishers, Hingham, MA, USA, v. 13, n. 2, p. 113–126, jun. 2010. ISSN 1386-7857. Disponível em: <http://dx.doi.org/10.1007/s10586-009-0110-2>.

LIAO, X.; JIN, H.; LIU, H. Towards a green cluster through dynamic remapping of virtual machines. *Future Gener. Comput. Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 28, n. 2, p. 469–477, feb 2012. ISSN 0167-739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2011.04.013>.

LIM, H. C.; BABU, S.; CHASE, J. S. Automated control for elastic storage. In: *Proceeding of the 7th international conference on Autonomic computing*. New York, NY, USA: ACM, 2010. (ICAC'10), p. 1–10. ISBN 978-1-4503-0074-2. Disponível em: <http://doi.acm.org/10.1145/1809049.1809051>.

MAGALHãES, D. M.; SOARES, J. M.; GOMES, D. G. Análise do impacto de migração de máquinas virtuais em ambiente computacional virtualizado. In: *FACOM-UFMS. Proceedings of XXIX Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2011)*. [s.n.], 2011. p. 235–248. Disponível em: <http://sbrc2011.facom.ufms.br/files/main/ST05_3.pdf>.

MANCINI, E. P.; RAK, M.; VILLANO, U. Perfcloud: Grid services for performance-oriented development of cloud computing applications. In: *Proceedings of the 2009 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*. Washington, DC, USA: IEEE Computer Society, 2009. (WETICE '09), p. 201–206. ISBN 978-0-7695-3683-5. Disponível em: <http://dx.doi.org/10.1109/WETICE.2009.47>.

MANETTI, V. et al. Dynamic virtual cluster reconfiguration for efficient iaas provisioning. In: *Proceedings of the 2009 international conference on Parallel processing*. Berlin,

Heidelberg: Springer-Verlag, 2010. (Euro-Par'09), p. 424–433. ISBN 3-642-14121-8, 978-3-642-14121-8. Disponível em: <http://dl.acm.org/citation.cfm?id=1884795.1884844>.

MARSHALL, P.; KEAHEY, K.; FREEMAN, T. Elastic site: Using clouds to elastically extend site resources. In: *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2010. (CCGRID '10), p. 43–52. ISBN 978-0-7695-4039-9. Disponível em: <http://dx.doi.org/10.1109/CCGRID.2010.80>.

MARSHALL, P.; KEAHEY, K.; FREEMAN, T. Improving utilization of infrastructure clouds. In: *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*. [s.n.], 2011. p. 205 –214. Disponível em: <http://dx.doi.org/10.1109/CCGrid.2011.56>.

MELL, P.; GRANCE, T. *The NIST Definition of Cloud Computing*. 26. ed. [S.l.], jul. 2009. Disponível em: <http://www.csrc.nist.gov/groups/SNS/cloud-computing/>.

MELLO, T. C. de et al. Uma análise de recursos virtualizados em ambiente de hpc. In: *Anais do VIII Workshop em Clouds, Grids e Aplicações, em conjunto com o XXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Gramado - RS: [s.n.], 2010. p. 17–30.

MERGEN, M. F. et al. Virtualization for high-performance computing. *SIGOPS Oper. Syst. Rev.*, ACM, New York, NY, USA, v. 40, n. 2, p. 8–11, apr 2006. ISSN 0163-5980. Disponível em: <http://doi.acm.org/10.1145/1131322.1131328>.

MONTERO, R. S.; MORENO-VOZMEDIANO, R.; LLORENTE, I. M. An elasticity model for high throughput computing clusters. *J. Parallel Distrib. Comput.*, Academic Press, Inc., Orlando, FL, USA, v. 71, n. 6, p. 750–757, jun. 2011. ISSN 0743-7315. Disponível em: <http://dx.doi.org/10.1016/j.jpdc.2010.05.005>.

MUKHERJEE, T. et al. Measurement-based power profiling of data center equipment. In: *Proceedings of the 2007 IEEE International Conference on Cluster Computing*. Washington, DC, USA: IEEE Computer Society, 2007. (CLUSTER '07), p. 476–477. ISBN 978-1-4244-1387-4. Disponível em: <http://dx.doi.org/10.1109/CLUSTR.2007.4629270>.

MURATA, T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, v. 77, n. 4, p. 541–580, abr. 1989. ISSN 00189219. Disponível em: <http://dx.doi.org/10.1109/5.24143>.

NAZIR, A.; LIU, H.; SORENSEN, S.-A. A cost efficient framework for managing distributed resources in a cluster environment. In: *High Performance Computing and Communications, 2009. HPCC '09. 11th IEEE International Conference on*. [s.n.], 2009. p. 29 –35. Disponível em: <http://dx.doi.org/10.1109/HPCC.2009.16>.

NETTO, M. A. S.; BUYYA, R. Coordinated Rescheduling of Bag-of-Tasks for Executions on Multiple Resource Providers. *Concurrency and Computation: Practice and Experience*, Wiley, 2011. Disponível em: <http://www.buyya.com/papers/CoordSched-Grid5000-CCPE.pdf>.

ONG, H. et al. Vccp: A transparent, coordinated checkpointing system for virtualization-based cluster computing. In: *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*. [s.n.], 2009. p. 1 –10. ISSN 1552-5244. Disponível em: <http://dx.doi.org/10.1109/CLUSTR.2009.5289183>.

OPRESCU, A.-M.; KIELMANN, T. Bag-of-tasks scheduling under budget constraints. In: *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*. Washington, DC, USA: IEEE Computer Society, 2010. (CLOUDCOM '10), p. 351–359. ISBN 978-0-7695-4302-4. Disponível em: <http://dx.doi.org/10.1109/CloudCom.2010.32>.

OPRESCU, A.-M.; KIELMANN, T.; LEAHU, H. Budget estimation and control for bag-of-tasks scheduling in clouds. *Parallel Processing Letters*, v. 21, n. 2, p. 219–243, 2011.

PAREKH, S. et al. Using control theory to achieve service level objectives in performance management. *Real-Time Syst.*, Kluwer Academic Publishers, Norwell, MA, USA, v. 23, n. 1/2, p. 127–141, jul. 2002. ISSN 0922-6443. Disponível em: <http://dx.doi.org/10.1023/A:1015350520175>.

PENHA, D. O. da; FREITAS, H. C. de; MARTINS, C. A. P. da S. Modelagem de sistemas computacionais usando redes de petri: aplicação em projeto, análise e avaliação. In: *Anais da IV Escola Regional de Informática RJ/ES*. [s.n.], 2004. p. 40 p. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/erirjes/2004/001.pdf>.

PRUEKSAAROON, S.; VARAVITHYA, V. A framework for management of virtualization clusters. *International Journal of Advancements in Computing Technology*, v. 4, n. 4, p. 182–191, 2012. Disponível em: <http://www.aicit.org/IJACT/ppl/IJACTVOL4NO4MAIN_part23.pdf>.

REGO, P. et al. Faircpu: Architecture for allocation of virtual machines using processing features. In: *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*. [s.n.], 2011. p. 371 –376. Disponível em: <http://dx.doi.org/10.1109/UCC.2011.62>.

RIEDEL, M. et al. Computational steering and online visualization of scientific applications on large-scale hpc systems within e-science infrastructures. In: *Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2007. (E-SCIENCE '07), p. 483–490. ISBN 0-7695-3064-8. Disponível em: <http://dx.doi.org/10.1109/E-SCIENCE.2007.21>.

ROSENBLUM, M. The reincarnation of virtual machines. *Queue*, ACM, New York, NY, USA, v. 2, n. 5, p. 34–40, jul. 2004. ISSN 1542-7730. Disponível em: <http://doi.acm.org/10.1145/1016998.1017000>.

Sá, T.; SOARES, J. M.; GOMES, D. G. Cloudreports: uma ferramenta gráfica para a simulação de ambientes computacionais em nuvem baseada no framework cloudsim. In: *IX Workshop em Clouds, Grids e Aplicaçies (WCGA 2011), SBRC 2011*. [S.l.: s.n.], 2011. p. 103–116.

SEVIOR, M.; FIFIELD, T.; KATAYAMA, N. Belle monte-carlo production on the amazon ec2 cloud. *Journal of Physics: Conference Series*, v. 219, n. 1, 2010. Cited By (since 1996): 2. Disponível em: <http://dx.doi.org/10.1088/1742-6596/219/1/012003>.

SONNEK, J.; CHANDRA, A. Virtual Putty: Reshaping the Physical Footprint of Virtual Machines. In: *Proc. of Workshop on Hot Topics in Cloud Computing (HotCloud'09)*. [S.l.: s.n.], 2009.

SOTOMAYOR, B. *Provisioning computational resources using virtual machines and leases*. Tese (Doutorado) — The University of Chicago, Illinois, USA, August 2010.

SOTOMAYOR, B.; KEAHEY, K.; FOSTER, I. Combining batch execution and leasing using virtual machines. In: *Proceedings of the 17th international symposium on High performance distributed computing*. New York, NY, USA: ACM, 2008. (HPDC '08), p. 87–96. ISBN 978-1-59593-997-5. Disponível em: <http://doi.acm.org/10.1145/1383422.1383434>.

SOUSA, F. R. C. et al. Gerenciamento de dados em nuvem: Conceitos, sistemas e desafios. In: *Tópicos em Sistemas Colaborativos, Interativos, Multimídia, Web e Bancos de Dados, SWIB 2010, 1. ed. SBC, Belo Horizonte*. [S.l.: s.n.], 2010. v. 1, p. 101–130.

STANTCHEV, V. Performance evaluation of cloud computing offerings. In: *Proceedings of the 2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences*. Washington, DC, USA: IEEE Computer Society, 2009. (ADVCOMP '09), p. 187–192. ISBN 978-0-7695-3829-7. Disponível em: <http://dx.doi.org/10.1109/ADVCOMP.2009.36>.

SULEIMAN, B. et al. On understanding the economics and elasticity challenges of deploying business applications on public cloud infrastructure. *J. Internet Services and Applications*, v. 3, n. 2, p. 173–193, 2012. Disponível em: <http://dx.doi.org/10.1007/s13174-011-0050-y>.

VECCHIOLA, C.; PANDEY, S.; BUYYA, R. High-performance cloud computing: A view of scientific applications. In: *Proceedings of the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks*. Washington, DC, USA: IEEE Computer Society, 2009. (ISPAN '09), p. 4–16. ISBN 978-0-7695-3908-9. Disponível em: <http://dx.doi.org/10.1109/I-SPAN.2009.150>.

WERSTEIN, P.; SITU, H.; HUANG, Z. Load balancing in a cluster computer. In: *Parallel and Distributed Computing, Applications and Technologies, 2006. PDCAT '06. Seventh International Conference on*. [s.n.], 2006. p. 569 –577. Disponível em: <http://dx.doi.org/10.1109/PDCAT.2006.77>.

XU, C.; BAI, Y.; LUO, C. Performance evaluation of parallel programming in virtual machine environment. In: *Network and Parallel Computing, 2009. NPC '09. Sixth IFIP International Conference on*. [s.n.], 2009. p. 140 –147. Disponível em: <http://dx.doi.org/10.1109/NPC.2009.22>.

XU, C. et al. Pvc: A novel personal virtual cluster based on multi-core platform. In: *Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*. [s.n.], 2009. p. 363 –368. Disponível em: <http://dx.doi.org/10.1109/ISCIS.2009.5291807>.

YAMASAKI, S.; MARUYAMA, N.; MATSUOKA, S. Model-based resource selection for efficient virtual cluster deployment. In: *Proceedings of the 2nd international workshop on Virtualization technology in distributed computing*. New York, NY, USA: ACM, 2007. (VTDC '07), p. 6:1–6:7. ISBN 978-1-59593-897-8. Disponível em: <http://doi.acm.org/10.1145/1408654.1408660>.

YAN, B. et al. An efficient and stable cluster system based on improved load balancing algorithm. In: *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*. [s.n.], 2010. v. 2, p. 360 –363. Disponível em: <http://dx.doi.org/10.1109/ICCSIT.2010.5564731>.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *J. Internet Services and Applications*, v. 1, n. 1, p. 7–18, 2010. Disponível em: <http://dx.doi.org/10.1007/s13174-010-0007-6>.

# APÊNDICE A – REDES DE PETRI

As Redes de Petri foram criadas a partir tese de doutorado de Carl Adam Petri, intitulada *Kommunication mit Automaten* (Comunicação com Autômatos), apresentada à Universidade de Bonn em 1962. Seu objetivo era desenvolver um modelo em que as máquinas de estado fossem capazes de se comunicar. A característica principal alcançada por este modelo foi a possibilidade de representar a concorrência. Este modelo recebeu o nome de Redes de Petri em homenagem ao próprio autor e hoje podemos encontrar variações do modelo original sendo usados em diversas áreas da ciência para ajudar no estudo do comportamento e desempenho de diferentes sistemas. Podemos encontrar as Redes de Petri (RdP) (MURATA, 1989) sendo utilizadas em áreas como economia, biologia, engenharia, computação, entre outras.

As variações de RdP tais como as Redes de Petri de alto Nível (Coloridas, Temporizadas, Numéricas ou Estocásticas) são modelos importantes e que não fazem parte da teoria original. Tais redes acrescentaram uma grande força descritiva ao processo de modelagem, através do uso de fichas com identidade e, consequentemente, do uso de conjuntos de fichas. No entanto, são representações que ganharam relevância por serem capazes de modelar sistemas de forma mais simplificada ou intuitiva ou simplesmente porque ainda não havia um modelo capaz de representar sistemas que envolviam temporizações ou comportamentos probabilísticos (estocásticos).

Antes de apresentar os conceitos e elementos que compõem a RdP é importante responder uma pergunta: Por que precisamos modelar um sistema computacional? [Jain 1991].

• Torna-se mais fácil o entendimento do funcionamento ou comportamento de um sistema computacional existente?

• O projeto do sistema computacional pode ser verificado e validado através de um modelo antes da construção de um protótipo?

• A avaliação do funcionamento, comportamento e por consequência do seu desempenho em relação a um histórico de projeto ou em relação a outros modelos de sistema existentes, também em fase de projeto é facilitada?

• É possível definir métricas de desempenho através do modelo?

São várias as respostas para justificar a modelagem de um sistema computacional. É possível afirmar que o uso de um modelo facilita e justifica o prosseguimento de novas

etapas de projeto e análise antes da etapa final de avaliação. Entre estas etapas podemos citar uma diretamente dependente da existência física do sistema e que muitas vezes só deve ser realizada após uma fase anterior de validação, antes do protótipo ser construído:

• Medição: A medição depende da existência física do sistema ou protótipo do sistema. Você implementaria seu sistema antes de verificá-lo através de um modelo? É como programar sem projetar o algoritmo!

Imagine que o sistema a ser medido (uma coleta de dados para avaliação de resultados) é *cluster* de computadores (BUYYA et al., 2009). Qual o custo financeiro para montar este *cluster*? Este *cluster* foi modelado anteriormente? Existe uma verificação/validação anterior à montagem do protótipo (*cluster* de servidores)?

Existem alguns modelos analíticos e de simulação [Jain 1991] para validação de um projeto. As RdP fazem parte destes dois modelos. Podemos representar uma RdP através de descrições matemáticas e gráficas. Podemos também simular uma Rede de Petri através de ferramentas específicas de simulação como o CPN Tools [1]. Existe um repositório de informações e simuladores de Redes de Petri que pode ser encontrado no site *Petri Nets World* [2].

## A.1 Composição de uma Rede de Petri

Uma RdP é um grafo bipartido cujos nós são divididos em duas categorias: lugares e transições. Lugares são utilizados para especificar um estado ou condição de um componente do sistema. As transições representam eventos ou operações. Para interligar os componentes de uma RdP, existem dois tipos de arcos direcionados, que podem ser simples ou múltiplos: Arcos de entrada que ligam uma transição a um lugar, e os arcos de saída que ligam um lugar a uma transição. A figura 23 mostra os elementos básicos da estrutura de uma RdP.

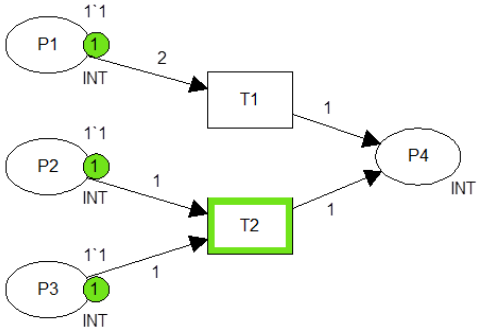


Figura 23: Notação gráfica de uma Rede de Petri.

A dinâmica da rede se apoia no uso de fichas que são armazenadas nos lugares. As fichas podem representar recursos ou condições, descrevendo o estado da rede em um

[1] www.cpntools.org
[2] www.informatik.uni-hamburg.de/TGI/PetriNets/

instante no tempo. As fichas são inseridas ou retiradas dos lugares através do disparo de uma transição. Para uma transição ser disparada, é necessário que haja uma quantidade de fichas nos lugares de entrada igual ou superior aos pesos dos seus respectivos arcos de entrada. Após o disparo, fichas são inseridas nos lugares de saída de acordo com o peso dos arcos de saída. E através dos disparos das transições que se pode visualizar o fluxo de controle e de informação no sistema.

Na RdP dada como exemplo na Figura A.1, apenas a transição t2 está habilitada; t1 não está habilitada porque seriam necessárias duas fichas em P1 para dispará-la, já que o peso do arco que liga P1 a t1 é igual a 2. Quando t2 for disparada, os fichas em P2 e P3 são retiradas e colocadas em P4. Notemos que o número de fichas não é necessariamente conservado.

## A.2 Redes de Petri Coloridas

As Redes de Petri Coloridas (RPC) (JENSEN; KRISTENSEN, 2009) são uma extensão das RP em que as fichas podem representar tipos e estruturas de dados mais complexos e possuem uma linguagem de programação associada. O principal objetivo das RPC é a redução do tamanho do modelo, permitindo que fichas individualizadas (coloridas) representem diferentes processos ou recursos em uma mesma sub-rede. Nas RPCs as fichas são representadas por estruturas de dados complexas. Deste modo, as fichas podem conter informações. Além disso, cada lugar armazena fichas de um tipo (ou cor) definido e arcos realizam operações sobre elas. As transições determinam a dinâmica da RPC e podem apresentar "expressões de guarda". Estas, por sua vez, indicam os tipos de fichas que possibilitam o disparo de uma transição.

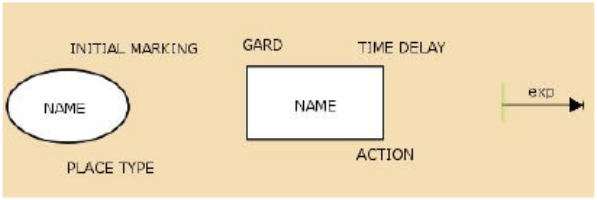


Figura 24: Notação gráfica de uma rede de Petri Colorida.

Uma RPC é composta por três partes: estrutura, inscrições e declarações. A estrutura é semelhante à de uma RP básica, ou seja, composta de lugares e transições, com arcos interconectando nós de tipos diferentes. As inscrições são associadas aos lugares, transições e arcos.

• Cada lugar tem as seguintes inscrições: Nome (para identificação); ColorSet (especificando os tipos de fichas que podem residir no lugar); Marcação inicial (multiconjunto de fichas coloridos).

• Cada transição tem as seguintes inscrições: Nome (para identificação); Guarda (expressão booleana contendo algumas das variáveis).

• Cada arco tem as seguintes inscrições: Expressão do arco (contendo algumas das variáveis).

As declarações são tipos, funções, operações e variáveis. Quando a expressão do arco é avaliada, ela gera um multi-conjunto de fichas coloridas. Expressões podem conter constantes, variáveis, funções e operações definidas nas declarações.

A figura 24 apresenta os principais atributos dos elementos de uma rede de petri colorida no CPN Tools.

# ANEXO A – DESCRIÇÃO DAS CORES DE FICHAS UTILIZADAS NA MODELAGEM

| Nome da Cor: | Usuario |
|---|---|
| Descrição: | Modela usuários do sistema. |
| Declaração: | colset Usuario = with A \| B; |

| Nome da Cor: | idJob |
|---|---|
| Descrição: | Modela identificadores (IDs) dos jobs |
| Declaração: | colset idJob = int with 1..1000; |

| Nome da Cor: | wallTimeJob |
|---|---|
| Descrição: | Representa *walltime* dos *jobs* |
| Declaração: | colset wallTimeJob = int with 1..100; |

| Nome da Cor: | idsJobs |
|---|---|
| Descrição: | Cria uma lista de IDs de jobs |
| Declaração: | colset idsJobs = list idJob; |

| Nome da Cor: | Job |
|---|---|
| Descrição: | Modela os *jobs* da rede |
| Declaração: | colset Job = record idJob: INT * usuario: Usuario * wallTime: INT * loadCPU: INT timed; |

| Nome da Cor: | Jobs |
|---|---|
| Descrição: | Modela uma lista de *jobs* na modelagem |
| Declaração: | colset Jobs = list Job timed; |

| Nome da Cor: | idWorkload |
|---|---|
| Descrição: | Modela IDs dos *workloads* |
| Declaração: | colset idWorkload = int with 1..1000; |

| Nome da Cor: | idsWorkloads |
|---|---|
| Descrição: | Modela uma lista de IDs dos *workloads* |
| Declaração: | colset idsWorkloads = list idWorkload; |

| Nome da Cor: | TypeWorkload |
|---|---|
| Descrição: | Contém chaves que identificam o tipo do padrão de submissão. A política proposta atua apenas para o tipo BURSTS |
| Declaração: | colset TypeWorkload = with NOTHING \| ONDEMAND \| STEADYSTREAM \| BURSTS; |

| Nome da Cor: | TypeProposal |
|---|---|
| Descrição: | Contém chaves que identificam as propostas modeladas na rede |
| Declaração: | colset TypeProposal = with NULL \| ELASTIC_SITE \| ELASTIC_CLUSTER; |

| Nome da Cor: | WorkloadLoad |
|---|---|
| Descrição: | Modela os valores da utilização que os *workloads* podem estabelecer no *cluster* |
| Declaração: | colset WorkloadLoad = int with 5..95; |

| Nome da Cor: | Workload |
|---|---|
| Descrição: | Modela uma carga de trabalho (*workload*) |
| Declaração: | colset Workload = record idWorkload: INT * typeProposal: TypeProposal * typeWorkload: TypeWorkload* idsJobs: idsJobs * workloadLoad: STRING * wallTimeTotal: INT timed; |

| Nome da Cor: | Workloads |
|---|---|
| Descrição: | Modela uma lista de *workloads* |
| Declaração: | colset Workloads = list Workload; |

| Nome da Cor: | Queue |
|---|---|
| Descrição: | Contém uma fila de *workloads* |
| Declaração: | colset Queue = list Workload; |

| Nome da Cor: | Hardware |
|---|---|
| Descrição: | Modela o *hardware* que compõem uma MV |
| Declaração: | colset Hardware = record processor : INT; |

| Nome da Cor: | idMachine |
|---|---|
| Descrição: | Modela IDs das MVs |
| Declaração: | colset idMachine = int with 1..1000; |

| Nome da Cor: | idsMachines |
|---|---|
| Descrição: | Modela uma lista IDs de MVs |
| Declaração: | colset idsMachines = list idJob; |

| Nome da Cor: | Machine |
|---|---|
| Descrição: | Modela uma máquina virtual |
| Declaração: | colset Machine = record idMachine: INT * hw : Hardware * idsWorkloads: idsWorkloads * timeMachine: INT; |

| Nome da Cor: | Machines |
|---|---|
| Descrição: | Modela uma lista de MVs |
| Declaração: | colset Machines = list Machine; |

| Nome da Cor: | MachinesList |
|---|---|
| Descrição: | Modela uma lista de uma lista de MVs |
| Declaração: | colset MachinesList = list Machines; |

| Nome da Cor: | NumbersOfMachines |
|---|---|
| Descrição: | Contém os números de MVs disponíveis, utilizadas e reusadas |
| Declaração: | colset NumbersOfMachines = product INT * INT * INT; |

| Nome da Cor: | AllocatedMachines |
|---|---|
| Descrição: | Possui uma lista de MVs que foram alocadas para um BoT que está sendo escalonado |
| Declaração: | colset AllocatedMachines = product Machines * Machines * NumbersOfMachines; |

| Nome da Cor: | MachinesAlocated_ES |
|---|---|
| Descrição: | Possui uma lista de MVs que foram alocadas de acordo com a política "ElasticSite"para um BoT que está sendo escalonado |
| Declaração: | colset MachinesAlocated_ES = Machines; |

| Nome da Cor: | AdaptableCluster |
|---|---|
| Descrição: | Contém o *cluster* adaptável |
| Declaração: | colset AdaptableCluster = record clusterID: INT * clusterNodes: INT* clusterLoad: STRING; |

| Nome da Cor: | WorkloadxClusterStatus |
|---|---|
| Descrição: | Possui o BoT que será executado e o *cluster* adaptável |
| Declaração: | colset WorkloadxClusterStatus = product Workload * AdaptableCluster; |

| Nome da Cor: | Cluster |
|---|---|
| Descrição: | Modela um a*cluster* |
| Declaração: | colset Cluster = record clusterID: INT * clusterNodes: Machines* clusterLoad: STRING * walltime: INT timed; |

| Nome da Cor: | Clusters |
|---|---|
| Descrição: | Modela uma lista de *clusters* |
| Declaração: | colset Clusters = list Cluster; |

| Nome da Cor: | ClusterID |
|---|---|
| Descrição: | Modela IDs dos *clusters* |
| Declaração: | colset ClusterID = int with 1..10; |

| Nome da Cor: | NotaCluster |
|---|---|
| Descrição: | Modela a nota de um *cluster* |
| Declaração: | colset NotaCluster = product INT*STRING; |

| Nome da Cor: | NotasClusters |
|---|---|
| Descrição: | Modela uma lista de notas de *clusters* |
| Declaração: | colset NotasClusters = list NotaCluster; |

| Nome da Cor: | RetornoOrdenacao |
|---|---|
| Descrição: | Utilizada como uma cor auxiliar durante o processo de desligamento de MVs. Contém uma lista de *clusters* e suas respectivas notas |
| Declaração: | colset RetornoOrdenacao = product NotasClusters * Clusters; |