UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Diogo Tavares da Silva

Abordagem icônica para modelagem e simulação de ambientes de computação em nuvem

São José do Rio Preto
2015

Diogo Tavares da Silva

Abordagem icônica para modelagem e simulação de ambientes de computação em nuvem

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Campus de São José do Rio Preto.

Orientador: Prof. Dr. Aleardo Manacero Jr.

São José do Rio Preto
2015

Diogo Tavares da Silva

Abordagem icônica para modelagem e simulação de ambientes de computação em nuvem

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Comissão Examinadora

Profª. Drª. Aleardo Manacero Jr.
UNESP – São José do Rio Preto
Orientador

Prof. Dr. Norian Marranghelo
UNESP – São José do Rio Preto

Prof. Dr. Ricardo Menotti
UFSCar – São Carlos

São José do Rio Preto
17 de agosto de 2015

Aos meus amados pais Pedro e Benedita
Ao meu irmão Daniel
À Tatiana.

Agradecimentos

Primeiramente agradeço a Deus, à intercessão de N.S. Aparecida e a todos que direta e indiretamente colaboraram para a escalada de mais este degrau na busca pelos meus objetivos.

Gostaria de agradecer aos meus amados pais Pedro e Benedita por toda força, orações e apoio incondicional que têm me dado hoje e sempre. Reconheço todas as dificuldades que passaram pra que pudesse alcançar essa meta. Agradeço também ao meu irmão Daniel por ser sempre o alívio cômico dos meus dias, meus primos (irmãos de vida) José e Joelmir, pela parceria profissional. Agradeço ainda a todos os familiares que me apoiam, especialmente a minha amada "*nona*" (vó) Carmem, por quem tenho o maior carinho do mundo.

Gostaria de agradecer a *mi novia* Tatiana que tem sido sempre companheira e principalmente paciente, apoiando meu crescimento pessoal e profissional com todo seu amor e dedicação.

Agradecer ainda ao meu orientador Prof. Dr. Aleardo Manacero Jr. e minha coorientadora Prof. Dra. Renata Spolon Lobato, por toda a dedicação, orientação e amizade adquirida ao longo destes anos. Além dos meus companheiros de projeto Arthur Jorge e Denison Menezes, que se tornaram grandes amigos no decorrer deste trabalho. Agradeço ainda de maneira maior à UNESP e ao CNPq pela bolsa de mestrado institucional investida em minha pesquisa e também à FAPESP pelo investimento em equipamentos.

Não poderia de deixar fazer um agradecimento especial também para os amigos que me ajudaram nos momentos em que mais precisei de alguém que me estendesse a mão. Desta forma, agradeço ao Igor S. Butarello, Gabriel Covello e ao Lucas A. Cazarote pela valorosa amizade.

Por fim agradeço a todos os amigos que fiz ao longo do tempo em que estive no laboratório do Grupo de Sistemas Paralelos e Distribuídos (GSPD): Denison (Denilson), Igor (Igão), Lúcio (Comprometimento), Brunno (Navarro), Renan (Nanzinho), Victor (Pala), Fernanda (Fer), Fernando (Kaway), João (Lentra A), Arthur (Fatal), Saraiva (Muringa), Covello (Cokinha Na Padoca), Mário (Pai de família) e Lucas (Tesouro). Esses últimos cinco obrigado por serem meus "patos" no bilhar durante todo o mestrado. Pra todos vocês pago uma "cokinha" e uma partida de bilhar em qualquer boteco em que a gente se encontrar por essa vida.

“Porque carros e aviões se tens sonhos e pernas?”

-Vander Lee

RESUMO

O uso de computação em nuvem tem se tornado cada vez mais popular nos últimos anos, impulsionado por fatores como portabilidade de aplicações e a tendência de redução nos investimentos em estrutura física de TI. Neste contexto, a avaliação de desempenho de sistemas de computação em nuvem é útil tanto para clientes, que precisam encontrar a melhor configuração de recursos para sua aplicação, quanto para provedores de serviço, que precisam estudar políticas mais eficientes de escalonamento e alocação de recursos e máquinas virtuais. Apesar de mais preciso, o uso de *benchmarking* não é a alternativa mais indicada para avaliar esses sistemas, uma vez que é caro usar o sistema apenas para medições de desempenho. Isso faz da simulação uma alternativa mais interessante, pois possui menor custo de implementação e maior facilidade de reconfiguração de parâmetros e reprodutibilidade de experimentos. Infelizmente, os simuladores de computação em nuvem conhecidos possuem problemas relativos a usabilidade e capacidade de modelagem. Este trabalho apresenta o desenvolvimento de uma abordagem icônica para modelagem e simulação de computação em nuvem no iSPD. A escolha do iSPD como base do projeto se justifica pois ele é um simulador que busca prover facilidade de uso, oferecendo uma interface icônica para a modelagem de sistemas. Resultados obtidos com a simulação de computação em nuvem mostram o sucesso do projeto, pois o simulador consegue representar corretamente execuções em nuvens reais, com custos de modelagem e execução bastante baixos.

Palavras-chave: Sistemas de computação, Computação em nuvem, Avaliação de desempenho, Simulação de sistemas.

ABSTRACT

The use of Cloud Computing is becoming increasingly more popular, driven by application´s portability and the trend of reduction of investments in IT's physical infrastructure. These developments create scenarios where it is hard to know if the use of such environments is efficient or not. In this context, evaluate the performance of cloud computing systems is useful both for clients, that need to find the best resource configuration for their applications, as well as for providers, who need to evaluate which scheduling and resource and virtual machine allocation policies are more efficient. Although more accurate, the use of benchmarking is not an adequate option for this evaluation since it is expensive to use the system just for performance measuring. This makes simulation the most attractive option because it has a lower deployment cost and it is easier to reconfigure model parameters and reproduce the measurements in a system's model. Unfortunately, the known cloud computing simulators have issues related to their usability and modeling capability. This work presents the development of an iconic approach for modeling and simulation of cloud computing with the iSPD. The choice for iSPD as foundation for this project is justified because it is a simulator aimed to be user-friendly, offering an iconic interface to systems' modeling. Results achieved with cloud computing simulation show success, since the simulator was able to correctly mimic executions in a real cloud, with a reasonably low cost for modeling and execution.

Keywords**:** *Computing systems, Cloud computing, Performance evaluation, system simulation.*

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
DOM	*Document Object Model*
DTD	*Document Type Definition*
Flop	*FLoating-point OPerations*
Flops	*FLoating-point Operations Per Second*
IaaS	*Infrastructure-as-a-Service*
Mbps	*Megabits per second*
PaaS	*Platform-as-a-Service*
SaaS	*Software-as-a-Service*
TCP	*Transmission Control Protocol*
TI	Tecnologia da Informação
VM	*Virtual Machine*
XML	*eXtensible Markup Language*

Sumário

1 Introdução

A computação em nuvem (*Cloud Computing*, em inglês) tem se tornado um modelo de computação distribuída muito popular. O principal conceito desta tecnologia é disponibilizar recursos computacionais sob demanda, por meio da internet, utilizando um modelo utilitário em que se paga pelo tempo e quantidade de recursos utilizados.

Aponta-se como principais estímulos ao desenvolvimento da computação em nuvem os avanços de uma série de tecnologias (processadores multinúcleo, virtualização de *hardware*, sistemas autônomos, tecnologias de internet e computação em grade) (BUYYA; BROBERG; GOSCINSKI, 2011). Outro fator crucial para o desenvolvimento desta tecnologia é que a maioria dos *datacenters* em uso são projetados para suportar picos teóricos, permanecendo subutilizados na maior parte do tempo. Isto estimulou o desenvolvimento de tecnologias que utilizem esse potencial subutilizado para obter maior aproveitamento do sistema.

Vários autores consideram a popularização do uso de computação em nuvem como uma evolução natural do uso de sistemas de computação. Assim como, no século passado, as fábricas deixaram de possuir estações de geração de energia próprias para utilizar de maneira mais simples e barata a rede elétrica recém criada, atualmente as empresas têm deixado de manter recursos computacionais em sua infraestrutura para utilizar o modelo de computação em nuvem (RITTINGHOUSE; RANSOME, 2009).

Desta forma, a computação em nuvem apresenta-se como uma alternativa vantajosa tanto para clientes, que reduzem custos de TI utilizando serviços de provedores externos, quanto para os provedores de serviço, que desenvolvem uma infraestrutura de computação que permite servir vários perfis distintos de usuários, aumentando a utilização e a eficiência do sistema, o que resulta em melhores custos operacionais (BUYYA; BROBERG; GOSCINSKI, 2011).

1.1 Motivação

A determinação das reais vantagens no uso de computação em nuvem depende da avaliação das alternativas possíveis na execução de uma certa carga computacional, o que é obtido por meio da avaliação de desempenho. Avaliar o desempenho é importante em sistemas de computação em nuvem tanto para os clientes que necessitam avaliar e selecionar a melhor configuração de recursos necessárias para a execução de sua aplicação, quanto para provedores de serviço, que precisam avaliar o desempenho de políticas de escalonamento, alocação e provisionamento de máquinas virtuais (VMs) e recursos.

Em ambientes de nuvem a avaliação por meio de *benchmarking* não é a alternativa mais viável, devido principalmente aos custos de sua aplicação. Isso porque do ponto de vista

do usuário não é possível conhecer, *a priori*, a quantidade de recursos que serão utilizados pela aplicação. Isso implica em realizar uma grande quantidade de experimentos, variando-se as quantidades de recursos contratados para estimar a melhor configuração necessária. Isto representa um custo extra, já que o custo de sistemas de nuvem é dado por utilização. Do ponto de vista do provedor de serviço essa avaliação torna-se inadequada pelo fato dos testes consumirem recursos da nuvem, o que significa prejuízo, dado que esses recursos não estão sendo alugados (CASTANE; NUNEZ; CARRETERO, 2012) (BUYYA; RANJAN; CALHEIROS, 2009).

Nesse contexto, o uso de simulação para avaliação de desempenho se torna fundamental. Simular ambientes em nuvem tem custo de implementação menor, dado que não necessitam de acesso a um sistema real. Além disso, modelos a serem simulados podem ser reconfigurados mais facilmente, bastando mudar os parâmetros desejados. Por fim, se modelo e simulador forem adequados é possível garantir com mais confiança a reprodutibilidade dos experimentos.

Embora existam simuladores voltados para computação em nuvem, eles apresentam, em sua maioria, problemas como falta de documentação, incapacidade de modelagem de certas características fundamentais de computação em nuvem, falta de interfaces de auxílio à modelagem para o usuário e exigência de conhecimentos avançados em diversos *frameworks* de desenvolvimento e linguagens de programação. A soma desses fatores acaba por reduzir a experiência dos usuários com essas ferramentas, tornando a simulação de sistemas de computação em nuvem menos atraente.

1.2 Objetivo

Dados os problemas quanto ao uso dos simuladores de computação em nuvem existentes, o objetivo deste trabalho foi desenvolver uma abordagem icônica para a modelagem e simulação de computação em nuvem para o simulador iSPD (*iconic Simulator of Parallel and Distributed systems*). O iSPD é um simulador de eventos discretos voltado para a simulação de sistemas paralelos e distribuídos, que possui como foco principal prover facilidade de modelagem.

Para atingir este objetivo foi necessário estender as capacidades de modelagem do iSPD, buscando a construção de uma caracterização dos aspectos de computação em nuvem que se encaixasse na proposta de modelagem icônica simplificada proposta pelo simulador. O principal intuito dessa abordagem foi permitir que o usuário que não possua conhecimentos avançados em linguagens de programação, possa utilizar uma ferramenta simples e intuitiva para simular sistemas de computação em nuvem. Essa abordagem permite ainda que usuários experientes desenvolvam seus modelos mais facilmente por meio do uso de uma interface visual para modelagem.

Adicionalmente, buscou-se desenvolver uma ferramenta que atendesse tanto os interesses de modelagem de usuários simples, que desejem avaliar o desempenho de um serviço contratado de computação em nuvem, como provedores de serviços (sejam eles grandes ou pequenos)

que desejam avaliar o desempenho dos métodos empregados na implementação de sua nuvem pública ou privada, de modo a encontrar o ajuste necessário para melhorar o desempenho da infraestrutura física que possui.

Como objetivo secundário desta abordagem destaca-se a geração de um conjunto de métricas com visualização de resultados baseada em relatórios e gráficos. A importância desta funcionalidade está em prover uma melhor visualização das medidas de desempenho, obtendo-se de maneira nativa informações sobre o desempenho de políticas de alocação de máquinas virtuais e estimativas de custos de utilização do sistema. Ressalta-se que estas informações não estão disponíveis em sua totalidade nos simuladores existentes, podendo ser obtidas somente por meio da implementação direta do usuário.

Por fim, buscou-se ainda desenvolver uma ferramenta útil para o estudo de políticas de alocação de máquinas virtuais, por meio da geração de métrica e resultados e da implementação das principais políticas de alocação de máquinas virtuais existentes. A importância deste aspecto de modelagem é permitir o estudo de estratégias de alocação de máquinas virtuais para um determinado conjunto de trabalho, de modo a melhorar a utilização do sistema segundo algum critério como justiça, redução do uso de recursos, economia de energia, etc.

1.3 Organização do texto

Este trabalho está estruturado em seis capítulos. No capítulo 2 apresenta-se a fundamentação teórica sobre computação em nuvem e simuladores de computação em nuvem. Apresenta-se no capítulo 3 um maior detalhamento do iSPD, ferramenta base deste trabalho. No capítulo 4 apresenta-se o desenvolvimento deste trabalho, descrevendo a implementação da modelagem e simulação de ambientes de computação em nuvem para o iSPD. No capítulo 5 apresentam-se os testes de validação da ferramenta desenvolvida. Por fim, apresentam-se no capítulo 6 as conclusões sobre este trabalho.

2 Computação em nuvem: Definições e Conceitos

Neste capítulo, apresentam-se os principais conceitos de computação em nuvem que são necessários para uma melhor compreensão do trabalho desenvolvido. Deste modo, investigam-se as origens e definições mais relevantes para esta tecnologia, assim como suas principais características e classes de serviços, além de um breve estudo sobre o processo de alocação de máquinas virtuais. Por fim, apresentam-se os principais simuladores de computação em nuvem disponíveis na literatura, realizando um comparativo entre suas características de projeto e utilização.

2.1 As origens da terminologia e do conceito de computação em nuvem

Historicamente é comum o uso do termo "nuvem" para *internet*. Isso se dá principalmente pela popularização da representação gráfica de uma rede complexa pelo desenho de uma nuvem (RITTINGHOUSE; RANSOME, 2009). Quanto ao termo "computação em nuvem", segundo a publicação eletrônica *Technology Review* (MIT, 2015), o primeiro uso se deu em um relatório de negócios da Compaq que abordava o oferecimento de serviços de *software* por meio da *internet* (COMPAQ, 1996).

O conceito teórico de computação em nuvem, por sua vez, é atribuído a John McCarthy. Ele teria sugerido, em 1961, que os avanços tecnológicos gerariam no futuro um modelo de negócio em que poder computacional e o uso de algumas aplicações seriam comercializados de forma utilitária, como ocorre com a água ou a energia elétrica por exemplo (RITTINGHOUSE; RANSOME, 2009).

Embora a ideia de computação utilitária tenha sido muito discutida nos anos subsequentes, somente em meados da década de 70 ficou claro que as tecnologias existentes até então eram incapazes de implementar esse modelo de computação "futurista". Somente nos anos 2000, com o amadurecimento de tecnologias como processadores multinúcleo, grades computacionais, virtualização de *hardware* e automação de *datacenters* é que esta ideia tornou-se novamente popular, utilizando agora o nome de "computação em nuvem" (BUYYA; BROBERG; GOSCINSKI, 2011).

2.2 Definição de computação em nuvem

Diversos autores têm buscado uma definição formal para o termo "computação em nuvem" (BUYYA; BROBERG; GOSCINSKI, 2011). Apresentam-se as definições mais relevantes nos parágrafos subsequentes.

Primeiramente um relatório da McKinsey and Co. (MCKINSEY; CO., 2009) define, observando um ponto de vista da indústria, computação em nuvem como "serviços baseados em *hardware* que oferecem computação, rede e capacidade de armazenamento de forma que: (i) o gerenciamento de *hardware* é altamente abstraído para o consumidor; (ii) consumidores veem os custos de infraestrutura como uma despesa operacional variável; (iii) a capacidade da infraestrutura é altamente elástica".

Por sua vez, Armbrust et al. (2009) apresentam um ponto de vista da academia em um relatório da Universidade da Califórnia, em Berkeley. Neste documento se enumera como principais características da computação em nuvem: "(i) A ilusão de recursos computacionais infinitos; (ii) eliminação da necessidade de submissão para um *front-end* pelos usuários da "nuvem" (iii) a capacidade de pagar pelo uso... como necessário...".

Por fim, observando uma visão governamental, o NIST (*National Institute of Standards and Technology*) define computação em nuvem como "... um modelo de negócio *pay-per-use* (pago por utilização) que permite, convenientemente, o acesso, por meio de rede, sob demanda, a um arranjo compartilhado de recursos computacionais configuráveis (rede, servidores, armazenamento, aplicações, serviços) que podem ser rapidamente providos e liberados com o mínimo esforço de gerenciamento ou interação do provedor de serviço" (MELL; GRANCE, 2011).

Embora existam diferentes graus de detalhamento entre essas definições, elas apontam um conjunto comum de características que serão detalhadas a seguir.

2.3 Características de Computação em Nuvem

Como discutido na seção anterior, encontra-se na maioria das definições de computação em nuvem um conjunto comum de características, descritas a seguir:

- **Autoatendimento:** Consiste em disponibilizar aos usuários uma interface que permita requisitar, configurar, utilizar e pagar por recursos sem a necessidade de intervenção de um operador humano (BUYYA; BROBERG; GOSCINSKI, 2011) (MELL; GRANCE, 2011);
- **Medição e pagamento por utilização:** Consiste em implementar soluções eficientes para a monitoração, cobrança e pagamento dos recursos utilizados por meio de medidas adequadas e de fácil compreensão para cada tipo de serviço (BUYYA; BROBERG; GOSCINSKI, 2011) (MELL; GRANCE, 2011);

- **Elasticidade:** Consiste na abstração de que o usuário tem acesso a recursos infinitos disponibilizados sob demanda. Desta forma, espera-se que o serviço forneça recursos em qualquer quantidade e a qualquer momento, cobrindo completamente as necessidades do sistema hospedado. É importante ainda que a estrutura possua capacidade para tanto aumentar o volume de recursos disponíveis quanto reduzi-los, quando não forem mais necessários (BUYYA; BROBERG; GOSCINSKI, 2011);

- **Estrutura configurável:** Consiste em oferecer um serviço que permita uma grande variabilidade de instalação de pilhas de *software* e configuração de parâmetros, de modo que um mesmo sistema possa atender a uma vasta gama de usuários com necessidades distintas (BUYYA; BROBERG; GOSCINSKI, 2011);

- **Amplo acesso:** Consiste em padronizar e permitir o acesso aos serviços oferecidos por meio de diferentes dispositivos (*tablets*, *smartphones*, *notebooks* e *desktops*, etc.) dado que a disponibilidade de tais serviços é provida por meio da *internet*. (MELL; GRANCE, 2011).

2.4 Computação em nuvem: uma soma de esforços

Como indicado na seção 2.1, o paradigma de computação em nuvem tornou-se possível nos últimos anos devido ao amadurecimento de uma série de tecnologias que, juntas, contribuíram para a elaboração deste novo modelo de computação utilitária, como é ilustrado na Figura 1. Deste modo, pode-se destacar como responsáveis pelo advento da computação em nuvem, segundo Buyya (BUYYA; BROBERG; GOSCINSKI, 2011), os avanços das seguintes tecnologias:

- **Tecnologias de *internet*:** Destaca-se como principal contribuição desta tecnologia a criação dos *web services* e, consequentemente, de arquiteturas orientadas a serviços (*Service-Oriented Architectures* - SOA), permitindo a comunicação de aplicações por meio da *internet*;

- **Computação autonômica:** Destacam-se como principais contribuições desta tecnologia inovações no gerenciamento de grandes *datacenters*, principalmente no desenvolvimento de *softwares* de automação com o intuito de executar tarefas como gerenciamento de níveis de serviço e de capacidade da infraestrutura, recuperação de catástrofe proativa e automação do provisionamento de máquinas virtuais;

- **Computação em grade:** Destaca-se como maior contribuição desta tecnologia o esforço na elaboração de padrões e protocolos públicos com intuito de oferecer acesso a computação distribuída de alto desempenho por meio da *internet* pública, utilizando-se do conceito de aglomeração de recursos;

- **Virtualização de *hardware*:** Esta tecnologia destaca-se como um conceito chave de computação em nuvem, e consiste em permitir a execução de múltiplos sistemas operacionais com pilhas de *software* heterogêneas em uma mesma plataforma física.

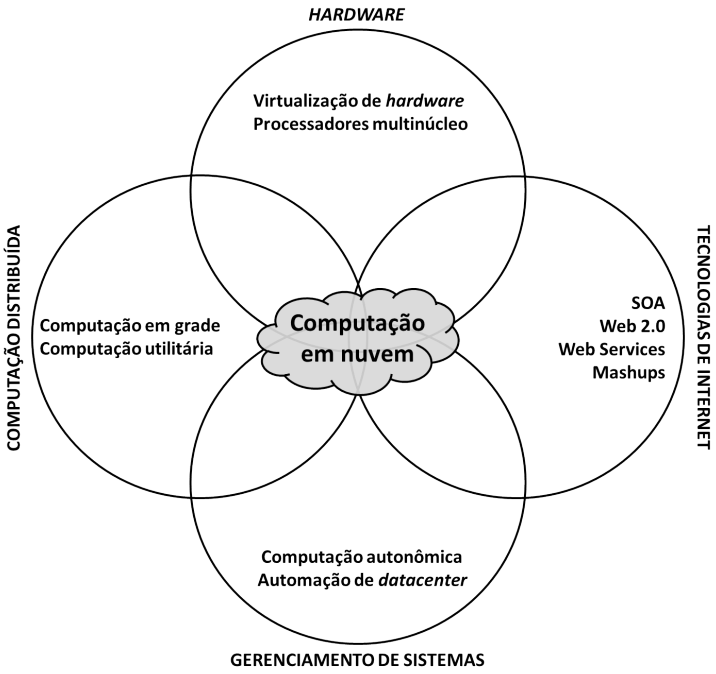


Figura 1 – Soma de esforços que contribuíram para o advento da computação em nuvem (adaptado de (BUYYA; BROBERG; GOSCINSKI, 2011))

Dada a importância da virtualização de *hardware* para a compreensão dos conceitos de computação em nuvem e consequentemente das atividades realizadas neste trabalho, apresenta-se na próxima seção uma abordagem mais detalhada sobre este conceito.

2.4.1 Conceitos de virtualização de *hardware*

A ideia de virtualização de *hardware* consiste em executar múltiplos sistemas operacionais com pilhas de *software* heterogêneas em uma mesma plataforma física. Isso é possível por meio da inserção de uma camada de virtualização entre o sistema físico e os sistemas virtualizados (chamados de máquinas virtuais, ou VMs). O papel da camada de virtualização é gerenciar o acesso e controle sobre os recursos de *hardware* utilizados por todas as máquinas virtuais hospedadas, por meio de um *software* conhecido como monitor de máquinas virtuais (*Virtual Machine Monitor* - VMM) ou *hypervisor*. Existem atualmente um grande número de VMMs disponíveis, dentre os quais destacam-se o VMware ESXi (VMWARE, 2013), o Xen (XEN, 2013) e o KVM (KVM, 2013).

O conceito de virtualização é ilustrado na Figura 2, em que se apresenta um VMM gerenciando o acesso das máquinas virtuais com sistemas heterogêneos aos recursos físicos de um *hardware* qualquer.

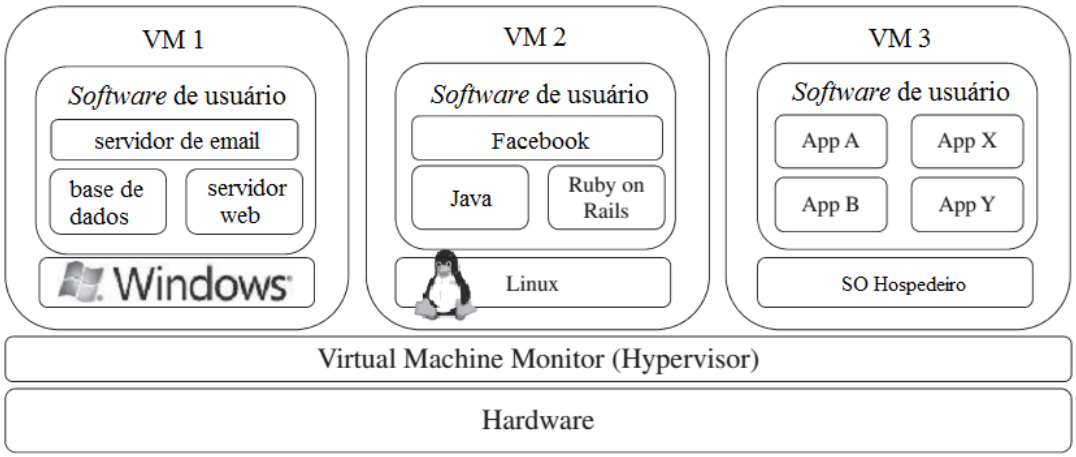


Figura 2 – Exemplo de uma plataforma de *hardware* executando uma série de sistemas virtualizados, gerenciados por um VMM (BUYYA; BROBERG; GOSCINSKI, 2011)

Ressalta-se que embora o conceito de virtualização exista há décadas, ele tornou-se interessante para a utilização em sistemas servidores somente com a popularização de tecnologias como processadores com múltiplos núcleos de processamento (*multicore*), paravirtualização e virtualização assistida por *hardware*. Destacam-se ainda os esforços de padronização, como a elaboração do *Open Virtualization Format* (OVF) (DMTF, 2012). Como consequência de tais avanços tecnológicos, a virtualização tornou-se a ferramenta ideal de implementação de sistemas computação em nuvem.

Segundo (BUYYA; BROBERG; GOSCINSKI, 2011), aponta-se tradicionalmente como benefícios da virtualização, melhorias no compartilhamento e utilização de um sistema, melhor gerenciamento e alta confiabilidade. Entretanto, com a adoção de virtualização tanto por sistemas clientes quanto servidores, diversos autores têm destacado três capacidades básicas em relação ao gerenciamento de carga de trabalho em um sistema virtualizado (THOMSEN, 2002):

- **Isolamento:** Consiste no fato de que todas as instruções de programa estão confinadas dentro do espaço da VM na qual ele se encontra, o que representa maior segurança e confiabilidade, pois erros ficam isolados dentro da VM em que aconteceram. Outro ponto importante desta capacidade é que o desempenho de uma VM não afeta diretamente o desempenho de outra;

- **Consolidação:** Consiste em hospedar vários servidores virtualizados, que possuem cargas de trabalho heterogêneas em uma mesma plataforma física. Esta capacidade é muito útil por permitir utilizar o potencial de recursos de *hardware* geralmente subutilizados em servidores dedicados. Outro aspecto positivo desta capacidade é a possibilidade de manter em uma mesma estrutura física um servidor atualizado e outro mais antigo, a fim de resolver problemas de compatibilidade de *software*;

- **Migração:** Consiste em salvar o estado de uma VM (disco virtual utilizado, imagem

da memória RAM e arquivos de configuração) e permitir que tal estado seja suspenso, serializado, migrado para outra plataforma e novamente restaurado. Observa-se que tal capacidade permite realizar balanceamento de carga de trabalho e recuperação de falha, além de simplificar a manutenção física do sistema.

2.5 Classes de serviços de computação em nuvem

Os serviços oferecidos por computação em nuvem dividem-se tradicionalmente em três classes de serviços conhecidas como IaaS, PaaS e SaaS, que oferecem infraestrutura de *hardware*, plataformas de desenvolvimento e disponibilidade de aplicações, respectivamente, como descrito a seguir:

- ***Infrastructure-as-a-Service* (IaaS):** Esta classe de serviço consiste em prover toda infraestrutura de *hardware* (processamento, armazenamento, rede e quaisquer outros recursos necessários e disponíveis) para que se possa instalar e manter operacional uma estrutura de *software* arbitrária, como sistemas operacionais e/ou aplicações (MELL; GRANCE, 2011).

  Como exemplos de serviço de IaaS pode-se destacar o Amazon Elastic Computing (EC2) (AMAZON, 2015) e o Google Compute Engine (GCE) (GOOGLE, 2015a), que oferecem o acesso a máquinas virtuais completamente configuráveis, de maneira bastante similar a servidores físicos.

- ***Platform-as-a-Service* (PaaS):** Esta classe de serviço tem como objetivo disponibilizar plataformas de desenvolvimento e hospedagem de aplicações, oferecendo um alto nível de abstração. A ideia é isentar o programador de preocupar-se com detalhes inerentes à infraestrutura (como o número de processadores ou a quantidade de memória que será utilizada pela aplicação desenvolvida, por exemplo). Alcança-se este nível de desenvolvimento por meio de padronizações do uso de linguagens, bibliotecas, serviços e ferramentas indicadas e suportadas pelo provedor de serviço (BUYYA; BROBERG; GOSCINSKI, 2011).

  Como exemplo de serviço de PaaS, pode-se destacar o Google AppEngine (GOOGLE, 2013), que é um ambiente escalável em que desenvolvedores podem implementar e hospedar aplicações *web* desenvolvidas em linguagens de programação específicas, como Python ou Java.

- ***Software-as-a-Service* (SaaS):** Esta classe de serviço tem como objetivo tornar disponíveis, por meio da interface de um navegador *web* ou de dispositivos móveis, aplicações de uso rotineiro, como aplicativos de escritório e editores de imagem. Esta classe de serviço mostra-se útil por eliminar a necessidade de instalação e manutenção de programas correspondentes em disco nos recursos computacionais de uma instituição.

Como exemplo de serviço de SaaS destacam-se o Google Docs (GOOGLE, 2015b) e o Office Web Apps (MICROSOFT, 2015), que oferecem uma série de aplicações de "escritório", como editor de texto, processador de planilhas, editor de apresentações, editor de imagens, etc.

2.6 Políticas de alocação de máquinas virtuais em ambientes de computação em nuvem

Dado que o objetivo deste trabalho é implementar um simulador de ambientes de nuvem, mais especificamente da classe de serviço de IaaS, é necessário apresentar alguns conceitos preliminares sobre o processo de alocação de máquinas virtuais em ambientes de computação em nuvem. Deste modo apresenta-se nesta seção uma definição para o processo de alocação de máquinas virtuais, assim como algumas heurísticas de implementação mais comumente utilizadas para esta finalidade.

Primeiramente, define-se como alocação de máquinas virtuais o processo de traçar estratégias para hospedar um conjunto de máquinas virtuais em um conjunto de recursos que integram uma infraestrutura física. O processo de alocação de máquinas virtuais divide-se basicamente em duas etapas, descritas a seguir (MISHRA; SAHOO, 2011):

- **Estimativa de recursos:** Primeiramente, toma-se um conjunto de máquinas virtuais e estima-se a quantidade de recursos computacionais utilizados (processamento, memória, disco, banda de rede, etc.) por cada VM. Embora a quantidade de recursos possa ser difícil de ser estimada, uma abordagem comum é a verificação de um histórico de uso de recursos pela VM;
- **Mapeamento das VMs para os servidores físicos:** Dado que a demanda de recursos tenha sido estimada, mapeia-se o modo como as VMs serão hospedadas pelo conjunto de servidores físicos que integram um determinado *datacenter*, alocando-as a partir de algum critério de seleção (política de alocação)

Ressalta-se que o problema de alocar um número n de VMs em um número m de servidores é um problema de complexidade NP-completo, dado que equivale ao problema das múltiplas mochilas (*multiple knapsack problem*). Por isso, utilizam-se comumente soluções aproximadas para a alocação de VMs, baseadas em heurísticas simples. Segundo (NATHANI; CHAUDHARY; SOMANI, 2012), as principais políticas de alocação de VMs para IaaS seguem o conceito de alocação imediata ou de melhor esforço para realizar a alocação.

As heurísticas utilizadas nas principais políticas de alocação consistem basicamente em selecionar uma VM assim como uma máquina física onde ela será alocada, segundo algum critério de seleção. Após a seleção da VM a ser alocada e da máquina hospedeira, realiza-se

a verificação de recursos. Se a máquina hospedeira possuir recursos suficientes para atender a demanda de recursos da VM selecionada, a VM em questão será então alocada nesta máquina, caso contrário seleciona-se a próxima máquina segundo o mesmo critério de seleção, até que a a máquina física selecionada atenda a demanda de recursos da VM, que será então alocada. Caso nenhuma máquina física possua recursos suficientes para alocar a VM selecionada, ela será rejeitada.

Apresenta-se a seguir uma relação das principais heurísticas utilizadas para a alocação de máquinas virtuais (CAMATI, 2013):

- ***Round-Robin*:** Também conhecido como alocação circular, este método consiste em alocar as VMs uniformemente entre as máquinas físicas, que são selecionadas por meio de uma lista circular;
- ***First-Fit*:** Este método é muito simples e consiste em alocar a VM na máquina física que primeiro atender seus requisitos de recursos, rejeitando-a caso nenhuma máquina física atenda sua demanda por recursos;
- ***First-Fit Decreasing (FFD)*:** Este método é uma variante do *first-fit*, em que primeiramente se ordena a lista de VMs a serem alocadas em ordem decrescente de acordo com a quantidade de recursos solicitados, para depois aplicar o critério de alocação utilizado pelo *first-fit* original. A ideia desta heurística é a de tentar alocar primeiramente as VMs que necessitam de mais recursos, dado que a alocação de VMs menores primeiro pode diminuir as chances de encontrar posteriormente máquinas físicas com recursos disponíveis suficientes para atender as VMs maiores;
- ***Volume*:** Este método pode ser considerado uma variante geométrica do *first-fit*. Ao contrário do *first-fit decreasing*, em que a ordenação é realizada no conjunto de VMs, neste método ordena-se a lista de máquinas físicas em ordem decrescente de acordo com o volume de recursos disponíveis (multiplicando-se as dimensões de processamento, memória, disco, etc.). Após a ordenação, aplica-se o critério de alocação do *first-fit* original. O conceito desta heurística é realizar a alocação das VMs primeiramente nos recursos físicos maiores, dado que é mais provável reduzir a utilização de recursos físicos deste modo, alocando mais máquinas virtuais em um mesmo recurso.

2.7 Simuladores de computação em nuvem

Nesta seção apresenta-se um levantamento sobre simuladores de computação em nuvem, assim como uma análise comparativa de suas principais funcionalidades e características arquiteturais.

Existe uma quantidade relativamente pequena de simuladores de computação em nuvem, sendo que alguns deles apresentam ainda problemas como documentação escassa ou inexistente, enquanto outros encontram-se em fase de protótipo ou exigem a instalação de ferramentas de desenvolvimento e linguagens de programação que não são muito populares para que se possa utilizá-los. Além disso, existem ainda vários simuladores que apenas estendem algumas características e funcionalidades de simuladores mais genéricos.

Destacam-se, desta forma, três simuladores de computação em nuvem que apresentam boa popularidade e documentação bem estabelecida. São eles o CloudSim (BUYYA; RANJAN; CALHEIROS, 2009), iCanCloud (CASTANE; NUNEZ; CARRETERO, 2012) e o GreenCloud (KLIAZOVICH et al., 2010). Apresenta-se uma descrição mais detalhada desses simuladores nas próximas seções.

2.7.1 CloudSim

O simulador CloudSim é desenvolvido pelo grupo CLOUDS (*The **Clo**ud Computing and **D**istributed **S**ystems*) da Universidade de Melbourne na Austrália, sob a coordenação de Rajkumar Buyya. O CloudSim é desenvolvido em Java e consiste basicamente em um conjunto de bibliotecas que implementam as classes de elementos necessários para a simulação de computação em nuvem. Desta forma, para utilizar este simulador, o usuário precisa obrigatoriamente possuir conhecimento prévio de programação em Java e consequentemente conhecer o paradigma de orientação a objeto.

O CloudSim possui uma capacidade de modelagem bastante ampla, permitindo ao usuário modelar diversos elementos constituintes da estrutura de computação em nuvem, como *datacenters*, máquinas individuais, escalonadores (*brokers*), possuindo ainda a capacidade de simular virtualização, por meio da modelagem de máquinas virtuais.

Como a modelagem no CloudSim é realizada por meio de codificação Java, o usuário pode ainda implementar suas próprias políticas de alocação de recursos, provisionamento de VMs e escalonamento de tarefas, caso uma política desejada não esteja implementada na ferramenta. No entanto, destaca-se como ponto negativo desta abordagem a inexistência de uma interface gráfica de auxílio à modelagem.

Destaca-se ainda como desvantagem o fato do CloudSim não possuir nenhum gerador de números aleatórios, o que limita a simulação do comportamento dos tempos de chegada das tarefas que integram uma carga de trabalho, por exemplo.

Por fim, ressalta-se que o CloudSim é utilizado como base de desenvolvimento de diversos outros projetos que estendem suas funcionalidades para nichos de utilização específicos. Dentre estes projetos citam-se o CloudAnalist (WICKREMASINGHE; CALHEIROS; BUYYA, 2010), CloudSimEx (GROZEV, 2015), RealCloudSim (ROCHA, 2013) e DynamicCloudSim (BUX; LESER, 2013).

2.7.2 iCanCloud

O simulador iCanCloud é desenvolvido pelo grupo ARCOS (*Grupo de **Ar**quitectura de Computadores, **Co**municaciones y **S**istemas*) da universidade Carlos III de Madrid, na Espanha, sob coordenação de Jesús Carretero Pérez. O iCanCloud é desenvolvido por meio dos *frameworks* OMNeT++ e INET (OMNET++, 2013), exigindo de seus usuários conhecimento prévio dessas ferramentas. Destaca-se, como ponto positivo, que este simulador apresenta uma interface gráfica de auxílio a modelagem, embora seja ainda necessário para a modelagem utilizar as ferramentas de desenvolvimento (necessidade de programar).

O principal objetivo do iCanCloud é avaliar o compromisso entre custo e desempenho de um conjunto de aplicativos executados em uma configuração de *hardware* específica, fornecendo aos usuários informações úteis sobre esses custos (NUNEZ et al., 2012) (CASTANE; NUNEZ; CARRETERO, 2012). Para isto este simulador se baseia no modelo de computação em nuvem provido pela Amazon (*Amazon Elastic Computing* - EC2) (AMAZON, 2015).

O foco do simulador iCanCloud está na simulação da classe de serviços de IaaS. Destaca-se entre suas funcionalidades a modelagem de virtualização, capacidade de modelagem de sistemas de armazenamento (podendo modelar armazenamento local, remoto e paralelo), suporte à modelagem e simulação de aplicações por meio de uma API adaptada da biblioteca de MPI (*Message Programming Interface*), uso de *traces* de aplicações reais, grafos de estado e programação da aplicação diretamente na plataforma de simulação.

2.7.3 GreenCloud

O simulador GreenCloud é desenvolvido pela Universidade de Luxemburgo, sob a coordenação de Dzmitry Kliazovich. O GreenCloud é uma extensão da plataforma Network Simulator NS2 (ISI, 2013) e suas linguagens de desenvolvimento e modelagem são o C++ e *scripts* de Tcl (*Tool command language*). O simulador possui uma interface gráfica, embora a mesma não seja nativa do simulador e deva ser baixada como uma extensão.

O foco do GreenCloud é a análise de desempenho energético, envolvendo principalmente a análise da comunicação da nuvem (KLIAZOVICH et al., 2010), muito em função do simulador ter sido desenvolvido como uma extensão de um simulador de redes. Com isso faltam ao GreenCloud capacidades para a modelagem de virtualização e análise de desempenho, exceto métricas seguindo o conceito de computação verde[1].

2.7.4 Comparativo entre ferramentas

A comparação entre simuladores pode ser sistematizada com a análise de parâmetros importantes do ponto de vista de usabilidade e arquitetura. Nesse sentido os principais pontos de

[1] Termo empregado para designar projetos de T.I. com foco em consumo consciente de energia e recursos, a fim de reduzir o uso de energia e, consequentemente, os custos operacionais

análise são:

- ***Framework* base de desenvolvimento do simulador**, o que reflete questões de desempenho, portabilidade e grau de dificuldade de utilização da ferramenta;
- **Linguagem de programação em que foi implementado**, o que reflete principalmente em questões de desempenho e portabilidade da ferramenta;
- **Licença de utilização**, que reflete em disponibilidade da aplicação para uso e disponibilidade de seu código-fonte, o que permite uma melhor capacidade de compreensão de funcionamento, além de permitir sua extensão;
- **Suporte a interface gráfica**, o que simplifica e melhora a experiência do usuário com a ferramenta;
- **Capacidade de simular virtualização**, o que reflete em uma melhor caracterização da arquitetura de nuvem propriamente dita.

Apresenta-se na Tabela 1 o comparativo entre os simuladores estudados, considerando os pontos discutidos.

Tabela 1 – Comparativo entre simuladores de computação em nuvem

	CloudSim	iCanCloud	GreenCloud
Framework base	—	OMNeT++, MPI	NS2
Linguagem de programação	Java	C++	C++, Tlc
Licença de uso	*Open Source*	*Open Source*	*Open Source*
Suporte a interface gráfica	Limitado (por meio do uso de extensões)	Existente	Limitado (por meio do uso de uma extensão)
Suporte à virtualização	Sim	Sim	Não

Realizando uma breve análise, com relação ao *framework* de desenvolvimento, os simuladores iCanCloud e GreenCloud possuem como desvantagem, do ponto de vista do usuário, a necessidade de possuir conhecimentos adicionais sobre os *frameworks* para utiliza-los.

Quanto à linguagem de programação utilizada, destaca-se principalmente o simulador CloudSim, pelo desenvolvimento em Java, o que garante portabilidade entre sistemas.

O suporte a interfaces gráficas é um ponto muito importante, por refletir na facilidade de modelagem. Com base neste fator, destaca-se positivamente o simulador iCanCloud, que

possui uma interface gráfica de auxílio a modelagem própria, enquanto CloudSim e GreenCloud realizam a modelagem por meio de programação, o que exige um nível maior de conhecimentos do usuário.

Por fim, analisando a capacidade de modelagem, caso se queira utilizar modelos que caracterizem virtualização, deve-se utilizar os simuladores CloudSim ou iCanCloud, dada a incapacidade do GreenCloud de modelagem desta característica.

2.8 Considerações finais

Buscou-se neste capítulo apresentar o embasamento teórico necessário sobre computação em nuvem para uma melhor compreensão deste trabalho. Além disso apresentou-se um levantamento dos principais simuladores de computação em nuvem presentes na literatura. Dessa apresentação conclui-se que nenhum simulador atende satisfatoriamente aos requisitos de facilidade de modelagem, uso e amplitude de aplicação. Com isso, no próximo capítulo apresenta-se o simulador iSPD (***i**conic **S**imulator of **P**arallel and **D**istributed Systems*) que é a ferramenta base deste trabalho, para que depois se possa iniciar a descrição deste trabalho, buscando permitir a modelagem e simulação de computação em nuvem por meio de uma abordagem icônica.

3 iSPD: *iconic Simulator of Parallel and Distributed Systems*

Apresenta-se neste capítulo o simulador iSPD (***i**conic **S**imulator of **P**arallel and **D**istributed systems*) (MANACERO et al., 2012), que é a ferramenta base deste trabalho. Dado que o objetivo deste trabalho é o desenvolvimento da capacidade de simulação de computação em nuvem, busca-se apresentar os aspectos mais importantes e pontos de projeto em que devem ser realizadas modificações no iSPD para atingir esse objetivo.

3.1 Visão Geral do iSPD

O iSPD é um simulador de grades computacionais desenvolvido pelo GSPD (**G**rupo de **S**istemas **P**aralelos e **D**istribuídos) da UNESP (GSPD, 2012). A linguagem de desenvolvimento do simulador é o Java, principalmente pela portabilidade apresentada por aplicações desenvolvidas nesta linguagem.

O principal objetivo do iSPD é oferecer facilidade de uso, de modo a tornar o processo de modelagem e simulação mais intuitivo e simplificado. Esse objetivo é obtido por meio de um conjunto de interfaces de auxílio à modelagem, como uma interface icônica para modelagem da grade computacional e janelas de configuração de carga de trabalho e geração de políticas de escalonamento. Tais facilidades isentam o usuário da necessidade de possuir conhecimentos prévios em programação.

A ferramenta iSPD é composta por módulos que desempenham funções bem definidas. O modo como esses módulos se relacionam é exibido na Figura 3 e uma descrição geral de cada um dos módulos é dada a seguir:

- **Interface icônica:** Módulo responsável pela modelagem do sistema por meio do uso de ícones, que representam os elementos que constituem a grade computacional. Outra função deste módulo é a configuração de usuários e da carga de trabalho;
- **Interpretador de modelos internos:** Módulo responsável por interpretar o modelo icônico gerado a partir da interface icônica, convertendo-o para o modelo de filas compreendido pelo motor de simulação;
- **Interpretador de modelos externos e exportador de modelos internos:** Módulo responsável por interpretar modelos gerados a partir de outros simuladores (atualmente Simgrid (CASANOVA, 2001) e GridSim (BUYYA; MURSHED, 2002)) para modelos icônicos do

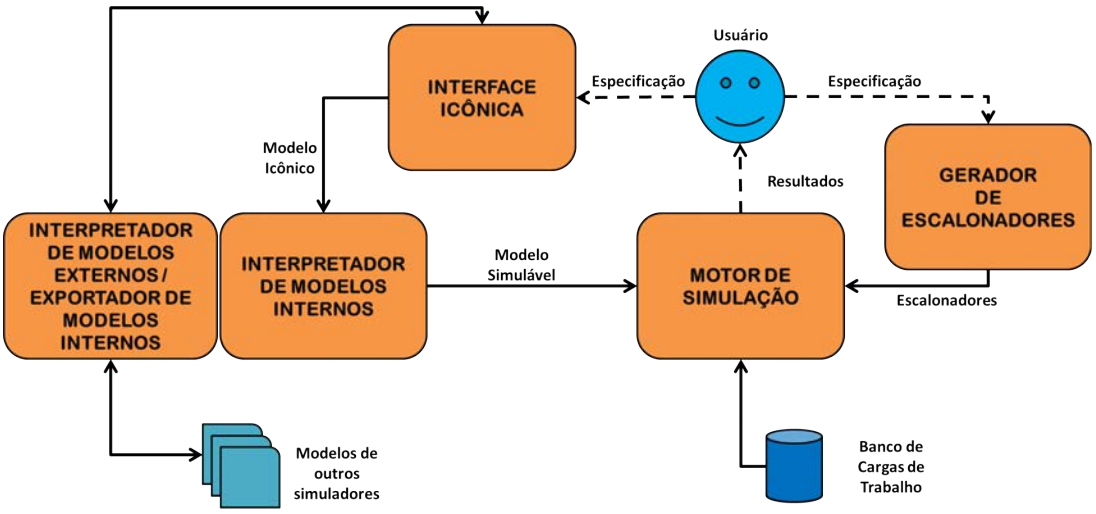


Figura 3 – Diagrama conceitual do iSPD

iSPD. Já o exportador de modelos internos é responsável por exportar modelos icônicos do iSPD para modelos correspondentes para esses simuladores.

- **Motor de simulação:** Módulo responsável por realizar a simulação de eventos discretos a partir da leitura de um modelo de filas. Outra função deste módulo é calcular as métricas e apresentar os resultados da simulação realizada;
- **Gerador de escalonadores:** Módulo responsável por gerenciar os escalonadores disponíveis para uso na simulação, além de permitir a geração de novas políticas de escalonamento de maneira facilitada, por meio de formulações matemáticas simples.

Apresenta-se nas próximas seções uma melhor descrição dos módulos de interface icônica, interpretador de modelos internos e motor de simulação, pois neles ocorreram mudanças para permitir a simulação de computação em nuvem.

3.2 Interface icônica

Como já apresentado, o módulo de interface icônica do iSPD é responsável pela modelagem e configuração do ambiente computacional a ser simulado, assim como da configuração dos usuários e da carga de trabalho aplicada.

Na Figura 4 é exibida a interface icônica do iSPD. Nela estão destacados, pelos retângulos marcados de 1 a 7, os painéis que constituem a interface icônica auxiliando o usuário no trabalho de modelagem. A função das áreas destacadas na interface é descrita a seguir:

1. **Área de seleção de ícones:** Em que se encontram os ícones utilizados pelo usuário para modelar o ambiente computacional. Existem quatro ícones de modelagem:

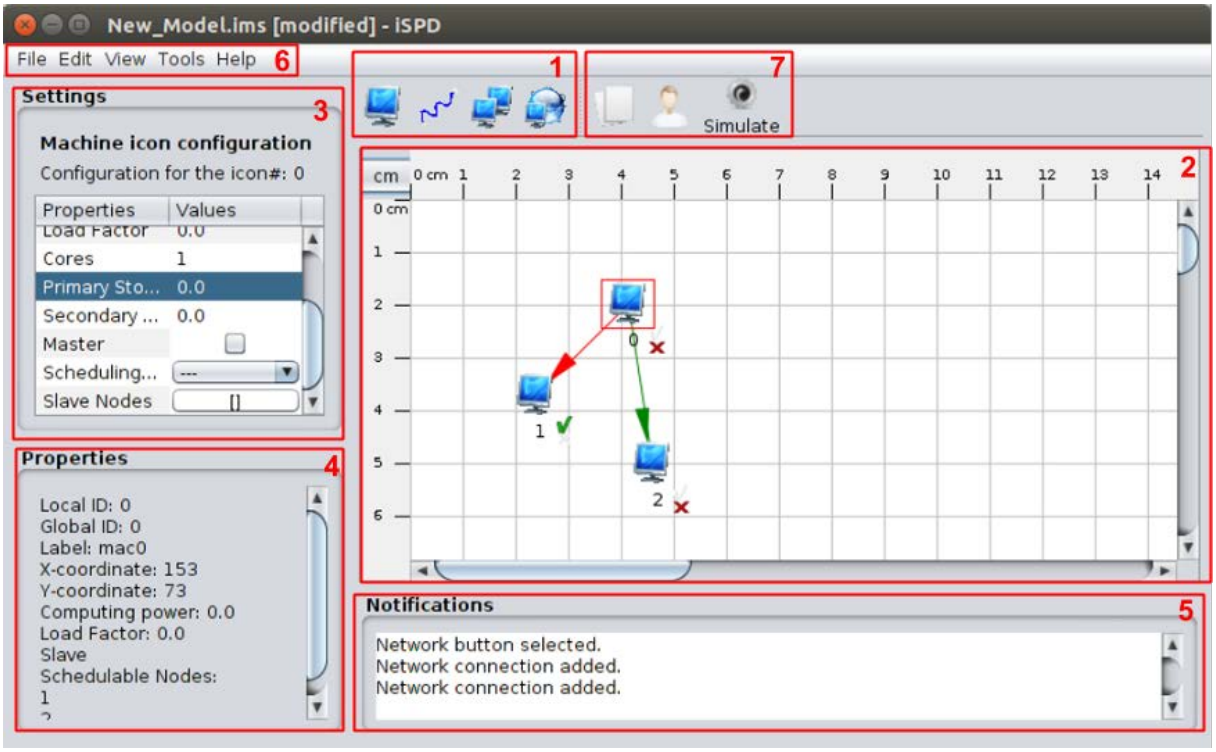


Figura 4 – Interface icônica do iSPD

- **Máquina individual:** Representado pelo ícone de monitor, tem como objetivo simular o comportamento de uma única máquina, com suas respectivas capacidades de processamento e armazenamento;
- ***Cluster* (Aglomerado de máquinas):** Representado pelo ícone de monitor duplo, tem como objetivo simular o comportamento de um aglomerado de duas ou mais máquinas, que possuem configuração de *hardware* homogênea e estão interligadas por um *switch*;
- **Enlace lógico:** Representado pelo ícone de cabo de rede, que serve para representar o comportamento de um enlace de rede entre dois outros elementos;
- ***Internet*:** Representado pelo ícone de globo, representando o comportamento de uma rede de *internet* pública no caminho entre dois ícones de computação (máquina individual ou *cluster*).

2. **Área de desenho:** Área em que o usuário realiza a modelagem do ambiente computacional. O processo de modelagem consiste em selecionar os ícones correspondentes aos elementos que se deseja simular na área de seleção de ícones e "arrastá-lo" para esta área da interface. Após desenhar o modelo, a configuração das características inerentes a cada ícone é realizada na área de configuração ou por meio de uma janela aberta por um duplo "clique" sobre o ícone;

3. **Área de configuração:** Parte da interface em que o usuário realiza a configuração das

características inerentes a cada ícone, atribuindo valores referentes à capacidade de computação para os elementos de processamento e capacidade de transferência de dados e atrasos para os elementos de comunicação;

4. **Área de propriedades:** Parte da interface em que o usuário visualiza o conjunto de propriedades referentes ao ícone selecionado;

5. **Área de notificações:** Parte da interface em que o usuário visualiza notificações sobre as ações que realiza no simulador;

6. **Barra de *menus*:** Parte da interface em que o usuário acessa os menus de abertura, salvamento, importação e exportação de arquivos, edição de modelos, opções de visualização, ferramentas de gerenciamento e opções de ajuda;

7. **Configurações complementares e botão "Simular":** Parte da interface em que o usuário realiza configurações complementares ao ambiente computacional simulado e, por fim, dá início a simulação. Esta área da interface é constituída por três botões, cujas funções são:

   - **Configuração da carga de trabalho:** Representada pelo ícone de folhas em branco, este botão dá acesso a janela de configuração da carga de trabalho que será aplicada ao modelo simulado. Esta janela está exibida na Figura 5. Observa-se que o iSPD permite ao usuário modelar a carga de trabalho a partir de três formas distintas: carga totalmente randômica, carga aplicada a nós específicos do modelo e carga extraída de *traces* de aplicações reais;

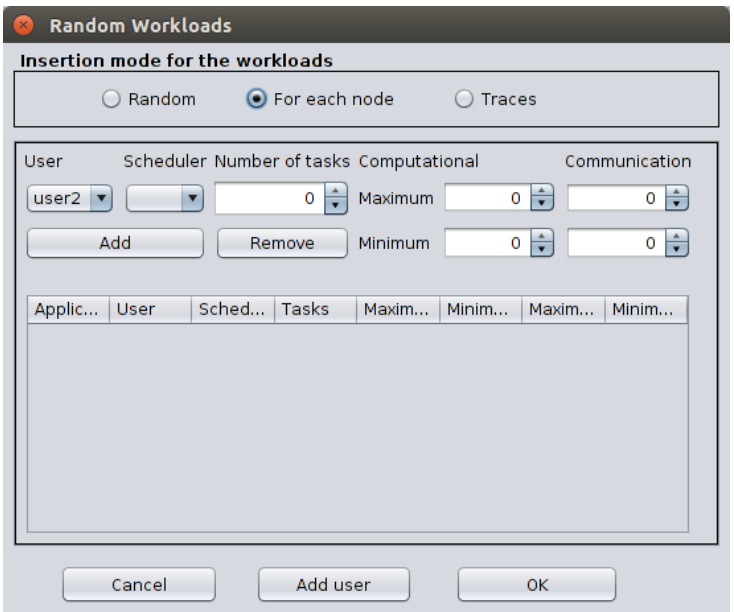


Figura 5 – Janela de configuração de carga de trabalho do iSPD

- **Configuração de usuários:** Representada pelo ícone de humanóide, este botão dá acesso a uma janela de configuração de usuários, exibida na Figura 6, em que se pode enumerar o conjunto de usuários do sistema a ser simulado.

Figura 6 – Janela de configuração de usuários do iSPD

- **Simular:** Representado pelo ícone de engrenagem, este botão dispara o processo de simulação do ambiente computacional modelado.

Pode-se perceber que o trabalho todo de modelagem é feito na interface icônica. Quando o usuário aciona o botão "*simulate*" o controle passa para o módulo interpretador de modelos internos, que verifica se não existem parâmetros faltando e, se todos os parâmetros estiverem defininos, converte o modelo icônico gerado pelo usuário em um modelo de filas simulável. Este modelo de filas é, então, executado no motor de simulação. Detalha-se na próxima seção o funcionamento do interpretador de modelos internos, apresentando o processo de conversão do modelo gerado na interface icônica para o modelo simulado no motor.

3.3 Interpretador de modelos internos

Como já apresentado, o papel do interpretador de modelos internos é o de receber um modelo icônico gerado por meio da interface icônica e convertê-lo para um modelo de filas que possa ser executado pelo motor de simulação. Descreve-se, nas próximas subseções, a estrutura do arquivo de modelo icônico e das estruturas de dados do modelo simulável, para que após conhecer com mais detalhes as linguagens objeto e alvo, se possa descrever mais claramente o processo de conversão entre esses formatos.

3.3.1 Arquivo de modelo icônico do iSPD

Quando o usuário salvar um modelo após configurá-lo na interface icônica, é gerado um arquivo de saída com toda informação refente ao modelo. Arquivos de modelos icônicos do iSPD chamam-se arquivos IMS (*Iconic Model of Simulation*) e utilizam a extensão "`.imsx`". Na

prática, um modelo IMS é um arquivo XML (*extensible Markup Language*) cuja especificação e verificação é realizada por meio de um arquivo DTD (*Document Type Definition*) chamado "`ispd.dtd`".

Por meio da estrutura definida no DTD "`ispd.dtd`", um modelo icônico divide-se em duas partes, descritas a seguir, que são a definição do ambiente, constituída pelo conjunto de ícones do modelo (máquinas, *clusters*, enlaces e *internets*) e carga de trabalho, destacada pela marca "`load`", como exibido na Figura 7.

```
...
<!ELEMENT system (owner*,
        (machine|link|cluster|internet)*,
        load?)>
...
```

Figura 7 – Trecho do DTD responsável pela verificação dos elementos que compõe o modelo icônico

- **Modelagem do ambiente:** Parte do modelo que transcreve o ambiente computacional a ser simulado, descrevendo cada ícone e suas respectivas características, como seu conjunto de atributos, seu identificador na grade (*label*) e sua posição na área de desenho. Os elementos de XML que representam cada ícone são descritos a seguir:

  - **Máquina:** Indicado pela marca "`machine`", os atributos que constituem esse elemento são o usuário proprietário, o poder computacional da máquina (em MFlops), o número de núcleos de processamento disponíveis, a utilização, que é a porcentagem de processamento da máquina que está sendo utilizada em computação de fundo[1] e a informação se a máquina é um nó mestre ou escravo na grade computacional, sendo necessário armazenar ainda a política de escalonamento de tarefas e uma lista de nós escravos caso a máquina seja um nó mestre. Apresenta-se na Figura 8 o trecho do DTD responsável pela verificação deste ícone;
  - ***Cluster*:** Indicado pela marca "`cluster`", os atributos que constituem esse elemento são o usuário proprietário, o número de máquinas que compõe o *cluster*, poder computacional (em MFlops), número de núcleos por máquina, banda de passagem de dados entre máquinas (em Mbps), latência de transmissão (em segundos) e por fim a política de escalonamento de tarefas utilizada. Apresenta-se na Figura 9 o trecho do DTD responsável pela verificação deste ícone;

[1] Quantidade de computação necessária para a execução de processos, como a carga de execução do S.O. e processos de usuário que não podem ser prejudicados pelo processamento cedido à grade computacional em que a máquina está vinculada.

```
...
<!ELEMENT machine (master?,position,icon_id,characteristic?)>
  <!ATTLIST machine id ID #REQUIRED>
  <!ATTLIST machine power CDATA "0.0">
  <!ATTLIST machine load CDATA "0.0">
  <!ATTLIST machine owner CDATA "user1">
...
```

Figura 8 – Trecho do DTD responsável pela verificação dos ícones de máquina

```
...
<!ELEMENT cluster (position,icon_id,characteristic?)>
  <!ATTLIST cluster id ID #REQUIRED>
  <!ATTLIST cluster nodes CDATA "0">
  <!ATTLIST cluster power CDATA "0.0">
  <!ATTLIST cluster bandwidth CDATA "0.0">
  <!ATTLIST cluster latency CDATA "0.0">
  <!ATTLIST cluster scheduler CDATA "---">
  <!ATTLIST cluster vm_alloc CDATA "---">
  <!ATTLIST cluster owner CDATA "user1">
  <!ATTLIST cluster master CDATA "true">
...
```

Figura 9 – Trecho do DTD responsável pela verificação dos ícones de *cluster*

– **Enlace:** Indicado pela marca "`link`", os atributos que constituem esse elemento são a banda de passagem (em Mbps), a latência de transmissão (em segundos), a taxa de ocupação do enlace (%) e a informação sobre os ícones que são origem e destino da conexão. Apresenta-se na Figura 10 o trecho do DTD responsável pela verificação deste ícone;

```
...
<!ELEMENT link (connect,position?,position?,icon_id)>
  <!ATTLIST link id ID #REQUIRED>
  <!ATTLIST link bandwidth CDATA "0.0">
  <!ATTLIST link load CDATA "0.0">
  <!ATTLIST link latency CDATA "0.0">
...
```

Figura 10 – Trecho do DTD responsável pela verificação dos ícones de enlace lógico

– **Internet:** Indicado pela marca "`net`", os atributos que constituem esse elemento são a banda de passagem (em Mbps), a latência de transmissão (em segundos) e a taxa de ocupação do enlace (%). Apresenta-se na Figura 11 o trecho do DTD responsável

pela verificação deste ícone;

```
...
<!ELEMENT internet (position,icon_id)>
  <!ATTLIST internet id ID #REQUIRED>
  <!ATTLIST internet bandwidth CDATA "0.0">
  <!ATTLIST internet load CDATA "0.0">
  <!ATTLIST internet latency CDATA "0.0">
...
```

Figura 11 – Trecho do DTD responsável pela verificação dos ícones de *internet*

- **Modelagem da carga de trabalho:** Parte do modelo que transcreve a carga de trabalho aplicada ao ambiente computacional. Pode ser definida de três modos:

  - **Carga Randômica:** Indicada pela marca "`random`", caracteriza-se pelos atributos de número de tarefas a serem criadas, tempo médio de chegada entre tarefas (em segundos) e por fim as probabilidades e tamanhos máximos, médios e mínimos de computação (em MFlop) e comunicação (em Mb). Apresenta-se na Figura 12 o trecho do DTD responsável pela verificação desta opção;

```
...
<!ELEMENT random (size,size)>
  <!ATTLIST random owner CDATA "user1">
  <!ATTLIST random tasks CDATA "0">
  <!ATTLIST random time_arrival CDATA "0">
...
```

Figura 12 – Trecho do DTD responsável pela verificação da opção de gerar carga randômica

  - **Carga definida por nó:** Indicada pela marca "`node`", caracteriza-se por meio de uma ou mais aplicações, em que cada aplicação tem como atributos a informação sobre o usuário que submeteu a aplicação, nó mestre em que as tarefas serão submetidas, número de tarefas a serem criadas, tamanhos máximos e mínimos de computação e comunicação, assim como na carga randômica. Apresenta-se na Figura 13 o trecho do DTD responsável pela verificação desta opção;
  - **Carga obtida de *trace* de aplicação real:** Indicada pela marca "`trace`", caracteriza-se por meio da informação do caminho absoluto em que o arquivo de *trace* com a descrição do conjunto de tarefas está armazenado, assim como o número de tarefas existentes e o formato em que os dados estão armazenados. Apresenta-se na Figura 14 o trecho do DTD responsável pela verificação desta opção.

```
...
<!ELEMENT node (size,size)>
  <!ATTLIST node application CDATA "app0">
  <!ATTLIST node owner CDATA "user1">
  <!ATTLIST node id_master CDATA "???">
  <!ATTLIST node tasks CDATA "0">
...
```

Figura 13 – Trecho do DTD responsável pela verificação da opção de gerar carga configurada por nó mestre

```
...
<!ELEMENT trace EMPTY>
  <!ATTLIST trace file_path CDATA #REQUIRED>
  <!ATTLIST trace tasks CDATA "0">
  <!ATTLIST trace format CDATA "iSPD">
...
```

Figura 14 – Trecho do DTD responsável pela verificação da opção de gerar carga a partir de um arquivo de *trace*

3.3.2 Modelo de filas do iSPD

O iSPD é um simulador orientado a eventos discretos e baseado em teoria de filas. Desta forma, um modelo simulável é constituído por uma rede de filas que possui um conjunto de centros de serviço e a informação sobre as ligações entre centros de serviço (conexões de entrada e saída, formando a rede de filas propriamente dita) e um conjunto de clientes (tarefas e mensagens) que percorrem as filas para serem atendidos.

A estrutura de dados que armazena a rede de filas no iSPD é constituída pelo conjunto de elementos que implementam os centros de serviço. Existem cinco tipos de centro de serviços (máquina, mestre, enlace, internet e *switch*), cabendo a cada tipo implementar o modelo de filas que mimetiza o comportamento do elemento real correspondente. Cabe ainda a cada centro de serviço armazenar as informações sobre as conexões de entrada e saída, ou seja, quais centros de serviço direcionam clientes para a sua fila e quais centros de serviço possuem as filas alimentadas por seus resultados. Por fim, cada centro de serviço deve ainda implementar os métodos de chegada de evento, atendimento, saída de evento e atendimento de requisição, que especificam o modo como cada centro de serviço deve realizar o tratamento dos clientes.

Pode-se dividir os centros de serviço do iSPD em duas classes, de acordo com a função que exercem dentro do processo de simulação do ambiente de grade. Essas classes são:

- **Centros de processamento:** Classe de centros de serviço que simulam os processos de escalonamento e processamento de tarefas, contabilizando métricas relativas ao processa-

mento das tarefas. Existem dois tipos de centros de processamento:

- **Máquina:** Este centro de serviço segue um modelo de filas M/M/n ("uma fila, vários servidores"), em que o número de servidores é igual ao número de núcleos de processamento que a máquina contém. A função deste centro de serviço é receber uma tarefa, computar seu tempo de processamento, marcá-la como tarefa concluída e reenviá-la para o centro de serviço que a escalonou;
- **Mestre:** Este centro de serviço segue o modelo de filas M/M/1 ("uma fila, um servidor"). A função deste centro de serviço é exercer o papel de *front-end* da grade, recebendo as tarefas submetidas na modelagem da carga de trabalho e realizando o escalonamento de tarefas, segundo a política de escalonamento especificada pelo usuário. Também é papel deste centro de serviço ainda contabilizar o retorno das tarefas concluídas;

- **Centros de comunicação:** Classe de centros de serviço que simulam o processo de comunicação de dados, contabilizando métricas relativas à transferência de dados. Existem três tipos de centros de comunicação:
  - **Enlace:** Este centro de serviço segue o modelo de filas M/M/1 ("uma fila, um servidor"). A função deste centro de serviço é simular a transmissão de dados (tarefa ou mensagem) por um enlace lógico entre dois elementos de processamento, contabilizando os atrasos de latência de transmissão, tempo de espera na fila e tempo de transmissão do cliente;
  - ***Internet*:** Este centro de serviço segue o modelo de filas M/M/∞ ("uma fila, infinitos servidores"). A função deste centro de serviço é simular a transmissão de dados por uma infraestrutura de *internet* pública. A diferença deste centro de serviço para o centro de serviço de enlace é que ele funciona apenas como um centro de atraso, não contabilizando tempo de espera na fila dado que existem infinitos servidores;
  - ***Switch*:** Este centro de serviço segue o modelo de filas M/M/1 ("uma fila, um servidor"). A função deste centro de serviço é simular a comutação de dados entre elementos que constituem a rede de filas construída a partir do elemento icônico de *cluster*, contabilizando os atrasos de espera na fila durante este processo.

Quanto ao conjunto de clientes da rede de filas, que é constituído pelas tarefas que constituem a carga de trabalho aplicada ao sistema, as estruturas de dados que implementam as tarefas no iSPD são caracterizadas pelos atributos de identificador da tarefa, usuário submetente, identificador do recurso em que foi submetida, instante de chegada no sistema (em segundos), *status* de execução (em espera, em execução, cancelada ou concluída), tamanho de computação (em MFlop) e tamanho de comunicação (em Mb).

3.3.3 Tradução do modelo icônico para o modelo de filas

A tradução do conteúdo do arquivo IMS (linguagem objeto) para o modelo de filas (linguagem alvo) divide-se em três etapas:

1. Leitura o modelo icônico baseando-se na formatação dos dados especificada no arquivo DTD;
2. Análise sintática do modelo utilizando bibliotecas de *parsing* de XML para identificar as marcas que caracterizam os ícones e tarefas;
3. Gerar as estruturas de dados que constituem os centros de serviços (rede de filas) e indicar os parâmetros de geração do conjunto de tarefas para o motor de simulação.

A Figura 15 ilustra como o iSPD realiza esse processo, realizando a leitura do IMS e gerando as estruturas que compõe o modelo simulável.

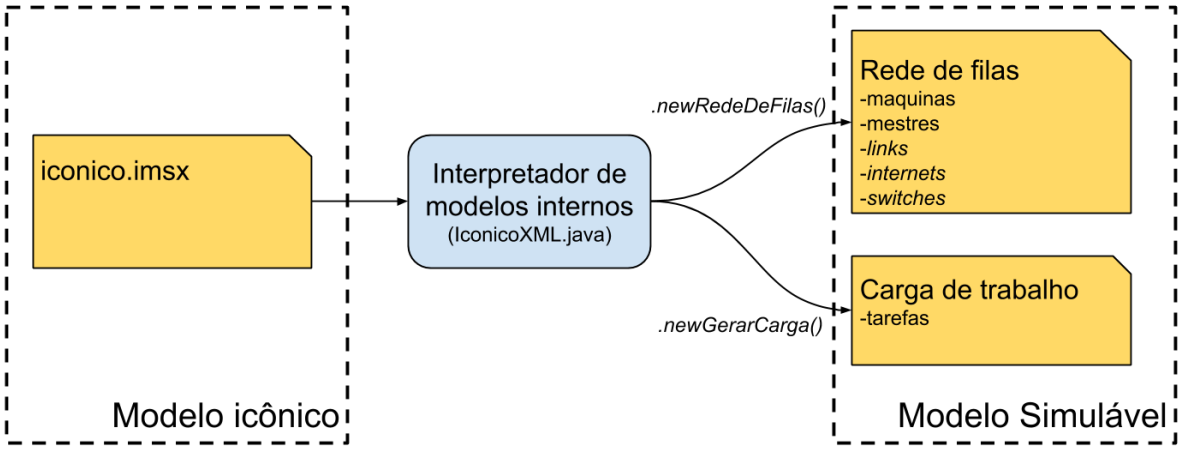


Figura 15 – Processo de conversão entre modelo icônico e modelo simulável

Para realizar a tradução dos ícones do modelo icônico para os centros de serviços correspondentes e a criação da lista de tarefas de acordo com o modo de geração de carga de trabalho indicado necessita-se, como já dito, realizar primeiro o *parsing* da estrutura XML do arquivo IMS. Este processo resulta em uma estrutura de árvore de derivação sintática construída a partir da hierarquia das marcas XML definida pelo arquivo DTD. Assim, o próximo passo na tradução deve identificar nessa árvore os elementos que contêm as marcas que descrevem cada ícone de modelagem e gerar os centros de serviços correspondentes a partir da leitura dos atributos desses ícones. Detalha-se a seguir o processo de identificação e tradução para cada tipo de ícone:

- **Máquina:** Identifica-se na árvore os elementos que possuem a marca "`machine`". A partir deste ponto existem duas abordagens diferentes, caso esse ícone possua ou não o atributo de mestre, indicado pela marca "`master`", como é ilustrado na Figura 16.

Caso o ícone não possua o atributo de mestre, ele será convertido como um centro de serviço de máquina. Por outro lado, se o ícone possui o atributo de mestre, ele será convertido para um centro de serviço do tipo mestre e posteriormente será adicionada em sua estrutura uma lista de escravos, formada pelos centros de serviço do tipo máquina que ele coordena;

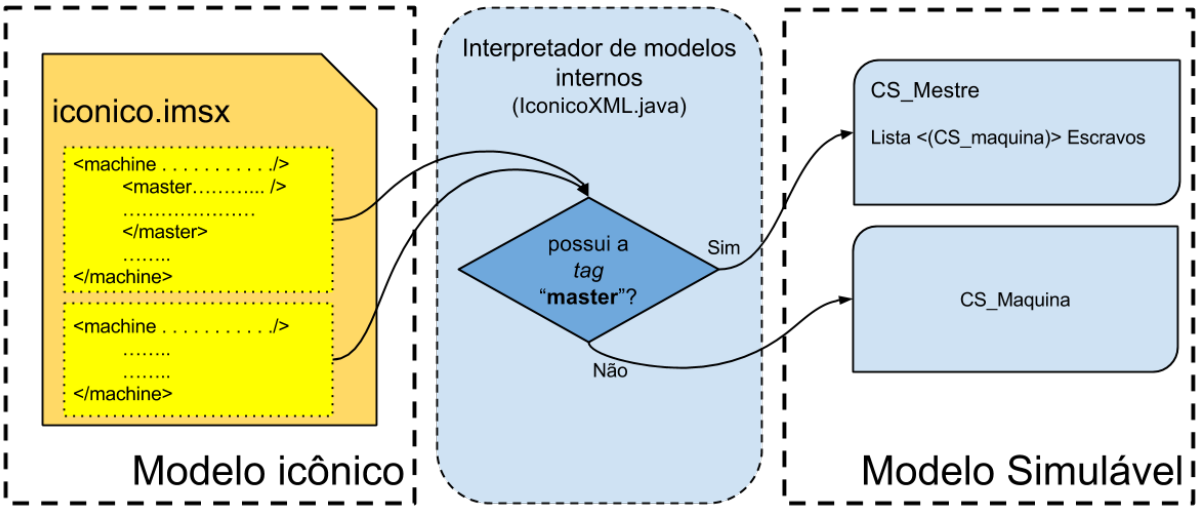


Figura 16 – Processo de conversão do ícone de máquina para os centros de serviço de mestre ou máquina, dependendo da existência do atributo de mestre.

- ***Cluster*:** Identifica-se na árvore os elementos que possuem a marca "`cluster`". A partir disso o ícone de *cluster* é traduzido para uma rede de filas correspondente, conforme ilustrado na Figura 17. Nesse processo inicialmente é gerada a quantidade de objetos de centro de serviço de máquina correspondente ao atributo de número de elementos do *cluster*. Após isso, gera-se um centro de serviço de mestre e adiciona-se em sua lista de escravos os centros de serviço de máquina gerados. Por fim, gera-se um centro de serviço de *switch* utilizando os atributos de banda de passagem e latência de transmissão. O papel do centro de serviço de *switch* é exercer a função de centro de comunicação entre os centros de serviço de máquina (escravos) e de mestre;

- **Enlace:** Identifica-se na árvore os elementos que possuem a marca "`link`". Para cada elemento encontrado gera-se um centro de serviço de enlace lógico, utilizando os atributos de banda de passagem, latência de transmissão e taxa de ocupação da rede;

- ***Internet*:** Identifica-se na árvore os elementos que possuem a marca "`internet`". Para esse elemento gera-se um centro de serviço de *internet*, utilizando os atributos de banda de passagem, latência de transmissão e taxa de ocupação da rede.

Por fim, a tradução de todos estes ícones gera listas de cada tipo de centro de serviço que constituem a estrutura de dados que armazena a rede de filas.

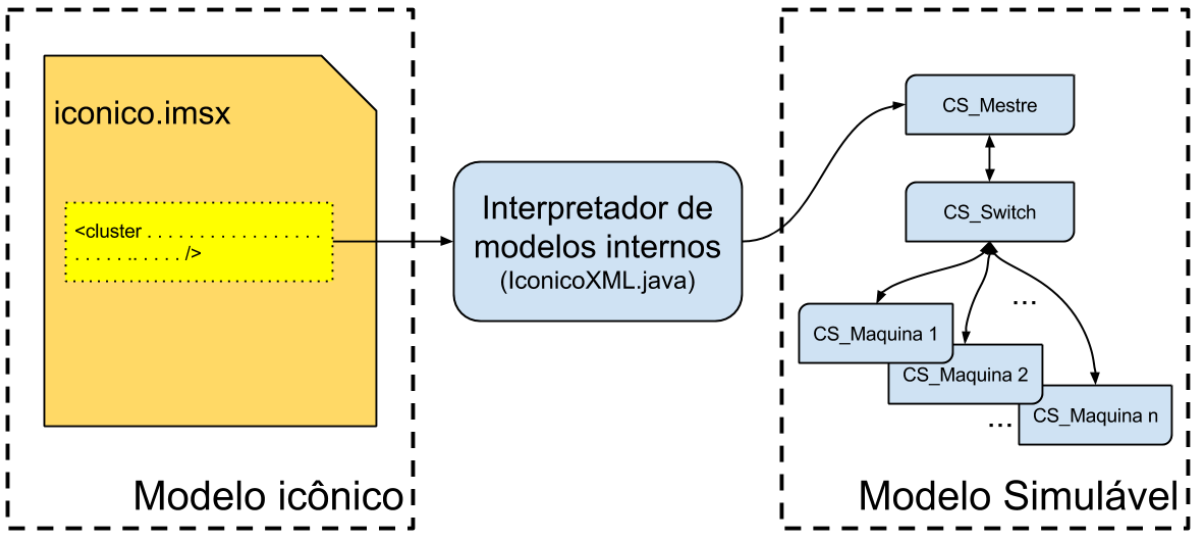


Figura 17 – Processo de conversão do ícone de *cluster* para a rede de filas correspondente

Quanto à tradução da carga de trabalho, o processo realizado é análogo ao ocorrido com os ícones. Deve-se identificar na árvore as marcas referentes ao modo de geração de tarefas selecionado no modelo icônico e a partir da leitura dos atributos, passar os parâmetros para o método do motor de simulação responsável por gerar a lista de tarefas correspondente, como descrito:

- **Carga randômica:** Identifica-se na árvore a marca "`random`". A partir disso, será gerado um número de tarefas equivalente ao atributo de número de tarefas. Quanto às características de cada tarefa, tem-se que o tamanho de processamento e o tamanho de comunicação são gerados a partir dos parâmetros de tamanho máximo, médio e mínimo e probabilidade de computação e comunicação, utilizando um gerador de valores aleatórios baseados na distribuição *Two-Stage Uniform* (LUBLIN; FEITELSON, 2001). Os tempos de chegada das tarefas, por sua vez, são gerados a partir de um gerador de números aleatórios baseado na distribuição exponencial, utilizando como parâmetro o atributo de tempo médio de chegada entre tarefas. Por fim, distribui-se uniformemente as tarefas geradas entre os centros de serviço de mestre;

- **Carga configurada por nó:** Identifica-se na árvore os elementos que contém a marca "`node`". Para cada elemento identificado, gera-se um conjunto de tarefas equivalente ao valor do atributo de número de tarefas, que são caracterizadas como uma aplicação. Para cada aplicação atribui-se o conjunto de tarefas ao centro de serviço de mestre indicado pelo atributo correspondente na modelagem icônica. Já os valores de tempo de chegada das tarefas e tamanhos de processamento e comunicação são gerados de maneira análoga à carga randômica;

- **Carga obtida de *trace*:** Identifica-se na árvore a marca "`trace`". Após isso, toma-se o

atributo de caminho do arquivo busca-se o arquivo de trace do iSPD, cujo formato é o WMS (*Workload Model of Simulation*) (SILVA et al., 2014). A partir da interpretação do arquivo, gera-se uma tarefa para cada linha, utilizando os atributos de identificador da tarefa, *status* de execução, tamanho computacional, tamanho de computação e tempo de chegada lidos a partir do arquivo de *trace*.

3.4 Motor de simulação

Dado que exista uma rede de filas com centros de serviço de mestre interligados aos seus escravos e um conjunto de tarefas, o iSPD pode dar início à simulação. O motor de simulação do iSPD utiliza o algoritmo *Event scheduling/Time-advance* (BANKS et al., 2001), que percorre uma lista de eventos futuros, repetindo o processo de atender o evento atual, atualizar o relógio de simulação e inserir novos eventos encadeados pelo evento atendido, até que a lista se esvazie, como ilustrado na figura 18.

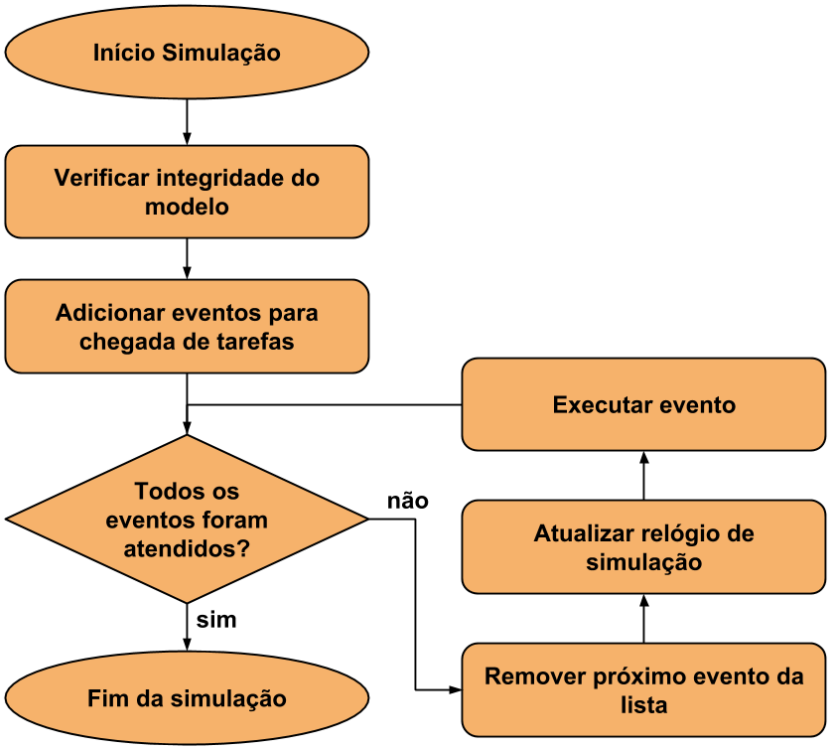


Figura 18 – Fluxograma de funcionamento do algoritmo de simulação de eventos discretos do iSPD (adaptado de (MANACERO et al., 2012))

Para executar o algoritmo *Event Schedule/Time Advance*, além da rede de filas, que contém os centros de serviço em que eventos são executados, e da lista de tarefas, que é utilizada para gerar os eventos de chegada de tarefas que dão início à simulação, o motor ainda necessita de mais duas estruturas, que são o relógio de simulação e a lista de eventos futuros. O relógio de simulação consiste em uma variável que armazena o tempo simulado, deste modo, cada vez que

o atendimento de um determinado evento avança o tempo de simulação, esta variável é alterada. Quanto a lista de eventos futuros (LEF), trata-se de uma estrutura de fila com prioridade, em que eventos da simulação são ordenados segundo o seu instante de ocorrência.

Quanto à estrutura dos eventos de simulação, cada elemento que representa um evento no iSPD é organizado por meio dos seguintes atributos:

- **Instante de ocorrência:** Utilizado para atualizar o relógio de simulação;
- **Centro de serviço de atendimento:** Determina qual centro de serviço deve executar o evento;
- **Cliente do evento:** Tarefa ou mensagem de controle que deu origem ao evento;
- **Tipo de evento:** Determina qual tipo de tratamento deve ser dado no centro de serviço que o executa (por meio dos métodos responsáveis por executar a chegada, atendimento, saída ou atendimento de requisição dos eventos).

Existem cinco tipos de eventos no iSPD:

- **Chegada:** Evento de chegada em um determinado centro de serviço que pode gerar um evento de atendimento ou um evento de chegada para outro centro de serviço, caso o evento não deva ser atendido no centro de serviço atual;
- **Atendimento:** Evento de atendimento em um determinado centro de serviço que dispara cálculos de tempo processamento ou comunicação e gera um evento de saída para uma tarefa ou um evento de mensagem para uma mensagem de controle;
- **Saída:** Evento de saída em um determinado centro de serviço que dispara cálculos de métricas de desempenho e gera possivelmente um evento de chegada para outro centro de serviço;
- **Escalonar:** Evento de escalonamento que ocorre em centros de serviço de mestre. Este evento dispara o processo de escalonamento que seleciona a próxima tarefa a ser executada e em qual centro de serviço o processamento ocorrerá, gerando um evento de saída no centro de serviço de mestre atual;
- **Mensagem e Saída de mensagem:** Evento de mensagem de controle que dispara eventos de chegada para centros de serviço que estão no caminho da mensagem ou evento de atendimento de requisição no centro de serviço que é destino da mensagem.

Dado que todas as estruturas necessárias para executar a simulação (rede de filas, tarefas, lista de eventos futuros e relógio de simulação) foram apresentadas, apresenta-se na próxima seção uma descrição detalhada do processo de simulação implementado pelo motor do iSPD.

3.4.1 Análise funcional do processo de simulação

O processo de simulação pode ser analisado não apenas do ponto de vista da simulação de eventos discretos como também do ponto de vista funcional, de modo a observar como acontecem as interações entre os centros de serviços que compõe rede de filas. Deste modo, apresenta-se na Figura 19 um diagrama que resume as interações entre os centros de serviço durante a simulação no iSPD. Essas interações ocorrem da seguinte maneira:

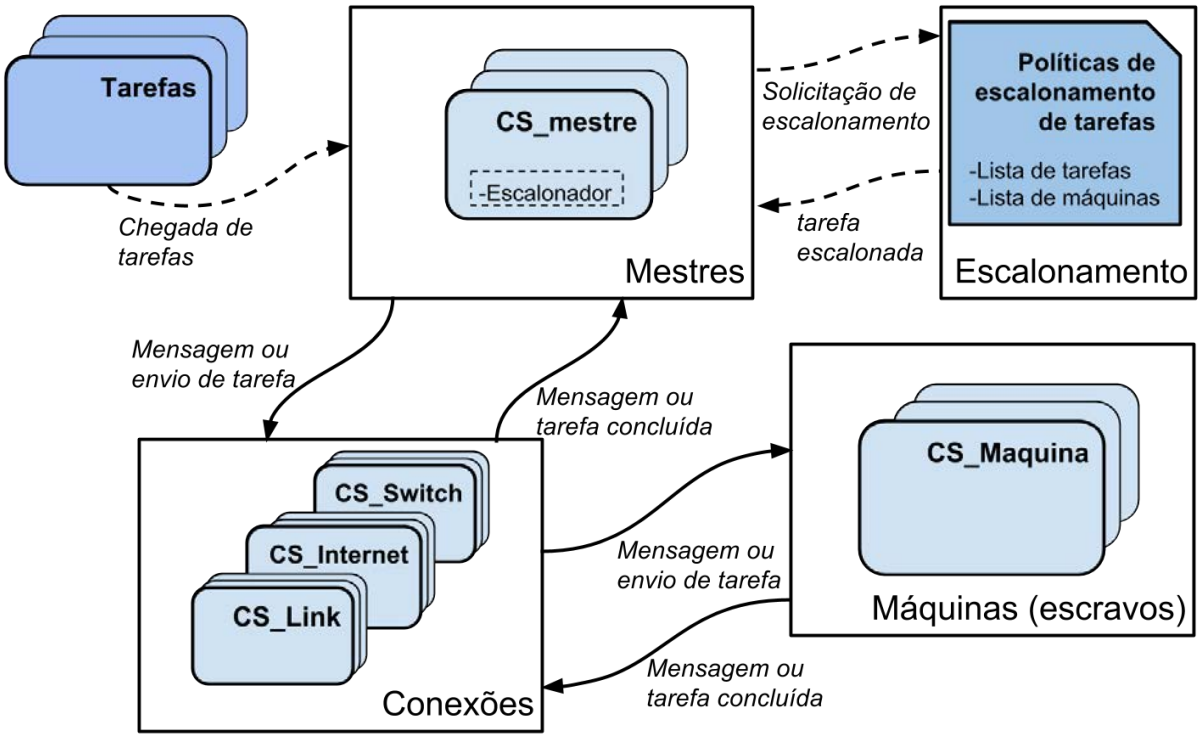


Figura 19 – Interações realizadas entre os centros de serviço durante o processo de simulação

1. Primeiramente as tarefas chegam aos centros de serviço de mestre que, por sua vez, executam o processo de escalonamento e enviam a tarefa selecionada para o centro de serviço que foi escalonado. Caso o centro de serviço selecionado seja outro mestre, a tarefa será reescalonada, caso seja um centro de serviço de máquina, ela será executada;

2. No caminho entre o mestre e o escravo a tarefa percorre os centros de serviço de comunicação, computando neste processo os tempos de espera nas filas, transmissão da tarefa e os atrasos de latência de transmissão no relógio de simulação.

3. Quando a tarefa chega ao centro de serviço de máquina que é seu destino, computam-se no relógio de simulação os tempos de espera na fila e de processamento da tarefa, calculando-se as métricas referentes ao processamento.

4. Por fim, o centro de serviço de máquina envia a tarefa concluída de volta para o mestre que marca a tarefa com o *status* de concluída.

Destaca-se que estas interações se repetem para todas as tarefas, até que todas cheguem ao estado de concluída ou cancelada, encerrando assim o processo de simulação.

3.5 Considerações finais

Apresentou-se neste capítulo os principais detalhes da estrutura do iSPD, de modo a compreender melhor o funcionamento do simulador, desde o processo de modelagem icônica, passando pelo processo de tradução de um modelo icônico para o modelo de filas e até o processo de simulação. Com isso é possível compreender de forma mais profunda os módulos do iSPD em que devem ocorrer as principais alterações para permitir a modelagem e simulação de computação em nuvem. Tais alterações são apresentadas de forma detalhada no próximo capítulo.

4 Modelagem icônica de computação em nuvem

O processo de simulação de computação em nuvem envolve características distintas das presentes no iSPD. Assim, durante este capítulo serão apresentadas as especificações da modelagem de computação em nuvem para o iSPD, bem como toda a implementação da simulação da classe de serviço de IaaS (*Infrastructure-as-a-Service*), que foi a classe de serviço de nuvem selecionada para ser desenvolvida neste trabalho.

4.1 Simulação de computação em nuvem com o iSPD

Como descrito na Seção 2.5, os serviços oferecidos pelo modelo de computação em nuvem são tradicionalmente divididos em três classes: serviços que oferecem infraestrutura de computação (IaaS), serviços que oferecem plataformas de desenvolvimento e de aplicações (PaaS) e serviços que oferecem *softwares* por meio da *web* (SaaS). Foi dito ainda, na Seção 2.7, que os principais simuladores disponíveis na literatura simulam apenas a classe de serviço de IaaS, exigindo ainda que os usuários possuam conhecimentos prévios em linguagens de programação ou *frameworks* de desenvolvimento.

Considerando as funcionalidades presentes no iSPD é possível especificar que as classes IaaS e PaaS apresentam características acrescentáveis ao modelo icônico. Os principais componentes a serem inseridos no simulador, para cada uma dessas classes de serviço são:

- **IaaS:** Devem ser acrescentados elementos para modelar máquinas virtuais (VM), gerenciador de máquinas virtuais (VMM), políticas de alocação de VMs, custos de utilização e disponibilidade de recursos;
- **PaaS:** Devem ser acrescentados elementos para definir um acordo de nível de serviço (*Service Level Agreement* - SLA) e um maior detalhamento da carga de trabalho, que deve indicar quais aplicações compõem a carga.

A fim de ilustrar a especificação das classes de serviço de computação em nuvem discutidas, apresenta-se na Figura 20 um diagrama de casos de uso da ferramenta iSPD destacando, em tom mais escuro, os casos de uso especificados ou desenvolvidos por este trabalho.

Primeiramente o usuário interage com o caso de uso de modelagem icônica que pode estender o caso de uso de configurar o conjunto de VMs, caso o usuário esteja modelando o serviço de IaaS. A modelagem icônica também deve incluir o caso de uso de configuração da

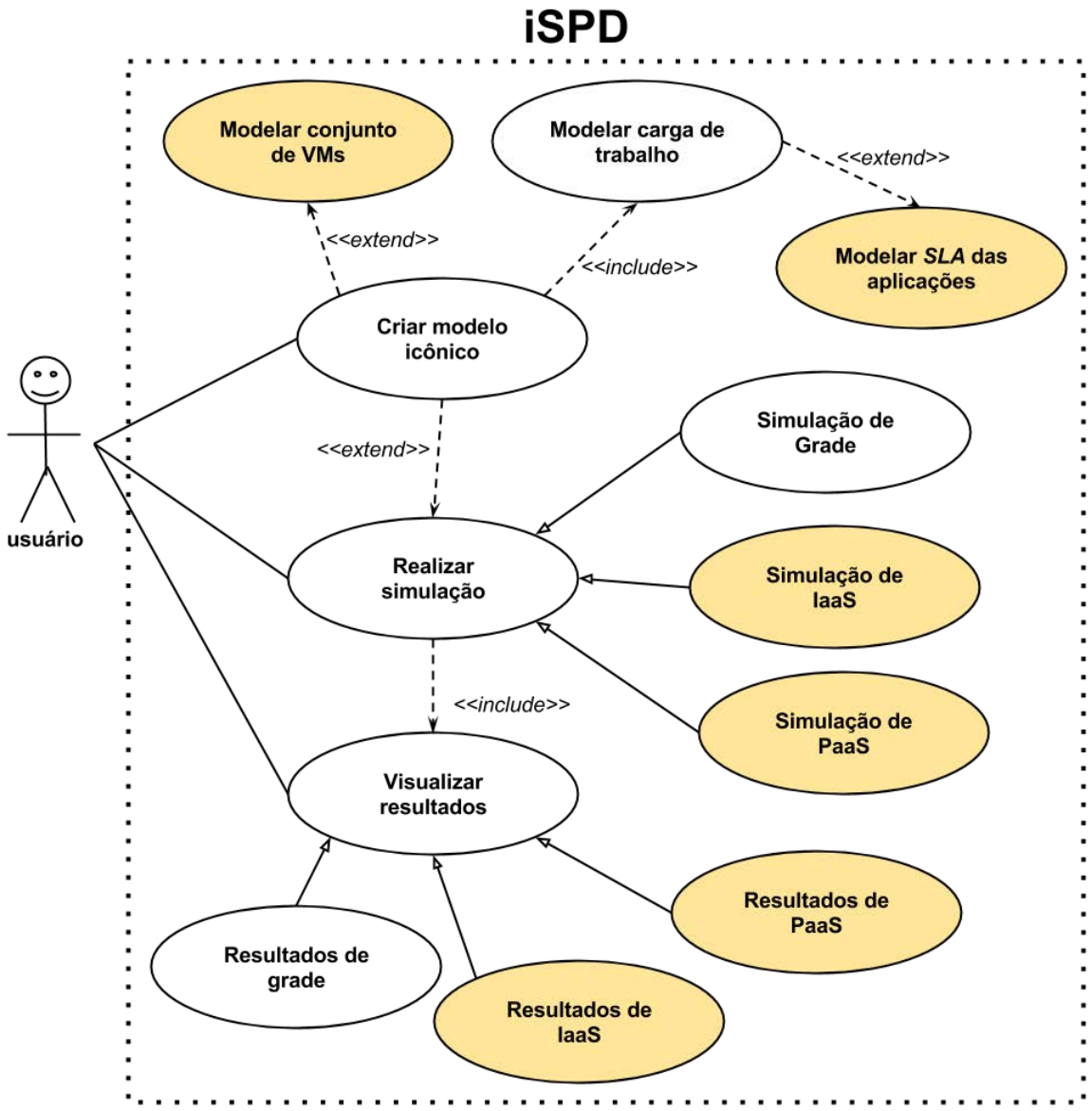


Figura 20 – Diagrama de casos de uso do iSPD, incluindo os novos casos decorrentes da modelagem de computação em nuvem

carga de trabalho, que por sua vez, pode estender a modelagem das SLAs de cada aplicação caso o usuário esteja modelando o serviço de PaaS.

Após a modelagem icônica, o usuário pode disparar o caso de uso de simulação do ambiente modelado. O caso de uso de simulação pode se especializar em três casos: simulação de ambiente de grade, já existente, simulação de IaaS, proposta neste trabalho, e simulação de PaaS, ainda em desenvolvimento.

Por fim, o caso de uso de simulação inclui o caso de uso de visualização dos resultados da simulação que se especializa em três casos de uso: a exibição dos resultados referentes a simulação de grades, de IaaS e de PaaS.

4.2 Caracterização dos recursos e da carga de trabalho para computação em nuvem

As principais diferenças entre computação em nuvem e computação em grade envolvem a definição de recursos e de carga de trabalho. No caso de IaaS a diferença de recursos está na caracterização de máquinas virtuais. Já quanto à carga de trabalho, ambientes IaaS não são diferenciáveis de ambientes de grade, o que implica em se poder utilizar a mesma carga nos dois casos.

Para a caracterização dos recursos buscou-se manter os ícones já existentes no simulador, apenas ampliando os atributos que os caracterizam. Para o suporte à virtualização criou-se o elemento de máquina virtual. Desta forma, a caracterização estendida dos recursos abrange os seguintes parâmetros:

- **Máquina física** - Caracterizada por meio das seguintes informações:
  - Usuário proprietário;
  - Número de núcleos de processamento, para simular a alocação de núcleos de processamento por máquinas virtuais;
  - Poder computacional por núcleo (em MFlops);
  - Quantidades de memória primária (em MB) e memória secundária (em GB), para caracterizar o espaço disponível (memória e disco) para alocar uma VM;
  - Custos de uso de processamento (em $/núcleos/hora), armazenamento primário (em $/Mb/hora) e armazenamento secundário ($/GB/hora), para gerar estimativas de custo de uso de um ou mais recursos em uma simulação;
  - Carga extra-nuvem (em %), que indica a porcentagem de computação utilizada por outros processos não ligados ao processamento da nuvem, como por exemplo gerenciamento de sistema operacional, rede ou processos locais;
  - Informação se o recurso é um elemento de processamento ou um VMM, sendo necessário especificar, caso seja um VMM, a lista de máquinas físicas coordenadas, a política de alocação de máquinas virtuais e a política de escalonamento de tarefas.
- ***Cluster*** - Caracteriza-se este recurso por meio das seguintes informações:
  - Número de máquinas físicas que compõe o *cluster*;
  - Conjunto de atributos que caracterizam cada máquina, como descritos pelo ícone de máquina física;
  - Banda de passagem (em Mbps) e latência de transmissão (em segundos), que caracterizam as ligações entre as máquinas que compõem o *cluster*;

  - Política de alocação de máquinas virtuais e política de escalonamento de tarefas do VMM que coordena o *cluster*.

- **Enlace e *internet*:** Manteve-se a modelagem original destes itens, descrita na Seção 3.3.1;
- **Máquina Virtual (VM):** Caracteriza-se este recurso por meio das informações:
  - Usuário contratante;
  - VMM gerenciador;
  - Número de núcleos de processamento necessários;
  - Quantidade de memória necessária (em MB);
  - Quantidade de disco (memória secundária) necessário (em GB);
  - Informação sobre o SO instalado na VM, incluído para a posterior implementação de PaaS, que utilizará esta informação.

4.3 Desenvolvimento da modelagem de IaaS no módulo de interface icônica

Como descrito na Seção 3.2, a interface icônica é o principal módulo de interação do usuário com o iSPD. Por meio dela o usuário pode modelar, utilizando ícones, o ambiente computacional que deseja simular, assim como acessar as interfaces de configuração da carga de trabalho e do conjunto de usuários. Deste modo, para permitir a modelagem de IaaS no iSPD, segundo a abordagem icônica existente, realizou-se uma série de alterações, estendo-se os atributos de configuração dos ícones de máquina e *cluster* e incluindo uma nova interface utilizada para a configuração das máquinas virtuais.

Primeiramente, como foi necessário incluir dois novos modos de modelagem (modelagem de IaaS e de PaaS) mantendo-se ainda a opção de modelagem para computação em grade, desenvolveu-se um mecanismo para selecionar o tipo de serviço a ser modelado, condicionando a visualização da interface gráfica de acordo com a opção selecionada. Deste modo, apresenta-se na Figura 21 a interface desenvolvida para seleção do serviço a ser simulado pelo iSPD. A escolha por "Cloud-IaaS" dispara novas janelas de configuração de elementos da nuvem.

4.3.1 Desenvolvimento das entradas de dados para a configuração dos ícones de processamento para IaaS

Como discutido na Seção 4.2, expandiu-se a caracterização dos ícones de máquina e *cluster* para incorporar atributos referentes à modelagem de computação em nuvem. Desta forma, desenvolveram-se novas tabelas de configuração de atributos para esses ícones as quais devem ser exibidas na área de configuração da interface icônica.

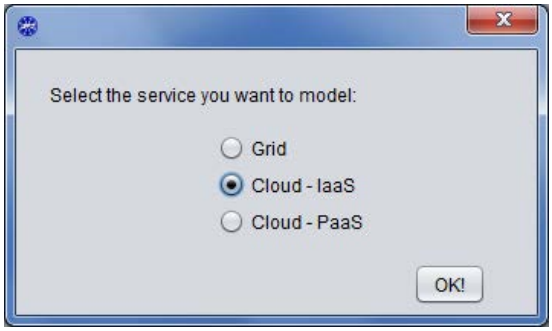
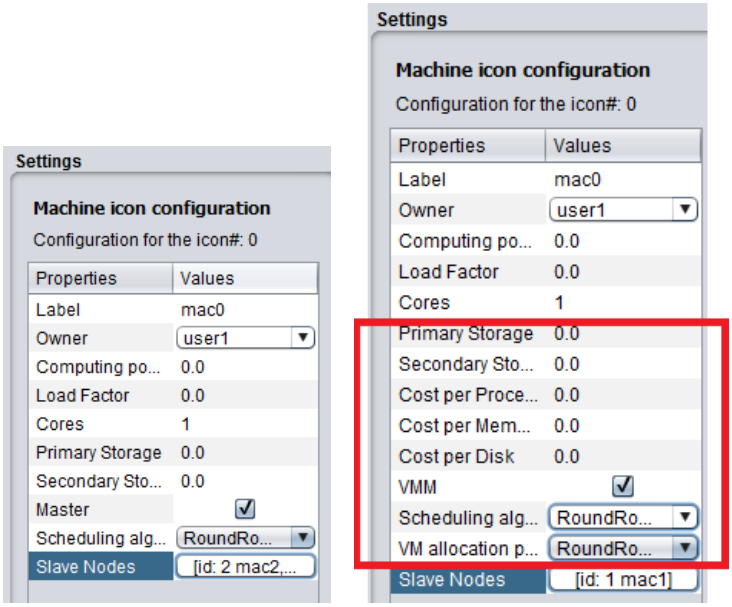


Figura 21 – Interface de seleção do serviço a ser simulado.

Deste modo, tanto para o ícone de máquina como para o de *cluster* foram adicionados os atributos referentes às quantidades de memórias primária (memória RAM) e secundária (disco rígido), custos de utilização de processamento, memória e disco e informação se o ícone é um VMM ou não, além dos atributos de políticas de alocação de VMs e escalonamento de tarefas selecionadas, caso o ícone seja um VMM.



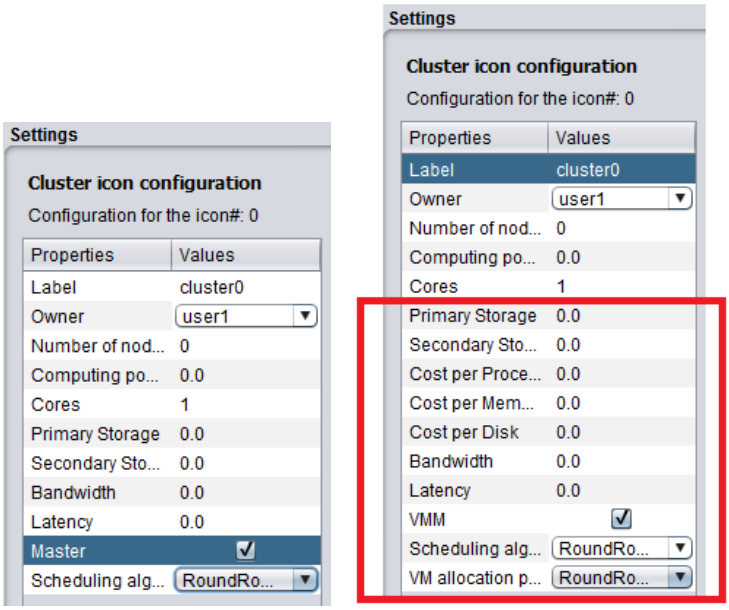
(a) Modelagem de grade (b) Modelagem de *cloud*

Figura 22 – Painel de configuração do ícone de máquina (a) para a modelagem de grades computacionais e (b) para a modelagem de computação em nuvem

A informação sobre um ícone de máquina ou *cluster* realizar o papel de VMM foi adaptada a partir da ideia do ícone exercer o papel de mestre, já existente na modelagem de computação em grade. Desta maneira, durante a configuração dos atributos, ao invés de escolher se o ícone é mestre de um conjunto de elementos da grade, seleciona-se se o ícone é um VMM que coordena um conjunto de elementos da infraestrutura de nuvem modelada.

As Figuras 22 e 23 apresentam os painéis de configuração dos ícones de máquina e *cluster* respectivamente, exibindo as diferenças entre a modelagem de grade computacional e a

modelagem de computação em nuvem, cujos atributos adicionais estão destacados dentro do retângulo.



(a) Modelagem de grade (b) Modelagem de *cloud*

Figura 23 – Painel de configuração do ícone de *cluster* (a) para a modelagem de grades computacionais e (b) para a modelagem de computação em nuvem

4.3.2 Interface de configuração de máquinas virtuais

Como a característica de virtualização não era tratada no iSPD, desenvolveu-se uma interface de modelagem desta característica fundamental de IaaS, apresentada na Figura 24. Essa interface é acionada a partir de um novo botão inserido na aba de configurações complementares. Para realizar a modelagem de uma máquina virtual, o usuário deve inserir os valores de cada um dos atributos referentes à VM nos campos indicados na figura, selecionando um usuário contratante e VMM responsável pelo gerenciamento, além de especificação das quantidades de recursos (núcleos de processamento, memória RAM e disco) necessários para a alocação da VM modelada.

Após o preenchimento de todas as características seleciona-se o botão "*Add VM*" para adicionar a VM modelada na lista de VMs configuradas, posicionada na parte inferior da interface. Para remover uma VM seleciona-se a linha correspondente na lista de VMs e em seguida aciona-se o botão "*Remove VM*". Por fim, para efetivar as configurações realizadas, seleciona-se o botão "OK!". Esta operação irá gerar a lista de máquina virtuais que será transcrita para o arquivo de modelo icônico, caso ele seja salvo.

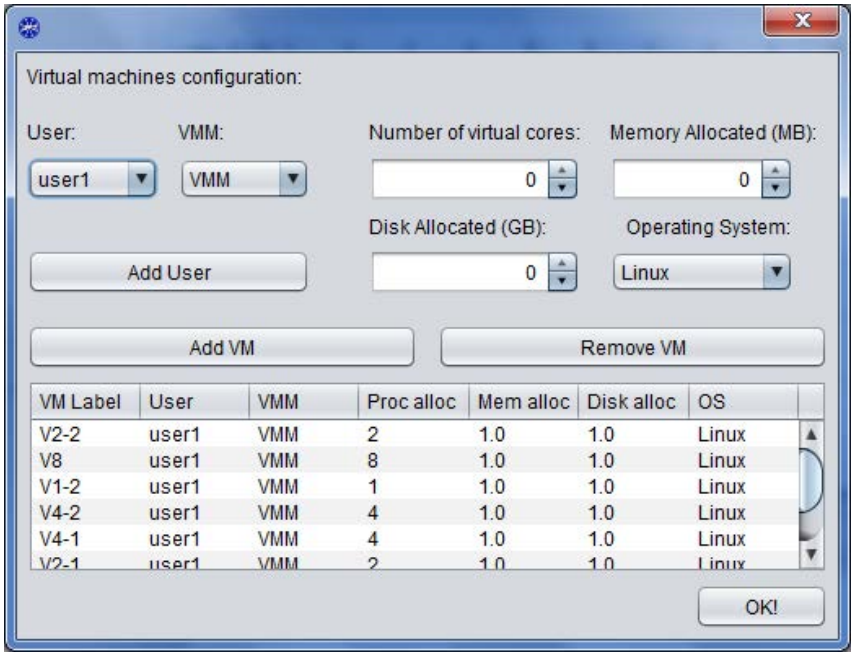


Figura 24 – Janela de configuração de VMs

4.4 Desenvolvimento da modelagem de IaaS no módulo interpretador de modelos internos

Como discutido na Seção 3.3, o papel do interpretador de modelos internos é realizar a tradução do modelo icônico para as estruturas de dados que compõe o modelo de filas. Descreve-se nesta seção o desenvolvimento das estruturas dos modelos icônico e simulável e do processo de tradução entre eles para permitir a simulação de IaaS.

4.4.1 Desenvolvimento do modelo icônico para IaaS

Como descrito na Seção 3.3.1, o modelo icônico é um arquivo XML cujo formato de dados é definido por um arquivo DTD chamado "`ispd.dtd`". A função do modelo icônico é armazenar, por meio de marca XML, toda informação referente aos elementos que compõem o ambiente computacional, assim como os parâmetros de geração da carga de trabalho.

Para desenvolver o trabalho proposto foi necessário alterar o arquivo "`ispd.dtd`" de modo a definir o padrão de dados utilizado para incluir as informações referentes à modelagem de IaaS. Destaca-se para esta finalidade a inclusão das estruturas necessárias para estender as características de máquina e *cluster*, além do desenvolvimento do formato de dados utilizado para armazenar as informações sobre máquinas virtuais.

Para realizar a inclusão da caracterização expandida dos ícones máquina e *cluster*, elaborou-se um elemento XML representado pela marca "`characteristic`", que constitui-se do conjunto de características de processamento, memória, disco e custos de utilização

referentes a um ícone, como descrito a seguir:

- **Processamento:** Identificado como "`process`", sendo caracterizado pelos atributos "`power`", que define o poder computacional em MFlops por núcleo de processamento, "`number`", que contém informação sobre o número de núcleos de processamento, e "`model`", que possui informação sobre a arquitetura do processador;
- **Memória:** Especificada pelo elemento "`memory`" e caracterizada pelo atributo "`size`", que apresenta a quantidade de memória primária disponível em MB;
- **Disco:** Especificado pelo elemento "`hard_disk`", por meio do atributo "`size`", que apresenta a quantidade de disco (memória secundaria) disponível em GB;
- **Custo:** Especificado pelo elemento "`cost`" por meio dos atributos "`cost_proc`" que armazena a informação sobre o custo de processamento, "`cost_mem`" que armazena informação sobre o custo de uso de memória e "`cost_disk`" que armazena informação sobre o custo de uso de disco.

Apresenta-se na Figura 25 o trecho do arquivo DTD que define a estrutura do elemento "`characteristic`", assim como os elementos que o constituem.

```
...
<!ELEMENT characteristic (process,memory,hard_disk,cost?)>

<!ELEMENT cost EMPTY>
    <!ATTLIST cost cost_proc CDATA "0.0">
    <!ATTLIST cost cost_mem CDATA "0.0">
    <!ATTLIST cost cost_disk CDATA "0.0">

<!ELEMENT process EMPTY>
    <!ATTLIST process power CDATA "0.0">
    <!ATTLIST process number CDATA "1">
    <!ATTLIST process model CDATA "x86">

<!ELEMENT memory EMPTY>
    <!ATTLIST memory size CDATA "0.0">

<!ELEMENT hard_disk EMPTY>
    <!ATTLIST hard_disk size CDATA "0.0">
...
```

Figura 25 – Trecho do DTD que define a caracterização das quantidades de recursos e custos de utilização para os ícones de máquina e *cluster*

Quanto à informação sobre um ícone ser VMM, adaptou-se para a representação desta característica a marca "`master`", existente no DTD original, que armazenava a política de

escalonamento por meio do atributo "`scheduler`" e a lista de escravos por meio de um ou mais elementos "`slave`". Deste modo, utiliza-se esta marca XML para armazenar os elementos de VMM, bastando incluir a informação sobre a política de alocação das máquinas virtuais, por meio da criação do atributo "`vm_alloc`".

Para o armazenamento da informação das máquinas virtuais, dado que não existia nenhuma estrutura que pudesse ser adaptada para esta finalidade, criou-se a marca "`virtualMac`", constituída pelos seguintes atributos:

- **Identificador:** Representado pela marca "`id`", que contém um identificador para o elemento;
- **Usuário:** Representado pela marca "`owner`", que contém o identificador do usuário contratante;
- **VMM coordenador:** Representado pela marca "`vmm`", que contém o identificador do VMM que coordena a VM;
- **Poder computacional:** Representado pela marca "`power`", que contém o número de núcleos de processamento alocados;
- **Memória alocada:** Representado pela marca "`mem_alloc`", que contém a quantidade de memória primária alocada em GB;
- **Disco alocado:** Representado pela marca "`disk_alloc`", que contém a quantidade de disco reservada para a VM em GB;
- **Sistema Operacional:** Representado pela marca "`op_system`", que contém informação sobre o sistema operacional instalado na VM.

Apresenta-se na Figura 26 o trecho do DTD que especifica a estrutura dos elementos de VM no iSPD.

4.4.2 Desenvolvimento do modelo de filas para simulação de IaaS

Devido a mudança de tipo de sistema modelado implementaram-se novos centros de serviço de processamento. Desta forma, primeiramente implementou-se o elemento de máquina virtual. O grande desafio de modelagem desse elemento está no fato de que durante o processo de simulação, a máquina virtual pode exercer duas funções diferentes (cliente das filas ou centro de serviço) caso ela esteja alocada em uma máquina física ou não. Deste modo, definiu-se um conjunto de estados em que a máquina virtual pode estar, o que inclui:

- **Livre:** neste estado a máquina virtual não se encontra alocada e comporta-se como um cliente da rede de filas. Desde que uma VM esteja livre, deverá ser submetida à política

```
...
<!ELEMENT virtualMac EMPTY>
    <!ATTLIST virtualMac id ID #REQUIRED>
    <!ATTLIST virtualMac owner CDATA "0.0">
    <!ATTLIST virtualMac vmm CDATA "0.0">
    <!ATTLIST virtualMac power CDATA "0.0">
    <!ATTLIST virtualMac mem_alloc CDATA "0.0">
    <!ATTLIST virtualMac disk_alloc CDATA "0.0">
    <!ATTLIST virtualMac op_system CDATA "0.0" >
...
```

Figura 26 – Trecho do DTD que define a caracterização das máquinas virtuais no iSPD

de alocação do VMM responsável e caso seja selecionada deverá percorrer a rede de filas até chegar ao centro de serviço de máquina onde deve ser alocada, passando assim para o estado de alocada;

- **Alocada:** neste estado a máquina virtual encontra-se alocada em uma máquina física e é anexada a rede de filas, passando a se comportar como um centro de serviço de processamento que segue o modelo de filas M/M/n (uma fila, *n* servidores), em que o valor de *n* é definido pelo número de núcleos de processamento virtuais que possui. Neste cenário, a função deste centro de serviço consiste em simular a execução das tarefas na nuvem, computando as métricas de processamento de maneira análoga ao centro de serviço de máquina utilizado para simulação de grades computacionais;

- **Rejeitada:** a máquina virtual passa para este estado quando não é possível alocá-la pela falta de recursos físicos suficientes nas máquinas físicas. Neste caso, a VM não exerce nenhuma função e apenas fica disponível para calcular métricas relativas à alocação, ao fim da simulação;

- **Destruída:** Este estado é alcançado apenas por máquinas virtuais que tenham sido alocadas. Se uma máquina virtual atuou como centro de processamento durante a simulação, ao fim deste processo o *status* da VM é alterado para este estado. Sua função é realizar uma transição de estados na VM de modo a permitir calcular os custos de utilização de uma VM em função do tempo em que a máquina virtual permaneceu alocada.

Para a simulação da infraestrutura física de processamento, desenvolveram-se os centros de serviço de VMM e máquina física. A necessidade de implementação destes centros de serviço surgiu pois os centros de serviço de processamento de grade computacional (máquina e mestre) não mimetizam corretamente o processo de execução de tarefas em IaaS. Deste modo, descrevem-se os centros de serviço de VMM e máquina física, apresentando caraterísticas de sua implementação e detalhes de seu funcionamento:

- **VMM:** Este centro de serviço segue o modelo de filas M/M/1 (“uma fila, um servidor”) e exerce o papel de gerenciador das máquinas virtuais hospedadas por um determinado conjunto de máquinas físicas que são coordenadas por este elemento. Dentre as funções deste centro de serviço está receber as máquinas virtuais e realizar a alocação delas entre as máquinas físicas, além de receber as tarefas submetidas e realizar o escalonamento de tarefas seguindo as políticas de alocação e escalonamento especificadas pelo usuário. É papel deste centro de serviço, ainda, contabilizar métricas referentes às máquinas virtuais rejeitadas no processo de alocação e ao conjunto de tarefas concluídas;

- **Máquina física:** Este centro de serviço segue um modelo de filas M/M/1 (uma fila, um servidor). A função deste centro de serviço baseia-se em ser um hospedeiro para o centro de serviço de máquina virtual, de modo a ceder recursos para as máquinas virtuais que estão alocadas nele e encaminhar as tarefas recebidas para a máquina virtual selecionada para executá-las.

Não houve alterações nos centros de serviço de comunicação (enlace, *switch* e *internet*) dado que eles atendiam as necessidades para a modelagem de IaaS. Deste modo, a especificação destes elementos manteve-se como descrito na Seção 3.3.2.

4.4.3 Tradução do modelo icônico para o modelo de filas para IaaS

Como discutido na Seção 3.3.3, o processo de interpretação de modelos internos consiste em utilizar um *parser* de XML para gerar uma estrutura de árvore a partir da identificação das marcas XML definidas no arquivo DTD de validação no arquivo IMS. Isso significa identificar os elementos que constituem o ambiente icônico e convertendo-os para os centros de serviço que integram o modelo simulável e a lista de tarefas a serem simuladas. Dado que foram definidos novos ícones e novos centros de serviço para a modelagem e simulação de IaaS, foi necessário realizar um conjunto de alterações no processo de tradução. Apresenta-se na Figura 27 um diagrama resumido desse processo, em que as principais alterações estão nos objetos que representem as redes de filas.

Detalha-se a seguir o conjunto de alterações na tradução de cada tipo de ícone que constitui o modelo icônico de computação em nuvem para os centros de serviço correspondentes:

- **Máquina:** Identifica-se na árvore os elementos que possuem a marca “`machine`”. Após a identificação, assim como ocorre no processo de conversão do ícone de máquina para grades computacionais, existem duas abordagens diferentes durante a interpretação, dado que seja identificado no elemento a marca “`master`” ou não, como ilustrado na Figura 28.

  Caso não tenha sido identificada a marca “`master`”, gera-se um centro de serviço de máquina física. Por outro lado, caso seja identificada a marca “`master`” será gerado um

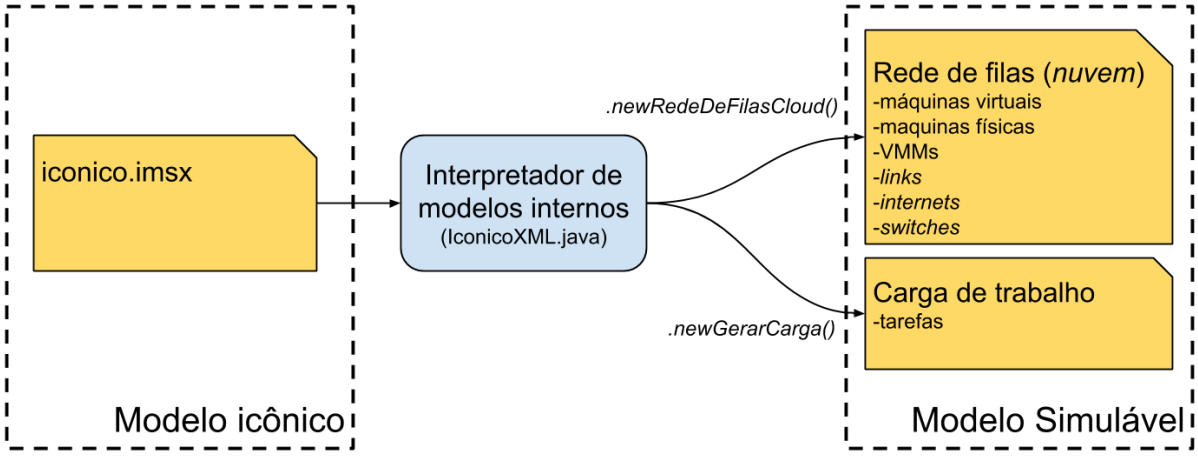


Figura 27 – Processo de conversão entre modelo icônico e simulável para computação em nuvem

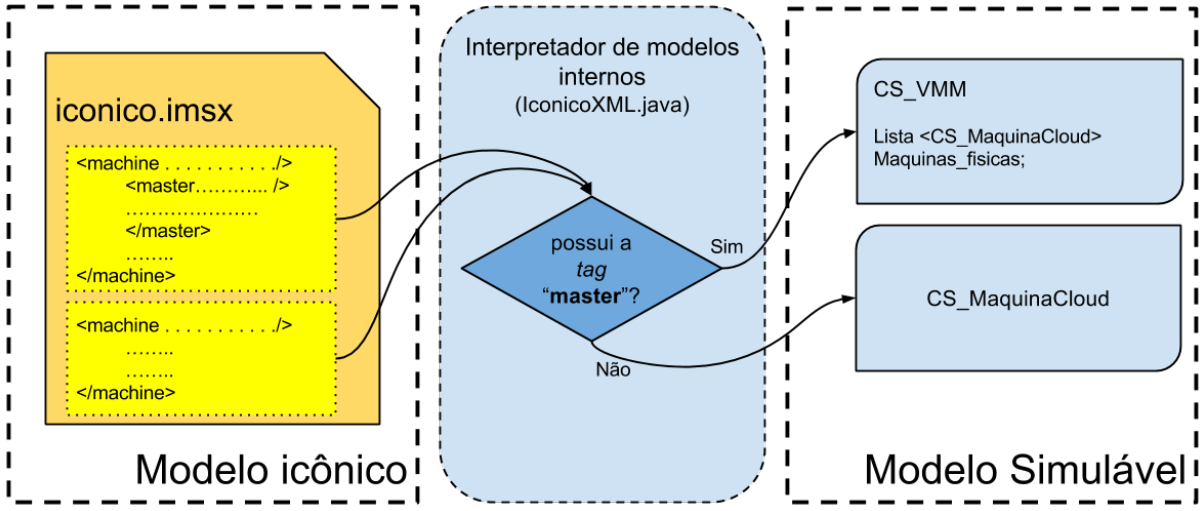


Figura 28 – Processo de conversão do ícone de máquina para os centros de serviço de VMM ou de máquina física

centro de serviço de VMM, adicionando posteriormente à lista de centros de serviço de máquina física coordenados pelo VMM;

- ***Cluster*:** Identifica-se na árvore os elementos que possuem a marca "`cluster`". A partir disso, assim como ocorre para grades computacionais, o ícone de *cluster* é traduzido para uma rede de filas correspondente, conforme ilustrado na Figura 29. Nesse processo inicialmente é gerada a quantidade de objetos de centro de serviço de máquina física correspondente ao atributo de número de elementos do *cluster*. Após isso, gera-se um centro de serviço de VMM e adiciona-se em sua lista de recursos coordenados os centros de serviço de máquina física gerados. Por fim, gera-se um centro de serviço de *switch* utilizando os atributos de banda de passagem e latência de transmissão.

- **Máquina Virtual:** Identifica-se os elementos que possuem a marca "`virtualMac`". A partir desse ponto, gera-se o centro de serviço de máquina virtual a partir da leitura dos

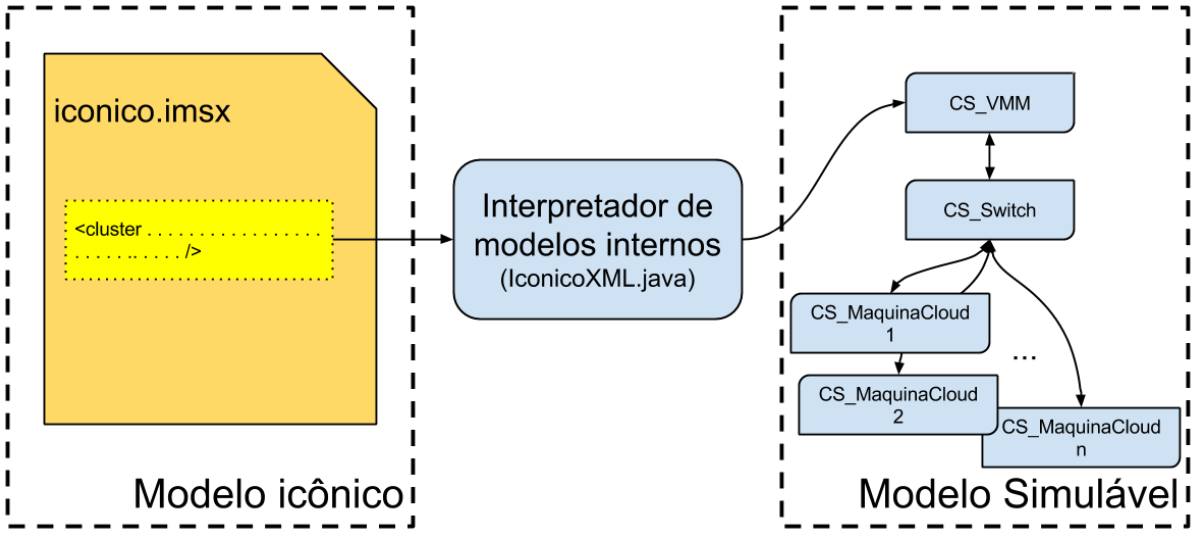


Figura 29 – Processo de conversão do ícone de *cluster* para a rede de filas correspondente

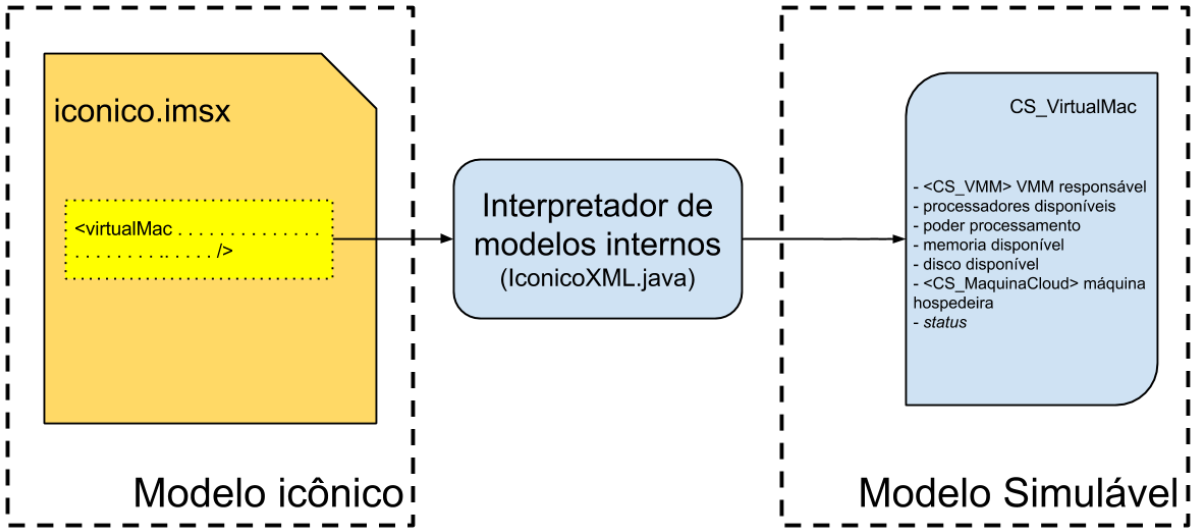


Figura 30 – Processo de conversão do elemento de máquina virtual para o centro de serviço correspondente

atributos do elemento correspondente na árvore, como ilustrado na Figura 30.

- **Enlace e *Internet*:** O processo de conversão destes ícones não sofreu alterações e mantém-se como descrito na Seção 3.3.3

4.5 Políticas de escalonamento de tarefas e de alocação de VMs

Dois componentes importantes na simulação de computação em nuvem são a forma como as máquinas virtuais são alocadas e a forma como as tarefas são escalonadas nessas máquinas virtuais. Considerando a classe de serviços de IaaS o escalonamento é semelhante ao feito para

grades computacionais, enquanto a alocação envolve aspectos não tratados em grades, como adequação de demanda e recursos. Assim temos:

- **Implementação do processo de escalonamento de tarefas em IaaS para o iSPD**

  Houve poucas alterações no modo como as políticas de escalonamento no iSPD para IaaS foram implementadas em relação ao processo de escalonamento de tarefas para grades computacionais. Como discutido na Seção 4.2, para simular a classe de serviço de IaaS, a modelagem da carga de trabalho mantêm-se a mesma, dado que a única diferença é que a tarefas passam a ser a executadas em ambiente virtualizado. Deste modo, a implementação das políticas de escalonamento de tarefas para nuvem alteraram somente os centros de serviço para onde as tarefas são escalonadas, que deixam de ser máquinas físicas e passam a ser os centros de serviço de máquina virtual.

  Para esta versão inicial implementou-se o algoritmo de escalonamento *round-robin*. Este algoritmo busca escalonar as tarefas homogeneamente entre as máquinas virtuais, selecionando qual recurso deve escalonar uma dada tarefa por meio de uma lista circular.

- **Alocação de máquinas virtuais**

  Como discutido na Seção 2.6, o processo de alocação consiste em estimar a quantidade de recursos computacionais (núcleos de processamento, memória RAM e disco) que uma máquina virtual utiliza e então selecionar, segundo algum critério, uma máquina física que possua recursos suficientes para hospedá-la.

  Desta forma, a alocação para o iSPD baseia-se em tomar um centro de serviço de VMM e selecionar um a um os centros de serviço de máquina virtual que ele gerencia. Para cada VM o mecanismo define a máquina física candidata, segundo o critério de alocação adotado, e verifica se ela tem recursos disponíveis para atender a demanda da VM. Caso a máquina física possua recursos disponíveis, dispara-se o processo de alocação e os recursos são reservados para a máquina virtual selecionada. Por outro lado, caso não exista nenhuma máquina física que comporte a VM selecionada, ela passa para o estado de rejeitada.

  Quanto aos algoritmos de alocação desenvolvidos nativamente no iSPD, implementaram-se os algoritmos de alocação *round-robin*, *first-fit*, *first-fit Decreasing* e *Volume*, descritos na Seção 2.6.

4.6 Desenvolvimento do motor de simulação para IaaS

O motor de simulação para IaaS usa a mesma base da simulação de grades. As principais diferenças são a inclusão de um evento representando a alocação de VMs e de mensagens para tratar essa alocação.

4.6.1 Desenvolvimento do processo de simulação de eventos discretos para IaaS

Assim como ocorre para grades computacionais, o processo de simulação desenvolvido para IaaS também possui etapas preparatórias de simulação e duas etapas conclusivas. Apresenta-se na Figura 31 um fluxograma de execução do processo de simulação de eventos discretos para IaaS, cujas etapas são descritas a seguir:

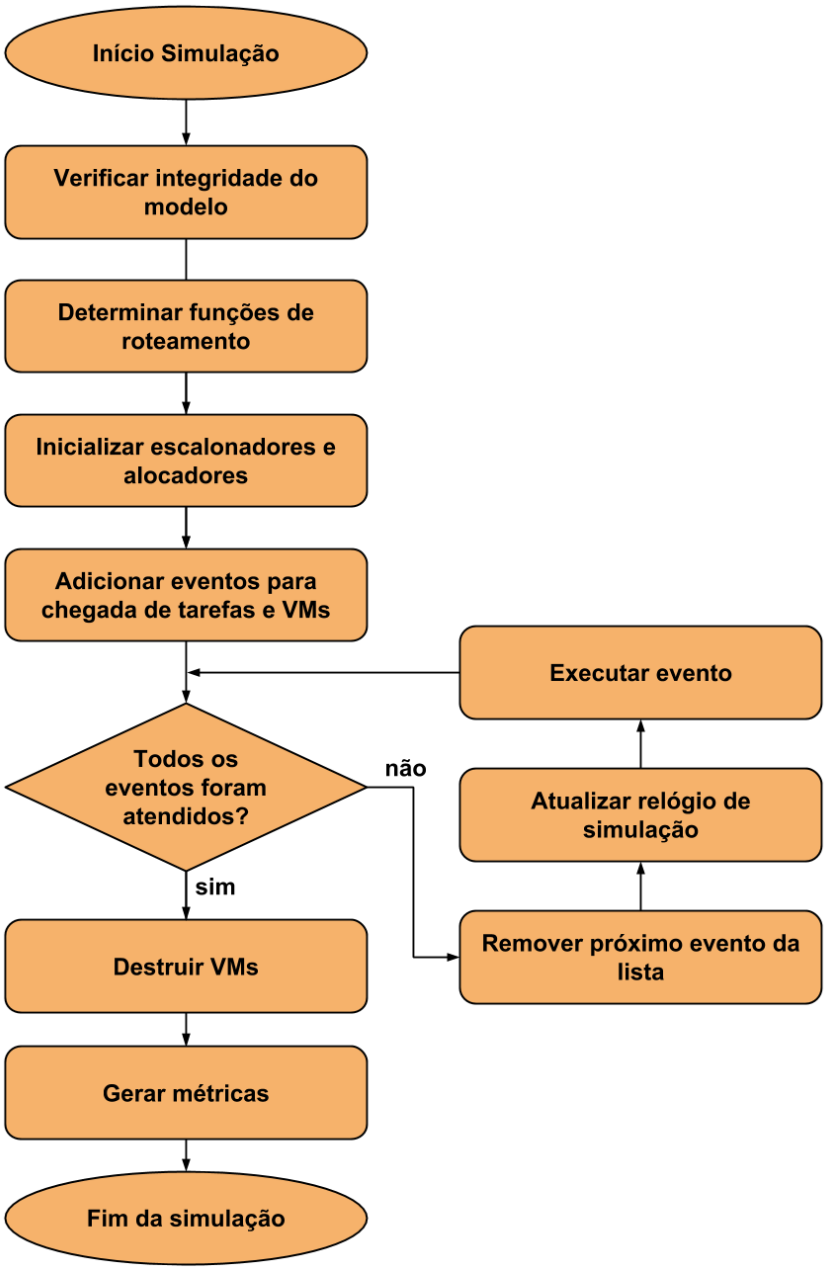


Figura 31 – Fluxograma de execução do processo de simulação de eventos discretos para a classe de serviço de IaaS

1. **Validação do modelo simulável:** Verifica-se se a rede de filas possui VMMs com conexões até os centros de serviço de máquina física que coordena. Verifica-se também se

existem centros de serviço de máquina virtual e se a lista de tarefas não se encontra vazia;

2. **Definição das funções de roteamento:** Utiliza-se algoritmo de Djikstra de caminhos mínimos para calcular e armazenar o caminho de um centro de serviço de VMM para cada uma das máquinas físicas que coordena, assim como para calcular o caminho de cada centro de serviço de máquina física para seu centro de serviço de VMM;

3. **Inicialização dos alocadores e escalonadores:** Percorre-se os centros de serviço de VMM para disparar os métodos responsáveis pela alocação das máquinas virtuais e pelo escalonamento das tarefas;

4. **Geração dos eventos de chegada das tarefas e VMs no sistema:** Percorrem-se as listas de tarefas e máquinas virtuais gerando-se os eventos que preenchem a LEF. Gera-se, para cada tarefa ou VM, um evento de chegada que possui como tempo de ocorrência o tempo de chegada da tarefa ou VM, como centro de serviço o mestre em que a tarefa ou VM foi submetida e como cliente a tarefa ou VM propriamente dita;

5. **Simulação:** Percorre-se a LEF removendo o primeiro evento da lista, atualizando o relógio de simulação para o instante de ocorrência do evento e verifica-se qual o tipo de evento e qual o centro de serviço indicado para atendê-lo. Após essa verificação ocorre a execução do evento e o correspondente cálculo de métricas e eventual geração de novos eventos. Esse processo continua até que a LEF esteja vazia.

6. **Destruição das máquinas virtuais:** Percorre-se os centros de serviço de máquinas físicas, alterando o estado dos centros de serviço de máquina virtual hospedados de "alocada" para "destruída". O objetivo desta transição de estado é calcular o tempo de utilização, pois temporizador é iniciado quando a VM é alocada e é finalizado durante esta etapa. Esta medida é utilizada para obter as métricas de custo de utilização das máquinas virtuais;

7. **Geração de métricas de desempenho:** Percorre-se os centros de serviço coletando todas as métricas de simulação, que são retornadas a interface gráfica, que as usará para a geração de relatórios e gráficos da simulação.

4.6.2 Análise funcional do processo de simulação

Como discutido na Seção 3.4.1, a análise do processo de simulação pode ser realizada não apenas do ponto de vista da simulação de eventos discretos mas também do ponto de vista funcional, de modo a observar como os centros de serviços que integram a rede de filas interagem entre si. Apresenta-se na Figura 32 um diagrama que resume as interações entre os centros de serviço que realizam a simulação de IaaS.

Podem-se dividir as interações que ocorrem entre os centros de serviços em dois processos diferentes, que são a alocação das máquinas virtuais e o escalonamento e execução das tarefas. Ao

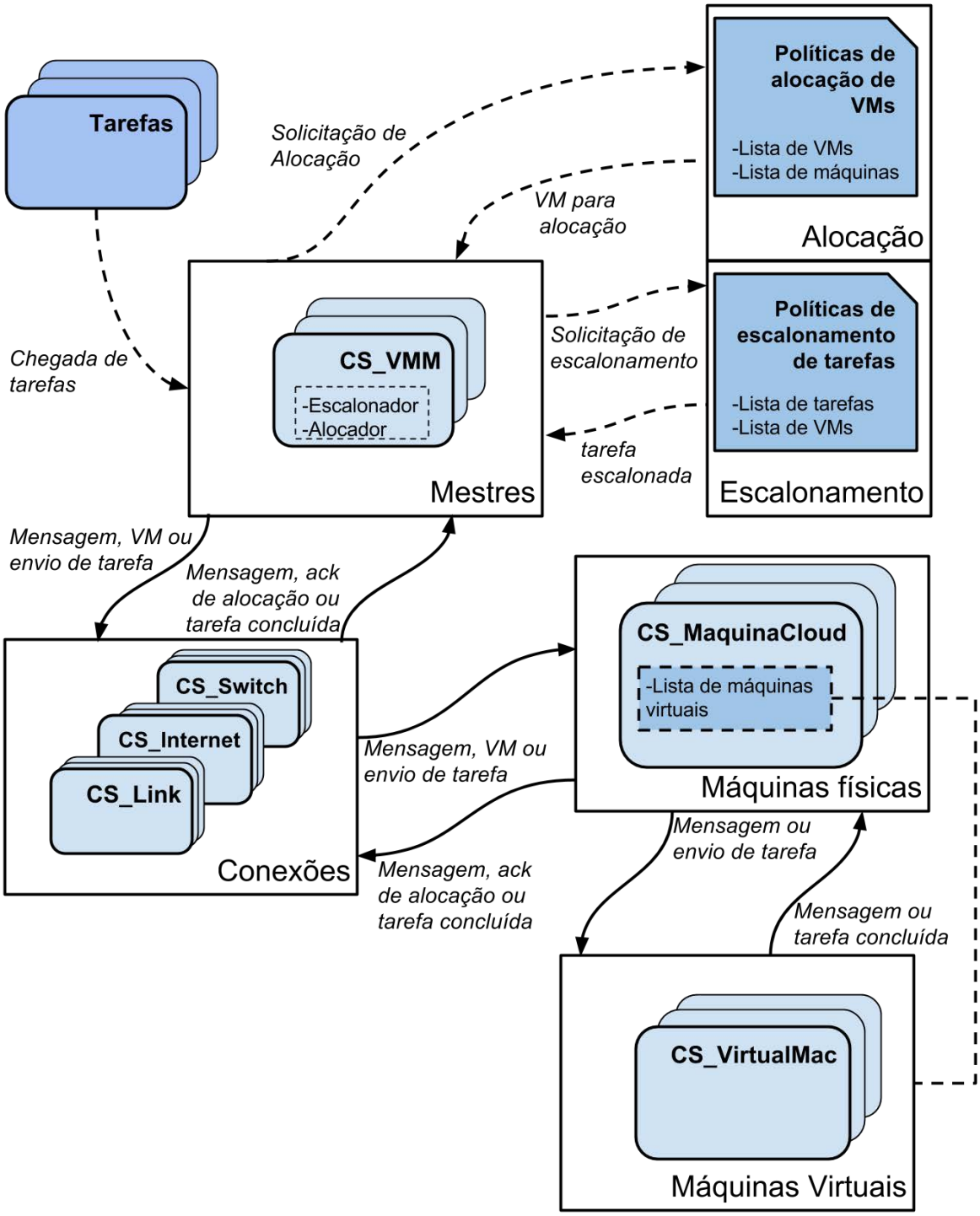


Figura 32 – Diagrama que resume as interações entre os centros de serviços de IaaS

iniciar a simulação, as primeiras relações entre os centros de serviços são referentes à alocação de máquina virtuais e podem ordenadas como segue:

1. Primeiramente as máquinas virtuais chegam aos centros de serviço de VMM que as gerenciam que, por sua vez, solicitam o processo de alocação e recebem a informação para a VM selecionada e o centro de serviço indicado para alocá-la. Caso o centro de serviço selecionado seja outro VMM, a tarefa será submetida a outra política de alocação. Caso seja um centro de serviço de máquina física, ela será hospedada;

2. No caminho entre o VMM e a máquina física, a máquina virtual percorre os centros de serviço de comunicação, computando neste processo os tempos de espera nas filas, transmissão da tarefa e os atrasos de latência de transmissão no relógio de simulação.

3. Quando a máquina virtual chega ao centro de serviço de máquina física que é seu destino, este centro de serviço irá incluir a máquina virtual na sua lista de máquinas virtuais hospedadas e irá enviar uma mensagem contendo o *ack* de alocação, que deve percorrer os elementos de comunicação na rede de filas até chegar novamente no VMM que coordena a VM;

4. Por fim, ao receber o *ack* de alocação, o centro de serviço de VMM irá determinar a função de roteamento para o centro de serviço de máquina virtual indicado, cujo *status* passará definitivamente para alocado e que será anexado na rede de filas, permitindo assim que o VMM possa começar a escalonar tarefas para essa VM.

Repetem-se estas interações para cada máquina virtual existente no sistema, até que todas encontrem-se no estado de alocada ou destruída.

Além das interações realizadas para a alocação das máquinas virtuais ocorrem as interações entre os centros de serviço necessárias para o escalonamento e execução das tarefas, cuja ordem de execução ocorre como segue:

1. Primeiramente as tarefas chegam aos centros de serviço de VMM, que executam o processo de escalonamento e enviam a tarefa selecionada para o centro de serviço que foi escalonado. Caso o centro de serviço selecionado seja outro VMM, a tarefa será reescalonada. Caso seja um centro de serviço de máquina virtual, ela será executada;

2. No caminho entre o VMM e a máquina física hospedeira da VM selecionada, a tarefa percorre os centros de serviço de comunicação, computando neste processo os tempos de espera nas filas, transmissão da tarefa e os atrasos de latência de transmissão no relógio de simulação;

3. Quando a tarefa chega no centro de serviço da máquina física onde a máquina virtual encontra-se hospedada, ele percorre sua lista de máquinas virtuais alocadas e dado que encontre a VM desejada, encaminha a tarefa para este centro de serviço de máquina virtual;

4. Quando a tarefa chega no centro de serviço de máquina virtual, computam-se no relógio de simulação os tempos de espera na fila e de processamento da tarefa e calculam-se as métricas referentes ao processamento.

5. Por fim, o centro de serviço de máquina virtual que executou a tarefa a devolve ao VMM que a submeteu, calculando a função de roteamento se necessário. Ao chegar no VMM a tarefa é marcada como concluída.

Estas interações se repetem para cada uma das tarefas, até que todas tenham chegado ao estado de concluída ou cancelada, encerrando assim o processo de simulação.

4.7 Geração de métricas e exibição de resultados

Para a análise de ambientes de computação em nuvem são necessárias algumas métricas que não são usadas em grades. Essas novas métricas são coletadas pelo motor de simulação, que repassa os valores medidos para a interface icônica para a apresentação gráfica dos resultados. Os próximos parágrafos descrevem primeiro a especificação das novas métricas e depois as interfaces para sua exibição.

4.7.1 Especificação das métricas de desempenho para IaaS no iSPD

Para computação em nuvem é necessário ter métricas de custo de utilização das VMs e métricas relativas à alocação de máquinas virtuais. Além disso, adaptou-se o conjunto de métricas já existentes no iSPD para obter as medidas de desempenho dos elementos de processamento a partir dos novos centros de serviço desenvolvidos neste trabalho (VMM, máquinas físicas e máquinas virtuais).

Deste modo, analisando a adaptação das métricas de processamento, implementou-se nos centros de serviço de máquina virtual o cálculo das métricas de quantidade de tarefas executadas, quantidade de carga computacional (em MFlop) processada e tempo de espera na fila de processamento. Implementaram-se ainda os métodos necessários para percorrer cada centro de serviço de máquina virtual ao fim da simulação, coletando dados de modo a gerar as séries numéricas que são passadas para a geração de resultados gráficos na interface de apresentação de resultados.

Em relação às métricas de alocação, desenvolveram-se as métricas de número total de VMs alocadas, número total de VMs rejeitadas e número de VMs alocadas por máquina. Para obter essas métricas se percorre, ao fim da simulação, os centros de serviço de máquina

física, contabilizando o valores de máquinas virtuais alocadas no total, e gerando a estrutura que armazena a série numérica que contém os valores de máquinas virtuais alocadas em cada máquina física. Por fim, para verificar o número de máquinas virtuais rejeitadas, percorre-se os centros de serviço de VMM, que contem essa informação ao fim do processo de alocação. Ao fim do procedimento, as séries numéricas referentes a esta métrica também são passadas para a interface icônica.

Quanto às métricas relativas aos custos de utilização dos recursos, desenvolveram-se as métricas de custo parcial, que referem-se ao custo de utilização para cada máquina virtual, e as métricas de custo total, que referem-se ao custo acumulado de utilização de todas as VMs. Para os dois casos são oferecidos valores de custos de processamento, memória e disco.

Para calcular os custos parciais e totais de utilização deve-se, primeiro, calcular o tempo de utilização das máquinas virtuais, medindo-se o tempo que cada centro de serviço de máquina virtual permanece no estado de alocado. Para calcular esta medida basta capturar os instantes de tempo simulado em que ocorreram as transições nos centros de serviço de VM para o estado de VM alocada e depois para o estado de VM destruída.

Deste modo, para calcular os custos parciais de utilização de processamento, memória e disco para cada VM, percorrem-se os centros de serviço de máquina virtual, calculando cada uma das métricas na forma a seguir:

- **Custo de processamento:** É o produto do tempo de alocação (em horas) pela taxa (custo) de uso de núcleos de processamento da máquina hospedeira ($/*cores*/hora), como segue:
  *Custo de processamento = taxa de processamento ** (*tempo em estado de alocada / 3600*)

- **Custo de memória:** É o produto do tempo de alocação (em horas) pela taxa de uso de memória da máquina hospedeira ($/Mb/hora), como segue:
  *Custo de memória = taxa de memória ** (*tempo em estado de alocada / 3600*)

- **Custo de disco:** É o produto do tempo de alocação (em horas) pela taxa de uso de disco da máquina hospedeira ($/GB/hora), como segue:
  *Custo de disco = taxa de disco ** (*tempo em estado de alocada / 3600*)

4.7.2 Interface de exibição dos resultados da simulação de IaaS

Com a inclusão e modificação de métricas para computação em nuvem é necessário desenvolver uma nova interface de exibição de resultados para IaaS, a partir da adaptação da interface existente para grades computacionais. A interface desenvolvida recebe as métricas calculadas pelo motor de simulação e gera os relatórios e gráficos referentes à simulação de IaaS, organizando-os em abas. Cada aba apresenta um tipo de resultado da simulação. As abas desenvolvidas para análise de resultados de IaaS foram:

- Relatório geral sobre a simulação;
- Relatório sobre atendimento e execução das tarefas;
- Relatório sobre quantidade de carga de trabalho processada por cada elemento;
- Gráficos do uso de processamento através do tempo;
- Gráficos de distribuição da transmissão de dados entre os recursos de comunicação;
- Gráficos sobre a distribuição da execução das tarefas pelas VMs;
- Gráficos sobre a distribuição das VMs alocadas pelas máquinas físicas;
- Gráficos sobre os custos de utilização das máquinas virtuais.

Destaca-se que os resultados referentes ao processamento das tarefas foram apenas refatorados a partir dos existentes para computação em grade, substituindo-se os centros de serviço de máquina pelos centros de serviço de máquina virtual. Desta forma, julga-se interessante dar um maior detalhamento nos resultados desenvolvidos especificamente por este trabalho, que são o relatório geral sobre a simulação, os gráficos sobre a alocação das VMs entre as máquinas físicas e sobre os custos de utilização das máquinas virtuais.

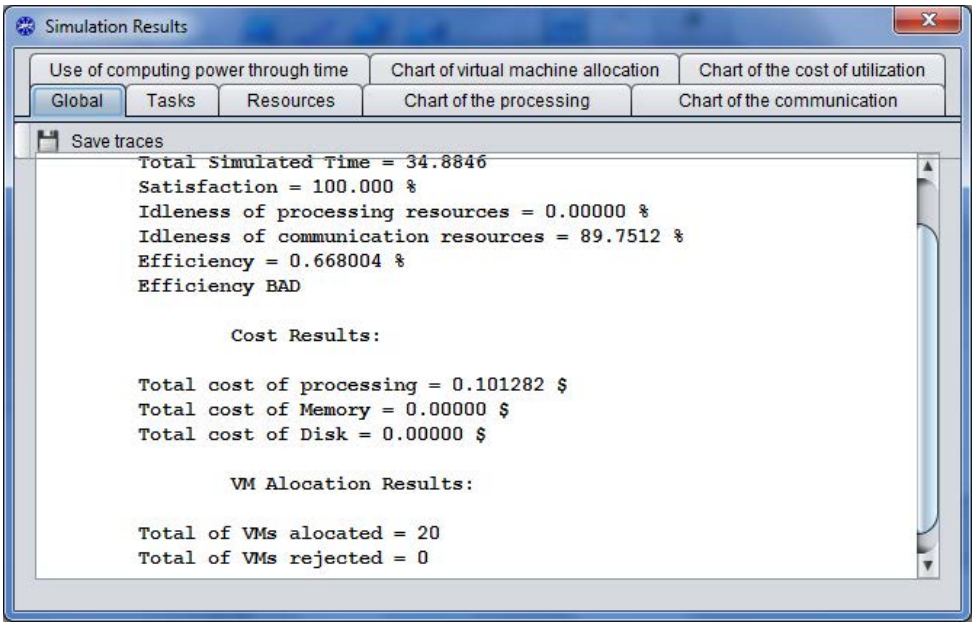


Figura 33 – Janela de exibição de resultados destacando a aba do relatório geral da simulação

Para o desenvolvimento do relatório geral da simulação, além das informações como o tempo total simulação, eficiência, satisfação do usuário e taxa de ociosidade dos recursos, incluiu-se as informações sobre os custos totais de utilização de processamento, memória e disco, além das informações sobre o número de VMs alocadas e rejeitadas. Apresenta-se na Figura 33

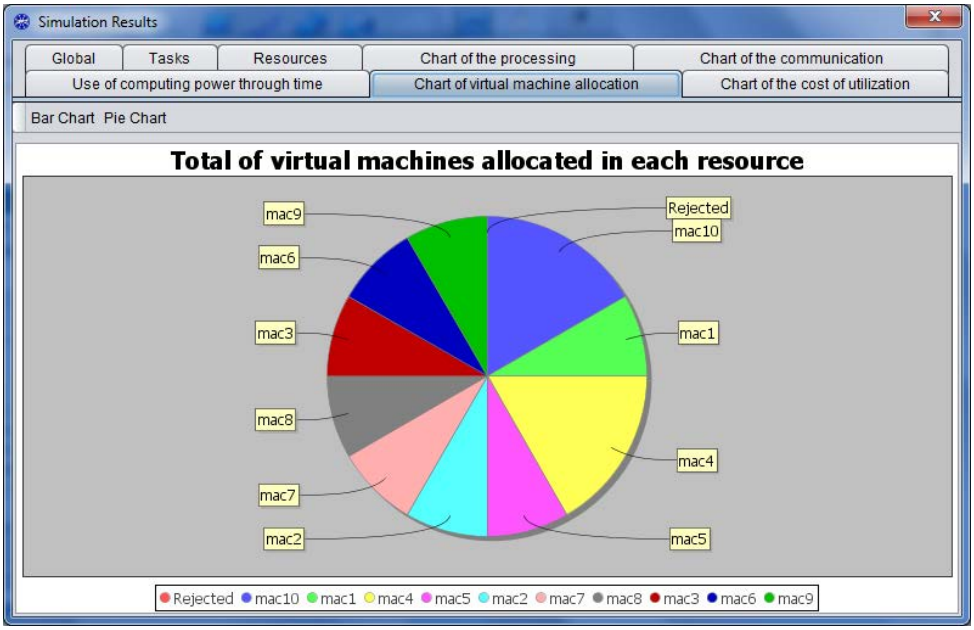


Figura 34 – Janela de exibição de resultados destacando a aba do gráfico de alocação das máquinas virtuais entre as máquinas físicas

a interface de exibição de resultados, em que se destaca a aba que apresenta o relatório geral da simulação, com dados sobre custos totais e de alocação de VMs, por exemplo.

Quanto à alocação de máquinas virtuais, implementaram-se gráficos de barra e pizza sobre a distribuição das máquinas virtuais entre os centros de serviço de máquina física, apresentando também o número de máquinas rejeitadas. Apresenta-se na Figura 34 a interface da janela de exibição de resultado, destacando a aba que apresenta o gráfico de distribuição de VMs entre as máquinas físicas sob o formato de gráfico de pizza. Observa-se ainda acima do gráfico, os botões onde o usuário pode alternar o formato de visualização do gráfico.

Por fim, em relação aos custos de utilização das máquinas virtuais, implementaram-se os gráficos de barras de custo de processamento, memória e disco, além do gráfico que apresenta o custo acumulado (soma dos três tipos de custos). Apresenta-se na Figura 35 a interface de exibição de resultados, destacando-se a aba de gráficos de custo de utilização. Exibe-se na figura o gráfico de custo total acumulado de utilização das máquinas virtuais. Destaca-se ainda, acima do gráfico, os botões para escolher entre as opções de custo de processamento, memória, disco e custo de utilização acumulado.

4.8 Especificação da modelagem de PaaS para o iSPD

Como já discutido, a modelagem da classe de serviço de PaaS para o iSPD encontra-se ainda em desenvolvimento. Deste modo, apresenta-se nesta seção uma especificação de como esta classe de serviço deve ser caracterizada para o iSPD.

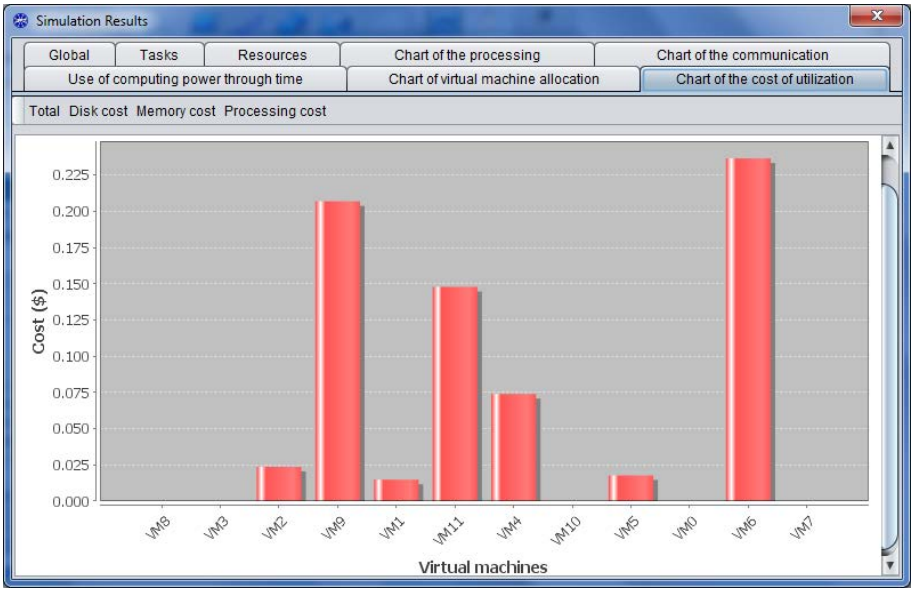


Figura 35 – Janela de exibição de resultados destacando a aba do gráfico de custo de utilização das máquinas virtuais

O objetivo da classe de serviço de PaaS é oferecer um plataforma de desenvolvimento e hospedagem de aplicações que isenta o usuário de realizar o gerenciamento da demanda de recursos utilizados. Desta forma, observando que o principal aspecto da classe de serviço de PaaS é a hospedagem de aplicações, para possibilitar a modelagem desta classe de serviço no iSPD deve-se caracterizar a carga de trabalho de uma maneira mais detalhada.

Assim, define-se a carga de trabalho como um conjunto de aplicações. Para cada aplicação, modela-se o acordo de nível de serviço (SLA), que determina as quantidades de recursos (núcleos de processamento, memória e disco) contratados pela aplicação, assim como o S.O. em que a mesma deve executar. Além da SLA, modela-se ainda para cada aplicação o conjunto de tarefas que a compõe, definindo para cada uma, além das informações já existentes sobre carga de computação e carga de comunicação, quantidades de núcleos de processamento, memória e disco necessários para sua execução. Apresenta-se na Figura 36 a estrutura de carga de trabalho definida para a modelagem de PaaS.

Quanto à modelagem do ambiente para PaaS, o usuário deve implementar apenas o ambiente físico, utilizando a abordagem icônica desenvolvida para IaaS. Deve-se configurar para os elementos de VMM a política adotada para o provisionamento de máquinas virtuais. Esta política deve identificar quais os requisitos de recursos das aplicações modeladas e elaborar estratégias para instanciar VMs com capacidade elástica, que possam atender as requisições da aplicação que irá hospedar. Observa-se que esta abordagem elimina a necessidade de modelar as VMs diretamente, através da interface de configuração de máquinas virtuais.

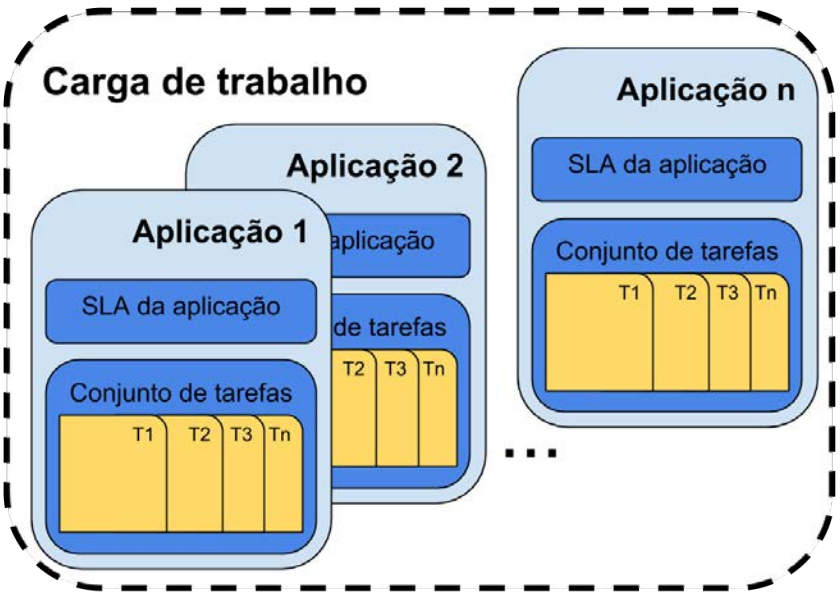


Figura 36 – Estrutura definida para a modelagem da carga de trabalho para PaaS

4.9 Considerações finais

Neste capítulo apresentou-se o desenvolvimento deste trabalho, com destaque para a especificação e implementação das alterações necessárias na interface gráfica, interpretador de modelos e motor de simulação do iSPD. Apresentou-se também as especificações de novas políticas para alocação de VMs e escalonamento de tarefas, assim como métricas de avaliação considerando-se a modelagem e simulação de computação em nuvem, mais especificamente de IaaS para o simulador iSPD. Apresenta-se no próximo capítulo o conjunto de testes realizados para validar o processo de simulação de computação em nuvem desenvolvido.

5 Testes e resultados

Para validar a simulação de computação em nuvem pelo iSPD adotou-se o processo de comparação dos resultados obtidos com o iSPD com resultados medidos em uma nuvem real e no simulador CloudSim (Seção 5.1). Outros testes foram executados para verificar se o processo de simulação é realizado em tempos factíveis, analisando o impacto do aumento da carga de trabalho no tempo de execução da simulação comparando os tempos de execução com o CloudSim (Seção 5.2). Por fim, realizaram-se ainda testes para verificar a utilidade do iSPD como uma ferramenta para o estudo de políticas de alocação de máquinas virtuais (Seção 5.3).

5.1 Teste de validação da simulação de IaaS pelo iSPD

Para validar a modelagem e a simulação de IaaS desenvolveu-se um teste para comparar o tempo de execução de um ambiente implementado em um serviço de IaaS e o tempo de execução simulado para o ambiente correspondente ao real modelado no iSPD. As características de implementação dos ambientes de testes, tais como especificação da carga de trabalho utilizada, especificação dos modelos correspondentes aos ambientes de testes nos simuladores são agora apresentadas.

Para servir como ambiente de nuvem real selecionou-se o serviço de IaaS Google Compute Engine (GCE)(GOOGLE, 2015a). O GCE é um serviço de IaaS que integra um conjunto de serviços de computação em nuvem chamado Google Cloud Platform, disponibilizado pela Google. Ele foi escolhido pois oferece uma versão de testes em que se recebe trezentos dólares virtuais que podem ser gastos nestes serviços durante o período de sessenta dias, permitindo sua avaliação sem custos.

Além do GCE também se incluiu o CloudSim nos testes. Selecionou-se este simulador por ele ser a ferramenta mais referenciada na literatura, com aproximadamente 3690 citações segundo o Google Scholar.

5.1.1 Implementação do ambiente no GCE

Para realizar os testes implementou-se, por meio do GCE, três ambientes de computação distribuída, construídos a partir de três perfis de máquinas virtuais disponíveis na versão de testes:

- **"g1-Small":** que conta com um núcleo virtual de 1,38 GCEU[1], 1,7 GB de memória RAM e 10 GB de disco;

[1] GCEU (*Google Compute Engine Unit*) é a unidade de capacidade da CPU utilizada pelo GCE para descrever o poder de computação de suas máquinas virtuais. Foi definido pelo provedor de serviço que 2,75 GCU representa

- **“n1-Standard-1”:** que conta com um núcleo virtual de 2,75 GCEU, 3,75 GB de memória RAM e 10 GB de disco;
- **“n1-highcpu-2”**: que conta com dois núcleos virtuais de 5,5 GCEU, 1,8 GB de memória RAM e 10 GB de disco.

A topologia dos ambientes de testes está apresentada na Figura 37. Para cada perfil de máquina virtual foram instanciadas cinco máquinas virtuais contendo uma imagem do sistema operacional Ubuntu 14.04 LTS “Trusty Tahr”. Uma dessas VMs realizou o papel de *front-end* do ambiente computacional, enquanto as outras quatro máquinas exercem a função de nós de processamento. Quanto às interconexões, a rede virtual é implementada como uma rede local virtual (*vlan*) com endereço de rede 10.240.0.0/16.

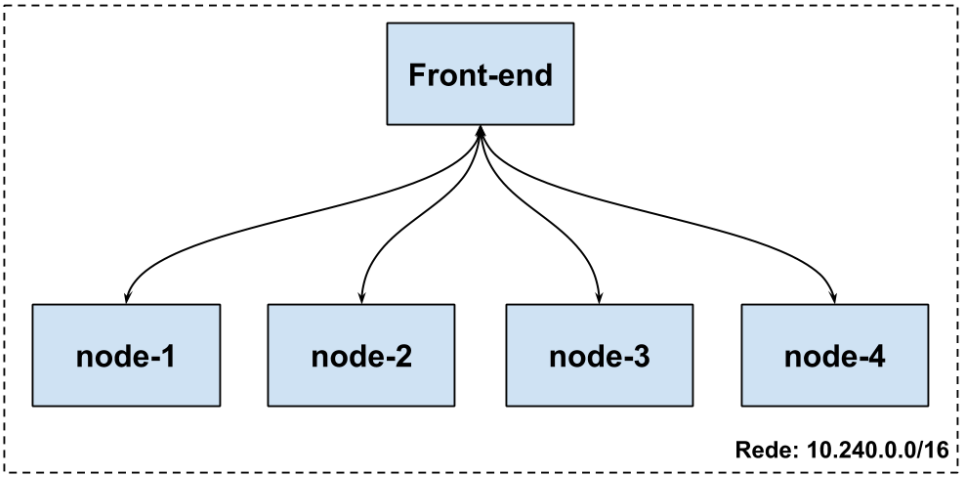


Figura 37 – Estrutura virtual dos ambientes de testes implementados no GCE

Quanto à carga de trabalho, implementaram-se dois programas em linguagem python, descritos a seguir:

- “`client.py`”: Disponível no Anexo A, este arquivo implementa o programa cliente executado pela VM de *front-end*. Seu funcionamento consiste em implementar quatro *threads* que disparam solicitações aos nós de processamento conforme o número programado de tarefas para serem executadas;
- “`server.py`”: Disponível no Anexo B, este arquivo implementa o programa servidor executado pelas VMs que desempenham a função de processamento. Seu funcionamento consiste em esperar por solicitações e executar vinte milhões de somas de ponto flutuante, devolvendo para o *front-end* o tempo decorrido em cada solicitação recebida.

a potência mínima de uma CPU virtual (com suporte a *hiperthreading*) de um processador da família Sandy Bridge, Ivy Bridge ou Haswell utilizado.

5.1.2 Modelos correspondentes no iSPD e CloudSim

Para modelar os ambientes implementados no GCE, tanto no iSPD como no CloudSim, é preciso dimensionar os parâmetros de processamento, memória, disco e banda de passagem dos perfis de VMs. Enquanto os valores de memória RAM e disco dos perfis de VM estavam especificados nas mesmas unidades das que devem ser utilizadas nos simuladores, foi preciso estimar os valores referentes às características de processamento e banda de rede para os elementos modelados. Foi necessário ainda estimar a carga computacional na unidade de MFlop das tarefas computadas pelos ambientes de teste.

Para estimar as capacidades de processamento dos perfis das máquinas virtuais utilizadas, deve-se observar que a unidade de GCEU é bastante subjetiva, podendo representar a capacidade de um núcleo virtual baseado em famílias de processadores da arquitetura x86 que possuem capacidades de processamento muito diferentes. Assim, foi preciso estimar a medida de capacidade de processamento dos perfis das VMs em MFlops, que é uma medida absoluta de capacidade de processamento. Para tanto avaliou-se os tempos de execução do programa "`server.py`" nas VMs e em um processador com capacidade conhecida. Assim temos que em um processador de 51,2 Gflops (Intel i5, 4 núcleos, 1,6 Ghz e 8 intruções/ciclo) o tempo médio de execução de "`server.py`" foi de 2,17s. Isso implica que sua carga computacional é de:

$$Carga = 51200\ MFlops * 2{,}17\ s = 111104\ MFlop$$

A partir destas informações foi possível estimar as capacidades de processamento dos processadores virtuais utilizados pelos perfis de VMs que implementaram os três ambientes de testes no GCE. Os resultados obtidos estão apresentados na Tabela 2.

Tabela 2 – Valores de poder de processamento estimados para cada perfil de VM do GCE utilizado na modelagem dos ambientes de testes

Ambiente	Tempo de execução	Capacidade de processamento (GFlops)
g1-small	5,095	21,805
n1-standard-1	2,419	45,938
n1-highcpu-2	2,338	47,518

Quanto à estimativa da banda de transmissão dos enlaces utilizados nos ambientes de testes, utilizou-se o comando "*traceroute*". Dado que o tamanho de mensagem enviada é fixo (60 bytes), foi possível estimar a banda de passagem por meio destas informações. Os resultados obtidos são apresentados na Tabela 3.

Após realizar a estimativa desses valores, implementou-se os modelos icônicos e programas em Java correspondentes aos ambientes do GCE no iSPD e no CloudSim, respectivamente. Ressalta-se que o iSPD e o CloudSim implementam a simulação de computação em nuvem tanto do ponto de vista do consumidor de computação, que é o caso implementado nos ambientes

Tabela 3 – Valores de banda de passagem estimados para os enlaces dos ambientes de testes implementados com os perfis de VMs do GCE

Ambiente	Tempo de transmissão (ms)	Banda de passagem (Mbps)
g1-small	1,360	0,366
n1-standard-1	1,679	0,273
n1-highcpu-2	1,550	0,294

modelados no GCE, como do provedor de serviço, que preocupa-se com a modelagem de um número maior de características, como a infraestrutura física do sistema e as políticas de alocação de VMs e escalonamento de tarefas utilizadas. Desta forma, para modelar o ambiente virtualizado do GCE, implementou-se um ambiente físico dentre os possíveis para hospedá-lo.

Para o iSPD, implementaram-se os ambientes constituídos por cinco ícones de máquinas físicas, sendo um deles configurado como VMM, que corresponde à VM que implementa o *front-end*, e os outros como máquina físicas que devem hospedar o ambiente virtualizado do GCE. Foram selecionadas as políticas *first-fit*, para a alocação de VMs, e a política *round-robin*, para escalonar as tarefas. Por fim, modelou-se os ícones de enlace lógico de acordo com os valores obtidos na Tabela 3. Apresenta-se na Figura 38 o ambiente físico modelado.

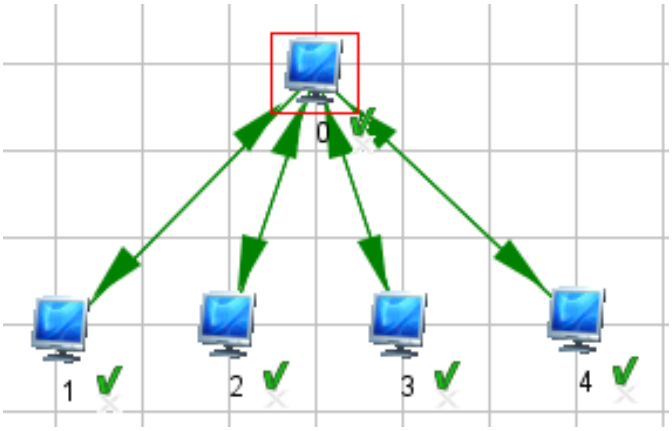


Figura 38 – Ambiente físico modelado no iSPD para simular os ambientes do GCE

Implementou-se para o CloudSim, por meio das classes e métodos correspondentes, os arquivos Java contendo a modelagem do ambiente físico e virtualizado, assim como feito no iSPD, para cada ambiente de teste. Ressalta-se que como o CloudSim não possui um gerador de números aleatórios incluiu-se, na programação dos modelos, o gerador de números aleatórios do iSPD para gerar os instantes de chegada das tarefas.

5.1.3 Análise dos tempos de execução do GCE e tempos simulados do iSPD e CloudSim

Durante os testes aplicaram-se cargas de trabalho com 20, 40, 60, 80, 100, 140, e 200 tarefas, respectivamente. Em cada caso as execuções foram repetidas um número mínimo de cinco vezes, de modo a verificar sua convergência. Apresentam-se a seguir os resultados obtidos para cada ambiente de testes.

Ambiente "g1-small"

Apresenta-se na Tabela 4 os valores obtidos para os testes de tempo de execução, em segundos, no GCE e tempos simulados correspondentes no iSPD e CloudSim para o ambiente de teste implementado por meio das instâncias de VM "g1-small". Apresenta-se também na Figura 39 um gráfico de barras ilustrando o padrão de crescimento dos valores do tempo real e simulados de acordo com o aumento da carga de tarefas.

Tabela 4 – Tempos médios, em segundos, de execução real (GCE) e simulados (iSPD e CloudSim) para VMs do perfil "g1-small"

Número de Tarefas	GCE	iSPD	Erro iSPD (%)	CloudSim	Erro CloudSim (%)
20	26,12	27,62	5,74	25,60	1,99
40	53,66	53,08	1,08	50,10	6,63
60	80,40	78,64	2,19	76,50	4,85
80	108,20	104,02	3,86	102,00	5,73
100	131,58	129,48	1,60	127,50	3,10
140	183,06	180,48	1,41	178,40	2,55
200	265,82	256,90	3,36	254,80	4,15

Observa-se que o crescimento dos tempos de execução no GCE com relação ao aumento do número de tarefas executadas apresenta um padrão linear. Esse era o comportamento esperado dado que as tarefas são independentes. Deste modo, observa-se que a carga de trabalho possui um grau de complexidade linear.

Quanto aos valores obtidos, observa-se que os tempos simulados são muito próximos aos tempos reais, apresentando apenas pequenas disparidades. Apresenta-se na Tabela 4 o erro relativo dos valores de tempos de execução simulados em relação ao tempo real obtido no GCE. Observa-se que embora o tempo simulado pelo iSPD se apresente menos preciso em relação ao tempo real para um número de tarefas pequeno (20 tarefas para este ambiente de testes), o erro relativo dos resultados obtidos por esse simulador tende a diminuir conforme se aumenta a carga de trabalho.

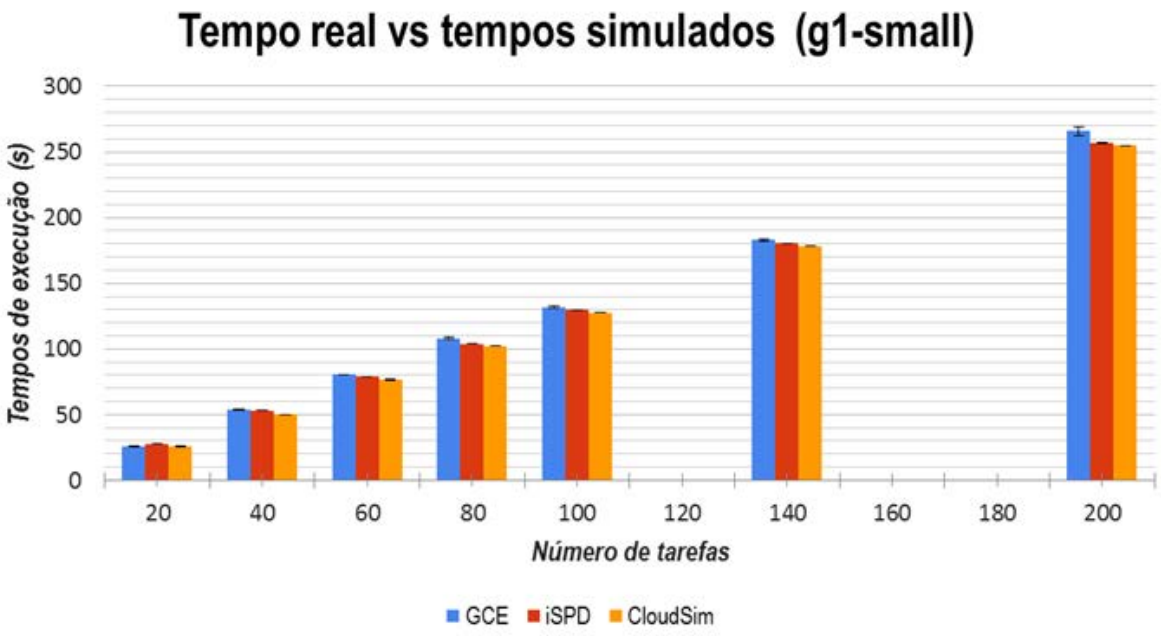


Figura 39 – Tempos de execução real e simulados para o ambiente de testes construído com VMs “g1-small”

Ambiente “n1-standard-1”

Apresentam-se na Tabela 5 os valores dos tempos de execução, em segundos, no GCE e dos tempos simulados correspondentes no iSPD e CloudSim para este ambiente de testes. Apresenta-se ainda na Figura 40 o gráfico de crescimento dos valores do tempo real e simulados de acordo com o aumento da carga de tarefas para este caso.

Pode-se perceber que o padrão de crescimento linear permanece, dado que trata-se da mesma carga de trabalho. No entanto, observa-se uma redução dos tempos de execução, pois o ambiente de testes foi implementado com um perfil de VMs que possui uma capacidade de processamento superior ao dobro do perfil de VMs “g1-small” utilizado no ambiente de teste anterior.

Tabela 5 – Tempo médio, em segundos, de execução real (GCE) e simulados (iSPD e CloudSim) para VMs do perfil “n1-standard-1”

Número de Tarefas	GCE	iSPD	Erro iSPD (%)	CloudSim	Erro CloudSim (%)
20	12,66	14,98	18,32	12,52	1,11
40	24,84	26,94	8,45	24,70	0,56
60	37,34	39,04	4,55	36,60	1,98
80	49,86	52,1	4,49	48,76	2,21
100	61,72	63,16	2,33	60,82	1,46
140	86,26	87,34	1,25	85,04	1,41
200	125,32	123,84	1,18	121,28	3,22

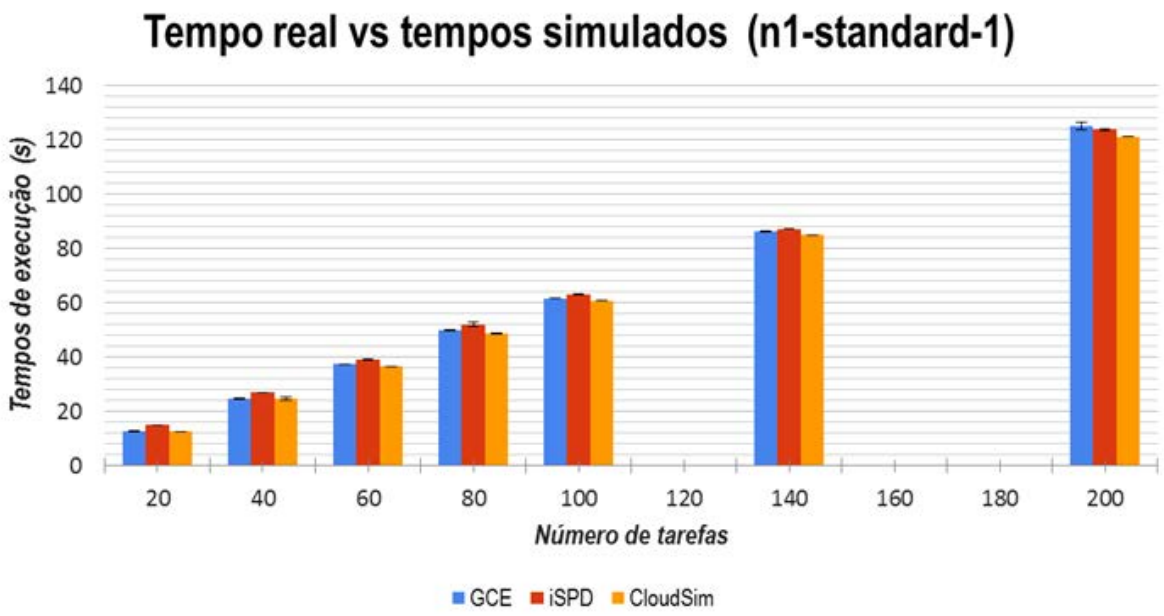


Figura 40 – Tempos de execução real e simulados para o ambiente de testes construído com VMs “n1-standard-1”

Observa-se que neste ambiente de testes houve um aumento considerável da taxa de erro relativo no tempo obtido no iSPD para 20 tarefas. No entanto, verifica-se também que a taxa de erro para os tempos obtidos pelo iSPD mantêm-se decrescente com o aumento do número de tarefas enquanto para o CloudSim esses valores mantêm um comportamento constante, de modo que o iSPD passa a apresentar erros relativos menores que o CloudSim para os casos de 140 e 200 tarefas.

Ambiente “n1-highcpu-2”

Apresentam-se na Tabela 6 os tempos médios de execução reais e simulados (em segundos) para o ambiente de testes implementados com VMs do perfil “n1-highcpu-2”, apresentando na Figura 41 o gráfico de barras que ilustra o comportamento de crescimento desses valores para este caso. Observa-se para este caso o mesmo padrão linear nos valores obtidos para os tempos de execução. Observa-se ainda que os tempos de execução real e simulados apresentam-se menores em relação ao caso anterior, embora a diferença entre estes tempos seja bem menor, dado que o perfil de VMs utilizada na implementação do ambiente de teste, o “n1-highcpu-2” é ligeiramente mais rápido que o perfil “n1-standard-1”, utilizado no modelo anterior.

Mais um vez observa-se que o iSPD apresenta um erro relativo considerável para o caso de 20 tarefas e passa a apresentar valores de erro relativos cada vez menores conforme aumentam-se as tarefas. Deste modo observa-se que mais uma vez, que enquanto o erro relativo obtido para os tempos simulados no CloudSim mantêm-se praticamente constantes, o erro relativo dos valores obtidos pela simulação do iSPD diminui fortemente, apresentando valores de erro relativos inferiores aos obtidos pelo CloudSim para os casos de teste de 100, 140 e 200 tarefas.

Tabela 6 – Tempo médio, em segundos, de execução real (GCE) e simulados (iSPD e CloudSim) para VMs do perfil "n1-highcpu-2"

Número de Tarefas	GCE	iSPD	Erro iSPD (%)	CloudSim	Erro CloudSim (%)
20	12,06	14,3	18,57	12,4	2,82
40	24,18	25,84	6,87	23,74	1,82
60	36,1	37,54	3,99	35,42	1,88
80	48,08	49,98	3,95	47,14	1,96
100	60,1	61	1,50	58,8	2,16
140	84,38	84,42	0,05	82,16	2,63
200	119,74	119,46	0,23	117,24	2,09

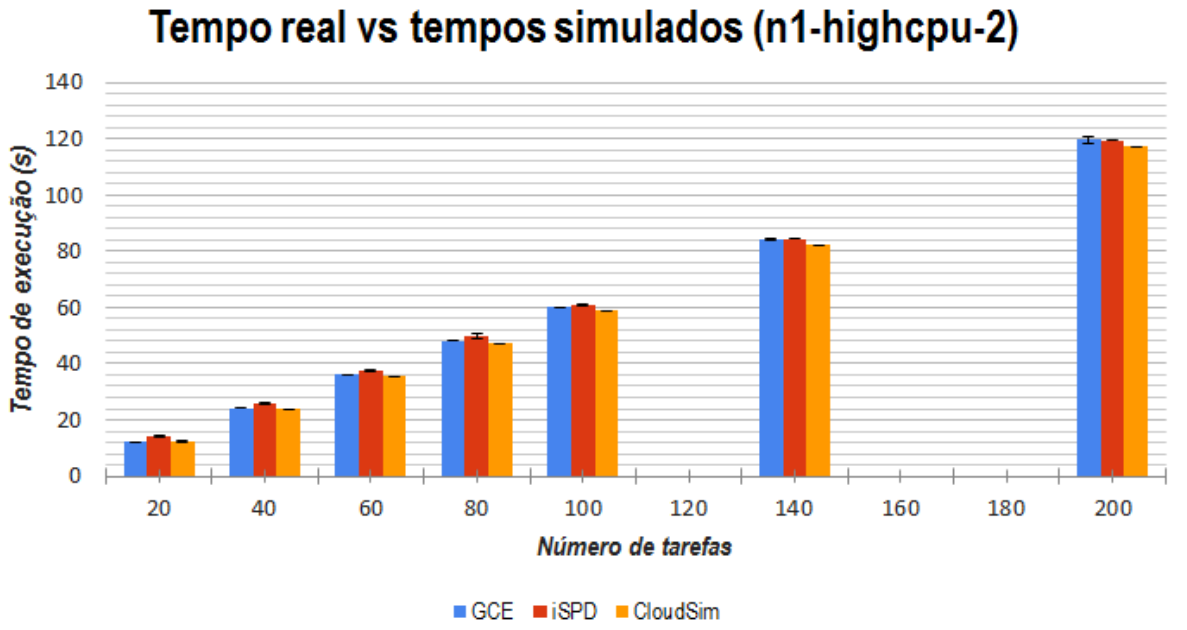


Figura 41 – Tempos de execução real e simulados para o ambiente de testes construído com VMs "n1-highcpu-2"

5.1.3.1 Análise qualitativa dos resultados

Observou-se em todos os casos de teste um comportamento de decaimento dos valores do erro relativo iSPD dos tempos de execução simulados no iSPD. Por esta razão, buscou-se estudar a causa deste comportamento.

Primeiramente, aponta-se como um dos fatores da disparidade dos valores simulados pelo iSPD, em relação aos tempos de execução obtidos no GCE, o fato deste simulador contabilizar não apenas o tempo de execução das tarefas, como também o tempo decorrido na alocação das VMs e no escalonamento das tarefas, enquanto no GCE e na simulação realizada pelo CloudSim estes valores não podem ser contabilizados. Deste modo, observa-se que existe um tempo constante para alocar um número fixo de máquinas virtuais e um tempo que varia diretamente em função do número de tarefas decorrente do escalonamento que é contabilizado apenas no

iSPD. Isso pois o iSPD diferencia aspectos da simulação de IaaS interessantes para a avaliação do provedor de serviço, ao invés de contabilizar apenas o tempo decorrente da execução das tarefas.

Outro aspecto que pode ter influenciado o comportamento obtido pelo iSPD é o modo como a modelagem da capacidade de processamento foi realizada. Como a medida utilizada para descrever a capacidade de processamento das VMs do GCE é subjetiva, a abordagem utilizada para obter um valor absoluto, que foi obter os tempos de execução e então calcular a capacidade de MFlops em relação a um processador conhecido, pode gerar discrepâncias na capacidade de processamento modelada em relação à capacidade real dos ambientes de teste.

Deste modo, para o ambiente "g1-small" pode-se observar que os tempos simulados no iSPD foram sistematicamente menores que os tempos obtidos no GCE, evidenciando que os processadores modelados no simulador foram estimados com uma capacidade um pouco acima da real. Do mesmo modo, para os ambientes "n1-standard-1" e "n1-highcpu-2" os tempos simulados são sistematicamente maiores que os obtidos pelo GCE, equiparando-se apenas para cargas de trabalho maiores, o que evidencia que os processadores modelados para estes ambientes foram melhor estimados, apesar de ainda terem maior poder computacional.

Ressalta-se ainda como um possível fator de discrepância, porém de menor impacto nos resultados, a estimativa da banda de passagem da rede, que pode oscilar em relação ao valor modelado durante a execução dos testes nos ambientes reais.

5.2 Custo temporal do processo de simulação

Um aspecto importante no processo de simulação é o tempo de resposta do simulador, ou seja, quanto tempo é consumido pela ferramenta para simular um determinado modelo. Desta forma, realizaram-se testes para medir os tempos de execução do processo de simulação de IaaS pelo iSPD, comparando-os com os tempos correspondentes no CloudSim. Para realizar estes testes o modelo usado, mostrado na Figura 42, é composto por:

- Ambiente físico:
  - Dez máquinas físicas, com dois núcleos de processamento de 50 GFlops, 1024 MB de memória RAM e 10 GB de disco.
  - Um VMM, com configuração de recursos igual às máquinas físicas, política *first-fit* para alocação de VMs e política *round-robin* para escalonamento das tarefas;
  - topologia de rede em estrela, com o VMM como nó central.
- Ambiente virtualizado:
  - Vinte máquinas virtuais, que requisitam dois núcleos de processamento, 512 MB de memória RAM e 5 GB de disco.

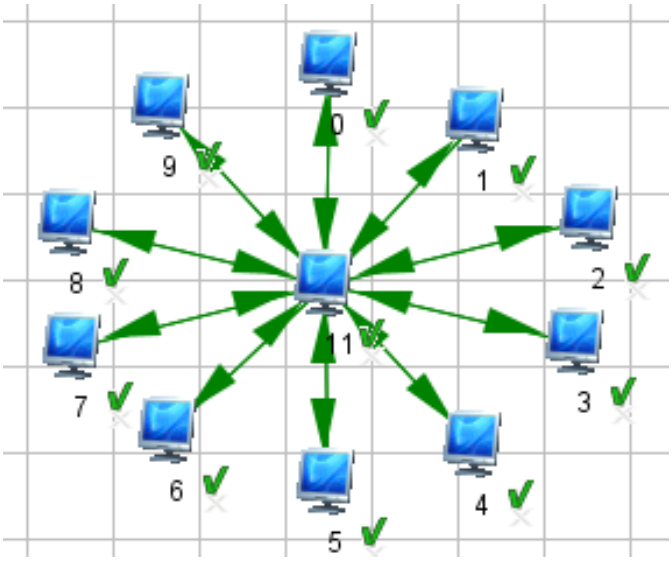


Figura 42 – Ambiente físico modelado no iSPD para o teste de tempo de execução da simulação

Implementou-se o modelo correspondente no CloudSim por meio de um programa Java como dez objetos de máquina física, um objeto do tipo *broker* (VMM), e vinte objetos do tipo máquina virtual, utilizando as configurações correspondentes às modeladas no iSPD.

Para a modelagem da carga de trabalho definiram-se tarefas com carga computacional variando entre 50 e 100 Gflop e com carga de comunicação variando entre 5 e 30 Mb. O número de tarefas simuladas ficou entre 10 e 100000 tarefas, com variações de um fator de 10 vezes.

Quanto ao ambiente de execução das simulações, tanto para o iSPD quanto para o CloudSim, utilizou-se uma estação de trabalho com as seguintes características de *software* e *hardware*:

- Processador Intel® Core™ i5-4200U CPU @ 1.60GHz x 4;
- Memória RAM com capacidade de 4 GB;
- Sistema Operacional Ubuntu 14.04 LTS “Trusty Tahr”;
- Máquina Virtual Java (JVM) executando Java na versão “1.7.0_79”;

5.2.1 Análise dos resultados

Após a realização de todos os testes, apresenta-se na Tabela 7 os valores dos tempos de execução do processo de simulação para os simuladores iSPD e CloudSim. Apresenta-se ainda Figura 43 o gráfico com os tempos de execução da simulação obtidos no iSPD e no CloudSim, em escala logarítmica. Observa-se que com o aumento do número de tarefas os tempos de execução no CloudSim crescem de forma muito mais drástica que no iSPD, como pode-se notar no caso de execução de 100 mil tarefas, quando o tempo médio de execução da simulação no Cloudsim chega a ser cem vezes maior do que o tempo médio de execução no iSPD.

Tabela 7 – Tempos de execução da simulação, em segundos, no iSPD e no CloudSim

Número de tarefas	iSPD	CloudSim
10	0,068	0,022
100	0,285	0,097
1000	2,284	0,451
10000	21,870	27,770
100000	204,432	21061,264

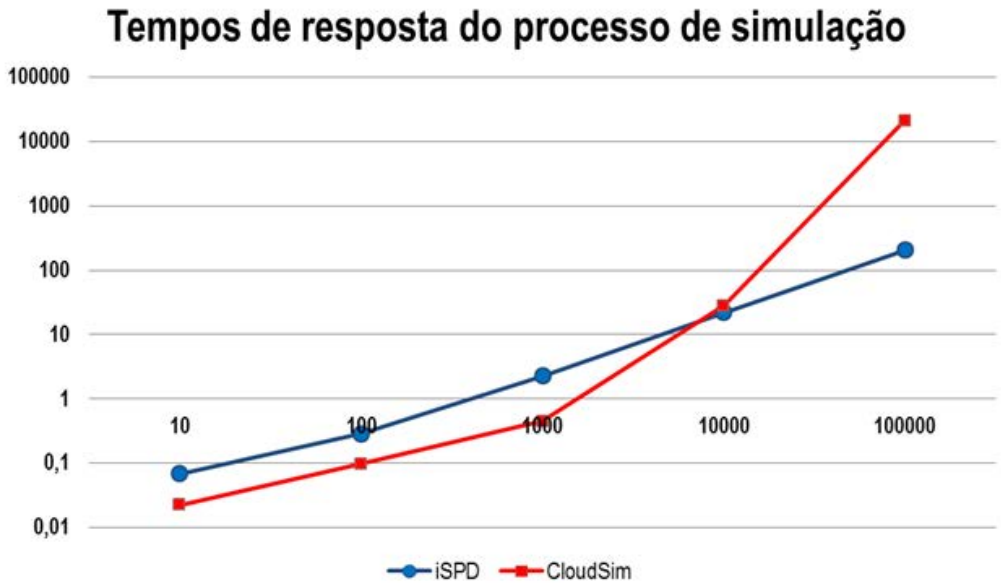


Figura 43 – Tempo médio de execução da simulação para os simuladores iSPD e CloudSim

A partir dos valores medidos foi possível observar o comportamento de crescimento desses tempos com o aumento da carga de trabalho. Pode-se perceber que enquanto para o iSPD a taxa de crescimento foi praticamente constante, convergindo para um valor entre 9 e 10 vezes a cada fator de 10 vezes mais tarefas, os tempos de execução para o CloudSim cresceram de forma exponencial. Por exemplo, para o caso de 100 mil tarefas, o tempo médio do CloudSim foi 758 vezes maior que o tempo médio para 10 mil tarefas, enquanto para o iSPD esse aumento foi de 9,3 vezes.

5.3 Testes de políticas de alocação com o iSPD

Um aspecto importante de simulação de computação em nuvem é o suporte à análise do processo de alocação, desde a implementação das políticas de alocação mais utilizadas, até a geração de métricas e resultados sobre a alocação. Assim, para analisar a capacidade do iSPD como ferramenta de estudo do processo de alocação de máquinas virtuais, foram realizadas simulações com diferentes políticas de alocação.

5.3.1 Definição do ambiente de testes desenvolvido e da metodologia empregada

Para avaliar diferentes políticas de alocação de máquinas virtuais é necessário utilizar um modelo heterogêneo de máquinas físicas e virtualizadas. Deste modo, para a modelagem da infraestrutura física, implementou-se um modelo contendo sete máquinas físicas, sendo que seis realizam o papel de máquinas hospedeiras enquanto uma desempenha o papel de VMM da infraestrutura modelada. Cada máquina física foi configurada com 50 GFlops de processamento por núcleo, 1024 Mb de memória e 10 GB de disco, variando-se o número de núcleos de processamento. Deste modo configurou-se uma máquina com oito núcleos de processamento (chamada "8core"), uma máquina com seis núcleos ("6core"), uma máquina com quatro núcleos ("4core"), duas máquinas com dois núcleos ("2core-1" e "2core-2") e uma máquina com um núcleo somente ("1core"). Apresenta-se na Figura 44 o ambiente modelado na interface icônica do iSPD.

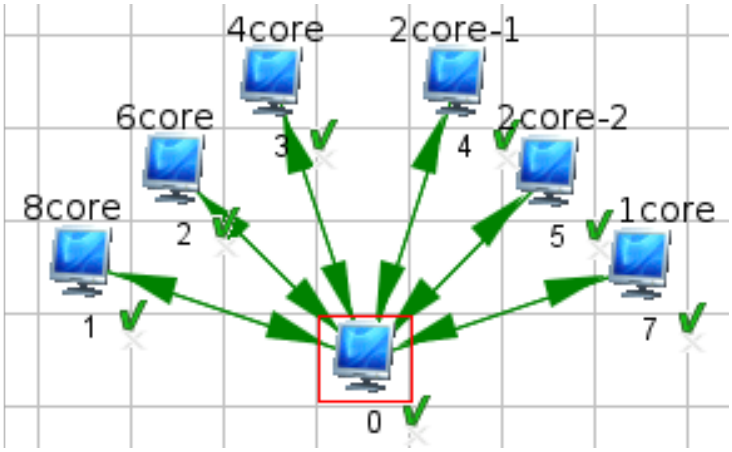


Figura 44 – Modelo físico de teste para o estudo de políticas de alocação

Modelaram-se também oito máquinas virtuais, com requisitos pequenos de uso de memória e disco (1Mb de memória e 1Gb de disco) de modo a deixar o processo de análise de requisitos de recursos restrito apenas ao número de núcleos virtuais necessários. Deste modo, modelou-se uma VM com requisito de oito núcleos virtuais (chamada "V8"), duas VMs com requisitos de quatro núcleos ("V4-1" e "V4-2"), duas VMs com requisitos de dois núcleos ("V2-1" e "V2-2") e três máquinas com requisitos de um núcleo virtual ("V1-1", "V1-2" e "V1-3").

Duante os testes, realizou-se o processo de alocação variando-se a política de alocação de máquinas virtuais utilizada pelo VMM. As políticas de alocação implementadas foram *round-robin* (RR), *first-fit* (FF), *first-fit decreasing* (FFD) e *volume*.

5.3.2 Análise dos resultados

Para estes testes considerou-se que a alocação das máquinas virtuais ocorria após todas serem requisitadas. A ordem de requisição das VMs era aleatória e a ordem das máquinas físicas era fixa, determinada pela modelagem no iSPD. Para cada política se repetiu a alocação um total de 100 vezes, conferindo-se, ao fim das execuções, o número médio de VMs rejeitadas, número médio de núcleos ocupados e taxa de ocupação média dos núcleos que compõe a infraestrutura física, como apresentado na Tabela 8. Uma análise dos valores obtidos para cada política é apresentada a seguir.

Tabela 8 – Estatísticas de alocação de VMs segundo diferentes políticas

Política de alocação	VMs rejeitadas	Núcleos ocupados	Taxa de ocupação
RR	0,90	17,34	0,75
FF	0,89	18,62	0,81
FFD	0,00	23,00	1,00
Volume	1,00	15,73	0,68

Política *round-robin*

Observa-se, pela análise das simulações realizadas e pelos dados da Tabela 8, que esta política apresentou uma média de rejeição de 0,9 máquinas virtuais, por rodada de alocação, com no máximo uma VM rejeitada por instância. Seu desempenho relativamente fraco se justifica pelo comportamento da heurística utilizada pela política, que se preocupa apenas em distribuir uniformemente as VMs entre os recursos existentes, não verificando informações sobre a demanda de recursos solicitada. Isto contribui para que esta política apresente menores taxas de ocupação dos recursos em relação às políticas FF e FFD, pois a alocação das VMs menores para os recursos acaba por afetar a alocação das VMs maiores, que solicitam 4 ou 8 núcleos.

Para verificar o comportamento descrito apresenta-se na Figura 45 a porcentagem de VMs rejeitadas de acordo com o número de núcleos solicitados. Observa-se que na maioria dos casos (85%), a VM rejeitada é a "V8" que solicita 8 núcleos (55%), seguida das VMs "V4-1" e "V4-2", com 4 núcleos (30%).

Política *first-fit*

Assim como para a política RR, a política FF não atendeu todas as requisições de VMs, obtendo um número médio de VMs rejeitadas de 0,89, como apresentado na Tabela 8. Aqui também houve no máximo a rejeição de uma VM em cada instância de teste.

Ressalta-se que como esta política aloca as VMs na primeira máquina física que atender seus requisitos de recursos, sem considerar informações sobre a demanda de recursos de cada

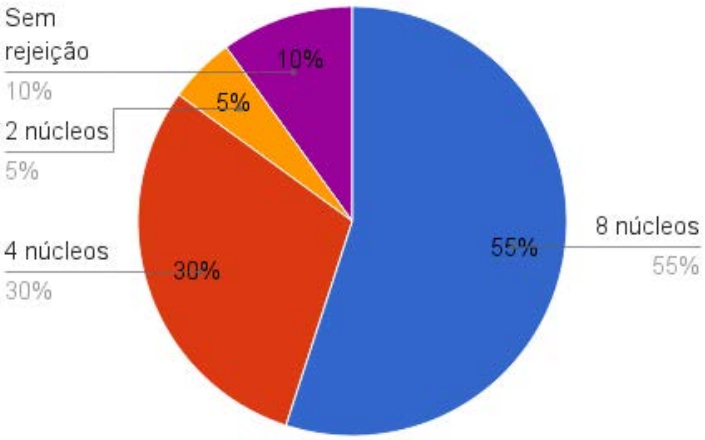


Figura 45 – Taxa de rejeição de VMs de acordo com o número de núcleos de processamento solicitados para a política *round-robin*

VM, esta política pode alocar VMs menores e que estão primeiro na ordem de atendimento em máquinas físicas maiores, fazendo com que VMs maiores, que solicitam quatro ou oito núcleos possam acabar sendo rejeitadas. No entanto, também existe uma tendência de mais máquinas menores serem alocadas para um mesmo recurso físico que as atenda simultaneamente, o que faz com que exista a possibilidade de máquinas maiores estarem disponíveis para atender VMs maiores, aumentando assim a taxa de utilização em relação à política RR.

Pode-se visualizar o comportamento descrito por meio da Figura 45, em que se apresenta a porcentagem de VMs rejeitadas de acordo com o número de núcleos solicitados. Observa-se neste caso que a maioria de rejeições ocorreram para as VMs “V4-1” e “V4-2”, com 4 núcleos (64%), enquanto a VM “V8” foi rejeitada apenas 22% das vezes.

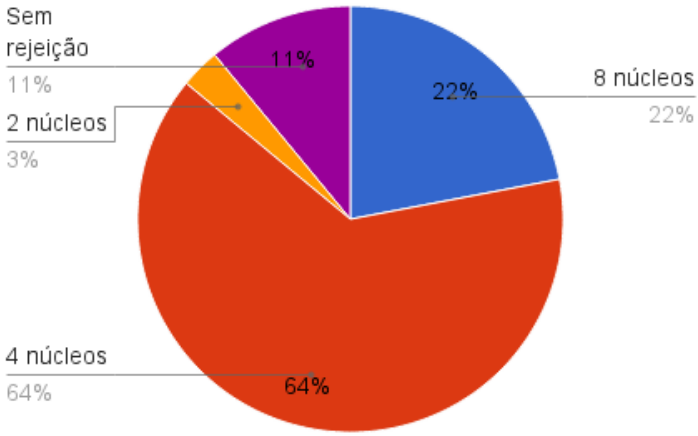


Figura 46 – Taxa de rejeição de VMs de acordo com o número de núcleos de processamento solicitados para a política *first-fit*

Política *first-fit decreasing*

Destaca-se como principal diferença desta política de alocação, em relação às políticas RR e FF, o fato dela considerar o volume de recursos solicitados pelas VMs, ordenando-as em ordem decrescente para realizar a alocação. Isto permite alocar primeiro as VMs que necessitam de um volume maior de recursos para serem atendidas, contornando os problemas encontrados por políticas que não consideram o volume de recursos solicitados pelas VMs para realizar a alocação.

Desta forma, verifica-se na Tabela 8 que esta política de alocação apresenta taxa de ocupação média de 100%, não ocorrendo rejeições para o ambiente de teste executado.

Política *Volume*

A política *volume* foi a política testada que obteve os piores resultados, com um número médio de uma VM rejeitada, como apresentado na Tabela 8. Houve apenas uma execução em que a alocação não resultou em rejeições e um caso em que duas máquinas virtuais foram rejeitadas, resultando em uma taxa de rejeição em 99% dos casos.

O alto índice de rejeição desta política se justifica pela heurística utilizada, dado que ordena as máquinas físicas para atendimento em ordem decrescente de recursos disponíveis. Isso faz com que a primeira máquina física selecionada para alocação seja a máquina "8core", fazendo com que a alocação de VMs menores nesta máquina física impossibilite o atendimento da a VM "V8", que solicita 8 núcleos.

Seu comportamento de rejeições é visto na Figura 47. Observa-se neste caso que a grande maioria das execuções (82,2%) acaba por rejeitar a VM "V8".

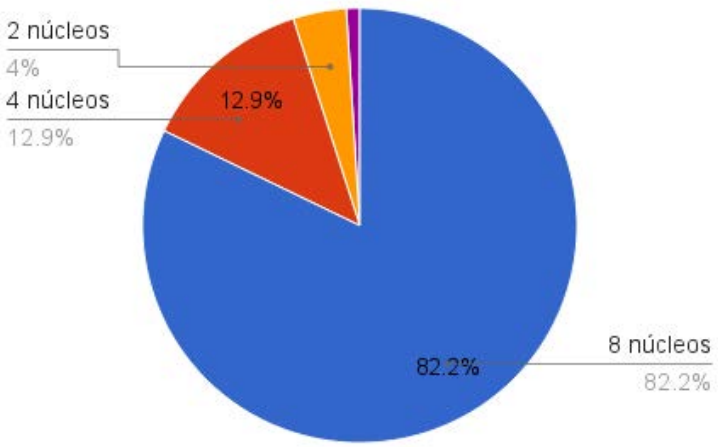


Figura 47 – Taxa de rejeição de VMs de acordo com o número de núcleos de processamento solicitados para a política *volume*

5.4 Considerações finais

Este capítulo apresentou uma variedade de testes com o intuito de validar a ferramenta desenvolvida neste trabalho. Os resultados obtidos mostraram que a simulação de IaaS com o iSPD é eficiente, com boa precisão e velocidade. Por fim, o teste de avaliação das políticas de escalonamento revelou que o iSPD se apresenta como uma ferramenta muito útil para o estudo de políticas de alocação.

6 Conclusões

Este trabalho apresentou o desenvolvimento de uma abordagem icônica para a modelagem e simulação de sistemas de computação em nuvem. O foco foi atender ao requisito de facilidade de uso, permitindo ao usuário modelar ambientes de computação em nuvem por meio de uma interface visual, utilizando ícones para especificar a infraestrutura física dos sistemas modelados e formulários de configuração para modelar o ambiente virtualizado (máquinas virtuais) e a carga de trabalho. Os resultados obtidos são resumidos a seguir.

6.1 Principais Resultados

O principal resultado obtido foi a inclusão da capacidade de simulação de computação em nuvem, da classe de IaaS, no iSPD. A vantagem em fazer a modelagem e simulação de IaaS através de uma abordagem icônica é a simplificação do processo isentando o usuário de possuir conhecimentos avançados de linguagens de programação e *frameworks* de desenvolvimento para a modelar esses sistemas.

Em paralelo também se desenvolveu um conjunto amplo de medidas e gráficos de desempenho, que permite avaliar com mais confiança os resultados da simulação, como alocação das VMs, escalonamento das tarefas, processamento da carga de trabalho, taxas de utilização dos recursos e tempo de simulação. Ressalta-se que esta funcionalidade não é oferecida, com este nível de detalhamento, pelos simuladores de computação em nuvem estudados, sendo que na maioria dos casos o usuário deve programar as métricas e resultados que deseja obter.

Outro ponto importante a ser destacado é o oferecimento de uma maior variedade de políticas de alocação de máquinas virtuais presentes nativamente no iSPD. Em comparação o CloudSim apresenta apenas a política "*first-fit*" implementada. Destaca-se ainda o desenvolvimento de janelas de edição para a implementação de novas políticas de alocação, caso as existentes não atendam as necessidades do usuário. A soma desses fatores positivos revela o simulador iSPD como uma ferramenta importante para o estudo de políticas de alocação de máquinas virtuais para ambientes de IaaS.

Os testes apresentados no capítulo anterior indicam ainda que a simulação de computação em nuvem com o iSPD é eficiente pois:

- Permite modelagem de forma rápida e simplificada;
- Tem boa precisão, uma vez que o erro médio ponderado pelo número de tarefas executadas, quando comparado a ambientes reais ficou abaixo de 2,2%;

- Tem boa velocidade de simulação, principalmente para testes com cargas de trabalho maiores, apresentando ordem de complexidade temporal linear contra uma complexidade exponencial do CloudSim;
- Tem boa flexibilidade de aplicação, pois permite investigar o impacto de diferentes políticas de alocação de VMs com facilidade.

Deve-se destacar, entretanto, que a especificação da classe de IaaS, bem como a de PaaS, poderia ser melhor e mais simples se a literatura sobre computação em nuvem fosse mais consistente. Detalhes sobre as classes de serviço e políticas de alocação são difíceis de encontrar, e em vários casos, inexistentes. Outro problema de informação se relaciona aos ambientes reais de computação em nuvem, que não apresentam dados claros, sobre políticas de alocação de VMs, custos de implementação, etc. Isso dificulta a sua modelagem nos simuladores de nuvem.

6.2 Trabalhos futuros

Destacam-se como trabalhos futuros diversas melhorias na simulação de computação em nuvem no iSPD. Ressalta-se que algumas destas ações se encontram em fase de desenvolvimento dentro do grupo de pesquisa:

- **Desenvolvimento de mais políticas de alocação:** Para este ajuste deseja-se investigar e incluir políticas de alocação utilizadas por provedores de computação em nuvem que ainda não estejam implementadas no iSPD;
- **Simulação de falhas de execução do sistema:** A implementação desta característica permite a modelagem de sistemas de forma mais fidedigna, já que a simulação implementada atualmente é otimista e não considera falhas na rede, nos recursos de processamento e na execução das tarefas;
- **Implementação do conceito de migração:** Esta característica é importante por se tratar de uma das três capacidades básicas de virtualização. Desta forma, a implementação desta característica é importante para a posterior implementação de políticas de alocação e de provisionamento de VMs que utilizem esta capacidade, assim como para a implementação da simulação de métodos de recuperação de catástrofe.
- **Implementação da classe de serviços de PaaS:** Uma breve especificação da modelagem desta classe de serviço foi apresentada neste trabalho. A implementação desta classe de serviço para o iSPD já se encontra em desenvolvimento. A importância de desenvolver este tópico está no fato de existir uma demanda não atendida pelos simuladores existentes para a avaliação de desempenho desta classe de serviço, dado que usuários desejam estimar custos de hospedagem de sua aplicação em provedores de serviço de PaaS, além dos

provedores necessitarem estudar estratégias eficientes para provisionamento de recursos virtualizados que atendam os níveis de serviço das aplicações hospedadas.

6.3 Publicações

Resultados preliminares associados a este trabalho foram apresentados em em artigos já publicados, incluindo:

- "Development of a marked structure for traces of parallel and distributed systems" com a autoria de Diogo Tavares da Silva, Aleardo Manacero Jr., Denison Menezes, Renata Spolon Lobato e Roberta Spolon. Publicado no *Annual Simulation Symposium*, realizado na cidade de San Diego - CA, nos EUA, no ano de 2014. Conferência com *qualis* B1;
- "Modelagem e simulação de sistemas de computação em nuvem para a ferramenta iSPD" com a autoria Diogo Tavares da Silva, Aleardo Manacero Jr., Arthur Jorge, Renata Spolon Lobato, Denison Menezes e Roberta Spolon. Selecionado para sessão de apresentações orais na Escola Regional de Alto Desempenho de São Paulo do ano de 2014 (ERAD-SP 2014) realizado na UFABC, na cidade de São Bernardo do Campo;
- "Simulação de sistemas de computação em nuvem para o iSPD" com autoria de Diogo Tavares da Silva, Aleardo Manacero Jr., Arthur Jorge, Renata Spolon Lobato, Denison Menezes e Roberta Spolon. Publicado na 4ª edição da revista Interciência e Sociedade (ISSN impresso nº 2236-0468 e ISSN online nº 2238-1295).
- "Abordagem icônica de modelagem e simulação de sistemas de computação em nuvem utilizando o iSPD" com autoria de Diogo Tavares da Silva, Aleardo Manacero Jr., Arthur Jorge, Renata Spolon Lobato, Denison Menezes e Roberta Spolon e João Antônio Magri Rodrigues. selecionado para sessão de apresentações orais na Escola Regional de Alto Desempenho de São Paulo do ano de 2015 (ERAD-SP 2014) realizado na UNESP, na cidade de São José do Rio Preto;

Destaca-se que ainda que estão sendo elaborados artigos pra serem publicados em conferências ou revistas com *qualis* igual ou superior a B1.

Referências

AMAZON. Amazon EC2. Disponível em <http://aws.amazon.com/pt/ec2/>. 2015. Citado 2 vezes nas páginas 24 e 28.

ARMBRUST, M. et al. *Above the Clouds: A Berkeley View of Cloud Computing*. [S.l.], 2009. Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>. Citado na página 20.

BANKS, J. et al. *Discrete-Event System Simulation*. 3nd edition. ed. [S.l.]: Prentice-Hall, 2001. Citado na página 44.

BUX, M.; LESER, U. Dynamiccloudsim: Simulating heterogeneity in computational clouds. In: *Proceedings of the 2Nd ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*. New York, NY, USA: ACM, 2013. (SWEET '13), p. 1:1–1:12. ISBN 978-1-4503-2349-9. Disponível em: <http://doi.acm.org/10.1145/2499896.2499897>. Citado na página 27.

BUYYA, R.; BROBERG, J.; GOSCINSKI, A. *Cloud Computing: Principles and Paradigms*. Wiley, 2011. ISBN 9781118002209. Disponível em: <http://books.google.com.br/books?id=S1NvRRd77rQC>. Citado 8 vezes nas páginas 9, 16, 19, 20, 21, 22, 23 e 24.

BUYYA, R.; MURSHED, M. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Pract. and Exper.*, v. 14, p. 1175–1220, 2002. ISSN 0038-0644. Citado na página 31.

BUYYA, R.; RANJAN, R.; CALHEIROS, R. N. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. *CoRR*, abs/0907.4878, 2009. Citado 2 vezes nas páginas 17 e 27.

CAMATI, R. S. *Novos algoritmos para alocação de máquinas virtuais e um novo método de avaliação de desempenho*. Dissertação (Dissertação de mestrado) — Pontifícia Universidade Católica do Paraná - PUC, 2013. Disponível em http://<https://www.ppgia.pucpr.br/pt/arquivos/mestrado/dissertacoes/2013/ricardo-stegh.pdf>. Citado na página 26.

CASANOVA, H. Simgrid: a toolkit for the simulation of application scheduling. In: *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001*. [S.l.: s.n.], 2001. p. 430–437. Citado na página 31.

CASTANE, G. G.; NUNEZ, A.; CARRETERO, J. icancloud: A brief architecture overview. In: *ISPA*. IEEE, 2012. p. 853–854. ISBN 978-1-4673-1631-6. Disponível em: <http://dblp.uni-trier.de/db/conf/ispa/ispa2012.html#CastaneNC12>. Citado 3 vezes nas páginas 17, 27 e 28.

COMPAQ. *Internet Solutions Division Strategy for Cloud Computing*. [S.l.], 1996. Disponível em <http://www.technologyreview.com/sites/default/files/legacy/compaq_cst_1996_0.pdf>. Citado na página 19.

DMTF. *Open Virtualization Format Specification*. [S.l.], 2012. Disponível em <http://dmtf.org/sites/default/files/standards/documents/DSP0243_1.1.0.pdf>. Citado na página 23.

GOOGLE. Google AppEngine, Power your business with google cloud. Disponível em <https://developers.google.com/appengine/>. 2013. Citado na página 24.

GOOGLE. Google Compute. Disponível em <https://cloud.google.com/compute/>. 2015. Citado 2 vezes nas páginas 24 e 72.

GOOGLE. Google docs. Disponível em <docs.google.com/?hl=pt-BR>. 2015. Citado na página 25.

GROZEV, N. Cloudsim and cloudsimex. Disponível em <https://nikolaygrozev.wordpress.com/2014/06/08/cloudsim-and-cloudsimex-part-1/>. 2015. Citado na página 27.

GSPD. Gspd's homepage. Disponível em <http://www.dcce.ibilce.unesp.br/spd/>. 2012. Citado na página 31.

ISI, I. S. I. Network simulator - ns2. Disponível em <http://www.isi.edu/nsnam/ns/>. 2013. Citado na página 28.

KLIAZOVICH, D. et al. GreenCloud: A packet-level simulator of energy-aware cloud computing data centers. In: *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. [S.l.: s.n.], 2010. p. 1–5. ISSN 1930-529X. Citado 2 vezes nas páginas 27 e 28.

KVM. Kernel based virtual machine. Disponível em <http://www.linux-kvm.org/page/Main_Page>. 2013. Citado na página 22.

LUBLIN, U.; FEITELSON, D. G. The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, v. 63, p. 2003, 2001. Citado na página 43.

MANACERO, A. et al. iSPD: an iconic-based modeling simulator for distributed grids. In: *Annals of 45th Annual Simulation Symposium*. Orlando, USA: [s.n.], 2012. (ANSS12, CDROM), p. 1–8. Citado 3 vezes nas páginas 9, 31 e 44.

MCKINSEY; CO. *Clearing the Air on Cloud Computing*. [S.l.], 2009. Technical Report. Citado na página 20.

MELL, P.; GRANCE, T. *The NIST Definition of Cloud Computing*. Gaithersburg, MD, 2011. Disponível em: <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>. Citado 3 vezes nas páginas 20, 21 e 24.

MICROSOFT. Office web apps. Disponível em <http://office.microsoft.com/pt-br/web-apps/>. 2015. Citado na página 25.

MISHRA, M.; SAHOO, A. On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In: *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*. Washington, DC, USA: IEEE Computer Society, 2011. (CLOUD '11), p. 275–282. ISBN 978-0-7695-4460-1. Disponível em: <http://dx.doi.org/10.1109/CLOUD.2011.38>. Citado na página 25.

MIT. Who coined “cloud computing”? Disponível em <http://http://www.technologyreview.com/news/425970/who-coined-cloud-computing/>. 2015. Citado na página 19.

NATHANI, A.; CHAUDHARY, S.; SOMANI, G. Policy based resource allocation in iaas cloud. *Future Gener. Comput. Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 28, n. 1, p. 94–103, jan. 2012. ISSN 0167-739X. Disponível em: <http://dx.doi.org/10.1016/j.future.2011.05.016>. Citado na página 25.

NUNEZ, A. et al. iCanCloud: A flexible and scalable cloud infrastructure simulator. *J. Grid Comput.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 10, n. 1, p. 185–209, mar. 2012. ISSN 1570-7873. Disponível em: <http://dx.doi.org/10.1007/s10723-012-9208-5>. Citado na página 28.

OMNET++. Welcome to the omnet++ community! Disponível em <http://www.omnetpp.org/>. 2013. Citado na página 28.

RITTINGHOUSE, J.; RANSOME, J. F. *Cloud Computing: Implementation, Management, and Security*. [S.l.]: CRC, 2009. Citado 2 vezes nas páginas 16 e 19.

ROCHA, L. A. Realcloudsim version 9.8 documentation. Disponível em <http://download2.polytechnic.edu.na/pub4/sourceforge/r/re/realcloudsim/RealCloudSim /REALcloudSimV98_documentation.pdf>. 2013. Citado na página 27.

SILVA, D. T. et al. Development of a marked structure for traces of parallel and distributed systems. In: *Proceedings of the 2014 Annual Simulation Symposium*. San Diego, CA, USA: Society for Computer Simulation International, 2014. (ANSS ’14), p. 6:1–6:8. Disponível em: <http://dl.acm.org/citation.cfm?id=2664292.2664298>. Citado na página 44.

THOMSEN, E. *OLAP Solutions: Building Multidimensional Information Systems*. Wiley, 2002. ISBN 9780471400301. Disponível em: <http://books.google.com.br/books?id= vVZIYeVHOcwC>. Citado na página 23.

VMWARE. WMware ESXi e ESX. Disponível em <http://www.vmware.com/br/products/esxi-and-esx/overview.html>. 2013. Citado na página 22.

WICKREMASINGHE, B.; CALHEIROS, R. N.; BUYYA, R. Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 446–452, 2010. ISSN 1550-445X. Citado na página 27.

XEN. Why Xen Project? Disponível em <http://www.xenproject.org/users/why-the-xen-project.html>. 2013. Citado na página 22.

Anexos

ANEXO A – Arquivo de programa cliente utilizado no teste de validação frente ao *Google Compute Engine*

Segue em anexo o código fonte do programa "`client.py`", escrito em Python, que é utilizado para executar o programa cliente na VM de *front-end* da infraestrutura de nuvem desenvolvida na plataforma *Google Compute Engine*. Uma breve descrição do processo implementado é descrito a seguir:

- Primeiramente, nas linhas 1,2 e 3, realiza-se a importação das bibliotecas que disponibilizam os métodos que oferecem suporte a *sockets*, medição de tempo e *multi threading*, respectivamente;
- Entre as linhas 4 e 19 é definido o método *"cliente(ip)"*, que recebe como parâmetro o endereço IP utilizado pela VM que está hospedando o programa servidor (descrito no Anexo B) e que tem como função implementar umas das *threads* que deverá enviar solicitações ao programa servidor. A sequência de ações realizadas por este método é descrita como segue:
  - Primeiramente define-se nas linhas 5 e 6 que o endereço a ser acesso é o endereço recebido pelo parâmetro `ip` e que a comunicação ocorrerá na porta 5000;
  - Na linha 7 instância-se um *socket*, configura-se os parâmetros da comunicação, que ocorrerá utilizando-se o protocolo TCP;
  - Na linha 8 define-se como endereço de destino do *socket* instanciado a tupla constituída pelo endereço IP e a porta definidos nas linhas 5 e 6;
  - Na linha 9 abre-se a conexão definida nas linhas anteriores;
  - Entre as linhas 13 e 18 implementa-se o laço de repetição que envia um número `n` de solicitações (que deve ser inserido no código). A cada solicitação envia-se o valor 0.2 e se recebe como resultado o tempo decorrido na execução de cada solicitação pelo programa servidor;
  - Na linha 19 encerra-se a conexão realizada, encerrando-se a implementação deste método.
- Por fim, entre as linhas 23 e 34, são criadas e disparadas quatro *threads* que executam o método *"cliente(ip)"*, passando como parâmetro para cada *thread* o endereço IP (que deve

ser inserido no código) de cada uma das VMs que implementam os nós de processamento do ambiente de testes.

Apresenta-se o código do programa cliente a seguir:

```
1.  import socket
2.  import time
3.  from threading import Thread
4.  def cliente(ip):
5.          HOST = ip     # Endereco IP do Servidor
6.          PORT = 5000            # Porta que o Servidor esta
7.          tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8.          dest = (HOST, PORT)
9.          tcp.connect(dest)
10.         print 'Para sair use CTRL+X\n'
11.         # onde está a letra n deve-se entrar o número de ..
12.         # .. repetições de solicitações de processamento
13.         for i in range(n):
14.                 msg = '0.2'
15.                 tcp.send (msg)
16.                 rec = tcp.recv(1024)
17.                 result = float(rec)
18.                 print "resultado:",result, "   de: ", ip
19.         tcp.close()
20.
21. # onde esta a string 'ip' deve-se entrar o ip do nó onde o ..
22. # .. arquivo "server.py" está executando
23. th1 = Thread(target = cliente, args = ('ip', ))
24. th1.start();
25. th2 = Thread(target = cliente, args = ('ip', ))
25. th2.start();
27. th3 = Thread(target = cliente, args = ('ip', ))
28. th3.start();
29. th4 = Thread(target = cliente, args = ("ip", ))
30. th4.start()
31. th1.join()
32. th2.join()
33. th3.join()
34. th4.join()
```

ANEXO B – Arquivo de programa servidor utilizado no teste de validação frente ao *Google Compute Engine*

Segue em anexo o código fonte do programa "`server.py`", escrito em Python, que é utilizado para executar o programa servidor nos nós que integram a infraestrutura de nuvem desenvolvida na plataforma *Google Compute Engine*. Uma breve descrição do processo implementado é descrito a seguir:

- Primeiramente, realiza-se nas linhas 1 e 2 a importação das bibliotecas que disponibilizam os métodos que oferecem suporte a *sockets* e medição de tempo, respectivamente;
- Nas linhas 3 e 4 define-se o endereço IP da VM com que irá se estabelecer conexão (que fica em aberto, dado que este programa implementa um servidor) e o número de porta em que se irá comunicar, configurado como 5000;
- Na linha 5 instancia-se um *socket* definindo que a conexão irá ocorrer utilizando o protocolo TCP;
- Na linha 6 define-se o endereço do elemento do elemento com quem se deseja comunicar (Endereço IP, porta);
- Nas linhas 7 e 8 configura-se o *socket* para escutar as mensagens recebidas na porta definida na linha 6;
- Na linha 9 define-se um laço infinito para aceitar mais de uma conexão simultânea e em seguida na linha 10 o método *"accept()"* retorna os objetos que implementam o *socket* utilizado para comunicação com o cliente que solicita conexão e o endereço IP desse cliente como retorno para as variáveis `con` e `cliente`, respectivamente;
- Por fim, implementa-se entre as linhas 12 e 23 o laço responsável por aguardar por mensagens através do *socket* instanciado no objeto `con` e a cada mensagem recebida executar 20 milhões de somas de ponto flutuante, contabilizando o tempo decorrido no processo de enviando este valor ao cliente que solicitou o trabalho.

Apresenta-se a seguir o código do programa servidor:

```
1.  import socket
2.  import time
3.  HOST = ''
4.  PORT = 5000
5.  tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6.  orig = (HOST, PORT)
7.  tcp.bind(orig)
8.  tcp.listen(10)
9.  while True:
10.         con, cliente = tcp.accept()
11.         print 'Conectado por', cliente
12.         while True:
13.                 msg = con.recv(1024)
14.                 if not msg: break
15.                 x = float(msg)
16.                 temp1=time.clock()
17.                 for i in range(4000):
18.                         for i in range(5000):
19.                                 y=x+x
20.                 temp2 = time.clock() - temp1
21.                 ret = str(temp2)
22.                 con.send(ret)
23.         con.close()
```