Avaliação de roteamento em redes P2P visando
obtenção de QoS na busca de serviços em nuvem

Dionisio Machado Leite Filho

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Avaliação de roteamento em redes P2P visando obtenção de QoS na busca de serviço em nuvem

Dionisio Machado Leite Filho

Orientadora: Profa. Dra. Regina Helena Carlucci Santana

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

USP – São Carlos
Maio de 2012

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

L533a Leite, Dionisio Machado
Avaliação de roteamento em redes P2P visando
obtenção de QoS na busca de serviço em nuvem /
Dionisio Machado Leite; orientadora Regina Helena
Carlucci Santana. -- São Carlos, 2012.
106 p.

Dissertação (Mestrado - Programa de Pós-Graduação em
Ciências de Computação e Matemática Computacional) --
Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2012.

1. Avaliação de Desempenho. 2. Qualidade de
Serviço. 3. Metaescalonadores. 4. P2P. 5. Simulação.
I. Santana, Regina Helena Carlucci, orient. II.
Título.

Agradecimentos

Agradeço primeiramente aos meus pais que sempre me motivam e incentivam tanto nos momentos felizes como nos momentos tristes. À professora Regina, minha orientadora, por ter aceitado me orientar e ter me guiado nos momentos em que eu necessitava e pela paciência em me orientar. Agradeço também os professores que contribuíram com a minha formação, em especial ao professor Marcos Santana que me deu conselhos valiosos no começo da minha pesquisa.

Aos meus amigos de república (Fabio, Stênio, Daniel, Koxonha) que me deram um apoio muito grande tanto nas dificuldades como nas alegrias.

Aos amigos de laboratório que me aguentam toda semana (Paulo, Bruno, Daniel, Fausto, Edvard, Roni, Edwin, Luis e os demais que estão chegando ou saindo) e ainda contribuem para discussões interessantes não só sobre trabalho mas também discussões do dia-a-dia.

Agradeço ao Maycon que, além de ser um amigo, me deu um apoio fundamental no início do meu mestrado e sempre esteve à disposição para discutir possíveis melhorias relacionadas ao meu trabalho.

Agradeço ao professor Julio pelas contribuições sempre bem vindas e pelas contribuições que ainda estão por vir.

Agradeço ao ICMC e principalmente ao LASDPC por proporcionar uma estrutura na qual eu pude crescer tanto como pessoa tanto como profissional.

Resumo

Este trabalho apresenta a avaliação de diferentes algoritmos de roteamento utilizados na camada lógica ponto a ponto (P2P) adotada por um Metaescalador que provê Qualidade de Serviços (QoS) na Computação em Nuvem. Experimentos mostram a superioridade de três algoritmos de roteamento P2P (BCR, Chord e Pastry) em relação à utilização de Round Robin, analisando-se o tempo de resposta e a variabilidade entre os resultados obtidos em diferentes testes. Os experimentos consideram, além dos algoritmos de roteamento, a influência do número de usuários e do tipo de serviço requisitado e como esses fatores interagem entre si. É apresentado ainda um estudo sobre a melhor métrica a ser adotada para representar as informações da rede. As métricas consideradas foram latência e número de saltos. Os resultados obtidos permitem determinar, com base nos objetivos especificados, qual o impacto dos sistemas P2P utilizados pelo metaescalador na busca e descoberta de serviços em relação à forma como a qualidade de serviços é abordada.

Abstract

This work presents an evaluation of different routing algorithms that are employed in a logical layer peer-to-peer (P2P) that are adopted by a Metascheduler that provides quality of services (QoS) in Cloud Computing. The experiments show the superiority of three P2P routing algorithms (BCR, Chord, Pastry) in relation to Round Robin utilization, analysing the response times and the variation between the results obtained results in different tests. The experiments consider, besides the routing algorithms, the influence of the number of the users and the type of requested services and how these factors interact between themselves. Besides of this, it is presented a study about the better metric to be adopted to represent the network information. The considered metrics were the latency and number of hops. The obtained results allow to determine, based on specific objectives, the impact of the utilization of P2P systems by the metascheduler in the search and discovery of services in relation to the way that the QoS is performed.

Sumário

Resumo	iii
Abstract	v
Lista de Siglas	xiii
1 Introdução	1
1.1 Considerações Iniciais	1
1.2 Motivação e Objetivos	3
1.3 Estrutura	3
2 Computação em Nuvem	5
2.1 Considerações Iniciais	5
2.2 Virtualização	6
2.3 Modelo Econômico	7
2.4 Características da Computação em Nuvem	8
2.5 Tipos de Nuvem	9
2.6 Provedores de Computação em Nuvem	10
2.7 Desafios da Computação em Nuvem	12
2.8 Plataforma de Desenvolvimento	13
2.8.1 CloudSim	14
2.8.2 Brite	15
2.9 Considerações Finais	17
3 Metaescalonadores	19
3.1 Considerações Iniciais	19
3.2 Escalonadores Locais e Metaescalonadores	20
3.3 MACC	22
3.4 Qualidade de Serviço	24
3.5 Considerações Finais	26
4 Comunicação nas Nuvens	27
4.1 Considerações Iniciais	27
4.2 Descoberta de Recursos	27
4.3 P2P	28

4.3.1	Características dos Sistemas P2P	29
4.3.2	Redes de Sobreposição	30
4.4	Roteamento P2P em Sistemas Distribuídos	32
4.5	Modelo de Comunicação do MACC	34
4.6	Qualidade de Serviço em Nível de Rede	36
4.7	Considerações Finais	37
5	Desenvolvimento do Projeto	39
5.1	Considerações Iniciais	39
5.2	Modelagem da Rede	40
5.3	Projeto e Implementação	43
5.4	Algoritmos de Roteamento	45
5.4.1	Pastry	47
5.4.2	Chord	48
5.5	Considerações Finais	50
6	Resultados	51
6.1	Considerações Iniciais	51
6.2	Seleção da Métrica	52
6.3	Planejamento	58
6.4	Comparação entre as Políticas de Roteamento	59
6.4.1	Política <i>Round Robin</i> e Política BCR	59
6.4.2	Política Chord e Política BCR	64
6.4.3	Política Pastry e Política BCR	67
6.5	Política Pastry e Política Chord	70
6.6	Considerações Finais	72
7	Conclusões	75
7.1	Considerações Iniciais	75
7.2	Contribuições	76
7.3	Dificuldades Relacionadas ao Projeto	77
7.4	Trabalhos Futuros	77

Lista de Figuras

2.1	Ambiente de nuvem	6
2.2	Camadas da Arquitetura em Nuvem. Adaptado de Rimal et al. (2009).	8
2.3	Formas de Desenvolvimento da Nuvem. Adaptado de (Furht, 2010)	9
2.4	Arquitetura do Simulador. Adaptado de (Calheiros et al., 2010)	14
2.5	Módulos que Compõe o BRITE. Adaptado de (Medina et al., 2001)	16
3.1	Modelo Centralizado de Troca de Informações	21
3.2	Modelo Descentralizado de Troca de Informações	21
3.3	Modelo de Troca de Mensagens Hierárquicas	22
3.4	Arquitetura do MACC	23
4.1	Taxonomia das Redes P2P. Adaptado de Ranjan et al. (2006)	29
4.2	Espaço de Busca da CAN. Adaptado de Ratnasamy et al. (2002).	31
4.3	Espaço de Busca do Chord. Adaptado de Stoica et al. (2001).	31
4.4	Espaço de Busca do Pastry. Adaptado de Rowstron e Druschel (2001).	32
4.5	Visão Geral do Modelo de Comunicação Adotado pelo do MACC.	35
4.6	Modelo de Comunicação do MACC.	35
5.1	Modelo Topológico Utilizado	40
5.2	Distribuição Geográfica dos <i>data centers</i>	41
5.3	Rede Utilizada nos Experimentos com 30 Clientes e 15 <i>data centers</i>	42
5.4	Rede Utilizada nos Experimentos com 60 Clientes e 15 <i>data centers</i>	42
5.5	Diagrama das Classes Utilizadas (Calheiros et al., 2010).	43
5.6	Fluxo de Comunicação Utilizado pelo CloudSim. Adaptado de Calheiros et al. (2010).	45
5.7	Busca Utilizando Round Robin	46
5.8	Busca Utilizando BCR	47
5.9	Topologia Pastry	48
5.10	Busca Utilizando Pastry	48
5.11	Topologia Chord	49
5.12	Busca Utilizando Chord	49
6.1	Diferença nos Tempos de Resposta	53
6.2	Distribuição dos Tempos de Resposta	54
6.3	Influência dos Fatores na Variável de Resposta.	54
6.4	Diferença nos Tempos de Resposta (política Chord).	55

6.5	Distribuição dos Tempos de Resposta em relação às Réplicas	55
6.6	Influência dos Fatores no Segundo Experimento.	56
6.7	Diferença nos Tempos de Resposta (política Pastry).	56
6.8	Distribuição dos Tempos de Resposta em relação às Réplicas	57
6.9	Influência dos Fatores no Segundo Experimento.	57
6.10	Tempos de tomada de decisão dos algoritmos.	58
6.11	Histograma dos Tempos de Resposta.	61
6.12	Intervalos de Confiança	61
6.13	Gráfico de Valores Individuais.	62
6.14	Influência dos Fatores.	62
6.15	Principais Efeitos no Tempo de Resposta.	63
6.16	Interação entre os Fatores.	63
6.17	Histograma com os Tempos de Resposta (Chord e BCR).	64
6.18	Intervalos de Confiança	65
6.19	Tempos de Resposta Individuais.	65
6.20	Influência dos Fatores nos Experimentos (Chord e BCR).	66
6.21	Efeitos dos Fatores nos Tempos de Resposta.	66
6.22	Interação dos Fatores (Chord e BCR).	67
6.23	Histograma com os Tempos de Resposta (Pastry e BCR).	68
6.24	Intervalos de Confiança	68
6.25	Tempos de Resposta Individuais (Pastry e BCR).	68
6.26	Influência dos Fatores nos Experimentos (Pastry e BCR).	69
6.27	Efeitos dos Fatores nos Tempos de Resposta (Pastry e BCR).	69
6.28	Interação dos Fatores (Pastry e BCR).	69
6.29	Histograma com os Tempos de Resposta.	70
6.30	Intervalos de Confiança	71
6.31	Influência dos Fatores nos Experimentos.	71
6.32	Efeitos dos Fatores nos Tempos de Resposta.	72
6.33	Interação dos Fatores.	72

Lista de Tabelas

2.1	Comparação das Ofertas Comerciais	11
3.1	Comparação entre Metaescaladores	25
6.1	Fatores Fixos definidos para todos os Experimentos	52
6.2	Fatores Variáveis (1º Experimento)	53
6.3	Planejamento de Experimentos (RR e BCR)	60
6.4	Planejamento de Experimentos (Chord e BCR)	64
6.5	Planejamento de Experimentos (Pastry e BCR)	67
6.6	Planejamento de Experimentos (Chord e Pastry)	70

Lista de Siglas

API	-	<i>Application Programming Interface</i>
AS	-	<i>Autonomous System</i>
AWS	-	<i>Amazon Web Services</i>
BCR	-	<i>Baseada na Capacidade da Rede</i>
BoT	-	<i>Bag of Task</i>
CAN	-	<i>Content Addressable Network</i>
CASA	-	<i>Community Aware Scheduling Algorithm</i>
CDN	-	<i>Content Delivery Network</i>
CPU	-	<i>Central Processing Unit</i>
DNS	-	<i>Domain Name Service</i>
EC2	-	<i>Elastic Compute Cloud</i>
GB	-	<i>Gigabyte</i>
Gb	-	<i>Gigabit</i>
GRIS	-	<i>General Resource Information System</i>
IaaS	-	<i>Infrastructure as a Service</i>
IP	-	<i>Internet Protocol</i>
LAN	-	<i>Local Area Network</i>
LRAM	-	<i>Local Resource of Allocation Management</i>
LRMS	-	<i>Local Resource Management System</i>
MACC	-	<i>Metascheduler Architecture to provide QoS in Cloud Computing</i>
MIPS	-	<i>Million of Instructions Per Second</i>
MSDN	-	<i>Monitoring and Discovery System Manager</i>
NIST	-	<i>National Institute of Standards and Tecnology</i>
NWIRE	-	<i>Net-Wide-Resources</i>
PaaS	-	<i>Platform as a Service</i>
QoS	-	<i>Quality of Services</i>
RAM	-	<i>Random Access Memory</i>
RR	-	<i>Round Robin</i>
SaaS	-	<i>Software as a Service</i>
SGBD	-	<i>Sistema de Gerenciamento de Banco de Dados</i>
SLA	-	<i>Service Level Agreement</i>
SLS	-	<i>Service Location Service</i>
TCP	-	<i>Transmission Control Protocol</i>
TI	-	<i>Tecnologia da Informação</i>
UDDI	-	<i>Universal Description, Discovery and Integration</i>
VM	-	<i>Virtual Machine</i>

Introdução

1.1 Considerações Iniciais

Nos últimos anos, a forma como as aplicações e serviços são ofertados aos consumidores vem passando por mudanças significativas. Antes da concepção dos sistemas que utilizam a rede para o compartilhamento de recursos, para acessar um conversor de arquivos ou mesmo um editor de texto, os usuários precisavam realizar instalações locais desses aplicativos.

Uma mudança relacionada a esse cenário refere-se a utilização de redes locais (LANs) que permitem que programas possam ser acessados a partir de outros computadores sem a necessidade de instalação e configuração em todos os computadores participantes da rede. Esse foi um dos primeiros passos para a adoção de sistemas distribuídos para o compartilhamento de dados e recursos entre computadores (Tanenbaum, 2003).

Um modelo de sistemas distribuídos mais utilizado para o compartilhamento e processamento de serviços é o modelo cliente-servidor, onde um ou mais computadores com maior capacidade tem a finalidade de armazenar, processar e prover acesso aos dados e serviços (Kurose et al., 2010).

Com o surgimento da *World Wide Web* (Berners-Lee, 2001) os sistemas cliente e servidor passaram a ser cada vez mais utilizados e a partir dessa evolução surgiram novas formas de oferecimento de serviços como os portais Web e correios eletrônicos (e-mail) (Kurose et al., 2010).

Uma evolução dos sistemas oferecidos sobre a Web foi o surgimento da computação em nuvem, onde o fornecimento de recursos e serviços é feito através da Internet e os usuários

necessitam, na maior parte das vezes, utilizar um navegador Web para acessar serviços e dados que são armazenados e processados na nuvem (Armbrust et al., 2009).

A computação em nuvem é uma forma de utilizar recursos computacionais, tais como, memória, aplicativos, processadores e etc. como serviços. A forma de funcionamento da nuvem, se assemelha aos sistemas operacionais de rede (Wu et al., 2010), onde os recursos computacionais são fornecidos como um serviço regular. A diferença entre a computação em nuvem e sistemas operacionais de rede não é o objetivo mas sim as tecnologias que se uniram para realizar a formação da nuvem (Buyya et al., 2010).

De uma maneira geral, a computação em nuvem favorece a migração dos servidores de dentro de uma empresa para uma nuvem. Os requisitos necessários para o desenvolvimento de uma infraestrutura ou de um serviço são definidos e o fornecedor de serviços em nuvem cria essa infraestrutura (Zhang et al., 2010).

Com a migração para a nuvem os custos ficam relacionados apenas aos recursos que são utilizados, o que caracteriza o modelo **pague o que utilizar**. Esse modelo de negócios visa reduzir os preços com infraestrutura e manutenção de equipamento uma vez que os gastos com manutenção ficam a cargo dos provedores da computação em nuvem (Gupta, 2010). Com isso, uma das vantagens da computação em nuvem, além da redução dos preços, é a possibilidade de aumentar a capacidade de seus servidores de acordo com o aumento da demanda de seus serviços (Zhang et al., 2010).

No entanto, para a utilização da nuvem, é necessário que existam mecanismos de monitoramento para auxiliar no fornecimento e descoberta de serviços para os usuários. Devido a essa necessidade, Peixoto et al. (2010) propõe um metaescalador que auxilia o monitoramento e entrega de serviços aos usuários da computação em nuvem com qualidade de serviço (QoS).

Qualidade de Serviço (QoS) pode ser definida como a percepção do usuário em relação à eficiência de um dado serviço. QoS significa prover algum tipo de garantia sobre um serviço, tal como: perdas mínimas, desempenho máximo, pequeno tempo de resposta, exatidão e consistência dos dados recebidos (Vasiliou, 2000). Dentre essas características, o metaescalador proposto utiliza o fornecimento de baixos tempos de resposta como garantia de QoS na comunicação entre nuvens.

O metaescalador tem como objetivo realizar a manutenção dos critérios de QoS, fazendo com que os serviços sejam entregues de forma transparente aos clientes. Por se preocupar com a qualidade de serviço fim-a-fim, o metaescalador proposto por Peixoto et al. (2010), auxilia em prover QoS na execução das requisições dos usuários e na entrega dos serviços. O metaescalador, para realizar a entrega e a descoberta de recursos, propõe a utilização de um modelo de comunicação que utiliza sistemas P2P para o fornecimento de QoS fim-a-fim.

Esse metaescalador é desenvolvido de forma modular e dentre os seus módulos está o escalonamento. No módulo de escalonamento estão os escalonadores locais e as políticas de roteamento que são utilizadas por esse metaescalador em sua camada de comunicação.

Sendo assim, este projeto de mestrado visa a avaliar essa camada de comunicação do meta-escalador e a partir de metodologias de avaliação de desempenho determinar quais as vantagens da adoção dos sistemas P2P na obtenção de QoS na computação em nuvem.

1.2 Motivação e Objetivos

A motivação para a proposta desse projeto de mestrado foi verificar que há vários trabalhos com foco na definição dos conceitos relacionados à computação em nuvem e em como os serviços são executados. Jones (2008), Rimal et al. (2009), Armbrust et al. (2009), Furht (2010), Zhang et al. (2010), Mell e Grance (2011) entre outros, se preocupam em definir quais são as características, funções e lacunas apresentadas pela computação em nuvem.

Trabalhos como os apresentados por Li et al. (2009), You et al. (2009), Li (2009) entre outros, tem o objetivo de analisar as cargas de trabalho e como as formas de virtualização são aplicadas na nuvem. Como o ambiente está na Internet, um dos pontos a ser analisado é a comunicação entre os clientes dos serviços disponibilizados pela nuvem. Graffi et al. (2010), Lai et al. (2010), Ranjan et al. (2010) e Zhou et al. (2011) apresentam conceitos relacionados à busca de serviços utilizando sistemas P2P.

Apesar de haver estudos relacionados à comunicação entre as nuvens e utilização de sistemas P2P, não existe uma avaliação de desempenho que demonstre a razão da seleção de um sistema P2P (Pastry e/ou Chord) em detrimento de outro e muito menos como é feita a integração entre esses sistemas e as demais funcionalidades oferecidas pela nuvem para auxiliar na obtenção de QoS.

O objetivo deste trabalho de mestrado é avaliar os sistemas P2P que podem ser utilizados para a descoberta de serviços e levantar quais são suas vantagens e desvantagens dependendo do domínio da aplicação que está sendo avaliada e como os sistemas P2P contribuem na obtenção de baixos tempos de resposta.

Para alcançar o objetivo proposto, foi realizada em cada etapa do projeto uma avaliação de desempenho para verificar quais métricas utilizar e quais sistemas avaliar a fim de obter bons tempos de resposta contribuindo assim para a definição de qual sistema P2P o metaescalador proposto por Peixoto et al. (2010) pode utilizar e em que casos específicos.

1.3 Estrutura

Os próximos capítulos serão estruturados da seguinte forma:

- No Capítulo 2 serão apresentados os conceitos relacionados à computação em nuvem. Os conceitos apresentados permitem verificar quais as funcionalidades oferecidas, bem como os desafios proporcionados por esse modelo computacional;

- O Capítulo 3 apresenta os conceitos referentes aos metaescalonadores. Nesse capítulo será apresentado o metaescalonador ao qual esta proposta está vinculada. Além de apresentar o metaescalonador, no capítulo serão discutidas quais as diferenças deste metaescalonador com os demais metaescalonadores encontrados na literatura;
- O Capítulo 4 apresenta os conceitos de comunicação em nuvem, com o foco em sistemas P2P. Esse capítulo irá auxiliar na definição de qual sistema P2P utilizar nos experimentos a serem realizados;
- No Capítulo 5 serão discutidos os métodos utilizados para o desenvolvimento das políticas de roteamento definidos para o metaescalonador. Ainda nesse capítulo será apresentado o ambiente de testes utilizado na elaboração deste projeto;
- O Capítulo 6 por sua vez, detalha os resultados de acordo com as políticas que foram simuladas. Os resultados devem ajudar a identificar dentre as políticas utilizadas, em quais condições uma política pode ser melhor que as demais;
- No Capítulo 7 são apresentadas as conclusões deste projeto bem como as dificuldades encontradas e os trabalhos futuros.

Por fim são apresentadas as referências bibliográficas.

Computação em Nuvem

2.1 Considerações Iniciais

A computação em nuvem é assim intitulada por se tratar de um modelo computacional que permite o acesso a recursos e informações por meio da Internet e não localmente. A nuvem facilita o uso de infraestrutura e plataforma computacional que pode ser dinamicamente configurada e reconfigurada de acordo com as necessidades dos usuários (Zhang et al., 2010).

De acordo com o NIST (*National Institute of Standards and Technology*), a computação em nuvem é um modelo que permite o acesso conveniente a conjuntos de recursos compartilhados, tais como: redes, servidores, armazenamento, aplicações e serviços, que podem ser disponibilizados ou liberados com certo esforço de gestão ou de interação com o provedor de serviços (Mell e Grance, 2011). Nesse modelo, os usuários acessam os serviços (*software*, plataforma e infraestrutura) utilizando, na maior parte das vezes, um navegador Web (Zhang et al., 2010).

A nuvem é uma analogia a um conjunto de componentes que executam serviços para os clientes sob demanda e, geralmente, o consumidor não tem ideia de onde estes serviços estão, por isso é dito que os serviços estão nas nuvens (Rimal et al., 2009) conforme apresentado na Figura 2.1.

O funcionamento do sistema em nuvem se assemelha aos sistemas operacionais de rede, sistemas de tempo real e grades computacionais, onde os recursos computacionais são fornecidos como um serviço regular (Kim et al., 2009b) (Wu et al., 2010). A diferença entre a nuvem dos demais sistemas que a compõe não é o objetivo, mas sim as tecnologias existentes que se uniram para realizar a sua formação (Buyya et al., 2010).

Figura 2.1: Ambiente de nuvem

Além dos sistemas de grades computacionais e sistemas de tempo real, os serviços de e-mail gratuito, serviços de busca, portais de Internet, serviços de hospedagem Web e alguns serviços de infraestrutura computacional são exemplos de serviços oferecidos pela nuvem e que já eram utilizados antes da concepção do termo (Kim et al., 2009b).

Além disso, esse sistema computacional incorpora virtualização, desenvolvimento sob demanda e entrega de serviços Web na Internet. Não existe uma inovação em arquitetura e infraestrutura, uma vez que esse sistema utiliza abordagens, conceitos e práticas que já foram estabelecidos há certo tempo (Carolan et al., 2009).

O que a difere dos outros sistemas que entregam serviços é a utilização da virtualização, que explora ao máximo o uso do hardware disponível (Carolan et al., 2009). A virtualização pode ser definida como a emulação de vários computadores lógicos em um único computador real. Essa característica faz com que os usuários tenham a impressão de usarem um recurso físico ao invés de um recurso virtual (Chieu et al., 2009).

2.2 Virtualização

De uma maneira geral, a computação em nuvem é a migração dos servidores de dentro de uma empresa para a nuvem. O usuário define os requisitos necessários para que seus recursos sejam executados, como: processamento, rede de longa distância e largura de banda. E o provedor cria virtualmente esses servidores dentro da infraestrutura de nuvem (Zhang et al., 2010).

A facilidade de acesso aos serviços sob demanda conduz a classificação da nuvem como um paradigma para a computação de serviços dinâmicos que são geralmente apoiados por *data centers*. Os *data centers* podem ser considerados como um conjunto de máquinas virtuais em rede (Buyya et al., 2010).

O conjunto de máquinas virtuais em rede só é possível devido à virtualização, que é a diferença tecnológica da computação em nuvem para os demais sistemas computacionais. A

virtualização se refere principalmente à melhor utilização de recursos físicos pelos usuários e por suas aplicações. Além disso, permite que os servidores, dispositivos de armazenamento e hardware sejam considerados como um único sistema, ao invés de sistemas separados, permitindo que esses recursos possam ser alocados dinamicamente pelos servidores (Chieu et al., 2009). A virtualização é adotada pela nuvem, pois, oferece vantagens no compartilhamento, gerenciamento e isolamento de recursos físicos. Cada máquina virtual pode ter seu próprio sistema operacional, aplicações e serviços de armazenamento e rede (Carissimi, 2008) (Chieu et al., 2009).

A computação em nuvem, com a utilização de virtualização, permite que um único servidor físico execute vários outros servidores lógicos que podem ter objetivos diferentes e sistemas diferenciados uns dos outros (Chieu et al., 2009). Ao permitir essa execução, o uso dos recursos na nuvem se torna mais eficiente, reduzindo os custos operacionais e de gestão da infraestrutura (Zhang et al., 2010).

2.3 Modelo Econômico

O pagamento pelos recursos da nuvem utilizados é um diferencial quando comparado a outras abordagens computacionais para a entrega de serviços. Do ponto de vista empresarial, a natureza sob demanda desse sistema ajuda a incorporar aspectos de desempenho e capacidade em nível de serviço (Carolan et al., 2009). Essa natureza econômica permite que as organizações criem ambientes dinâmicos onde recursos podem ser aumentados ou diminuídos com base na carga de trabalho e nos parâmetros de desempenho do serviço a ser executado. O pagamento por recursos computacionais pode assumir a forma de contratos de locação de equipamentos garantindo um nível mínimo de serviço por parte do provedor de nuvem (Carolan et al., 2009).

Com a migração para a nuvem o usuário passa a pagar apenas pelos recursos efetivamente utilizados, o que caracteriza o modelo **paye o que utilizar**, onde as garantias são oferecidas pelo provedor de infraestrutura por meio de acordos em nível de serviço personalizado ou SLA (*Service Level Agreement*). O SLA serve como um nível de contrato de serviços, entre o provedor e o cliente, que especifica o nível em que o serviço será prestado (Gupta, 2010).

Assim, ao invés de negociar com uma organização de tecnologia de informação (TI) os recursos de infraestrutura ou plataforma necessários para a implantação de um serviço, é proposto o modelo de auto atendimento onde o cliente compra ciclos de computação e recebe uma interface Web ou uma interface de programação de aplicativos (API) para a criação de máquinas virtuais (Carolan et al., 2009).

Não é necessário um contrato de longo prazo entre o cliente e os prestadores de serviços, os custos relacionados aos serviços são cobrados por utilização, e um aplicativo pode ser criado para executar uma tarefa por poucos minutos ou poucas horas. Nesse sistema os aplicativos ou serviços são considerados temporários e o pagamento é baseado no consumo de recursos como:

CPU, volumes de dados movidos ou quantidade de dados armazenados (Furht, 2010) (Carolan et al., 2009).

A regra de negócios utilizada pela nuvem transfere os custos de manutenção da infraestrutura que antes eram dos clientes para o provedor de serviços (Carolan et al., 2009). Essa inversão da responsabilidade tem consequências significativas. No passado, os clientes determinavam como os diversos componentes de uma aplicação seriam definidos em um conjunto de servidores. Agora, um cliente pode usar a API de um provedor para criar não apenas a composição inicial de uma aplicação, mas também como ela irá evoluir para acomodar as mudanças de carga de trabalho (Carolan et al., 2009).

2.4 Características da Computação em Nuvem

Os serviços na nuvem podem ser considerados como uma arquitetura de camadas onde vários recursos estão disponíveis em cada uma das camadas que formam a nuvem. Essas camadas podem ser consideradas como um conjunto escalável de recursos virtualizados que são capazes de hospedar aplicações e fornecer serviços aos usuários (Zhang et al., 2010). No âmbito da computação em nuvem pode-se considerar que diferentes tipos de serviços podem ser oferecidos, tais como: *software*, *hardware*, plataforma, dados, infraestrutura, dentre outros (Rimal et al., 2009). Os serviços podem ser classificados em várias camadas, que podem ser agrupadas em três principais categorias (Ohlman et al., 2009) (Mell e Grance, 2011) como apresenta a Figura 2.2.

Figura 2.2: Camadas da Arquitetura em Nuvem. Adaptado de Rimal et al. (2009).

A primeira forma apresentada, **SaaS**, corresponde aos serviços que estão hospedados na nuvem. Esses serviços podem variar desde um serviço de e-mail até editores de texto. O usuário utiliza o serviço que pode ser pago ou gratuito (Peng et al., 2009) (Armbrust et al., 2009).

A segunda forma apresentada é o **PaaS**, também conhecida como plataformas para a computação em nuvem. Nesse nível o usuário interage na criação de aplicações e na publicação das

mesmas, ficando assim a estrutura definida pelo fornecedor, como as tecnologias que serão utilizadas (.NET, Java, SGBDs), de forma fixa. Cabe ao usuário apenas escolher dentre as opções de plataforma a que melhor se enquadra nas suas necessidades (Armbrust et al., 2009) (Zhang et al., 2010).

Por ultimo é apresentada a **IaaS** que é a forma mais básica de serviço oferecido aos usuários (Zhang et al., 2010). Nessa forma os usuários contratam a infraestrutura (máquinas virtuais) e armazenamento e a partir deste ponto o usuário tem liberdade para criar sua plataforma bem como interagir com os serviços (Rimal et al., 2009) (Zhang et al., 2010).

2.5 Tipos de Nuvem

As organizações podem optar por implantar aplicativos em três tipos de nuvens: públicas, privadas ou híbridas, como apresenta a Figura 2.3, cada qual com características diferentes (Mell e Grance, 2011).

Figura 2.3: Formas de Desenvolvimento da Nuvem. Adaptado de (Furht, 2010)

As **Nuvens Públicas** são gerenciadas por terceiros. Quem mantém as nuvens públicas fornece infraestrutura e serviços aos usuários. Este modelo é interessante para usuários que precisam de flexibilidade e serviços temporários para executar suas tarefas, ou programas, e reduzir custos com compra e manutenção de hardware (Furht, 2010). Os serviços prestados pelas nuvens públicas podem ser gratuitos como editores de texto, por exemplo, ou pagos como a Amazon EC2.

Nuvens Privadas são constituídas para prover recursos específicos para clientes particulares, isto é, a infraestrutura pode ser alugada ou mesmo pertencer ao cliente (Furht, 2010)

(Zhang et al., 2010). Os clientes são, em sua maioria, empresas ou instituições públicas ou privadas e neste modelo a infraestrutura pode ser modificada de acordo com as necessidades desses clientes.

A **Nuvem Híbrida** é um conjunto de funções ou utilização das duas abordagens citadas anteriormente. Este modelo pode compartilhar temporariamente o uso de recursos das nuvens públicas para garantir o desempenho de todos os serviços, quando uma empresa privada tem uma alta carga de trabalho (Furht, 2010) (Mell e Grance, 2011).

Os tipos de nuvens, como apresentado na Figura 2.3, não representam necessariamente a localização física da nuvem. As nuvens públicas estão tipicamente na Internet e nuvens privadas estão normalmente localizadas nas dependências de uma organização privada que pode não ser prestadora de serviços em nuvem.

Uma nuvem privada pode ser hospedada não só nas dependências da organização, como pode também estar presente em provedores de infraestrutura como a Amazon (Zhang et al., 2010).

As organizações podem fazer uma série de considerações a respeito do modelo ideal de nuvem e que atenda aos seus interesses e pode ser adotado mais de um modelo de nuvem para resolver problemas diferentes.

Um serviço temporário pode ser desenvolvido e utilizado de forma mais adequada em uma nuvem pública, pois evita a compra de equipamentos adicionais para resolver uma necessidade temporária (Carolan et al., 2009). Da mesma forma, um serviço permanente, ou que tem requisitos específicos sobre a qualidade do serviço ou da localização dos dados, pode ter uma implementação mais adequada se for considerada uma nuvem privada ou híbrida.

Em qualquer tipo de nuvem, os usuários não precisam saber onde os dados estão sendo processados ou armazenados, a ideia é que esses recursos estão na nuvem e podem ser acessados por meio da Internet (Carolan et al., 2009).

2.6 Provedores de Computação em Nuvem

As principais aplicações no domínio de nuvem incluem: Amazon Web Services, Microsoft Azure e *Google App Engine*. Esses provedores de serviços oferecem uma variedade de pacotes de serviços de monitoramento, gerenciamento e provisionamento de recursos e serviços (Furht, 2010).

Os *Webservices* da Amazon Web Services (AWS)¹: *Elastic Load Balancer*, *Auto Scaling* e *CloudWatch*, possuem funcionalidades que são necessárias para a alocação de serviços e recursos do Amazon EC2.

O serviço *Elastic Load Balancer* dispõe automaticamente a carga dos aplicativos de entrada em várias instâncias do EC2 disponibilizados pela Amazon. Já o serviço de *Auto Scaling* pode

¹<http://aws.amazon.com/ec2/>

ser usado para aplicar dinamicamente ou ampliar o número de instâncias do Amazon EC2, para tratar alterações na demanda dos serviços (Ranjan et al., 2010).

Finalmente, o serviço *CloudWatch* é integrado com os outros *Webservices* para tomada de decisões estratégicas com base em informações em tempo real dos recursos agregados e informações sobre o desempenho dos serviços (Ranjan et al., 2010).

O Windows Azure tem uma estrutura composta de nodos formados por servidores com balanceamento de carga. O *Fabric Controller* gerencia o nodo através de uma base de serviços denominado agente controlador Azure, que monitora o estado do servidor. Se uma falha é encontrada, o gerenciador pode administrar uma reinicialização do servidor ou a migração dos serviços do atual servidor para outros servidores que estejam funcionando normalmente (Zhang et al., 2010).

Ao contrário de outras plataformas em nuvem, o *Google App Engine* oferece aos programadores uma plataforma escalável. O acesso ao sistema operacional é restrito pelo *Google App Engine*. As estratégias de balanceamento de carga, serviço de configuração e dimensionamento são gerenciados automaticamente pelo sistema em segundo plano (Buyya et al., 2010).

A Tabela 2.1 apresenta uma comparação entre os principais fornecedores comerciais de computação em nuvem.

Tabela 2.1: Comparação das Ofertas Comerciais

	Amazon	Google	Microsoft
Tipo de Serviço	IaaS	PaaS – IaaS	PaaS - IaaS
Serviço Ofertado	Processamento e Armazenamento	Processamento	Processamento e Armazenamento
Interface de Acesso	Interface Web e Linha de Comando	Interface Web e Linha de Comando	Portal Azure
Virtualizador	Xen	Contêiner de Aplicação	Contêiner de Serviço
Plataforma	Windows/Linux	Linux	Windows

Os fornecedores apresentados na Tabela 2.1 são os principais no fornecimento de serviços em nuvem, sendo os mesmo apresentados por (Vecchiola et al., 2009), (Buyya et al., 2010), (Ranjan et al., 2010), (Zhang et al., 2010).

Como apresentado na Tabela 2.1, enquanto a Amazon possui apenas soluções IaaS, a Google e a Microsoft fornecem infraestrutura e plataforma para desenvolvimento de aplicações (Zhang et al., 2010).

O *Google App Engine* fornece um conjunto de APIs e um modelo de aplicação que permite aos desenvolvedores utilizar serviços adicionais fornecidos pelo Google, como o e-mail, *Google sites*, editores de texto entre outros (Vecchiola et al., 2009).

Devido à natureza dos serviços ofertados pelo Google, apenas serviços Web, o mesmo não fornece armazenamento. Diferentemente da Google, a Microsoft com a plataforma Azure ofe-

rece serviços de armazenamento com o *Azure Storage* e *SQL Data Service* e a Amazon oferece serviços de armazenamento com o S3 (Zhang et al., 2010).

Além de oferecer serviços, os fornecedores precisam se preocupar com questões relacionadas a segurança, tolerância a falhas e bons tempos de resposta para os usuários, fazendo com que a nuvem seja adotada com confiança.

2.7 Desafios da Computação em Nuvem

A computação em nuvem oferece uma série de benefícios em relação aos paradigmas computacionais anteriores a ela. No entanto, ainda há uma série de desafios, que estão abertos a pesquisa, tais como (Carolan et al., 2009) (Furht, 2010):

- **Desempenho** – Os serviços executados na nuvem apresentam sobrecargas adicionais em relação aos serviços executados localmente. Devem ser consideradas as sobrecargas devido ao gerenciamento da nuvem e das máquinas virtuais. Além disso, os usuários que estão a distâncias consideráveis dos provedores podem ter alta latência e atrasos;
- **Segurança** – As aplicações devem fornecer acesso apenas a usuários autorizados e autenticados e os usuários precisam ser capazes de confiar que seus dados não foram manipulados por outras pessoas. A segurança no ambiente de nuvem deve ser estabelecida por meio de autenticação forte, autorização e procedimentos nas contas dos usuários, estabelecendo a segurança dos dados no início das transações, tráfego das informações e finalização das sessões. A segurança deve ser integrada em todos os aspectos da aplicação e sua implantação na arquitetura das nuvens deve ocorrer em todos os processos;
- **Escalabilidade** – Os serviços projetados para a computação em nuvem precisam ser escaláveis de acordo com as cargas de trabalho que chegam a esses serviços. Para alcançar isso, os serviços e os dados devem ser flexíveis para maximizar a escalabilidade;
- **Disponibilidade** – Como o modelo de computação em nuvem segue o modelo de pague o que utilizar, o usuário espera que os serviços contratados estejam em contínuo funcionamento;
- **Confiabilidade** – Os componentes de um sistema que oferece serviço na nuvem devem ter as falhas minimizadas e a sua manutenção deve ser possível sem interrupção dos serviços prestados. Um ponto crucial em termos de confiabilidade está relacionado com a perda de dados. A ideia é que as aplicações operem os dados, e esses permaneçam intactos, mesmo se apresentar falha em um ou mais servidores ou máquinas virtuais no qual os dados são armazenados ou processados;

- **Flexibilidade e agilidade** – A computação em nuvem pode melhorar o processo de criação de serviços, criando servidores de plataforma com serviços mais flexíveis, ou seja, suportando um grande número de frameworks de desenvolvimento e suportando várias linguagens de programação;
- **Manutenção** – Depois que um aplicativo é implantado, ele precisa ser mantido. Isso quer dizer que quando um serviço tem que ser reparado o tempo de inatividade tem que ser mínimo. Os componentes de um aplicativo ou de infraestrutura devem ser atualizados ou mesmo substituídos sem interromper o fornecimento do serviço;
- **Interoperabilidade** – As plataformas normalmente oferecem APIs de acordo com sua infraestrutura. É necessário que existam linguagens padronizadas entre os diversos fornecedores de serviço tanto em relação à infraestrutura como em relação a plataformas e serviços.

Avanços na arquitetura de aplicação ajudam a promover a computação em nuvem. Como pode ser observado nesta seção, os desafios e as preocupações com a utilização desse sistema computacional são similares às aquelas observadas na utilização de novos aplicativos para aplicações críticas. A maior diferença entre os dois casos é a confiança dos clientes em deixar seus dados e os recursos necessários para executar seus serviços sob a responsabilidade de terceiros.

2.8 Plataforma de Desenvolvimento

No desenvolvimento de serviços é necessário, além do desenvolvimento e avaliação dos métodos utilizados, avaliar a arquitetura que será utilizada e suas especificações. Como apresentado na Seção 2.6, os provedores de serviço possuem características referentes aos serviços ofertados e consequentemente esses serviços devem ser distribuídos entre os diversos *data centers* pertencente a esses provedores.

Como a nuvem apresenta desafios no desenvolvimento de novas políticas, sejam elas para escalonamento de máquinas virtuais, escalonamento de *hosts*, descoberta de serviços entre outras, torna-se complexo desenvolver tais políticas e executar testes controláveis, que possibilitem a repetição dos mesmos em um ambiente homogêneo e inalterado (Calheiros et al., 2010).

Além disso, o presente projeto de mestrado aborda características que são relacionadas à infraestrutura da nuvem (descoberta de serviços) tornando assim mais custoso o processo de desenvolvimento de políticas e testes, uma vez que a manipulação de uma infraestrutura privada, por terceiros, é praticamente impossível devido a diferentes questões, dentre elas a mais significativa é a segurança.

Assim, considerando as limitações impostas pela aferição nos sistemas de nuvem, para o desenvolvimento de políticas para a nuvem, a metodologia mais adequada é a utilização de modelagem e simulação.

Alguns simuladores para sistemas distribuídos e para grades computacionais estão disponíveis (Bell et al., 2003) (Casanova et al., 2008) (Ostermann et al., 2011). No entanto, o que mais se adequa para simulação de computação em nuvem e que é o mais amplamente utilizado é o CloudSim (Calheiros et al., 2010).

Trabalhos como os apresentados por (Kim et al., 2009a), (Beloglazov e Buyya, 2010), (Jeyarani et al., 2010), (Sindhu e Mukherjee, 2011) e (Shi et al., 2011) são exemplos da flexibilidade e possibilidades oferecidas pelo CloudSim.

2.8.1 CloudSim

O CloudSim adota uma arquitetura de multicamadas modulares para realizar a gerência dos seus componentes de forma separada (Calheiros et al., 2010). Estes componentes são apresentados na Figura 2.4.

Figura 2.4: Arquitetura do Simulador. Adaptado de (Calheiros et al., 2010)

Como apresentado na Figura 2.4, o CloudSim possui vários módulos que são, por sua vez, agrupados em três categorias:

- **Código do Usuário** – onde são definidos os cenários da simulação como: configuração dos *data centers*, quantidade de hosts, valor monetário dos recursos, número de clientes e a quantidade de brokers ou metaescalonadores;

- **CloudSim** – módulo para definição das políticas que serão desenvolvidas na nuvem. Como é separado em módulos, o desenvolvimento de novas políticas pode ser feita de forma independente;
- **Núcleo de Simulação** – nessa categoria são encontrados os gerenciadores de eventos e as filas utilizadas pelo simulador.

Dentre os módulos que compõe a arquitetura do CloudSim está o módulo de rede (categoria CloudSim). A presença desse módulo demonstra que o simulador não se preocupa apenas com os serviços (denominados *cloudlets*), máquinas virtuais e *data centers*, mas também em como a comunicação é realizada na nuvem (Calheiros et al., 2010).

De acordo com Calheiros et al. (2010) o simulador não conta com entidades reais para a simulação de entidades de rede, como roteadores ou switches. Em vez disso, é utilizada a latência que uma mensagem possui entre sua entidade de origem (cliente) para a entidade destino (*data center*). A latência entre essas entidades é armazenada em uma matriz do tipo $N \times N$ e a informação de *delay* é utilizada sempre que uma mensagem é enviada entre a origem e o destino.

O CloudSim necessita de uma topologia de rede para a descrição das características da rede como: a quantidade de nodos disponíveis e os pesos das arestas de ligação entre um nodo e outro. Essas informações são necessárias para que o simulador possa utilizar essas informações para gerar a sobrecarga da rede (Calheiros et al., 2010). A topologia de rede, que auxilia nas atividades relacionadas à rede, não são geradas pelo CloudSim e sim por um gerador de topologias de rede denominado BRITE (Medina et al., 2001).

O BRITE é utilizado para compor arquivos de configuração que possuem as características da rede como quantidade de nodos presente na rede e o peso das arestas de ligação entre os nodos (Calheiros et al., 2010).

2.8.2 Brite

O BRITE é um gerador de topologias que não se restringe a apenas uma forma de gerar topologias (Medina et al., 2001). Por se tratar de um gerador de topologias genérico, ele fornece várias formas de construir topologias e de modelos de distribuição de nodos (Medina et al., 2001). A Figura 2.5 apresenta uma visão geral do BRITE.

O BRITE possui o modelo de distribuição de nodos (1) e as derivações do modelo utilizado. Os modelos que podem ser utilizados pelo BRITE são: Waxman (Waxman, 1988) e Barabasi-Albert (Albert e Barabási, 2000).

Segundo Naldi (2005), o modelo Waxman é um modelo de geração de grafos aleatórios para modelar topologias de rede com o propósito de avaliar algoritmos de roteamento. Esse modelo utiliza propriedades espaciais para gerar a conectividade do grafo.

Figura 2.5: Módulos que Compõe o BRITE. Adaptado de (Medina et al., 2001)

O grafo contém um número N de nodos que são distribuídos uniformemente sobre um plano retangular. Cada nodo possui coordenadas cartesianas inteiras que são utilizadas para gerar a conectividade do grafo, ou seja, a conectividade dos nodos está relacionada com a relação espacial entre eles (suas distâncias no plano) (Waxman, 1988) (Naldi, 2005).

Albert e Barabási (2000) propõe um mecanismo gerador de grafos que usa a lei de distribuições de potência, também chamada de apego preferencial. Nesse modelo a rede cresce de forma incremental e a interconexão entre os nodos é feita de acordo com a proximidade dos nodos no plano cartesiano.

Esse modelo sugere duas formas possíveis para a geração das topologias de rede: crescimento incremental e conectividade preferencial. Crescimento incremental refere-se a adição contínua de novos nodos, e, portanto, o aumento gradual no tamanho da rede. Conectividade preferencial refere-se à tendência de um nodo ligar-se a outros nodos que estão próximos a ele (Albert e Barabási, 2000).

O grafo é gerado com as informações sobre o tamanho do plano e a quantidade de nodos que irão compor a simulação (2). As ligações entre esses nodos são feitas utilizando um dos algoritmos citados anteriormente.

Assim, de posse da topologia gerada pelo BRITE, é gerado um arquivo de topologia (3) que pode ser salvo como BRITE, SSF, NS e JSim (Medina et al., 2001).

Com a utilização das ferramentas descritas nessa seção é fornecido, por meio de simulação, um ambiente onde a nuvem pode ser avaliada considerando várias características. Dentre as características que podem ser analisadas está a rede, possibilitando assim o uso do CloudSim para o desenvolvimento de políticas de roteamento que podem ser utilizadas por provedores de computação em nuvem.

2.9 Considerações Finais

A computação em nuvem é dependente da qualidade de conexão disponível para o utilizador dos diversos serviços (Pokharel e Park, 2009), uma vez que todos os serviços são acessados por meio da Internet. É importante investigar maneiras de prover QoS, com relação à comunicação, para as aplicações e serviços desse ambiente.

Essa forma de computação agrega oportunidades para que serviços sejam migrados de servidores locais para servidores remotos. Contudo, é necessário avaliar o funcionamento das camadas que a compõe, uma vez que a tendência é que este modelo computacional cresça e atinja grandes proporções. Serviços e políticas de computação em nuvem se utilizados sem passar por avaliações de desempenho, para verificar a viabilidade de utilização em determinadas atividades, podem não prover um QoS satisfatório aos clientes.

Por parte dos fornecedores, é necessário monitorar a nuvem para escalonar os serviços dos usuários. Assim, surge a necessidade de criar interfaces para que os usuários possam definir as configurações mínimas para que suas aplicações possam ser executadas com qualidade.

Essas interfaces podem ser *brokers*, escalonadores ou metaescalonadores que são desenvolvidos para monitorar os recursos disponíveis. Entender como os escalonadores dos fornecedores funcionam pode trazer benefícios na identificação de possíveis gargalos e assim melhorar a qualidade na forma como os serviços são ofertados e entregues pela nuvem.

Metaescalonadores

3.1 Considerações Iniciais

Como a nuvem é um ambiente de computação distribuído, é necessário que haja mecanismos para o gerenciamento dos recursos desse ambiente (Buyya et al., 2010). Como esse sistema é formado por um conjunto de *data centers* (Zhang et al., 2010) deve-se ter dois tipos de escalonadores: um para monitorar os recursos da nuvem e outro para monitorar as diversas nuvens (Xhafa e Abraham, 2010).

O primeiro tipo de escalonador é o escalonador local (Christodoulopoulos et al., 2009). Esse escalonador controla os recursos dos *data centers* em nível de recursos disponíveis, como armazenamento e processamento (Christodoulopoulos et al., 2009).

Quando há várias nuvens a serem monitoradas, o escalonamento pode ser feito utilizando um *broker* ou um metaescalonador (Yamini et al., 2011). Um metaescalonador esconde a complexidade do mecanismo de escalonamento utilizado pela nuvem provendo assim uma forma transparente de alocação de recursos para a execução dos serviços dos clientes (Liu et al., 2008) (Heidt et al., 2008) (Mann, 2005).

Os metaescalonadores são utilizados para distribuir os serviços em servidores, baseados nos requisitos do cliente (Buyya et al., 2010). Diferentemente dos escalonadores locais, os metaescalonadores não possuem controle direto sobre os recursos da nuvem, ou seja, ele repassa ao escalonador local as solicitações realizadas pelos clientes e os escalonadores locais repassam aos metaescalonadores informações de disponibilidade (Yamini et al., 2011).

3.2 Escalonadores Locais e Metaescalonadores

Metaescalonadores e escalonadores locais são duas soluções, que normalmente coexistem, para o escalonamento em sistemas distribuídos como, por exemplo, em grades computacionais e nuvens (Xhafa e Abraham, 2010).

Weissman e Grimshaw (1996) propõe uma forma de escalonamento para sistemas distribuídos baseado em gerenciamento de recursos locais (LRMS - *Local Resource Management System*). Cada participante desse ambiente precisa iniciar um gerenciador local e um gerenciador global de recursos. O gerenciador global conta com duas interfaces: a interface do gerenciador de escalonamento para os escalonadores locais e, o escalonador da grade para os gerenciadores de escalonamento remoto. O compartilhamento de informações pode ser acessado a qualquer momento e os arquivos são estáticos.

Seguindo a linha de utilização de metaescalonadores, Schwiegelshohn e Yahyapour (1999) apresentam um metaescalonador denominado NWIRE (*Net-Wide-Resources*). O NWIRE consiste em um meta-gerenciador que é responsável por controlar o conjunto de domínios ou meta-domínios que possui acesso ao gerenciador de recursos (LRMS). O NWIRE utiliza várias características de escalonamento incluindo a existência de escalonadores convencionais e reserva de recursos.

Subramani et al. (2002) apresenta um modelo de escalonamento de computação distribuída que se adapta às mudanças do uso dos recursos. O objetivo central foi propor um metaescalonador que considera a distribuição de tarefas em vários servidores ao invés de encaminhá-los ao servidor menos sobrecarregado.

O metaescalonador proposto utiliza a técnica de *backfilling*, que consiste em balancear a utilização e manutenção das tarefas com base em fila (Subramani et al., 2002).

Outra infraestrutura de escalonamento baseada nas mudanças do uso de recursos é apresentada pelo *OurGrid*, que utiliza como critério de escalonamento a reputação e a disponibilidade dos recursos (Andrade et al., 2003).

Com a difusão dos metaescalonadores as técnicas utilizadas, até então para grades computacionais, foram sendo aprimoradas como é apresentado por Iosup et al. (2007). O metaescalonador proposto utiliza uma arquitetura hierárquica no qual os recursos de mesmo nível podem cooperar entre si. Essa abordagem descentralizada permite a cooperação entre várias organizações diferentes sem a necessidade de um ponto central de controle.

Leal et al. (2009) apresenta o *GridWay* e discute o seu modelo descentralizado de escalonamento para múltiplas organizações que trabalham de forma cooperativa. Esse esquema propõe um metaescalonador para cada infraestrutura da organização. O método foi uma alternativa para as limitações apresentadas por metaescalonadores com modelo centralizado.

Uma abordagem de escalonamento dinâmico descentralizado é proposta por Huang et al. (2011). Essa abordagem é denominada CASA (*Community Aware Scheduling Algorithm*) e

funciona como um escalonamento de duas fases. Na primeira fase, uma tarefa é submetida ao nodo mais adequado entre os participantes do ambiente. Na segunda fase, onde ocorre o escalonamento dinâmico, o objetivo é melhorar as decisões do escalonamento de forma interativa.

Como apresentado nos trabalhos descritos anteriormente, há basicamente três modelos de troca de informação entre os metaescalonadores e os escalonadores locais. A Figura 3.1 apresenta o modelo centralizado de comunicação entre os metaescalonadores e os escalonadores locais.

Figura 3.1: Modelo Centralizado de Troca de Informações

No modelo apresentado na Figura 3.1, há um metaescalonador principal que mantém informações de carga de trabalho sobre todas as organizações participantes (Weissman e Grimshaw, 1996). Os serviços são enviados para o metaescalonador que decide qual distribuidor local irá receber os serviços. Este modelo apresenta algumas desvantagens causadas pela centralização, tais como: difícil escalabilidade e problemas com tolerância a falhas (Subramani et al., 2002).

A Figura 3.2 apresenta o modelo distribuído (Leal et al., 2009) (Huang et al., 2011).

Figura 3.2: Modelo Descentralizado de Troca de Informações

No modelo apresentado na Figura 3.2, cada ambiente possui seu próprio metaescalonador que periodicamente consulta os outros metaescalonadores para obter informações da carga instantânea de recursos (Subramani et al., 2002). Os serviços são enviados para o metaescalonador

local que decide, dentre os escalonadores locais, qual deles irá executar os serviços ou se há necessidade de migrar os serviços para outro ambiente.

E por fim há o modelo hierárquico, Figura 3.3, onde o fluxo de trabalho é compartilhado entre os metaescalonadores (Subramani et al., 2002). Os recursos do metaescalonador de mesmo nível cooperam entre si (Iosup et al., 2007).

Figura 3.3: Modelo de Troca de Mensagens Hierárquicas

Neste modelo os serviços são enviados para o metaescalonador mais próximo ao cliente e não há fila de serviços. O metaescalonador envia o serviço para o ambiente que estiver disponível e que satisfaça os requisitos de QoS dos clientes. Cada ambiente pode usar diretivas de agendamento diferentes devido ao fato de possuírem um metaescalonador local e cada metaescalonador local interagir com os escalonadores locais (Subramani et al., 2002) (Iosup et al., 2007).

Os modelos de troca de informação seguiram a evolução da computação em grade e hoje há modelos bem difundidos que derivam dos modelos apresentados anteriormente (Chard, 2011). A evolução dos metaescalonadores por sua vez, permite que esta abordagem seja utilizada além dos domínios de computação em grade (Chard, 2011). Assim, é proposto por Peixoto et al. (2010) um metaescalonador denominado MACC (*Metascheduler Architecture to provide QoS in Cloud Computing*) que considera a evolução dos metaescalonadores do ambiente de grades computacionais para o ambiente de nuvem.

3.3 MACC

Como foi discutido na seção 3.2, um metaescalonador é uma abordagem computacional que permite a alocação de serviços, com o objetivo de distribuir a carga entre os vários provedores de serviços de um ambiente ou de uma federação de ambientes (Peixoto et al., 2009).

O conceito de federações consiste na união de diversos provedores de computação em nuvem que possuem propósitos de negócios semelhantes, possibilitando assim o acesso a serviços

de nuvem de forma escalar, mesmo sobre diferentes provedores dessa federação (Buyya et al., 2010).

Assim, o MACC é um metaescalador que torna possível prover QoS na computação em nuvem. A sua principal função é realizar a manutenção dos critérios de QoS, fazendo com que os serviços sejam requisitados de forma transparente aos clientes.

O cliente encaminha as solicitações ao MACC, que segue o modelo hierárquico de comunicação entre escalonadores locais e metaescaladores. O MACC realiza um escalonamento em duas camadas. A primeira camada realiza uma pesquisa sobre qual federação pode oferecer os requisitos de qualidade de serviços ao cliente, encaminhando assim os serviços ao escalonador local. A segunda camada realiza o escalonamento de acordo com as informações de recursos da federação, que irá processar as solicitações do cliente.

Para realizar a atividade descrita anteriormente, o MACC utiliza vários módulos como é apresentado na Figura 3.4. Cada módulo é orientado à QoS, sendo assim, o MACC possui um módulo que se preocupa com os serviços e requisições dos clientes. Esse tratamento é importante, uma vez que é necessário analisar quais os atributos de QoS deverão ser abordados no atendimento dos serviços.

O módulo de controle de admissão é dependente do módulo de controle de valores que é responsável por verificar os contratos (SLAs) estabelecidos com o cliente. Nesse módulo é encontrado o modelo econômico utilizado pelo MACC.

Figura 3.4: Arquitetura do MACC

O núcleo do MACC apresenta vários outros módulos que são responsáveis diretos na obtenção da QoS que será oferecida aos clientes. Esses módulos são descritos como (Peixoto et al., 2010):

- **Cache UDDI** - Armazena informações sobre serviços locais e sobre os serviços remotos facilitando a busca por esses serviços;

- **LRAM** - Gerenciador de Alocação de Recursos Local. Esse módulo utiliza os módulos *hypervisor* e de recursos apresentados na Figura 3.4 para o gerenciamento dos recursos locais das federações;
- **Políticas de Escalonamento** - Esse módulo é responsável pelas políticas de roteamento utilizadas na primeira fase de escalonamento e pelas políticas de alocação de máquinas virtuais utilizadas na segunda fase de escalonamento;
- **Workload Engine** - Esse módulo faz a estimativa do tempo necessário que um serviço necessita para a utilização dos recursos da federação;
- **Monitoring and Discovery System Manager (MDSN)** - Esse módulo é utilizado para a localização dos registros de serviços disponíveis. Para essa localização esse módulo conta com:
 - **Trigger Service** - Esse submódulo é responsável por notificar as mudanças dos recursos como expansão ou diminuição de recursos.
 - **Index Service** - Esse submódulo desempenha o endereçamento de cada recurso requisitando dados de desempenho.

Os módulos apresentados têm como objetivo trabalharem em conjunto formando assim uma infraestrutura em que cada módulo se preocupa com a obtenção de QoS relacionado à atividade particular do módulo e geral do MACC.

3.4 Qualidade de Serviço

Segundo Chard (2011) uma das formas de manter os níveis de QoS é estabelecer acordos, como uma SLA entre clientes e provedores. Sendo assim, An et al. (2010) propõe o uso de modelos econômicos para a alocação de recursos com foco na negociação de mercado dinâmica.

Comparado ao modelo apresentado por An et al. (2010), o MACC também se preocupa com o estabelecimento de um preço inicial, porém os preços são regidos por uma SLA, que conta com uma definição de obrigações por parte do provedor de serviços para os clientes (Peixoto et al., 2010).

Além das regras de mercado, há trabalhos (Gmach et al., 2009) (Zhu et al., 2008) que realizam a alocação dinâmica de CPU para cumprir os objetivos de QoS. Além disso, é oferecida a diferenciação de serviço por meio da alocação baseada em prioridades.

O que diferencia os trabalhos citados anteriormente do MACC é que o MACC possui um fornecimento de QoS fim-a-fim. Existe uma preocupação em garantir QoS tanto em nível de rede como em nível de aplicação, diferenciando assim o MACC de outros metaescaladores (Peixoto et al., 2010).

A Tabela 3.1 apresenta uma comparação entre metaescalonadores de grades computacionais apresentados em Peixoto et al. (2010) e interfaces de acesso utilizados pela nuvem apresentados em Zhang et al. (2010), Buyya et al. (2010) e Ranjan et al. (2010) e que vão ao encontro das funções do MACC.

Tabela 3.1: Comparação entre Metaescalonadores

Metaescalonador	Middleware	Balanceamento de Carga	Tipo de Serviços	Algoritmo de Escalonamento	Auto escalável
CSF	<i>Globus</i>	Reserva de Recursos	Independente	<i>Round Robin</i>	Não
<i>Gridway</i>	<i>Globus</i>	<i>Round Robin</i>	Independente	<i>Round Robin</i>	Não
Amazon	EC2	Definido por arquivos de configuração	Infraestrutura	DNS	Sim
Microsoft	<i>Azure</i>	Automático	Plataforma e Infraestrutura	CDN	Sim
<i>Google</i>	<i>App Engine</i>	Automático	Plataforma e Infraestrutura	Manual	Sim
Rackspace	CLB	Automático	Infraestrutura	<i>Round Robin</i>	Sim
<i>GoGrid Cloud Hosting</i>		Definido por arquivos de configuração	Infraestrutura	<i>Round Robin</i> e <i>Last connect algorithm</i>	Manual
MACC		Baseado em QoS	Independente	P2P	Sim

A Tabela 3.1 apresenta características referentes aos metaescalonadores e interfaces de configuração de serviços em nuvem que funcionam como escalonadores. Os metaescalonadores de grades apresentam vantagens como: independência de plataforma, porém não apresentam escalabilidade (Foster e Kesselman, 1997) (Foster e Kesselman, 1999).

As políticas de escalabilidade devem ser programadas pelo cliente de acordo com a aplicação (Foster e Kesselman, 1999). Diferente dos metaescalonadores para grade, os fornecedores de serviços nas nuvens possuem em sua maioria um nível mais alto de escalabilidade e balanceamento de carga automático (Zhang et al., 2010).

O MACC quando comparado a outros metaescalonadores apresenta vantagens relacionadas ao balanceamento de carga, que é avaliada de acordo com a qualidade de serviço que será oferecida aos clientes (Peixoto et al., 2010). Essa característica visa auxiliar o balanceamento de carga uma vez que os escalonadores que utilizam métodos automáticos podem não considerar os atributos corretos para esse controle.

O MACC apresenta vantagem sobre o Gogrid (Ranjan et al., 2010) em relação ao balanceamento de carga e auto escalabilidade, uma vez que, no Gogrid essas políticas precisam ser desenvolvidas de acordo com a aplicação. O MACC, por sua vez, possui suas políticas voltadas à qualidade de serviço (Peixoto et al., 2010).

Outro diferencial entre o MACC e os escalonadores específicos para nuvem é relacionado ao modelo de comunicação. A abordagem mais utilizada é o *Round Robin* devido ao balanceamento de carga feito no nível de roteamento (Ranjan et al., 2010). Na Amazon a localização de onde os serviços serão instanciados é de responsabilidade do cliente e o escalonamento das instâncias é feito de acordo com as especificações das aplicações. O tráfego das aplicações é automaticamente distribuído pelo EC2 utilizando *Domain Name Service* (DNS) fazendo um balanceamento de carga no nível de roteamento interno (Amazon, 2012).

O Windows Azure utiliza a técnica de *Content Delivery Network* (CDN) que consiste em criar cópias dos dados em servidores mais próximos dos clientes. Essa técnica consiste em caches de objetos estáticos para diminuir o tempo de carregamento das aplicações para os clientes (Microsoft, 2012).

Enquanto os escalonadores e metaescalonadores possuem, em sua maioria, apenas um algoritmo ou técnica para o roteamento das aplicações, o MACC possui um modelo de comunicação que pode utilizar algoritmos convencionais com o *Round Robin* ou mesmo sistemas P2P (Peixoto et al., 2010). Essa característica do MACC permite que o modelo de comunicação possa evoluir de acordo com a evolução da computação em nuvem.

3.5 Considerações Finais

Independentemente do contexto envolvido, os tipos de gerenciamento de QoS devem incluir requisitos, tais como: mapeamento e negociação de QoS, estabelecimento de uma SLA e monitoramento de QoS (Guo et al., 2007) (Foster et al., 1999). O mapeamento dos melhores atributos de QoS não é uma tarefa simples e depende dos objetivos do cliente, bem como da capacidade do provedor de serviço para prover as características necessárias a esses clientes.

Os metaescalonadores auxiliam nessa tarefa de mapeamento utilizando informações de outros metaescalonadores e de escalonadores locais, como foi apresentado nesse capítulo. Um metaescalonador é formado por várias partes que devem trabalhar em conjunto para atingir um objetivo comum (Peixoto et al., 2010).

Como apresentado até aqui, o MACC é composto por vários módulos, e dentre esses módulos será dada uma atenção maior ao módulo de escalonamento. Esse módulo deve conter algoritmos responsáveis pelo roteamento das solicitações dos clientes. Para que essa atividade seja executada de forma a obter um nível aceitável de QoS é necessário que as políticas de roteamento sejam avaliadas seguindo metodologias já consolidadas para que o desempenho do MACC não seja afetado.

Essa avaliação é necessária devido ao modelo de fornecimento de QoS do MACC que possui um modelo fim-a-fim. Dessa forma, a atividade de comunicação também deve ser abordada.

Comunicação nas Nuvens

4.1 Considerações Iniciais

A redução dos custos da Internet e a facilidade de acesso a esse meio, que pode ser acessado por dispositivos variando desde computadores pessoais até telefones celulares, tendem a impulsionar a computação em nuvem (Furht, 2010), uma vez que essa abordagem enfatiza a habilidade de oferecer serviços pela Internet com foco na prestação de serviços sob demanda (Buyya et al., 2010).

Nesse caso, os provedores de computação em nuvem estão espalhados em diferentes localizações geográficas, pela Internet, com o objetivo de oferecer melhores serviços aos clientes (Ranjan et al., 2010). Essa distribuição geográfica visa diminuir a distância entre os clientes e os provedores e com isso fornecer serviços com segurança, confiabilidade e tolerância a falhas (Peixoto et al., 2010).

Não é suficiente ocorrer apenas comunicação entre os provedores que estão dispersos geograficamente. Essa comunicação deve vir acompanhada de características que proporcionem aspectos de QoS como baixos tempos de resposta e tolerância a falhas tanto na descoberta quanto na entrega de recursos e serviços (Peixoto et al., 2010).

4.2 Descoberta de Recursos

A descoberta de serviços desempenha um papel fundamental na obtenção de QoS dentro da computação em nuvem. Chard (2011) apresenta o conceito de metaescaladores que, utili-

zando *middlewares* para descoberta de recursos e serviços, podem fornecer qualidade de serviços em atividades tanto de grades computacionais quanto de ambientes de nuvem.

É importante considerar o modelo que será seguido para a descoberta de serviços. Os modelos utilizados para essa descoberta seguem abordagens centralizadas ou descentralizadas (Chard, 2011).

O foco deste trabalho de mestrado é no estudo de modelos descentralizados, uma vez que os modelos centralizados apresentam, em sua maioria, problemas como ponto único de falha e nem sempre esse modelo proporciona tolerância a falhas e escalabilidade (Ranjan et al., 2010).

Já os sistemas descentralizados podem ser usados para minimizar algumas das limitações apresentadas pelo modelo centralizado. A descoberta de recursos na forma distribuída é geralmente baseada em abordagens hierárquicas ou P2P (ponto a ponto) (Peixoto et al., 2010). O Ganga (Massie et al., 2004) é uma arquitetura de descoberta de recursos hierárquica. A própria entidade responsável nomeia as informações sobre os recursos e gerencia onde essas informações devem ser armazenadas.

A descoberta de serviços pode ser baseada em sistemas P2P, onde as informações encontram-se distribuídas entre os nodos da rede (Ranjan et al., 2006). Em estruturas P2P puras, pode-se afirmar que, nenhum nodo tem função mais importante do que outro. Pela própria definição, essa arquitetura é tolerante a falhas, reorganizável e escalável. Por causa dessas características, há vários sistemas P2P, que podem ser utilizadas para auxiliar na descoberta de serviços em sistemas distribuídos, propostas na literatura (Ranjan et al., 2008).

4.3 P2P

Os sistemas ponto a ponto (P2P) são aplicações que surgiram como uma alternativa aos tradicionais sistemas cliente-servidor (Kurose et al., 2010). Diferente de aplicações Web, servidores de e-mail e até mesmo o *Domain Name Service* (DNS), que necessitam de servidores centralizados, os sistemas P2P possuem uma dependência mínima (nem sempre é necessário) de servidores centralizados para a troca de informação (Kurose et al., 2010).

Os sistemas P2P representam uma forma de construir sistemas e aplicativos distribuídos, onde os dados e recursos computacionais são derivados da colaboração de muitos pontos na Internet de maneira uniforme. Os pontos que compõem uma rede P2P podem atuar tanto como servidores quanto como clientes sem uma coordenação centralizada (Lua et al., 2005).

De acordo com Kurose et al. (2010) os sistemas P2P podem servir para a distribuição de arquivos, construção de um banco de dados distribuído também conhecido como *Distributed Hash Table* (DHT) e telefonia pela Internet (Voip).

Neste trabalho de mestrado será utilizado o conceito de DHT por considerar a descoberta de recursos e não o compartilhamento de arquivos ou aplicações.

4.3.1 Características dos Sistemas P2P

Segundo Ranjan et al. (2006), as redes P2P apresentam características relacionadas à organização da rede, organização dos dados e a forma de roteamento. A taxonomia da Figura 4.1 apresenta as formas de divisão feitas pelos sistemas P2P.

Figura 4.1: Taxonomia das Redes P2P. Adaptado de Ranjan et al. (2006)

Como os nodos que compõe uma rede P2P podem ser organizados logicamente, os mesmo podem utilizar duas categorias básicas (Milojicic et al., 2002): estruturadas e não estruturadas.

Um sistema **não estruturado** é descrito por um modelo de ligações aleatórias que são baseadas na popularidade das informações de cada nodo (Ranjan et al., 2006). Neste modelo não existe uma preocupação em criar e manter uma organização lógica (Milojicic et al., 2002). São exemplos de sistemas não estruturados o Napster, Gnutella e Kazaa.

Já os sistemas **estruturados** são caracterizados por um modelo de ligações que segue uma determinada hierarquia, como a apresentada pelo DHT (Ranjan et al., 2006). Além de seguir uma hierarquia, os sistemas estruturados se preocupam em manter uma organização lógica denominada *overlay* (rede de sobreposição) (Jin et al., 2008). Essa organização é mantida para que a busca por um recurso possua o menor número de passos dentro da rede de sobreposição. São exemplos de sistemas estruturados o CAN, Pastry, Chord e Tapestry (Schmidt e Parashar, 2005).

A **organização dos dados** segue o modelo da organização da rede (Ganesan et al., 2004). Se a rede for **não estruturada**, significa que os nodos não farão parte de uma topologia específica e nem de um domínio específico uma vez que os dados são espalhados pela rede de forma aleatória.

No modelo **estruturado**, os dados são organizados na rede seguindo uma topologia específica que depende da rede de sobreposição. Essa característica é utilizada para limitar a complexidade da busca, balanceamento de carga e limitação na sobrecarga do gerenciamento da localidade dos dados (Ganesan et al., 2004).

Os dados podem ser as informações referentes aos nodos, como capacidade de processamento e armazenamento, ou arquivos e esses dados formam a *hash* que é utilizada para localizar outros nodos dentro da rede (Bienkowski et al., 2005).

A **consulta e o roteamento** são relacionados à forma como a rede é estruturada. Normalmente as redes **não estruturadas** utilizam técnicas como busca em largura e profundidade na busca por um determinado recurso (Androutsellis-theotokis e Spinellis, 2004).

Já as redes que utilizam sistemas **estruturados** possuem um sistema de busca que utiliza a hierarquia da rede (Ganesan et al., 2004). Esse método proporciona o controle de carga no roteamento, uma vez que cada nodo recebe aproximadamente o mesmo número de consultas e mantém um número limitado de informações sobre os nodos da rede. As informações que um nodo mantém se referem apenas aos nodos mais próximos a ele (Ganesan et al., 2004).

4.3.2 Redes de Sobreposição

As redes de computadores possibilitam a troca de informações entre dois ou mais nodos, geograficamente separados, sem uma conexão direta entre eles. A infraestrutura de comunicação deve oferecer meios para conduzir os dados entre a origem e o destino (Tanenbaum, 2003).

Segundo Tanenbaum (2003), uma das principais funções da camada de rede (pilha de protocolos do TCP (*Transmission Control Protocol*) e IP (*Internet Protocol*) TCP/IP) é realizar o roteamento dos pacotes IP, permitindo que um dado seja transmitido entre um nodo origem a um nodo destino independentemente do trajeto físico a ser percorrido.

Devido a essa característica oferecida pela pilha TCP/IP é possível ter mecanismos de roteamento na camada de aplicação, que operam de maneira totalmente separada dos mecanismos situados na camada de rede (Coulouris e Dollimore, 1988). Esse mecanismo é chamado de roteamento com redes *overlay* ou redes de sobreposição. Visto que o roteamento é realizado na camada de aplicação, uma vez definido o endereço de rede do nodo para o qual uma mensagem deve ser enviada, os mecanismos de roteamento da camada de rede são utilizados e processados normalmente (Coulouris e Dollimore, 1988).

Como as redes de sobreposição são topologias lógicas, alguns sistemas P2P utilizam sua própria topologia lógica para estruturar suas respectivas redes como é o caso do Chord (Stoica et al., 2001), Pastry (Rowstron e Druschel, 2001) e CAN (Ratnasamy et al., 2002).

A CAN (*Content Addressable Network*) organiza um espaço lógico multidimensional de coordenadas, que é dividido em zonas. Cada zona é atribuída a um participante e conforme os participantes vão entrando na rede, as zonas vão sendo divididas entre os participantes. A Figura 4.2 representa a divisão das coordenadas utilizadas pela CAN.

Como apresentado na Figura 4.2, conforme os nodos forem entrando na rede eles são alocados de acordo com a proximidade aos outros nodos na rede formando um plano multidimensional (Ratnasamy et al., 2002).

Figura 4.4: Espaço de Busca do Pastry. Adaptado de Rowstron e Druschel (2001).

A Figura 4.4 apresenta a topologia que é utilizada pelo Pastry. Nessa topologia, a busca é realizada como a busca em uma árvore. Quando um nodo entra na rede, assim como nas outras abordagens, é gerado um identificador (ID) que é relacionado ao recurso que ele possui e sua localização. Com base no ID gerado, o nodo é atribuído a uma árvore, de forma a ficar próximo aos nodos com recursos semelhantes ou localidades próximas.

De acordo com Ranjan et al. (2006), as características de entrada e saída de nodos da rede passam pelos mesmos procedimentos e o que diferencia uma abordagem da outra é a forma como a busca é realizada e a topologia que é adotada. Sendo assim, quando um novo nodo entra na rede P2P, é preciso que o novo nodo conheça algum nodo que já esteja participando dessa rede, para que esse nodo, já participante, seja sua "porta de entrada" para a rede (Ratnasamy et al., 2002).

Uma vez que o novo nodo obtém sua zona na rede, é necessário realizar o ajuste das tabelas de roteamento. Para isso, o novo nodo observa a tabela de roteamento do nodo pré-existente e verifica, dentre os participantes, quais são as entradas referentes aos seus vizinhos. Essas entradas são retiradas da tabela do nodo pré-existente e inseridas na tabela do novo nodo, o qual adiciona ainda uma entrada para o nodo pré-existente (Ratnasamy et al., 2002) (Rowstron e Druschel, 2001). Após esse procedimento a tabela de rotas é atualizada. Quando o nodo sai da rede a tabela de rotas deve ser atualizada novamente para manter a consistência da rede (Stoica et al., 2001).

4.4 Roteamento P2P em Sistemas Distribuídos

Como apresentado na seção anterior, os sistemas P2P estruturados podem ser utilizados no encaminhamento e descoberta de recursos e serviços. Essa característica facilitou a adoção de sistemas P2P para roteamento em sistemas distribuídos.

Em Spence e Harris (2003) é proposta uma extensão de um sistema baseado em Pastry para um sistema de busca. O sistema de busca proposto trabalha em conjunto com o XenoServer

para a busca de nodos baseados em atributos como: localização topológica e atributos de QoS. A busca é dividida em várias dimensões. Em cada dimensão, de forma lógica, a busca pelas informações dos recursos é feita utilizando um modelo de árvore, onde as folhas são os Xeno-Servers (Spence e Harris, 2003).

Em Tam et al. (2004), é apresentada uma aplicação de troca de mensagens baseada em Pastry. O modelo proposto define diferentes esquemas para a publicação e subscrição de mensagens para vários domínios como bolsa de valores e mercados de leilão. O modelo procede ao processo de inscrição para um evento através de uma árvore de nodos que possuem os mesmos interesses. A raiz da árvore é o nodo que publica a inscrição e os nodos que irão se inscrever são as folhas da árvore (Tam et al., 2004).

Cai et al. (2004) apresenta uma abordagem para habilitar um componente de computação em grade denominado *General Resource Information System* (GRIS). Esse componente utiliza uma topologia em anel (Chord) para o gerenciamento de recursos gerais relacionados a grades computacionais. O modelo apresentado por Cai et al. (2004) faz o mapeamento dos valores dos atributos utilizados na busca por um recurso utilizando uma localidade uniforme.

Triantafyllou e Aekaterinidis (2004) propõe um sistema de inscrições baseado em CAN. Cada evento a ser publicado é mapeado para uma região da CAN, nesse modelo a região que foi mapeada fica conhecida como o ponto do evento ou a zona de evento. Todos os pares que fizerem parte da região do evento são notificados sobre ele. Assim como o sistema apresentado por Tam et al. (2004), o objetivo é diminuir os passos na inscrição para um determinado evento.

Yong Meng et al. (2005) propõe o uso de um GRIS baseado em Chord. Nesse trabalho, o GRIS atua como um metaescalonador que mantém informações relacionadas a cada domínio e cada domínio possui seu escalonador local. A busca por um recurso é baseada no sistema Chord e as IDs são geradas de acordo com a descrição do recurso e o servidor que o possui, evitando assim a geração de chaves repetidas.

Cheema et al. (2005) apresenta um GRIS que utiliza o Pastry como padrão de roteamento. Três heurísticas diferentes são propostas para esse GRIS: busca de uma só vez (one shot), busca recursiva e busca paralela. A primeira é baseada no recurso local para saber a disponibilidade de cada escalonador local, a segunda é baseada na busca pelo recurso mais próximo ao solicitante e a terceira é baseada em várias buscas já com o conhecimento da localidade do recurso para a exata alocação baseada nos parâmetros da requisição do solicitante (Cheema et al., 2005).

Com o surgimento da computação em nuvem algumas abordagens utilizadas em computação em grade foram absorvidas por esse modelo computacional. Em Lai et al. (2010) é proposto uma estratégia de descoberta de recursos em computação em nuvem utilizando sistemas P2P. A proposta desse trabalho é utilizar o sistema Chord (Stoica et al., 2001) na descoberta de recursos em nuvem. Nesse trabalho os autores sugerem uma modificação do Chord para funcionar com mais de um anel.

Lai et al. (2010) propõe a criação de novos anéis baseados nas regiões de cada solicitante. Lai et al. (2010) usa as descrições dos serviços para a construção da topologia. A vantagem do trabalho de Lai et al. (2010) é o uso de vários atributos na construção dos IDs dos nodos.

Ranjan et al. (2010) propõe o uso de sistemas P2P para o provisionamento de recursos da nuvem como a descoberta de serviços e o balanceamento de carga. Os autores consideram que a interconexão dos sistemas que compõe a nuvem (servidores, máquinas virtuais (VMs), serviços de aplicação), utilizando sistemas P2P, pode evitar os problemas de provisionamento de recursos e evitar possíveis gargalos e problemas de ponto único de falha.

Dentre as três camadas que compõe a nuvem, IaaS, PaaS e SaaS, os autores sugerem que a comunicação P2P deve estar na camada PaaS para possibilitar a comunicação das máquinas virtuais dentro do *data center*. Foi utilizado o sistema P2P Pastry como sistema base para o roteamento (Ranjan et al., 2010).

Diferente das abordagens apresentadas anteriormente, Zhou et al. (2011) apresenta em seu trabalho uma abordagem P2P híbrida por considerar que o custo de manutenção das redes de sobreposição é muito alto e pode causar sobrecarga nas atividades de descoberta de serviços. Zhou et al. (2011) utiliza o conceito de *supernodes* para a manutenção da rede P2P. Assim como os outros trabalhos relacionados a P2P e nuvem, nenhuma avaliação de desempenho é realizada entre os sistemas P2P.

O presente trabalho de mestrado aborda a avaliação de desempenho entre sistemas P2P para demonstrar qual a diferença entre as abordagens utilizadas. Essa avaliação permite identificar quais as diferenças e possíveis gargalos que podem ser proporcionados pelas redes de sobreposição na descoberta de serviços.

4.5 Modelo de Comunicação do MACC

O modelo de comunicação adotado pelo MACC (Peixoto et al., 2009) (Peixoto et al., 2010) é apresentado na Figura 4.5, que demonstra o fluxo de desenvolvimento de uma aplicação encaminhada pelo MACC.

Como apresenta a Figura 4.5, o cliente solicita uma aplicação ao MACC que irá consultar, em sua camada de interconexão, qual o melhor *data center* para atender a solicitação do cliente. O modelo de comunicação adotado pelo MACC é composto por:

- **Camada de Aplicação** - Essa camada consiste na aplicação que o cliente está solicitando ou irá solicitar, além de obter as informações de localidade referente ao cliente;
- **Interconexão** - É a camada onde são coletadas as informações sobre a rede e sobre a disponibilidade dos *data centers*. Essa camada possibilita a pesquisa baseada nas informações de rede do cliente e do *data center*. Essa camada permite o desenvolvimento das políticas de roteamento;

Figura 4.5: Visão Geral do Modelo de Comunicação Adotado pelo do MACC.

- **Federação** - é a camada onde os *data center* irão executar as solicitações dos clientes.

Na condução dos cenários simulados, utilizando o modelo apresentado na Figura 4.5, foi considerado o ambiente de nuvens híbridas. Nesse cenário, a carga de trabalho é enviada de uma nuvem privada (cliente) para uma nuvem pública (*data center*) simulando o comportamento de comunicação entre os dois tipos de nuvens. O MACC coleta informações sobre os clientes e *data centers* e utiliza o modelo apresentado na Figura 4.6 para realizar a descoberta de recursos e o encaminhamento das solicitações dos clientes.

Figura 4.6: Modelo de Comunicação do MACC.

O MACC adota um modelo P2P hierárquico baseado em DHT (Peixoto et al., 2010). Esse modelo possui uma distribuição lógica que é formada por uma camada de sobreposição. Como apresentado na Figura 4.6, cada região possui seu metaescalador que verifica as condições da rede para o correto encaminhamento das requisições dos clientes. Utilizando esse modelo o cliente é atendido pelo *data center* que estiver mais próximo a ele diminuindo, assim, a latência na comunicação entre o cliente e o *data center*.

O MACC, usando a camada de interconexão apresentada na Figura 4.5, utiliza políticas de roteamento para a descoberta dos *data centers* que possuem menores latências aumentando assim a qualidade com que os serviços são entregues aos clientes.

O modelo de comunicação proposto e avaliado neste projeto de mestrado pode ser utilizado tanto no MACC como em outros projetos com objetivos semelhantes ou não ao do considerado. Um exemplo de utilização do modelo é o desenvolvimento de componentes para alocação de recursos na nuvem (Vernekar e Game, 2012).

4.6 Qualidade de Serviço em Nível de Rede

A qualidade de serviço, em nível de serviços relacionados à rede, pode ser entendida como um conjunto de requisitos a serem cumpridos pela rede para a entrega de um serviço com qualidade a seus usuários (Crawley et al., 1998). Essa qualidade deve ser obtida observando algumas métricas como: largura de banda, *delay*, *jitter* ou a probabilidade de perda de pacotes (Linnolahti, 2004). A observação dessas métricas pode melhorar a qualidade com que os serviços são entregues aos usuários.

Dentre as preocupações relacionadas à QoS estão: como a conexão é abordada, qual métrica é utilizada (quantidade de saltos ou atraso da rede) e qual a forma de roteamento é utilizada (endereçamento por origem e destino ou por fluxo) (Linnolahti, 2004).

O uso de sistemas P2P, visando obter qualidade de serviço, deve lidar com questões de comunicação e colaboração entre redes distintas, escalabilidade (relacionada com a quantidade de nodos de uma rede P2P), compartilhamento de recursos e informações (Androutsellis-theotokis e Spinellis, 2004).

A confiança é um dos pontos abordados pelo MACC, pois utiliza sistemas P2P para evitar falhas relacionadas à rede, como falta de conectividade e ausência de recursos disponíveis (Peixoto et al., 2010). Normalmente esses são problemas relacionados com o crescimento do ambiente (Linnolahti, 2004).

A qualidade de serviço, no nível de serviços de rede, que é abordada pelo MACC visa diminuir a latência entre dois ou mais pares monitorados pelo MACC. Clientes que estão a longas distâncias dos provedores de computação em nuvem poderão sentir a influência da latência em suas aplicações, particularmente se houver muito tráfego na rede e se os serviços dos clientes forem solicitados em grande quantidade, causando sobrecargas nos servidores (Leavitt, 2009).

4.7 Considerações Finais

Este capítulo abordou as características e sistemas que utilizam P2P para roteamento e descoberta de serviços. Essas características auxiliaram na seleção de quais abordagens devem ser utilizadas e quais métricas utilizar para o desenvolvimento das políticas que podem ser utilizadas pelo MACC.

Como apresentado nos estudos relacionados à descoberta de serviços na nuvem utilizando P2P, duas abordagens foram utilizadas (Pastry e Chord). Sendo assim essas abordagens foram selecionadas para compor a camada de sobreposição que o MACC utiliza.

Dentre as métricas relacionadas à qualidade de serviço, as políticas baseadas em sistemas P2P utilizarão como métrica a quantidade de saltos e a latência para verificar dentre essas métricas qual a que proporciona um melhor rendimento ao sistema considerado.

Definido o escopo do trabalho, as políticas e as métricas a serem utilizados, os próximos capítulos irão apresentar como as políticas foram abordadas e como os resultados foram obtidos com o uso dessas políticas.

Desenvolvimento do Projeto

5.1 Considerações Iniciais

Este trabalho aborda a avaliação de políticas de roteamento em redes P2P no ambiente de computação em nuvem. A avaliação é realizada por meio de simulações considerando o MACC para modelagem do sistema a ser avaliado. Assim, na avaliação das políticas de roteamento considera-se a execução das políticas pelo MACC, utilizando o simulador CloudSim (Calheiros et al., 2010).

O CloudSim possui vários módulos, como é apresentado no Capítulo 2, e dentre esses módulos foram utilizados os módulos referentes ao **código do usuário** e os módulos de **rede** relacionados ao **CloudSim**.

No módulo do código do usuário são definidos os cenários da simulação, como a quantidade de *data centers* presentes na simulação, o número de clientes, a política de escalonamento, o número de *hosts* disponíveis, a quantidade de serviços e as políticas de roteamento a serem utilizadas.

As informações referentes à rede são definidas nos módulos do CloudSim que compõe a rede. Esses módulos utilizam uma topologia de rede, que é definida de acordo com a quantidade de *data centers* e clientes definidos para a simulação no código do usuário, para a localização dos *data centers*.

As definições do ambiente, no CloudSim, guiaram o desenvolvimento das políticas de roteamento que foram desenvolvidas para o MACC. Para o desenvolvimento dessas políticas de roteamento, foi necessário definir uma topologia com informações sobre a rede, para que as

mesmas pudessem ser utilizadas. Assim, antes de definir como utilizar as informações da rede foi necessário modelar a rede de acordo com os cenários de simulação que foram utilizados nos experimentos.

5.2 Modelagem da Rede

O CloudSim possui módulos relacionados à rede que permitem que a mesma seja utilizada para verificar qual o impacto da comunicação nas atividades da nuvem (Calheiros et al., 2010). As funcionalidades do CloudSim, que são relacionadas à rede, são: a topologia de rede e o cálculo do *delay*.

A topologia de rede foi gerada utilizando o Brite (Medina et al., 2001). Para a geração dessa topologia foi necessário definir alguns parâmetros como: o algoritmo para ligação dos nodos e a quantidade de nodos de cada topologia. O modelo de topologia utilizado foi o *top-down* (Medina et al., 2001).

Esse modelo é formado pela ligação de vários sistemas autônomos (AS) como apresentado na Figura 5.1.

Figura 5.1: Modelo Topológico Utilizado

O modelo de topologia, apresentado pela Figura 5.1, é composto por regiões de sistemas autônomos (em vermelho na figura) e a ligação dessas regiões com outras regiões de sistemas autônomos. Esse modelo foi escolhido por representar as regiões de forma separada.

A topologia foi utilizada para simular a distribuição dos *data centers* e clientes ao redor do mundo. Além de auxiliar no desenvolvimento das políticas de descoberta de serviço, uma vez

que as regiões possuem limites de domínios. A Figura 5.2 apresenta uma visão geral da possível distribuição dos *data centers* (DC) que podem estar dispersos geograficamente.

Figura 5.2: Distribuição Geográfica dos *data centers*

Como apresentado na Figura 5.2, os *data centers* com as descrições DC, podem estar distribuídos geograficamente em qualquer lugar do globo. A distribuição apresentada pela Figura 5.2 é uma visão geral de como podem estar dispostos os *data centers* e não representam a localização real dos mesmos.

Utilizando a topologia *top-down* como modelo topológico, foram geradas as topologias utilizadas na variação da quantidade de clientes. Foram utilizados para a composição da topologia: o Waxman como modelo de ligação de nodos e a geração do grafo utilizou a quantidade de nodos referentes ao número de usuários e *data centers* que compõem os modelos dos experimentos.

Além das informações anteriormente citadas, para gerar uma topologia BRITE é necessário definir parâmetros como: tamanho do plano, tamanho do quadrante do plano, número de nodos, tipo de ligação dos nodos e parâmetros utilizados pelo modelo de distribuição de nodos (Medina et al., 2001).

Na geração da topologia foram usados os valores padrão para todos os atributos exceto o número de nodos que é relacionado à soma de clientes e *data centers* que variou de acordo com os cenários simulados.

Na elaboração do planejamento de experimentos, um dos fatores utilizados foi o número de clientes que possui duas variações ou dois níveis. Por essa razão foram gerados dois arquivos de topologia.

Como foram utilizadas duas topologias, as mesmas são apresentadas na Figura 5.3 e Figura 5.4.

A Figura 5.3 se assemelha à Figura 5.4 apenas na quantidade de *data centers* e mostra que o BRITE permite a criação de topologias com ligações aleatórias que se aproximam das ligações

Figura 5.3: Rede Utilizada nos Experimentos com 30 Clientes e 15 *data centers*

Figura 5.4: Rede Utilizada nos Experimentos com 60 Clientes e 15 *data centers*

apresentadas por redes reais como mostra (Cheswick et al., 2000) e (Dodge, 2012) em estudos sobre mapeamentos de redes reais.

O CloudSim, usando as topologias geradas, faz um mapeamento de uma entidade (cliente ou *data center*) para um nodo da topologia (Calheiros et al., 2010). Dessa forma, os nodos que estão em vermelho representam os nodos que podem ser utilizados pelos *data centers* e os nodos em azul representam os nodos que podem ser utilizados pelos clientes. Essas topologias são utilizadas pelo MACC para o desenvolvimento e avaliação de seu modelo de comunicação.

5.3 Projeto e Implementação

Após a modelagem da rede, que utiliza a quantidade de clientes e *data centers* para essa modelagem, foi necessário integrar essas topologias ao simulador. Como o código do simulador é aberto, foi possível manipular algumas classes que são fundamentais para o desenvolvimento deste projeto de mestrado. As classes utilizadas são apresentadas na Figura 5.5.

Figura 5.5: Diagrama das Classes Utilizadas (Calheiros et al., 2010).

O diagrama de classes, Figura 5.5, é referente às classes dos módulos do CloudSim que foram utilizados para o desenvolvimento do projeto e, como discutido anteriormente, como o simulador é separado por módulos é possível simular cenários que não sejam dependentes. Por exemplo, se o objetivo é simular criação de máquinas virtuais, não é necessário que outros módulos sejam utilizados (Calheiros et al., 2010). Sendo assim, as classes que estão em vermelho, são as classes que tiveram que ser modificadas para que a influência da rede fosse simulada sobre o ambiente de computação em nuvem.

Devido a esse fato, o diagrama de classes apresentado não possui uma relação direta com as outras classes. Quando a rede é utilizada, na simulação, as demais classes que integram o módulo de rede do CloudSim são utilizadas.

A classe *CloudSim*, *Datacenter*, *VmAllocationPolicy*, *DatacenterCharacteristics* e *DatacenterBroker* são as principais classes utilizadas pelo MACC. O escalonamento local é feito na classe *Datacenter* bem como o cálculo dos recursos utilizados. As políticas de máquinas virtuais e as características do *data center* são utilizadas pelo MACC para analisar como a qualidade de serviço está sendo ofertada aos clientes e como está o cumprimento dos contratos ou SLAs.

A classe *CloudSim* é a responsável por manter as filas de eventos e o controle das entidades do simulador. Essa classe não sofreu qualquer modificação ao longo do desenvolvimento do projeto.

As classes relacionadas à rede e ao desenvolvimento deste projeto de mestrado são: Rede e *Networktopology*. A classe Rede foi criada para definir, dentre os nodos da topologia, quais os nodos disponibilizados para serem mapeados como *data center* e quais nodos disponibilizados para serem mapeados para os clientes.

Além de definir quais os nodos disponibilizados para as entidades, essa classe também é responsável por fazer o mapeamento entre as entidades do simulador e os nodos da topologia invocando assim métodos pertencentes à classe *Networktopology* para realizar essa função.

A classe *Networktopology* possui uma dependência de outras classes que são responsáveis pelo reconhecimento da rede e a geração das matrizes de distância.

A leitura do BRITE é realizada pela classe *GraphReaderBrite* que informa a composição do grafo à classe *NetworkTopology*. Essa leitura é feita da seguinte forma: é informado qual o arquivo de topologia e a partir dos identificadores dos nodos e das arestas o simulador conhece a topologia de rede.

A classe *GraphReaderBrite* está em vermelho no diagrama devido a uma modificação no seu modelo. Como a topologia é estática, arquivo de texto, e o presente projeto de mestrado considera a rede como uma estrutura dinâmica, foi necessário atribuir uma distribuição na leitura das arestas baseado na semente da simulação.

Essa modificação permitiu que a rede tivesse um comportamento dinâmico e que pode ser repetido e controlado, pois é baseado na semente da simulação.

Já a classe *DelayMatrix_Float* faz parte da construção da matriz de distâncias entre os nodos. Nessa classe, foi adicionado um método para contar a quantidade de saltos de um nodo para outro. Esse método consiste em verificar quantas iterações são necessárias para ir de um nodo origem para um nodo destino sem considerar os pesos das arestas.

Esse método auxiliou no desenvolvimento dos experimentos para avaliar qual métrica utilizar nos experimentos. Além de construir a matriz de distâncias, a classe *DelayMatrix_Float* é responsável pelo cálculo do atraso da rede.

O cálculo do atraso da rede é utilizado para fazer a integração entre as entidades do CloudSim e a topologia da rede e, a partir das informações de atraso de um nodo origem para um nodo destino, formar uma matriz de latência. Essa matriz de latência é utilizada pelas políticas de roteamento que usam a latência como métrica para a descoberta dos *data centers*.

A Figura 5.6 apresenta o fluxo de comunicação utilizado pelo CloudSim nas transações de envio e recebimento de mensagens (Calheiros et al., 2010).

O fluxo apresentado na Figura 5.6 é utilizado pelo simulador para acrescentar os tempos gastos na rede ao tempo final da simulação. Como apresentado na Figura 5.6, o cliente cria uma mensagem e a envia para ser processada. Nessa operação é acrescentado ao tempo de simulação o tempo de envio da mensagem (1). A mensagem é recebida e enviada para ser processada. O tempo de envio (2) é acrescentado ao tempo de simulação. Por fim, o tempo de entrega das solicitações (3) também é adicionado ao tempo de simulação.

Figura 5.6: Fluxo de Comunicação Utilizado pelo CloudSim. Adaptado de Calheiros et al. (2010).

Dessa forma, é utilizada a latência entre a origem e o destino para simular o comportamento do experimento considerando a execução e a comunicação da rede. Sempre que o cliente fizer uma solicitação é acrescentado ao tempo da simulação o seu atraso. Ao fim da simulação tem-se: $Tr = 3d + t$, onde, Tr é o tempo de resposta, d é o atraso (*delay*) obtido na comunicação e t representa o tempo de execução dos serviços pelos *data centers*.

Por fim, o desenvolvimento deste projeto de mestrado utilizou a classe *DatacenterBroker*, em vermelho na Figura 5.5, para o desenvolvimento das políticas de roteamento.

O simulador CloudSim não possui um modelo de roteamento, sendo necessário o desenvolvimento do mesmo. Sendo assim, baseado na literatura, foram desenvolvidas políticas de roteamento na classe *DatacenterBroker* possibilitando assim que as políticas, *Round Robin*, Baseada na Capacidade da Rede, Pastry e Chord fossem disponibilizadas ao código do usuário (*Main*) o que viabilizou a avaliação de desempenho das mesmas.

5.4 Algoritmos de Roteamento

A quantidade de saltos e as informações referentes ao estado da conexão entre dois nodos são dois parâmetros que podem ser utilizados como métricas para construção de algoritmos de roteamento (Linnolahti, 2004). Dessa forma, as políticas de roteamento desenvolvidas utilizaram como métricas: a quantidade de saltos entre a origem e o destino e a informação de latência entre um nodo origem e um nodo destino.

Dentre as políticas de roteamento, foram utilizadas: *Round Robin*, Busca Baseada na Capacidade da Rede (BCR) sem camada de sobreposição e duas abordagens P2P com camadas de sobreposição distintas.

O **Round Robin** é uma política bem difundida para a seleção de serviços que baseia-se em uma lista circular com os *data centers*. A cada solicitação de cliente, o próximo *data center* da lista será considerado. Nessa abordagem nenhuma característica da rede é considerada para a seleção do *data center*, não causando assim qualquer sobrecarga ao MACC na tomada de decisões.

Essa abordagem foi escolhida por ser utilizada atualmente por provedores de serviço de nuvem, principalmente os provedores que utilizam o *GoGrid*, e por possibilitar a comparação entre a observação ou não das características da rede para realizar o roteamento.¹

Figura 5.7: Busca Utilizando Round Robin

A Figura 5.7 representa um cliente fazendo uma solicitação ao MACC. O MACC possui uma lista de *data centers* disponíveis e envia o primeiro da lista ao cliente. Após essa disponibilização o *data center* é indicado como já utilizado, saindo assim, do início da lista.

O *data center* selecionado, representado pelo índice em vermelho, aguarda até que a lista seja esvaziada para voltar a ficar apto a receber solicitações.

A **Busca Baseada na Capacidade da Rede (BCR)** é uma política que considera as informações da rede para a busca do melhor *data center*. Nessa abordagem é analisada a métrica utilizada para a tomada de decisão. Se a métrica utilizada for a quantidade de saltos, o cliente irá consultar todos os *data centers* para descobrir, dentre os *data centers*, qual possui a menor quantidade de saltos. O mesmo ocorre quando a latência é utilizada como métrica.

Como apresentado na Figura 5.8, é verificado entre todos os *data centers* disponíveis, qual o *data center* que apresenta a menor distância para o cliente. No caso da Figura 5.8, o índice em vermelho não é retirado da lista de *data centers* aptos a receber solicitações.

A BCR sempre irá retornar a melhor opção para o cliente, contudo, ela possui uma desvantagem com relação à quantidade de interações geradas, pois cada cliente irá consultar todos os *data center* disponíveis. Para minimizar a sobrecarga apresentada pela BCR pode-se utilizar outras abordagens que considerem as informações da rede, como por exemplo as redes P2P.

¹[wiki.gogrid.com/wiki/index.php/\(F5\)_Load_Balancer#Load_Balancer_Type](http://wiki.gogrid.com/wiki/index.php/(F5)_Load_Balancer#Load_Balancer_Type) Apresenta duas políticas para realizar o roteamento. Uma dessas políticas é o Round Robin e a segunda Least Connect, envia serviços para os servidores menos sobrecarregados ou que possuam menos conexões ativas.

Figura 5.8: Busca Utilizando BCR

No Capítulo 4 foram apresentadas as características dos sistemas P2P. Dentre as características e abordagens referentes à P2P, foram selecionadas as abordagens baseadas em DHT para o desenvolvimento das políticas utilizadas pelo MACC. As políticas baseadas em DHT utilizadas neste projeto são: a Pastry e a Chord devido ao fato das duas possuírem um espaço de busca menor do que o apresentado pela BCR e apresentarem diferenças estruturais em si.

5.4.1 Pastry

Segundo Rowstron e Druschel (2001) uma rede de sobreposição Pastry deve possuir um identificador numérico único chamado *nodeid*. Esse *nodeid* é utilizado para enviar mensagens entre os nodos que possuam os *nodeids* numericamente mais próximos do valor da chave entre todos os nós ativos. Como não foram utilizadas descrições de serviço, os *nodeids* foram gerados através do mapeamento da rede facilitando, assim, a identificação das entidades do simulador. Essas entidades correspondem aos *data centers* e aos clientes. A topologia utilizada pela abordagem Pastry se assemelha a uma árvore e o processo de busca assemelha-se ao processo de busca binária. A Figura 5.9 ilustra a camada de sobreposição utilizada pelo Pastry.

A topologia apresentada na Figura 5.9 é uma sobreposição gerada sobre as topologias apresentadas nas Figura 5.3 e Figura 5.4. A busca na camada de sobreposição é voltada para a descoberta do *data center* com as melhores condições para o cliente. A busca é iniciada no nodo raiz da árvore e segue para a direita ou para a esquerda dependendo de qual nodo possui a menor distância no momento da busca. Essa informação é armazenada pelo MACC e ao final da operação o MACC tem conhecimento de qual *data center* é o mais adequado para fornecer melhores tempos de resposta.

O processo de busca é demonstrado na Figura 5.10, onde são apresentados os passos realizados na busca.

A busca é iniciada na metade da lista e continua até que o espaço de busca definido pelos *nodeids* termine. No caso da Figura 5.10, é demonstrada uma busca onde o *data center* adequado é o de identificação 101. O índice do *data center* continua na lista de aptos à execução de serviços.

Figura 5.9: Topologia Pastry

Figura 5.10: Busca Utilizando Pastry

A vantagem do Pastry em relação ao BCR é a diminuição no espaço de busca. Essa política não consulta todos os nodos para a tomada de decisão. Esse fato diminui a quantidade de iterações realizadas pelo MACC.

Como o objetivo é avaliar os sistemas P2P hierárquicos, mais de um sistema foi utilizado. A segunda abordagem utilizada foi a Chord que possui topologia e lógica de busca diferente das apresentadas pelo Pastry.

5.4.2 Chord

Uma abordagem diferente para a criação da camada de sobreposição e consequentemente a realização de roteamento em sistemas P2P, é a utilização do sistema Chord. A topologia formada pelo Chord é em forma de anel e a partir dessa topologia a busca é realizada (Stoica et al., 2001). O Chord trabalha de forma diferente da apresentada pelo Pastry. O MACC, de

posse das informações sobre a rede, forma a busca de acordo com a região em que o cliente se encontra. A Figura 5.11 apresenta essa topologia.

Figura 5.11: Topologia Chord

Os clientes são encaminhados a uma região de acordo com os *nodeids* do cliente aos *data centers* mais próximos (Stoica et al., 2001). Depois de verificar qual a região que o cliente melhor se encaixa, o MACC realiza a busca nos *data centers* da região em que o cliente faz parte. A Figura 5.12 apresenta o processo de busca realizado pela abordagem Chord.

Figura 5.12: Busca Utilizando Chord

A lista de data centers é ordenada de acordo com a região onde os *data centers* se encontram. Sendo assim, depois que o MACC identifica a região que o cliente pertence, a busca é realizada entre os *data centers* mais próximos ao cliente.

Assim como o Pastry, o Chord possui um espaço de busca menor se comparado ao BCR. O fato da busca ser concentrada em regiões próximas ao cliente faz com que a quantidade de iterações realizadas pelo MACC seja menor, já que quanto menor a quantidade de iterações menor a sobrecarga associada ao MACC.

5.5 Considerações Finais

Este capítulo apresentou uma descrição dos módulos que foram utilizados no desenvolvimento deste projeto de mestrado bem como as políticas desenvolvidas para o MACC. A identificação do ambiente utilizado é importante para a avaliação das políticas desenvolvidas, pois facilita a definição dos fatores a serem considerados nos experimentos.

A topologia de rede apresentada foi desenvolvida em relação ao número de entidades presentes no cenário de simulação. Essas entidades fazem parte do CloudSim e são utilizadas como parâmetros pelas políticas de roteamento para a definição do envio de serviços dos clientes.

Com a definição do modelo de comunicação proposto pelo MACC foi possível definir quais abordagens utilizar na definição das políticas de roteamento. Assim, foram desenvolvidas: abordagens que não se preocupam com a rede para demonstrar qual o impacto do roteamento sem informações da rede; abordagens de força bruta utilizando as informações da rede para definir qual o melhor *data center* para o cliente e dois sistemas P2P para serem comparadas com as abordagens de força bruta e, assim, verificar como a qualidade dos serviços podem ser afetadas de acordo com o modelo de comunicação adotado.

Ao invés de utilizar apenas um sistema P2P para comparar com as outras políticas, foram utilizadas duas abordagens com ordens topológicas diferentes para avaliar se existem diferenças entre as abordagens e suas topologias.

A avaliação de várias abordagens de roteamento permite definir qual será a abordagem que será utilizada pelo MACC. O uso de políticas de roteamento, que possuem baixos tempos de resposta, auxilia o MACC na entrega de serviços aumentando a qualidade de serviço oferecida aos clientes.

Resultados

6.1 Considerações Iniciais

A escolha das políticas de roteamento utilizadas para a avaliação do modelo de comunicação do MACC foi baseada nos trabalhos relacionados aos modelos de comunicação e algoritmos de roteamento que são utilizados pela computação em nuvem.

Como nos trabalhos relacionados não é apresentada uma avaliação de desempenho detalhada das políticas de roteamento, este trabalho de mestrado avaliou as políticas de roteamento apresentadas na literatura visando à seleção das políticas mais adequadas a serem utilizadas pelo modelo de comunicação do MACC.

De acordo com Fortier e Michel (2003), a avaliação de desempenho é necessária quando se deve comparar um número de alternativas para o desenvolvimento ou melhoria de um sistema e escolher dentre as alternativas qual satisfaz os objetivos do sistema. No Capítulo 5 foram apresentadas três políticas de roteamento que utilizam informações da rede. O primeiro passo para implementação dessas políticas é a definição de qual métrica utilizar.

Linnolahti (2004) apresenta duas métricas principais no desenvolvimento de políticas de roteamento: quantidade de saltos (*hops*) e latência entre um nodo origem e um nodo destino. Foi realizado um conjunto de experimentos para definir qual métrica utilizar no desenvolvimento e avaliação das políticas de roteamento.

Após a definição da métrica, serão apresentados os resultados da avaliação das políticas, utilizando como variável de resposta o tempo de resposta. Para identificar as diferenças com bases estatísticas, cada experimento foi executado dez vezes para identificar as possíveis variações

nos resultados obtidos. Para traçar os intervalos de confiança, todos os experimentos utilizaram um $\alpha = 0,05$ que corresponde a 5% de erro.

6.2 Seleção da Métrica

A seleção da métrica foi o primeiro passo antes de realizar os experimentos para observar a qualidade de serviço que o MACC oferece com relação à entrega de serviços aos clientes. Para a seleção da métrica, as políticas que utilizam informações da rede foram submetidas a uma avaliação de desempenho para a descoberta da melhor métrica a ser utilizada pelo MACC.

A metodologia de avaliação de desempenho utilizada foi o fatorial completo para a composição dos testes e análise dos dados. Como apresentado por Jain (1991) antes de iniciar uma avaliação de desempenho é preciso definir quais serão os fatores fixos (fatores que não irão sofrer mudanças em todos os experimentos, mantendo o mesmo ambiente para todo o conjunto de experimento) e fatores variáveis (fatores que passarão pela avaliação de desempenho para definir qual proporciona melhores resultados).

A Tabela 6.1 apresenta os fatores que foram fixados para os experimentos relacionados à avaliação de qual métrica utilizar.

Tabela 6.1: Fatores Fixos definidos para todos os Experimentos

Máquina	Valores
Processador	Intel <i>Quad 2 core</i>
Memória	4 GB RAM
Sistema Operacional	Ubuntu x64 10.04
Java	1.6.0_26
Simulador	CloudSim 2.1
Número de <i>Data Centers</i>	15
Memória do <i>Host</i>	16 GB
Processador do <i>Host</i>	12000 MIPS
Conexão <i>Host</i>	1 Gb
Virtualizador	Zen

A Tabela 6.1 apresenta os fatores fixos dos experimentos realizados. A primeira parte, relacionado à máquina, é o computador real onde a simulação foi realizada. Já as informações referentes ao simulador se referem a como foi simulado o ambiente da nuvem.

Os fatores variáveis são apresentados na Tabela 6.2 e representam o primeiro experimento realizado. Essa tabela é gerada de acordo com a metodologia apresentada por (Jain, 1991) e (Fortier e Michel, 2003).

A Tabela 6.2 apresenta os experimentos que foram realizados para verificar a influência das métricas nos tempos de resposta. Nesta avaliação a carga de trabalho foi fixada em dez serviços do tipo leve (carga de *Web Services*) para cada cliente (aqui representados pelos nodos). Assim,

Tabela 6.2: Fatores Variáveis (1º Experimento)

Experimento	Métrica	Nodos
1	Salto	30
2	Salto	60
3	Latência	30
4	Latência	60

para esse experimento consideram-se dois fatores. O primeiro é a métrica, que assume os níveis Número de saltos e latência da rede e o segundo considera o número de nodos ou clientes, que assume os níveis 30 e 60.

A política utilizada neste primeiro conjunto de experimento foi a BCR. A Figura 6.1 apresenta os tempos de resposta obtidos utilizando as métricas consideradas.

Figura 6.1: Diferença nos Tempos de Resposta

A Figura 6.1 apresenta o gráfico com as diferenças dos tempos de resposta mostrando o intervalo de confiança. O gráfico mostra que quando a latência é utilizada obtém-se melhores tempos de resposta tanto com trinta quanto com sessenta nodos. Além de menores tempos, a variação (representada pelo intervalo de confiança) da métrica latência é menor que a apresentada pela métrica salto mostrando que os tempos de resposta tiveram pouca variação.

A Figura 6.2 apresenta a distribuição dos tempos de resposta de acordo com as réplicas utilizadas.

Como mostra a Figura 6.2, os tempos de resposta utilizando a métrica latência possuem pouca variação enquanto a métrica saltos possui baixa oscilação com trinta usuários. No entanto, essa variação se torna maior com sessenta usuários.

A política BCR com saltos encontra a menor quantidade de saltos de uma origem para um destino, porém, a menor quantidade de saltos nem sempre reflete menor latência uma vez que o canal de comunicação pode estar congestionado ou possuir uma grande distância para o destino.

Figura 6.2: Distribuição dos Tempos de Resposta

Nos experimentos considerados, os níveis mais baixos são: saltos para a métrica e trinta para os nodos. Esses níveis são utilizados para demonstrar a influência dos fatores considerados na variável de resposta que, nesse experimento, foi o tempo de resposta.

Figura 6.3: Influência dos Fatores na Variável de Resposta.

A Figura 6.3 apresenta a influência dos fatores utilizados. Nota-se que existe uma linha traçada no plano que é definida por uma distribuição normal que é utilizada para verificar qual a influência dos fatores. Quanto mais distante da linha, maior a influência do fator.

Outro ponto a ser observado é que há valores positivos e negativos. Esses valores são definidos com relação à troca dos níveis. Como o nível mais baixo para o fator A foi a quantidade de saltos, na troca de níveis (latência como nível mais alto) os tempos de resposta diminuíram ao invés de aumentar.

O mesmo comportamento não é apresentado pelo fator B que vai do nível mais baixo (trinta) para o mais alto aumentando seu tempo de resposta. O fator AB, por se tratar de uma combinação entre os dois fatores citados anteriormente, diminui o tempo de resposta devido à influência do fator A.

Como a política Chord também utiliza informações da rede, uma avaliação de desempenho foi realizada seguindo os moldes da Tabela 6.2 para verificar a influência da métrica na utilização da política.

A Figura 6.4 apresenta as diferenças nos tempos de resposta obtidos no segundo conjunto de experimentos.

Figura 6.4: Diferença nos Tempos de Resposta (política Chord).

Diferente da política BCR, a política Chord possui tempos estatisticamente iguais entre as métricas utilizadas quando 30 nós são considerados. Para sessenta nós os tempos de resposta são estatisticamente diferentes. Mesmo havendo uma diferença, ela é baixa, o que mostra que a política Chord pode ser utilizada independente da métrica utilizada apresentando resultados similares e pequenos intervalos de confiança.

Figura 6.5: Distribuição dos Tempos de Resposta em relação às Réplicas

A Figura 6.5 mostra o comportamento de cada repetição com relação ao tempo de resposta. Como não há uma grande oscilação nos tempos de resposta, pode-se dizer que o mesmo comportamento é encontrado tanto com a métrica de nível mais baixo quanto com o nível mais alto.

A influência dos fatores considerados neste experimento é apresentada na Figura 6.6 onde é possível observar a influência na mudança dos níveis mais baixos para os níveis mais altos.

Figura 6.6: Influência dos Fatores no Segundo Experimento.

Por não apresentar uma diferença muito acentuada, a mudança do nível mais baixo para o mais alto resultou em uma pequena diminuição do tempo de resposta. Esse comportamento fez com que o fator *A* ficasse próximo à linha normal que representa a ausência de influência nas variáveis de resposta.

O fator *B*, assim como no primeiro experimento, teve uma grande influência devido à mudança significativa no número de nodos utilizados. O fator *AB*, por se tratar de um conjugado e como o fator *A* possui uma influência muito baixa, não apresentou influência significativa ficando assim no domínio da linha de erro.

Para finalizar a seleção da métrica, um conjunto de experimentos, também utilizando o modelo da Tabela 6.2, foi considerado. Nesse conjunto de experimentos foi utilizada a política Pastry. A Figura 6.7 apresenta as diferenças observadas nos tempos de resposta.

Figura 6.7: Diferença nos Tempos de Resposta (política Pastry).

A diferença apresentada nessa figura com relação às políticas já discutidas é que a métrica não apresenta uma grande diferença nos tempos de resposta, sendo que, como apresentado na

Figura 6.7 os tempos de resposta tanto para 30 quanto para 60 nodos são estatisticamente iguais. A Figura 6.8 apresenta o comportamento do tempo de resposta de acordo com a evolução das réplicas.

Nota-se que existem algumas oscilações entre as métricas, mas na média o comportamento é muito próximo uma da outra.

Figura 6.8: Distribuição dos Tempos de Resposta em relação às Réplicas

Esse comportamento, com baixas oscilações, é o que faz com que os intervalos de confiança sejam baixos.

A Figura 6.9 apresenta a influência dos fatores no terceiro conjunto de experimentos.

Figura 6.9: Influência dos Fatores no Segundo Experimento.

Diferente dos resultados anteriores, o fator *A*, quando alterado o nível mais baixo para o mais alto, possui influência positiva. Isso significa que mesmo não sendo estatisticamente diferentes, os tempos de respostas ficam levemente mais altos utilizando a métrica latência.

Como o fator *A* está no limite da normal, o fator *AB* não apresenta nenhuma influência, assim como o apresentado pela política Chord. O fator *B*, como nos outros casos, foi o mais impactante por ter um aumento considerável no número de nodos.

Como o objetivo da avaliação de desempenho é possibilitar comparações justas (Jain, 1991) a métrica escolhida foi a latência. Como nos sistemas que utilizam sobreposição a métrica

apresentou baixas influências, chegando sempre perto do limite do erro, qualquer uma das duas métricas pode ser utilizada como parâmetro para a política de roteamento.

A escolha pela latência teve como maior influenciador a política BCR, pois apresenta uma diferença acentuada entre as duas métricas.

A comparação entre as políticas de roteamento gera um grande volume de dados. Assim, restringir a métrica a ser utilizada torna-se importante.

6.3 Planejamento

Depois de definir a métrica a ser utilizada, apresenta-se um planejamento de experimentos para a correta avaliação das políticas e dos resultados obtidos. Nessa fase do projeto é utilizado o modelo fatorial completo para a modelagem do conjunto de experimentos e a avaliação dos dados.

Para a definição do conjunto de experimentos, os fatores fixos utilizados são os mesmos apresentados na Tabela 6.1, porém, o conjunto de experimentos utilizou fatores e níveis diferentes dos utilizados na seção anterior. Na elaboração dos experimentos e comparação das políticas de roteamento, são utilizados três fatores: política de roteamento, tipo de serviço e número de clientes.

As políticas de roteamento são comparadas com relação ao tempo de resposta com que cada uma das políticas entrega os serviços e não relacionado ao tempo de tomada de decisão proporcionado por cada política. Essa decisão foi tomada com base nas informações da Figura 6.10.

Figura 6.10: Tempos de tomada de decisão dos algoritmos.

A Figura 6.10 apresenta os tempos de tomada de decisão de cada algoritmo. Para chegar a esses valores, os algoritmos foram avaliados usando o tempo de início e o tempo de fim de cada algoritmo considerando a entrada e a saída de requisições. Inicialmente foram executadas 10 repetições do algoritmo BCR e os resultados obtidos apresentaram desvio padrão baixo, devido

a esse fato foi adotada a quantidade de 10 repetições para os demais algoritmos para traçar os intervalos de confiança e assim fazer uma comparação estatística entre eles. A política *Round Robin*, por possuir apenas uma operação de atribuição possui seus tempos em escala de nano segundos e por isso não está presente no gráfico.

Como apresentado na Figura 6.10, os tempos do algoritmo BCR são maiores que os apresentados pelos outros algoritmos por se tratar de um algoritmo de força bruta que possui mais iterações que os demais. Os algoritmos Chord e BCR, por possuírem classes assintóticas semelhantes, possuem tempos estatisticamente iguais.

Apesar de oferecer diferenças de tempos significativas entre si, nos tempos de resposta utilizados esses valores se tornam insignificantes, devido, principalmente, a escala que é considerada. Os tempos de resposta obtidos nos experimentos são na ordem de segundos enquanto o tempo de tomada de decisão dos algoritmos é na escala de milissegundos.

Os fatores tipo de serviço e número de clientes possuem dois níveis nos conjuntos de experimentos e as políticas de roteamento somam quatro níveis ao todo. Para uma análise mais detalhada, os níveis das políticas de roteamento foram agrupados dois a dois sendo assim comparado a política *Round Robin* com a política BCR, a política BCR com a política Chord, a política BCR com a política Pastry e finalmente a política Chord com a política Pastry.

Essa divisão possibilitou a utilização do modelo de organização de experimentos na forma N^f (Jain, 1991) onde o "N" representa a quantidade de níveis e "f" a quantidade de fatores.

O conjunto de experimentos utilizados foi então 2^3 possibilitando a criação de 8 experimentos. A combinação entre níveis e fatores auxilia na identificação da influência de cada fator na variável de resposta considerada. A variável de resposta utilizada neste projeto de mestrado foi o **tempo de resposta** na solicitação de um cliente por um serviço.

Para os gráficos que utilizam o erro (diferenças nos tempos de resposta utilizando intervalos de confiança e gráficos de influência de fatores) é considerado um $\alpha = 0,05$, ou seja, é considerado 5% de erro.

6.4 Comparação entre as Políticas de Roteamento

Nesta seção são apresentados os resultados obtidos. Todos os resultados apresentados nesta seção seguem o modelo de planejamento definido anteriormente.

6.4.1 Política *Round Robin* e Política BCR

Como apresentado nos trabalhos relacionados à descoberta de serviços, a política *Round Robin* é utilizada por fornecedores de computação em nuvem para a descoberta e escalonamento de serviços. Como este trabalho visa a descoberta de serviços, o *Round Robin* foi escolhido para compor um conjunto de experimentos.

O conjunto de experimentos que serão apresentados segue o modelo apresentado na Tabela 6.3.

Tabela 6.3: Planejamento de Experimentos (RR e BCR)

Experimento	Política	Clientes	Tipo do Serviço
1	RR	30	Leve
2	RR	30	Pesado
3	RR	60	Leve
4	RR	60	Pesado
5	BCR	30	Leve
6	BCR	30	Pesado
7	BCR	60	Leve
8	BCR	60	Pesado

O planejamento dos experimentos apresentados na Tabela 6.3 tem como objetivo demonstrar qual o efeito de escolher uma das duas políticas com relação aos tempos de resposta.

Como apresentado na Tabela 6.3, as políticas que serão analisadas são a *Round Robin* (RR) e a Baseada na Capacidade da Rede (BCR). Para verificar o comportamento do MACC com relação à quantidade de clientes, foram utilizados como níveis trinta e sessenta usuários e a carga de trabalho (tipo de serviço) foi definida, seguindo o modelo utilizado por Calheiros et al. (2011), como serviços leves e serviços pesados.

Os serviços leves correspondem às cargas geradas por acessos às páginas Web. Como apresentado por Calheiros et al. (2011), as cargas leves representam aplicações com pequena quantidade de processamento. Nos cenários de simulação, a definição do tamanho das *cloudlets* é que define esse tipo de serviço. No nível leve para o fator referente ao tipo de serviço, foi considerada uma distribuição uniforme com tamanhos de serviço que poderiam variar de 2000 até 2500 MIPS (Milhões de Instruções Por Segundo).

Já os serviços pesados representam as aplicações científicas. Normalmente esses serviços são relacionados a grandes quantidades de processamento como cálculo de números complexos (Calheiros et al., 2011). Nos cenários utilizados para a simulação, assim como para os serviços leves, foi considerada uma distribuição uniforme com tamanhos de serviço variando de 20000 até 25000 MIPS.

Tanto os serviços leves como os serviços pesados utilizam a semente da simulação para variar os tamanhos dos serviços de acordo com o limite superior e inferior definido para o tamanho das *cloudlets*. Nos dois casos a solicitação dos clientes é modelada com base na carga de trabalho de aplicações do tipo *Bag-of-Tasks* (BoT) onde as solicitações dos clientes são atendidas em paralelo (Iosup et al., 2008) (Calheiros et al., 2011).

A Figura 6.11 mostra as diferenças de tempos de resposta obtidos pelas duas políticas. As linhas verde (serviços leves) e azul (serviços pesados) representam a política BCR e as linhas preta (serviços leves) e vermelha (serviços pesados) representam a política *Round Robin*.

A Figura 6.11 apresenta dois eixos, onde o eixo *X* representa os tempos obtidos e o eixo *Y* representa a densidade dos tempos de resposta. Essa densidade é relacionada a concentração

Figura 6.11: Histograma dos Tempos de Resposta.

de tempos de resposta iguais. Quanto maior a densidade, menor o desvio padrão obtido. Na Figura 6.11 os desvios maiores são da política *Round Robin*. Isso demonstra que a variação nos tempos de resposta foi relativamente grande se comparado com a política BCR.

A Figura 6.12 apresenta as diferenças nos tempos de resposta observando os intervalos de confiança que foram obtidos a partir dos desvios padrões apresentados na Figura 6.11.

Figura 6.12: Intervalos de Confiança

Como é apresentado na Figura 6.12, os tempos obtidos pela política BCR são menores que os obtidos pela política *Round Robin*. Isso se deve a não observação das características da rede pela política *Round Robin*. Na média, os tempos obtidos com trinta e sessenta usuários na política BCR são mais próximos que os tempos médios obtidos pela política *Round Robin*. Além de menores tempos, a política BCR apresenta menores intervalos de confiança.

Os intervalos superiores e inferiores podem ser detalhados na Figura 6.13 que apresenta os valores individuais de cada experimento.

Como observado na Figura 6.13, a escala do gráfico começa em dez unidades para facilitar a identificação dos tempos de cada replicação. Os tempos da política *Round Robin* obtiveram apenas duas réplicas com tempos abaixo de 13 segundos com trinta clientes e duas réplicas

Figura 6.13: Gráfico de Valores Individuais.

abaixo de 16 segundos para sessenta clientes. Diferente da política *Round Robin* a política BCR obteve tempos parecidos em todas as replicações mostrando que todos os clientes foram atendidos de forma igual.

A influência dos fatores no tempo de resposta é apresentada na Figura 6.14.

Figura 6.14: Influência dos Fatores.

A Figura 6.14 apresenta uma linha normal que é construída de acordo com a quantidade de experimentos realizados. Quanto mais afastados da linha base, maiores serão os efeitos sobre a variável de resposta. Como apresentado, o fator A é o fator que possui maior influência. Este fator é negativo devido à mudança de nível que ocorre no fator. O nível mais baixo é a política *Round Robin* que possui maiores tempos de resposta.

Quando o nível passa da política *Round Robin* para a política BCR os tempos de resposta diminuem. Devido a esse fato o valor da influência do fator A ficou negativo.

A segunda maior influência é proporcionada pelo fator B que possui um crescimento considerável com a mudança de nível. Nesse caso, quando o nível mais baixo (trinta clientes) muda para o nível mais alto (sessenta clientes) os tempos de resposta aumentam. Devido a esse fato os efeitos do fator B são positivos, ou seja, eles aumentam os tempos de resposta com a mudança de nível.

Como o fator C está próximo à linha base, ele possui menos influência que os fatores A e B nos tempos de resposta. A combinação dos fatores A e B possuem uma influência significativa, pois, é a união dos dois fatores mais influentes. As outras combinações não possuem influência uma vez que estão todos sobre a linha base.

A Figura 6.15 apresenta os principais efeitos dos fatores na variável de resposta com relação à mudança de níveis.

Figura 6.15: Principais Efeitos no Tempo de Resposta.

Os efeitos observados na Figura 6.14 ocorrem devido às mudanças de níveis. A Figura 6.15 apresenta o gráfico com os efeitos. Quando a política de roteamento muda, o tempo de resposta diminui. O oposto ocorre com a quantidade de clientes.

A carga dos serviços varia menos se comparada com os outros fatores. Quanto mais afastado do ponto médio for o resultado, maior é a influência dos fatores. Os valores mostrados na Figura 6.14 são utilizados para descobrir qual a interação entre os fatores (AB , AC , BC , ABC).

Figura 6.16: Interação entre os Fatores.

Na Figura 6.16 a linha vermelha e a linha preta são relacionadas ao roteamento e são utilizadas em relação ao primeiro e ao segundo quadro superiores. No primeiro quadro, quando a política *Round Robin* é utilizada com trinta clientes ele apresenta um tempo de resposta médio de 13 segundos. Quando a quantidade de clientes dobra, os tempos de resposta da política *Round Robin* vão, em média, para 16 segundos.

Essa interação é apresentada na Figura 6.14. A política BCR não acompanha o comportamento da política *Round Robin*. A política BCR tem um leve aumento com relação à mudança no número de clientes.

As políticas de roteamento em relação ao tipo de serviço e os clientes em relação ao tipo de serviço mantêm as mesmas médias na mudança da carga de leve para pesada.

As linhas, quando estão paralelas, indicam que não houve interação entre os fatores observados, ou seja, os tempos de resposta médios não foram alterados com a interação entre o tipo dos serviços e a política de roteamento e nem com a mudança no número de clientes.

Como a política BCR apresentou melhores resultados, ela será utilizada na avaliação das políticas que utilizam redes de sobreposição. Essa avaliação irá permitir verificar quais são as vantagens e desvantagens apresentadas pelas políticas baseadas em P2P.

6.4.2 Política Chord e Política BCR

A política Chord utiliza uma topologia em anel que auxilia na descoberta e busca de recursos. Assim como os experimentos apresentados anteriormente, os experimentos apresentados nesta seção seguem a metodologia fatorial completo (Jain, 1991). A Tabela 6.4 apresenta a modelagem dos experimentos.

Tabela 6.4: Planejamento de Experimentos (Chord e BCR)

Experimento	Política	Clientes	Tipo do Serviço
1	Chord	30	Leve
2	Chord	30	Pesado
3	Chord	60	Leve
4	Chord	60	Pesado
5	BCR	30	Leve
6	BCR	30	Pesado
7	BCR	60	Leve
8	BCR	60	Pesado

As diferenças obtidas nos tempos de resposta são apresentadas no histograma da Figura 6.17.

Figura 6.17: Histograma com os Tempos de Resposta (Chord e BCR).

Apesar da política BCR apresentar tempos menores que os observados na política Chord, a diferença entre os tempos obtidos por essas duas políticas é bem menor que a diferença observada entre a BCR e a *Round Robin*. A política BCR por sua vez, apresenta desvios menores que refletem em intervalos de confiança menores como apresenta a Figura 6.18.

Figura 6.18: Intervalos de Confiança

Como apresentado nos intervalos de confiança, as duas políticas atendem aos clientes de forma igualitária, ou seja, os tempos de resposta de cada cliente são bem próximos como apresentado na Figura 6.19.

Figura 6.19: Tempos de Resposta Individuais.

Como as duas políticas atendem os clientes de forma equivalente, os tempos de respostas ficam agrupados, como apresentado na 6.19. Esse fato implica que mesmo com menos passos para a descoberta de um serviço, o Chord consegue atender os clientes com o mesmo comprometimento que a política BCR.

A influência dos fatores sobre a variável de resposta é apresentado na Figura 6.20. Diferente do comportamento do primeiro conjunto de experimentos (RR e BCR), o fator de maior influência foi a quantidade de clientes (fator *B*).

Figura 6.20: Influência dos Fatores nos Experimentos (Chord e BCR).

O fator *A* foi o segundo mais influente e assim como nos resultados anteriores possui valores negativos, pois utiliza como nível mais baixo o Chord que possui um tempo de resposta um pouco maior se comparado ao BCR. O conjugado *AB* tem uma influência significativa devido à influência tanto do fator *A* quanto do fator *B* nos tempos de resposta.

Um ponto importante apresentado na Figura 6.20 é que o fator *C* tem uma influência significativa com relação aos tempos de resposta. Isso mostra que a diferença entre as políticas de roteamento anteriores sobrepunham os tempos obtidos pela troca da carga de trabalho.

A Figura 6.21 apresenta os efeitos dos fatores com relação ao nível de cada fator.

Figura 6.21: Efeitos dos Fatores nos Tempos de Resposta.

Na Figura 6.21 a mudança de nível no fator cliente é o que possui maior distância com relação ao ponto médio, por isso esse fator possui uma influência maior. As políticas de roteamento apresentam tempos próximos, sendo que a política BCR apresenta os menores tempos.

Os serviços, nesse conjunto de experimentos, tiveram uma influência maior e isso reflete na posição das médias dos níveis em relação ao ponto médio. Baseado nas informações da Figura 6.21, a Figura 6.22 apresenta a interação entre os fatores.

Os dois quadrados superiores, Figura 6.22, são relacionados ao roteamento em conjunto com a quantidade de clientes (primeiro quadrado) e roteamento em conjunto com o tipo do serviço (segundo quadrado).

Quando o número de clientes muda, os tempos de resposta da política Chord são maiores que os tempos de resposta apresentados pela política BCR. Já os outros fatores não possuem interação uma vez que a mudança de nível forma uma linha paralela, ou seja, muda-se o nível e os tempos médios permanecem os mesmos.

Figura 6.22: Interação dos Fatores (Chord e BCR).

Para completar o cenário referente ao desempenho alcançado pelas diferentes políticas de roteamento consideradas neste trabalho é necessário realizar a avaliação da política Pastry e baseado nos resultados eleger qual a melhor política dentre as apresentadas.

6.4.3 Política Pastry e Política BCR

A política Pastry é uma política que utiliza uma topologia em árvore que auxilia na descoberta e busca por recursos. Como todos os experimentos apresentados até agora, o conjunto de experimentos foi modelado seguindo a metodologia fatorial completo (Jain, 1991). A Tabela 6.5 apresenta a modelagem dos experimentos.

Tabela 6.5: Planejamento de Experimentos (Pastry e BCR)

Experimento	Política	Clientes	Tipo do Serviço
1	Pastry	30	Leve
2	Pastry	30	Pesado
3	Pastry	60	Leve
4	Pastry	60	Pesado
5	BCR	30	Leve
6	BCR	30	Pesado
7	BCR	60	Leve
8	BCR	60	Pesado

As informações contidas na Tabela 6.5 orientaram a execução dos experimentos. A Figura 6.23 apresenta as diferenças nos tempos de resposta entre as duas políticas.

Como apresentado na Figura 6.23, os tempos de resposta da política Pastry são maiores que os tempos obtidos pela política BCR. Além disso, a densidade dos tempos obtidos pela política Pastry (eixo Y) é menor que as obtidas pelo BCR demonstrando que o desvio padrão dos tempos da política Pastry é maior do que o desvio da política BCR. A Figura 6.24 apresenta os intervalos de confiança obtidos.

Apesar de possuir uma densidade menor nos tempos de resposta, se comparado à política BCR, o intervalo de confiança apresentado pela política Pastry foi pequeno. A distribuição individual dos tempos de resposta dos clientes é apresentada na Figura 6.25.

Figura 6.23: Histograma com os Tempos de Resposta (Pastry e BCR).

(a) Intervalos dos Serviços Leves.

(b) Intervalos dos Serviços Pesados.

Figura 6.24: Intervalos de Confiança

Figura 6.25: Tempos de Resposta Individuais (Pastry e BCR).

Os tempos de resposta individuais não estão totalmente agrupados como é o caso da política BCR. Isso demonstra que mesmo com baixos tempos, houve uma diferenciação entre os tempos de resposta dos clientes. Essa diferença é mínima uma vez que a escala foi ampliada para a verificação individual dos pontos. A influência dos fatores é apresentada na Figura 6.26.

Figura 6.26: Influência dos Fatores nos Experimentos (Pastry e BCR).

Assim como na comparação entre a política Chord e a política BCR, a maior influência foi dada pela quantidade de clientes. Seguida do fator *A* e por último o fator *C*. Um ponto a ser observado é que a junção dos fatores *A* e *B* (*AB*) não proporcionou uma influência significativa como o apresentado nos experimentos realizados anteriormente.

A Figura 6.27 apresenta os principais efeitos dos fatores sobre a variável de resposta.

Figura 6.27: Efeitos dos Fatores nos Tempos de Resposta (Pastry e BCR).

A mudança de nível proporcionada pelo fator clientes, Figura 6.27, é a que mais se distancia da média, implicando que o fator cliente foi o fator que mais impactou nos tempos de resposta. Os outros fatores seguiram as tendências dos experimentos analisados anteriormente. A Figura 6.28 apresenta a interação entre os fatores que influenciam no tempo de resposta.

Figura 6.28: Interação dos Fatores (Pastry e BCR).

Na Figura 6.26 os fatores significantes foram apenas o A , B e C , pois como apresentado na Figura 6.28 não há interação significativa entre os fatores. Todos os resultados apresentados na Figura 6.28 estão paralelos, ou seja, o comportamento das médias se manteve proporcional com a mudança de nível.

6.5 Política Pastry e Política Chord

Como os resultados relacionados com as políticas baseadas em P2P tiveram um comportamento semelhante em relação à política BCR (tempos individuais, baixos intervalos de confiança e influência de fatores), é necessário realizar uma avaliação de desempenho entre as políticas que utilizam sistemas P2P para verificar em quais situações cada política apresenta melhores desempenhos.

O planejamento de experimentos utilizado, para obter os resultados a serem apresentados, foi obtido utilizando o modelo fatorial completo (assim como os demais experimentos) formando o conjunto de experimentos apresentados na Tabela 6.6.

Tabela 6.6: Planejamento de Experimentos (Chord e Pastry)

Experimento	Política	Clientes	Tipo do Serviço
1	Chord	30	Leve
2	Chord	30	Pesado
3	Chord	60	Leve
4	Chord	60	Pesado
5	Pastry	30	Leve
6	Pastry	30	Pesado
7	Pastry	60	Leve
8	Pastry	60	Pesado

As diferenças obtidas nos tempos de resposta entre as duas políticas são apresentadas no histograma da Figura 6.29.

Figura 6.29: Histograma com os Tempos de Resposta.

A Figura 6.29 mostra que para os serviços leves (linha preta e linha verde) a política Chord apresenta melhores tempos de resposta independente da quantidade de clientes. Para os serviços

do tipo pesado (linha azul e linha vermelha) a política Pastry apresenta melhores tempos de resposta.

Para poder afirmar se uma política realmente está melhor que a outra, a Figura 6.30 apresenta os intervalos de confiança para os dois tipos de serviço.

Figura 6.30: Intervalos de Confiança

Apesar dos tempos médios serem bem próximos, as figuras com os intervalos de confiança mostram que os intervalos não se sobrepõem, ou seja, há uma mudança nos tempos de resposta quando existe uma mudança de carga.

As diferenças nos tempos de resposta entre as duas políticas reflete no tempo de resposta para os clientes. Sendo assim, a Figura 6.31 apresenta a influência dos fatores considerados neste conjunto de experimento.

Figura 6.31: Influência dos Fatores nos Experimentos.

Diferente dos outros conjuntos de experimentos, na Figura 6.31, o fator *C* é o mais influente nos tempos de resposta. Seguido pelo fator *B* e por último a união dos fatores *A* e *C*. Esse fato mostra que, dependendo do domínio da aplicação, uma política é melhor que a outra. Por esse motivo, o fator *AC* possui influência nos tempos de resposta enquanto o fator *A* não é influente.

Figura 6.32: Efeitos dos Fatores nos Tempos de Resposta.

Os efeitos de cada fator em separado são apresentados na Figura 6.32.

Como apresentado na Figura 6.32, as mudanças de níveis mais significativos são relacionadas aos tipos de serviços e quantidade de clientes. O tempo médio entre as políticas de roteamento são muito próximos ao ponto médio. Devido a isso, o roteamento, de forma isolada, não apresenta influência nos tempos de resposta. A interação entre os fatores e seus respectivos níveis é apresentada na Figura 6.33.

Figura 6.33: Interação dos Fatores.

A Figura 6.33, na interação roteamento e tipos de serviços, apresenta as diferenças encontradas entre as duas políticas em relação aos tempos e resposta (eixo Y). A inclinação da reta entre roteamento e tipo de serviço mostra que a tendência da política Pastry é oferecer melhores tempos de resposta para aplicações do tipo pesada enquanto que a política Chord possui um desempenho melhor para aplicações de carga leve.

6.6 Considerações Finais

As políticas de roteamento consideradas neste trabalho dependem de uma métrica para avaliação do caminho a ser considerado entre o cliente e o *data center*. O primeiro experimento realizado avaliou qual a métrica mais adequada para utilização nos algoritmos.

Foram consideradas nessa avaliação a latência da rede e o número de saltos entre os pontos da rede. Desse experimento, pode-se concluir que para o algoritmo BCR (Baseada na Capacidade da Rede) a utilização da latência como métrica originou tempos de resposta menores e

uma variabilidade menor entre os diferentes testes realizados diferente do obtido com a métrica saltos.

Para os algoritmos Chord e Pastry observa-se uma pequena influência quando as diferentes métricas são consideradas. Estatisticamente, essas diferenças não são suficientes para concluir-se qual a melhor métrica. No entanto, nos testes realizados os tempos de resposta médios para a métrica latência sempre foram menores que os tempos para a métrica número de saltos.

Um ponto que deve ser ressaltado na análise dos resultados obtidos é que a sobrecarga gerada pelos algoritmos de roteamento é insignificante quando comparadas com os tempos envolvidos para a comunicação pela rede. Os tempos de resposta obtidos nos experimentos são da ordem de segundos. Os tempos obtidos para a determinação do caminho a ser considerado é da ordem de milissegundos, quando executado na mesma máquina utilizada para as simulações (ver Tabela 6.1). Mesmo considerando o algoritmo BCR que é o mais oneroso em termos de processamento, o tempo obtido foi, em média, 128,6mseg. Esse tempo comparado com os tempos para comunicação (em torno de 15 segundos) não é significativo.

A seleção da métrica latência auxiliou nas análises dos dados obtidos, sendo que, as políticas de roteamento foram avaliadas de acordo com as características que garantiam melhor desempenho.

Neste capítulo foram apresentados os resultados visando a avaliação de diferentes políticas de roteamento. Para isso foram avaliadas políticas que podem ser utilizadas em computação em nuvem considerando ou não informações da rede. A política *Round Robin* não utiliza informações da rede e as políticas BCR, Pastry e Chord utilizam informações da rede.

Comparando a política *Round Robin* com uma política que utiliza força bruta baseada nas informações da rede (BCR), observa-se que os tempos de resposta para a primeira política são superiores. Quando um pequeno número de usuários (30) é considerado observa-se que a política *Round Robin* apresenta tempos de resposta em torno de 26% maiores que os obtidos com a política BCR.

Quando o número de usuários aumenta, essa influência cresce e observa-se que os tempos na *Round Robin* são em torno de 38% maiores. Além de maiores tempos de resposta, a *Round Robin* apresenta ainda uma maior variabilidade entre os diferentes experimentos realizados.

As políticas baseadas em sistemas P2P (Chord e Pastry) foram avaliadas em relação à política BCR para avaliar qual a diferença entre usar uma política que possui menos passos para a tomada de decisão em relação a uma política que sempre analisa as informações de todos os participantes para a tomada de decisão.

Em relação à comparação entre as políticas BCR e Chord, observa-se que a BCR apresenta tempos de resposta menores mas bastante próximos dos obtidos com a Chord. Nesse caso, para 30 usuários os tempos da Chord são em torno de 3% maiores e para 60 usuários esses tempos são aproximadamente 7% maiores. A variabilidade entre os resultados nos diferentes testes foi pequena para as duas políticas embora a Chord tenha apresentado uma variação ligeiramente

maior. Observa-se nesse caso, que o fator que apresenta maior influência nos resultados é o número de clientes, devido principalmente a pequena diferença observada entre as duas políticas.

Quando é realizada a comparação entre a Pastry e a BCR observa-se uma influência mais uniforme no tempo de resposta, independente do número de usuários. Em todos os casos tem-se que a política Pastry resultou em tempos de resposta aproximadamente 6% maiores do que os obtidos na BCR. Nesse experimento não se observa interação entre os fatores considerados. A variabilidade dos resultados na Pastry é ligeiramente maior que a observada na BCR.

Por fim foi analisada qual a diferença entre as duas políticas que utilizam sistemas P2P (Chord e Pastry). Os resultados mostram que não existe uma diferença considerável proporcionada aos tempos de resposta. O domínio que cada política irá abordar é que faz a diferença, ou seja, na política Chord os serviços do tipo leve são entregues de melhor forma que a política Pastry (Pastry em torno de 6% pior). Em contra partida, a política Pastry possui melhores tempos de resposta para serviços do tipo pesado (Chord em torno de 2% pior).

Conclusões

7.1 Considerações Iniciais

A redução dos custos com Internet e as formas de acessá-la tendem a impulsionar a computação em nuvem, fazendo com que os clientes comecem a utilizar apenas serviços online e possibilitando também que empresas tenham suas infraestruturas computacionais na nuvem. Como os provedores de nuvem vêm crescendo e oferecendo cada vez mais facilidades para que os usuários migrem seus serviços para a nuvem, é importante observar qual a forma de comunicação que será empregada entre os provedores e os clientes com a finalidade de oferecer bons níveis de QoS.

Como a computação em nuvem agrega muitas tecnologias e conceitos já consolidados, como virtualização e sistemas operacionais de rede, melhorias utilizando conceitos relacionados a esses sistemas podem favorecer o fornecimento de serviços com um nível maior de qualidade.

Um dos conceitos abordados é a utilização de metaescalonadores, que a princípio eram utilizados na computação em grade para desenvolver o escalonamento e meta-escalonamento de serviços. Como os conceitos sobre metaescalonadores podem ser utilizados na computação em nuvem, Peixoto et al. (2010) propôs um metaescalonador com suporte ao oferecimento de qualidade de serviço.

Pelo fato da comunicação dentro das atividades da computação em nuvem poder se tornar um ponto de falhas, o metaescalonador proposto por Peixoto et al. (2010) utiliza um modelo hierárquico P2P que além de oferecer melhores tempos de resposta aos clientes também visa oferecer escalabilidade na nuvem, uma vez que, criando-se uma topologia entre os *data centers*

é possível fazer uma organização de acordo com a latência obtida, fazendo com que os clientes sejam servidos pelos *data centers* com melhores disponibilidades e menores latências.

Os resultados obtidos demonstram que as políticas tradicionalmente utilizadas em sistemas P2P (Chord e Pastry) apresentam tempos próximos à política BCR. Os tempos de resposta obtidos para o Chord e para o Pastry são, no máximo, 8% maiores que os obtidos na política BCR. O problema a ser considerado no caso do BCR é a sobrecarga imposta na tomada de decisões. Como mostrado no capítulo 6, para os experimentos realizados neste trabalho essa sobrecarga não é significativa, uma vez que a sobrecarga é da ordem de milissegundo enquanto os tempos envolvidos nas comunicações são da ordem de segundos.

No entanto, o problema observado em algoritmos que consideram busca exaustiva está relacionado a escalabilidade. Se o número de *data centers* aumentar consideravelmente pode-se vir a ter uma sobrecarga considerável. Para esse caso, os resultados apresentados neste trabalho mostram que as políticas Chord e Pastry podem ser consideradas e o aumento nos tempos de resposta é aceitável.

Com relação a escolha entre as políticas Pastry e Chord tem-se que para cargas leves a política Chord é mais adequada e para cargas pesadas a política Pastry apresenta melhores resultados. Desta forma, a escolha entre essas políticas depende do tipo de carga considerada.

Concluindo, os resultados obtidos pelas políticas que utilizam sistemas P2P tradicionais demonstra que com uma quantidade de iterações menor é possível proporcionar qualidade de serviço similar a da política de busca exaustiva. Com relação a qual das políticas é mais adequada para o MACC, deve-se observar que como o modelo de comunicação do MACC pode adotar tipos diferentes de políticas de roteamento, o mais adequado seria considerar as três políticas que utilizam informação da rede avaliada neste trabalho.

7.2 Contribuições

O projeto de mestrado apresenta contribuições relacionadas à computação em nuvem, metaescaladores e sistemas P2P. Dentre essas contribuições destacam-se:

- Uma revisão bibliográfica sobre computação em nuvem que permite avaliar quais suas funcionalidades, facilidades e passos futuros;
- Uma revisão sobre o estado da arte tanto com relação à metaescaladores quanto com relação a sistemas P2P. Essa revisão visa auxiliar a compreensão de como esses sistemas podem interagir para a obtenção de qualidade de serviço;
- O levantamento das diferenças entre os sistemas P2P utilizados. Como foram utilizadas duas cargas de trabalho, uma leve (carga de trabalho baseada em serviços Web) e uma

carga pesada (aplicações científicas). O desempenho da política Chord se mostrou eficiente para o primeiro cenário e a política Pastry mostrou-se eficiente para aplicações pesadas;

- Auxílio na formação de SLA, como foi realizada uma avaliação das políticas de roteamento, as mesmas podem ser utilizadas, dependendo do domínio da aplicação, ou seja, o MACC pode fornecer melhores contratos (SLAs) dependendo da carga utilizada e pode utilizar uma variação de políticas dependendo do estado da aplicação do cliente.

Além dos pontos citados acima, uma publicação referente ao metaescalador e aos modelos de comunicação apresentados neste documento são apresentados em:

PEIXOTO, M. L. M.; LEITE, D. M.; SANTANA, M. J.; SANTANA, R. H. C.. Gerenciamento de Máquinas Virtuais em uma Cloud Multimídia por meio do Metaescalamento. Este trabalho foi apresentado no Simpósio Brasileiro em Sistemas Multimídia (WebMedia) no ano de 2011.

Além da publicação, um artigo originado do trabalho foi enviado para a conferência: IEEE 2012 CLOUD Research.

LEITE, D. M.; PEIXOTO, M. L. M.; SANTANA, M. J.; SANTANA, R. H. C.. Evaluation of Routing Policies Performance to provide QoS in Cloud Computing. Esse artigo foi submetido em Março de 2012.

7.3 Dificuldades Relacionadas ao Projeto

Por se tratar de um trabalho que utilizou simulação, algumas dificuldades relacionadas a essa metodologia foram encontrados. Dentre elas estão:

- Pouca documentação do simulador; Por se tratar de um simulador em constante desenvolvimento, alguns erros de classes foram encontrados e a ausência de material de suporte causou problemas iniciais;
- Ausência de dinamicidade da rede; após uma quantidade razoável de testes foi verificado que a rede não possuía uma dinamicidade, ou seja, independente da semente da simulação os resultados eram sempre os mesmos. Uma solução foi verificar como era feita a distribuição pelo BRITE e a partir dessa distribuição atribuir pesos as arestas baseado nas sementes da simulação.

7.4 Trabalhos Futuros

Como trabalhos futuros, podem ser levantados os seguintes pontos:

- Avaliação de desempenho do modelo de comunicação do MACC utilizando técnicas como CDN em conjunto com os sistemas P2P;
- Desenvolvimento e testes da infraestrutura de comunicação utilizando servidores reais. Dessa forma, pontos como segurança em DHT podem ser avaliados e problemas como frequente entrada e saída de nodos da rede e como escalonar esses eventos;
- Estudo da influência das políticas de roteamento na elaboração e cumprimento de SLAs;
- Estudo baseado no tempo gasto por cada política de roteamento. Nesse estudo pode ser abordado o tempo gasto pelo metaescalonador para formar as redes de sobreposição e qual a sobrecarga gerada na criação e monitoramento dessas redes e a partir dos resultados sugerir novas políticas que não possuam uma sobrecarga grande.
- Utilização das políticas avaliadas neste trabalho em uma arquitetura orientada a serviços. Uma arquitetura que pode ser utilizada é a WSARCH (Estrella et al., 2011);

Referências Bibliográficas

- ALBERT, R.; BARABÁSI, A. Topology of evolving networks: local events and universality. *Physical review letters*, v. 85, n. 24, p. 5234–5237, 2000.
- AMAZON Amazon elb. [Http://aws.amazon.com/pt/elasticloadbalancing//188-4029135-7099348/](http://aws.amazon.com/pt/elasticloadbalancing//188-4029135-7099348/), 2012.
- AN, B.; LESSER, V.; IRWIN, D.; ZINK, M. Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2010, p. 981–988 (AAMAS '10, v.1).
Disponível em: <http://dl.acm.org/citation.cfm?id=1838206.1838338>
- ANDRADE, N.; CIRNE, W.; BRASILEIRO, F.; ROISENBERG, P. Ourgrid: An approach to easily assemble grids with equitable resource sharing. In: *Job scheduling strategies for parallel processing*, Springer, 2003, p. 61–86.
- ANDROUTSELLIS-THEOTOKIS, S.; SPINELLIS, D. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, v. 36, p. 335–371, 2004.
- ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R. H.; KONWINSKI, A.; LEE, G.; PATTERSON, D. A.; RABKIN, A.; STOICA, I.; ZAHARIA, M. *Above the clouds: A berkeley view of cloud computing*. Relatório Técnico UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.
Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>
- BELL, W.; CAMERON, D.; MILLAR, A.; CAPOZZA, L.; STOCKINGER, K.; ZINI, F. Optor-sim: A grid simulator for studying dynamic data replication strategies. *International Journal of High Performance Computing Applications*, v. 17, n. 4, p. 403–416, 2003.

- BELOGLAZOV, A.; BUYYA, R. Energy efficient allocation of virtual machines in cloud data centers. In: *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, Ieee, 2010, p. 577–578.
- BERNERS-LEE, T. Weaving the web: The original design and ultimate destiny of the world wide web, by its inventor. 1999. *L'architettura del nuovo Web. Dall'inventore della rete il progetto di una comunicazione democratica, interattiva e intercreativa*, 2001.
- BIENKOWSKI, M.; KORZENIOWSKI, M.; HEIDE, F. Dynamic load balancing in distributed hash tables. *Peer-to-Peer Systems IV*, p. 217–225, 2005.
- BUYYA, R.; RANJAN, R.; CALHEIROS, R. N. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010, Busan, South Korea, May 21-23), LNCS, Springer, Germany, 2010.*, v. abs/1003.3920, 2010.
- CAI, M.; FRANK, M.; CHEN, J.; SZEKELY, P. Maan: A multi-attribute addressable network for grid information services. *Journal of Grid Computing*, v. 2, n. 1, p. 3–14, 2004.
- CALHEIROS, R.; RANJAN, R.; BUYYA, R. Virtual machine provisioning based on analytical performance and qos in cloud computing environments. In: *Parallel Processing (ICPP), 2011 International Conference on*, IEEE, 2011, p. 295–304.
- CALHEIROS, R. N.; RANJAN, R.; BELOGLAZOV, A.; ROSE, C. A. F. D.; BUYYA, R. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. In: *Software: Practice and Experience*, Wiley Press, 2010.
- CARISSIMI, A. Virtualizacao: da teoria a solucoes. In: *26 Brazilian Symposium on Computer Networks and Distributed Systems, SBRC 2008, pag. 173-207. May 26th-30th, 2008, Rio de Janeiro, Brazil*, 2008.
- CAROLAN, J.; GAEDE, S.; BATY, J.; BRUNETTE, G.; LICHT, A.; REMMELL, J.; TUCKER, L.; WEISE, J. Sun microsystems - introduction to cloud computing architecture. In: *White paper 1st edition, pp.9-12, june 2009. Retrieved in December 07, 2009, from: <http://www.sun.com/featured-articles/CloudComputing.pdf>*, 2009.
- CASANOVA, H.; LEGRAND, A.; QUINSON, M. Simgrid: a generic framework for large-scale distributed experiments. In: *Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on*, IEEE, 2008, p. 126–131.

- CHARD, K. *Drive: A distributed economic meta-scheduler for the federation of grid and cloud systems*. Wellington, Nova Zelândia: Tese de Doutorado - Victoria University of Wellington, 2011.
- CHEEMA, A.; MUHAMMAD, M.; GUPTA, I. Peer-to-peer discovery of computational resources for grid applications. In: *Grid Computing, 2005. The 6th IEEE/ACM International Workshop on*, IEEE, 2005, p. 7–pp.
- CHESWICK, B.; BURCH, H.; BRANIGAN, S. Mapping and visualizing the internet. In: *In Proceedings of the 2000 USENIX Annual Technical Conference*, 2000, p. 1–12.
- CHIEU, T. C.; MOHINDRA, A.; KARVE, A. A.; SEGAL, A. Dynamic scaling of web applications in a virtualized cloud computing environment. In: *ICEBE '09: Proceedings of the 2009 IEEE International Conference on e-Business Engineering*, Washington, DC, USA: IEEE Computer Society, 2009, p. 281–286.
- CHRISTODOULOPOULOS, K.; SOURLAS, V.; MPAKOLAS, I.; VARVARIGOS, E. A comparison of centralized and distributed meta-scheduling architectures for computation and communication tasks in grid networks. *Comput. Commun.*, v. 32, p. 1172–1184, 2009.
Disponível em: <http://dl.acm.org/citation.cfm?id=1542555.1542766>
- COULOURIS, G. F.; DOLLIMORE, J. *Distributed systems: concepts and design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1988.
- CRAWLEY, E.; NAIR, R.; RAJAGOPALAN, B.; SANDICK, H. A framework for qos-based routing in the internet. *RFC2386*, v. 37, 1998.
- DODGE, M. An atlas of cyberspaces. [Www.personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/isp_maps.html](http://www.personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/isp_maps.html). Acesso: Janeiro/2012, 2012.
- ESTRELLA, J. C.; SANTANA, M. J.; SANTANA, R. H. C. *WSARCH: An Architecture for Web Services Provisioning with QoS Support: Performance Challenges*. VDM Verlag Dr. Müller, 140 p., 2011.
- FORTIER, P.; MICHEL, H. *Computer systems performance evaluation and prediction*. Digital Pr, 544 p., 2003.
- FOSTER, I.; KESSELMAN, C. Globus: A metacomputing infrastructure toolkit. *International Journal of High Performance Computing Applications*, v. 11, n. 2, p. 115–128, 1997.
- FOSTER, I.; KESSELMAN, C. Globus: A toolkit-based grid architecture. *Morgan Kaufmann*, p. 259–278, 1999.

- FOSTER, I.; KESSELMAN, C.; LEE, C.; LINDELL, B.; NAHRSTEDT, K.; ROY, A. A distributed resource management architecture that supports advance reservations and co-allocation. *Quality of Service, 1999. IWQoS '99. 1999 Seventh International Workshop on*, p. 27–36, 1999.
- FURHT, B. Cloud computing fundamentals. In: FURHT, B.; ESCALANTE, A., eds. *Handbook of Cloud Computing*, cap. 1, Boston, MA: Springer US, p. 3–19, 2010.
Disponível em: http://dx.doi.org/10.1007/978-1-4419-6524-0_1
- GANESAN, P.; YANG, B.; GARCIA-MOLINA, H. One torus to rule them all: multi-dimensional queries in p2p systems. In: *Proceedings of the 7th International Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004*, ACM, 2004, p. 19–24.
- GMACH, D.; ROLIA, J.; CHERKASOVA, L. Satisfying service level objectices in a self-managing resource pool. In: *Proceedings of the 2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, Washington, DC, USA: IEEE Computer Society, 2009, p. 243–253 (SASO '09, v.1).
Disponível em: <http://dx.doi.org/10.1109/SASO.2009.27>
- GRAFFI, K.; STINGL, D.; GROSS, C.; NGUYEN, H.; KOVACEVIC, A.; STEINMETZ, R. *Towards a P2P Cloud: Reliable Resource Reservations in Unreliable P2P Systems*. IEEE, 27–34 p., 2010.
Disponível em: <http://www.computer.org/portal/web/csd1/doi/10.1109/ICPADS.2010.34>
- GUO, L.; MCGOUGH, A. S.; AKRAM, A.; COLLING, D.; MARTYNIK, J.; KRZARNIC, M. Enabling qos for service-oriented workflow on grid. In: *CIT '07: Proceedings of the 7th IEEE International Conference on Computer and Information Technology*, Washington, DC, USA: IEEE Computer Society, 2007, p. 1077–1082.
- GUPTA, A. Cloud computing growing interest and related concerns. In: *Computer Technology and Development (ICCTD), 2010 2nd International Conference on*, 2010, p. 462–465.
- HEIDT, M.; DORNEMANN, T.; DORNEMANN, K.; FREISLEBEN, B. Omnivore: Integration of grid meta-scheduling and peer-to-peer technologies. In: *Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on*, 2008, p. 316–323.
- HUANG, Y.; BESSIS, N.; NORRINGTON, P.; KUONEN, P.; HIRSBRUNNER, B. Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm. 2011.

- IOSUP, A.; EPEMA, D. H. J.; TANNENBAUM, T.; FARRELLEE, M.; LIVNY, M. Inter-operating grids through delegated matchmaking. In: *Proceedings of the 2007 ACM/IEEE conference on Supercomputing, SC '07*, New York, NY, USA: ACM, 2007, p. 13:1–13:12 (SC '07, v.).
Disponível em: <http://doi.acm.org/10.1145/1362622.1362640>
- IOSUP, A.; SONMEZ, O.; ANOEP, S.; EPEMA, D. The performance of bags-of-tasks in large-scale distributed systems. In: *Proceedings of the 17th international symposium on High performance distributed computing*, ACM, 2008, p. 97–108.
- JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley, 685 p., 1991.
- JEYARANI, R.; RAM, R. V.; NAGAVENI, N. Design and implementation of an efficient two-level scheduler for cloud computing environment. In: *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID '10*, IEEE, Washington, DC, USA: IEEE Computer Society, 2010, p. 585–586 (CCGRID '10, v.).
- JIN, H.; TAO, Y.; WU, S.; SHI, X. Scalable dht-based information service for large-scale grids. In: *CF 08: Proceedings of the 5th conference on Computing frontiers*, New York, NY, USA: ACM, 2008, p. 305–312.
- JONES, M. T. *Computação em nuvem com linux*. 2008.
Disponível em: <http://www.ibm.com/developerworks/br/library/l-cloud-computing/>
- KIM, K.; BELOGLAZOV, A.; BUYYA, R. Power-aware provisioning of cloud resources for real-time services. In: *Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science*, ACM, 2009a, p. 1.
- KIM, W.; KIM, S.; LEE, E.; LEE, S. Adoption issues for cloud computing. In: *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, ACM, 2009b, p. 3–6.
- KUROSE, J.; ROSS, K.; MARQUES, A.; ZUCCHI, W. *Redes de computadores e a internet: uma abordagem top-down*. 5 ed. Pearson Addison Wesley, 61–139 p., 2010.
- LAI, K.-C.; HUANG, K. C.; KOONG, C. S. C.-S.; YU, Y. F.; HUANG, P. J.; CHEN, Q. J.; HUNANG, T. L. A P2P Resource Discovery Strategy for Cloud Computing Systems. *Journal of Computers*, v. 21, n. 1, p. 96–97, 2010.
- LEAL, K.; HUEDO, E.; LLORENTE, I. M. A decentralized model for scheduling independent tasks in federated grids. *Future Gener. Comput. Syst.*, v. 25, p. 840–852, 2009.
Disponível em: <http://dl.acm.org/citation.cfm?id=1550955.1551001>

- LEAVITT, N. Is Cloud Computing Really Ready for Prime Time? *Computer*, v. 42, n. 1, p. 15–20, 2009.
- LI, L. An optimistic differentiated service job scheduling system for cloud computing service users and providers. In: *Proceedings of the 2009 Third International Conference on Multimedia and Ubiquitous Engineering*, Washington, DC, USA: IEEE Computer Society, 2009, p. 295–299.
- LI, Q.; HAO, Q.; XIAO, L.; LI, Z. Adaptive management of virtualized resources in cloud computing using feedback control. In: *Proceedings of the 2009 First IEEE International Conference on Information Science and Engineering*, ICISE '09, Washington, DC, USA: IEEE Computer Society, 2009, p. 99–102 (ICISE '09, v.).
- LINNOLAHTI, J. Qos routing for p2p networking. In: *HUT T-110.551 Seminar on Inter-networking*, 2004.
Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.7192&rep=rep1&type=pdf>
- LIU, Y.; MASOUD SADJADI, S.; FONG, L.; RODERO, I.; VILLEGAS, D.; KALAYCI, S.; BOBROFF, N.; MARTINEZ, J. Enabling autonomic meta-scheduling in grid environments. In: *Autonomic Computing, 2008. ICAC '08. International Conference on*, 2008, p. 199–200.
- LUA, E.; CROWCROFT, J.; PIAS, M.; SHARMA, R.; LIM, S. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, v. 7, n. 2, p. 72–93, 2005.
- MANN, V. Metascheduling: an elixir for performance-tuning of compound sessions based on the cm-resource model. In: *Advanced Communication Technology, 2005, ICACT 2005. The 7th International Conference on*, 2005, p. 661–664.
- MASSIE, M.; CHUN, B.; CULLER, D. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, v. 30, n. 7, p. 817–840, 2004.
- MEDINA, A.; LAKHINA, A.; MATTA, I.; BYERS, J. Brite: An approach to universal topology generation. In: *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*, IEEE, 2001, p. 346–353.
- MELL, P.; GRANCE, T. *The NIST Definition of Cloud Computing*. Relatório Técnico, National Institute of Standards and Technology, Information Technology Laboratory, 2011.
Disponível em: <http://www.csrc.nist.gov/groups/SNS/cloud-computing/>
- MICROSOFT Microsoft windows azure. [Http://www.windowsazure.com/pt-br/home](http://www.windowsazure.com/pt-br/home), 2012.

- MILOJICIC, D.; KALOGERAKI, V.; LUKOSE, R.; NAGARAJA, K.; PRUYNE, J.; RICHARD, B.; ROLLINS, S.; XU, Z. Peer-to-peer computing. 2002.
- NALDI, M. Connectivity of waxman topology models. *Computer communications*, v. 29, n. 1, p. 24–31, 2005.
- OHLMAN, B.; ERIKSSON, A.; REMBARZ, R. What networking of information can do for cloud computing. *Enabling Technologies, IEEE International Workshops on*, v. 0, p. 78–83, 2009.
- OSTERMANN, S.; PLANKENSTEINER, K.; PRODAN, R. Using a new event-based simulation framework for investigating resource provisioning in clouds. *Scientific Programming*, v. 19, n. 2, p. 161–178, 2011.
- PEIXOTO, M. L. M.; SANTANA, M. J.; ESTRELLA, J. C.; TAVARES, T. C.; KUEHNE, B. T.; SANTANA, R. H. C. A Metascheduler architecture to provide QoS on the cloud computing. In: *ICT '10: 17th International Conference on Telecommunications*, Doha, Qatar: IEEE Computer Society, 2010, p. 650–657.
- PEIXOTO, M. L. M.; SANTANA, M. J.; SANTANA, R. H. C. A p2p hierarchical metascheduler to obtain qos in a grid economy services. In: *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 01*, Washington, DC, USA: IEEE Computer Society, 2009, p. 292–297 (CSE '09, v.1).
Disponível em: <http://dx.doi.org/10.1109/CSE.2009.385>
- PENG, J.; ZHANG, X.; LEI, Z.; ZHANG, B.; ZHANG, W.; LI, Q. Comparison of several cloud computing platforms. In: *ISISE '09: Proceedings of the 2009 Second International Symposium on Information Science and Engineering*, Washington, DC, USA: IEEE Computer Society, 2009, p. 23–27.
- POKHAREL, M.; PARK, J. S. Cloud computing: future solution for e-governance. In: *ICE-GOV '09: Proceedings of the 3rd International Conference on Theory and Practice of Electronic Governance*, New York, NY, USA: ACM, 2009, p. 409–410.
- RANJAN, R.; HARWOOD, A.; BUYYA, R. A study on peer-to-peer based discovery of grid resource information. *University of Melbourne, Australia, Technical Report GRIDS-TR-2006-17*, 2006.
- RANJAN, R.; HARWOOD, A.; BUYYA, R. Peer-to-peer-based resource discovery in global grids: a tutorial. *Communications Surveys & Tutorials, IEEE*, v. 10, n. 2, p. 6–33, 2008.
- RANJAN, R.; ZHAO, L.; WU, X.; LIU, A.; QUIROZ, A.; PARASHAR, M. Peer-to-peer cloud provisioning: Service discovery and load-balancing. *Cloud Computing*, p. 195–217, 2010.

- RATNASAMY, S.; STOICA, I.; SHENKER, S. Routing algorithms for dhts: Some open questions. In: *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, London, UK: Springer-Verlag, 2002, p. 45–52.
- RIMAL, B. P.; CHOI, E.; LUMB, I. A taxonomy and survey of cloud computing systems. In: *NCM '09: Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, IEEE, Washington, DC, USA: IEEE Computer Society, 2009, p. 44–51.
- ROWSTRON, A. I. T.; DRUSCHEL, P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, London, UK: Springer-Verlag, 2001, p. 329–350.
- SCHMIDT, C.; PARASHAR, M. Peer-to-peer information storage and discovery systems. *Peer-to-Peer Computing*, p. 79, 2005.
- SCHWIEGELSHOHN, U.; YAHYAPOUR, R. Resource allocation and scheduling in metasystems. In: *Proceedings of the 7th International Conference on High-Performance Computing and Networking*, HPCN Europe '99, London, UK: Springer-Verlag, 1999, p. 851–860 (*HPCN Europe '99*, v.).
Disponível em: <http://dl.acm.org/citation.cfm?id=645563.660323>
- SHI, Y.; JIANG, X.; YE, K. An energy-efficient scheme for cloud resource provisioning based on cloudsims. In: *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, IEEE, 2011, p. 595–599.
- SINDHU, S.; MUKHERJEE, S. Efficient task scheduling algorithms for cloud computing environment. *High Performance Architecture and Grid Computing*, p. 79–83, 2011.
- SPENCE, D.; HARRIS, T. Xenosearch: Distributed resource discovery in the xenoserver open platform. In: *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on*, IEEE, 2003, p. 216–225.
- STOICA, I.; MORRIS, R.; KARGER, D.; KAASHOEK, M. F.; BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for internet applications. In: *SIGCOMM 01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA: ACM, 2001, p. 149–160.
- SUBRAMANI, V.; KETTIMUTHU, R.; SRINIVASAN, S.; SADAYAPPAN, P. Distributed job scheduling on computational grids using multiple simultaneous requests. In: *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, Washington, DC, USA: IEEE Computer Society, 2002, p. 359.

- TAM, D.; AZIMI, R.; JACOBSEN, H. Building content-based publish/subscribe systems with distributed hash tables. *Databases, Information Systems, and Peer-to-Peer Computing*, p. 138–152, 2004.
- TANENBAUM, A. S. redes de computadores. são paulo: Ed. 2003.
- TRIANTAFILLOU, P.; AEKATERINIDIS, I. Content-based publish-subscribe over structured p2p networks. In: *Proceedings of the third international workshop on distributed event-based systems (DEBS)*, Citeseer, 2004, p. 104–109.
- VASILIOU, N. Overview of internet qos and web server qos. 2000, p. 1–37.
Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.6544>
- VECCHIOLA, C.; CHU, X.; BUYYA, R. Aneka: a software platform for .net-based cloud computing. *High Performance & Large Scale Computing, Advances in Parallel Computing. IOS Press, Amsterdam*, 2009.
- VERNEKAR, S.; GAME, P. Component based resource allocation in cloud computing. In: *Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam, India, January 2012*, Springer, 2012, p. 907–914.
- WAXMAN, B. Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on*, v. 6, n. 9, p. 1617–1622, 1988.
- WEISSMAN, J.; GRIMSHAW, A. A federated model for scheduling in wide-area systems. In: *High Performance Distributed Computing, 1996., Proceedings of 5th IEEE International Symposium on*, 1996, p. 542–550.
- WU, L.; CHEN, Z.; SU, J.; LUO, X. The based-role pmi model for access control in large scale netware system. In: *Computer Design and Applications (ICCD), 2010 International Conference on*, 2010, p. V2–81–V2–84.
- XHAFA, F.; ABRAHAM, A. Computational models and heuristic methods for grid scheduling problems. *Future Gener. Comput. Syst.*, v. 26, p. 608–621, 2010.
Disponível em: <http://dx.doi.org/10.1016/j.future.2009.11.005>
- YAMINI, L.; LATHASELVI, G.; MUKHERJEE, S. Efficient metascheduling in a cloud extended grid environment. In: *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, 2011, p. 667–672.
- YONG MENG, T.; MARCH, V.; WANG, X. A dht-based grid resource indexing and discovery scheme. In: *Singapore-MIT Alliance Symposium 2005*, Citeseer, 2005.
Disponível em: <http://hdl.handle.net/1721.1/7415>

- YOU, X.; XU, X.; WAN, J.; JIANG, C. Analysis and evaluation of the scheduling algorithms in virtual environment. In: *Embedded Software and Systems, 2009. ICESS '09. International Conference on*, 2009, p. 291–296.
- ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, v. 1, n. 1, p. 7–18, 2010.
- ZHOU, J.; ABDULLAH, N.; SHI, Z. A hybrid p2p approach to service discovery in the cloud. *International Journal of Information Technology and Computer Science (IJITCS)*, v. 3, n. 1, p. 1 – 9, 2011.
- ZHU, X.; YOUNG, D.; WATSON, B.; WANG, Z.; ROLIA, J.; SINGHAL, S.; MCKEE, B.; HYSER, C.; GMACH, D.; GARDNER, R.; CHRISTIAN, T.; CHERKASOVA, L. 1000 islands: Integrated capacity and workload management for the next generation data center. In: *Autonomic Computing, 2008. ICAC '08. International Conference on*, 2008, p. 172–181.