MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

MATHEUS BOUSQUET BANDINI

QUALIDADE DE SERVIÇO EM GRADES COMPUTACIONAIS UTILIZANDO ACORDOS EM NÍVEL DE SERVIÇO

Rio de Janeiro
2009

INSTITUTO MILITAR DE ENGENHARIA

MATHEUS BOUSQUET BANDINI

QUALIDADE DE SERVIÇO EM GRADES COMPUTACIONAIS UTILIZANDO ACORDOS EM NÍVEL DE SERVIÇO

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientadores:
Prof. Bruno Richard Schulze - D.Sc
Prof. Ronaldo Moreira Salles - Ph.D.

Rio de Janeiro
2009

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

B214 Bandini, M.
Qualidade de Serviço em Grades Computacionais utilizando Acordos em Nível de Serviço/ Matheus Bousquet Bandini.
– Rio de Janeiro: Instituto Militar de Engenharia, 2009.
81 p.: il., tab.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2009.

1. Tecnologias para Tratamento e Transmissão da Informação. 2. Tecnologias e Sistemas de Computação. I. Qualidade de Serviço em Grades Computacionais utilizando Acordos em Nível de Serviço. II. Instituto Militar de Engenharia.

CDD 621.319

INSTITUTO MILITAR DE ENGENHARIA

MATHEUS BOUSQUET BANDINI

QUALIDADE DE SERVIÇO EM GRADES COMPUTACIONAIS UTILIZANDO ACORDOS EM NÍVEL DE SERVIÇO

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientadores: Prof. Bruno Richard Schulze - D.Sc
Prof. Ronaldo Moreira Salles - Ph.D.

Aprovada em 6 de fevereiro de 2009 pela seguinte Banca Examinadora:

Prof. Ronaldo Moreira Salles - Ph.D. do IME - Presidente

Prof. Bruno Richard Schulze - D.Sc do LNCC

Prof. Antonio Roberto Mury - D.Sc do LNCC

Prof. Jose Neuman de Souza - D.Sc da UFC

Prof. Raquel Coelho Gomes Pinto - D.Sc do IME

Rio de Janeiro
2009

Dedico este trabalho à minha mãe Vera, cujo apoio incondicional nos momentos mais difíceis fizeram com que eu tivesse a perseverança para chegar até aqui.

AGRADECIMENTOS

Agradeço a Deus por me guiar no caminho do bem e me conceder o êxito nesta dura caminhada.

Agradeço à minha mãe Vera e meu irmão Thiago, por formar meu caráter e me incentivar incondicionalmente e por nunca me deixar desistir.

Agradeço à minha namorada Renata por compreender meus momentos de ausência e por me apoiar nos momentos de dificuldade.

Agradeço aos professores Bruno Schulze e Ronaldo Salles por aceitarem o desafio de me orientar neste trabalho e por colaborar na sua realização.

Agradeço aos amigos do LNCC: Antônio Roberto Mury, Fábio Licht, Luís Rodrigo Gonçalves, Thais Mello, Henrique Klôh, Carolina Areas e Bruno Barcellos pela prontidão em ajudar e pelo apoio moral.

Agradeço ainda, a todas as pessoas que conviveram comigo neste período e que, de alguma forma, me ajudaram ou me incentivaram.

Por fim, a todos os colegas e professores do curso de Mestrado em Sistemas e Computação, assim como aos funcionários do Departamento de Engenharia de Sistemas (SE/8) do Instituto Militar de Engenharia.

Matheus Bousquet Bandini

"*Corte sua própria lenha. Assim, ela aquecerá você duas vezes.*"

HENRY FORD

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABREVIATURAS

AC	-	*Autoridade Certificadora*
AD	-	*Administrative Domains*
ADC	-	*Administrative Domain Controller*
AI	-	*Active Interfaces*
AppLeS	-	*Application-Level Scheduling*
CPU	-	*Central Processing Unit*
DFS	-	*Distributed File System*
DiffServ	-	*Differentiated Services*
DNS	-	*Domain Name Service*
FTP	-	*File Transfer Protocol*
GAAS	-	*Global Access to Secondary Storage*
GARA	-	*General-purpose Architecture for Reservation and Allocation*
GRAM	-	*Grid Resource Allocation Manager*
Grid–QoS	-	*Grid Quality of Service*
GSI	-	*Grid Security Infrastructure*
G-QoSm	-	*Grid QoS Management*
HPC	-	*High Performance Computing*
IntServ	-	*Integrated Services*
IP	-	*Internet Protocol*
ISO	-	*International Organization for Standardization*
JVM	-	*Java Virtual Machine*
LAN	-	*Local Area Network*
MDS	-	*Monitoring and Discovery Service*
MPI	-	*Message Passing Interface*
MPLS	-	*Multi Protocol Labeling Switching*
NWS	-	*Network Weather Service*
OGSA	-	*Open Grid service Architecture*
OSI	-	*Open Systems Interconnection*
OV	-	*Organizações Virtuais*

PHP	-	*Personal Home Page*
PKI	-	*Public Key Infrastructure*
QGS	-	*QoS Grid Service*
QoS	-	*Quality os Service*
RMI	-	*Remote Method Invocation*
RMS	-	*Resource Management Systems*
RNP	-	*Rede Nacional de Ensino e Pesquisa*
RPC	-	*Remote Procedure Call*
RSL	-	*Resource Specification Language*
RSVP	-	*Resource reSerVation Protocol*
SBM	-	*Subnet Bandwidth Management*
SDD	-	*Servidor de Dados de Domínio*
SDL	-	*Servidor de Dados Locais*
SLA	-	*Service Level Agreements*
SSH	-	*Secure Shell*
TCP	-	*Transfer Control Protocol*
VCG	-	*Virtual Community Grid*
VMS	-	*VCG Management System*
WAN	-	*Wide Area Network*
WSRF	-	*Web Service Resource Framework*

RESUMO

Este trabalho apresenta um sistema que aplica um modelo de fornecimento de Qualidade de Serviço em Grades Computacionais (*Grid–QoS*) utilizando Acordos em Nível de Serviço (SLAs). Além disso, o sistema desenvolvido faz uso de uma interface *web* intuitiva que facilita a interação entre os usuários e o Sistema. Ao mesmo tempo, é proposto um novo processo de cobrança pelos serviços diferenciados. Essa cobrança é feita a partir da caracterização de Planos de Serviço que estabelecem níveis de prioridade e da classificação dos recursos disponíveis na Grade. O uso desses fatores foi considerado com o objetivo de garantir que os níveis de Qualidade de Serviço solicitados pelos usuários sejam, de fato, atendidos e para que os recursos da Grade não sofram degradação de desempenho em decorrência da elevada taxa de utilização de tais recursos.

ABSTRACT

This work presents a system that applies a model for the supply of Quality of Service for Grid Computing Applications (Grid–QoS) based on Service Level Agreements (SLAs). The proposed system also uses a web interface to improve the interaction between users and the system. At the same time, a new charging process for the use of differentiated services is proposed. This charging process is made through both service plans characterization and the sort of Grid available resources. The use of these factors has been considered to guarantee that the Quality of Service levels requested by users are, in fact, attended and to protect the Grid resources from performance degradation due to its high usage level.

1 INTRODUÇÃO

O desenvolvimento tecnológico das últimas décadas desencadeou um aumento considerável na capacidade computacional dos *hardwares*. Esse processo permitiu aos desenvolvedores e pesquisadores imaginarem novas formas de explorar ao máximo toda a capacidade computacional disponível (FOSTER et al., 2001). No entanto, os elevados preços de computadores de última geração podem tornar inviável uma atualização constante dos *hardwares* necessários para a realização de determinados experimentos (FOSTER et al., 2002).

Uma alternativa para solucionar o problema de custo é a utilização de Grades Computacionais. Neste tipo de infra–estrutura, um determinado número de recursos (computadores e dispositivos) é utilizado, conectados entre si através de redes privadas ou públicas, permitindo que o processamento de um ou mais recursos cooperem na realização de uma tarefa em comum.

As grades computacionais permitem o uso de recursos computacionais geograficamente distribuídos, como processamento e memória, através de esquemas de segurança que envolvem a utilização de certificados que identificam os usuários aptos a utilizar uma grade computacional (FOSTER et al., 2001).

Em sua forma nativa, uma grade funciona basicamente da seguinte forma: um usuário define o número de recursos que ele deseja para executar a sua aplicação e a envia, junto com os arquivos e parâmetros necessários para a execução. Em seguida, o *middleware* de grade utilizado se encarrega de escalonar a tarefa para o número de recursos especificado pelo usuário, sem se preocupar com necessidades especiais requeridas pela aplicação ou pelo próprio usuário.

Durante anos, as Grades Computacionais foram capazes de atender às necessidades da comunidade científica. Entretanto, devido ao crescimento da utilização de grades comerciais e até mesmo devido à necessidade de oferecer serviços de melhor qualidade para os usuários pioneiros no uso dessa infra–estrutura, conclui–se que é preciso realizar o estudo de métodos que possam ser aplicados com o objetivo de inserir Qualidade de Serviço no contexto de Grades Computacionais. As vantagens obtidas com o estabelecimento de métodos de *Grid–QoS* podem variar, desde a otimização do desempenho do ambiente

utilizado, até o melhor atendimento das necessidades dos usuários que utilizam uma Grade (SAHAI et al., 2002).

Baseado nisso, este trabalho propõe, desenvolve e aprensenta resultados de um sistema que aplica Qualidade de Serviço em um ambiente real de Grade Computacional.

1.1 JUSTIFICATIVA

Existem diversos fatores que devem ser levados em consideração ao se buscar soluções para resolver as limitações relacionadas à Qualidade de Serviço em Grades Computacionais. Dentre essas abordagens, as definidas como justificativas para a realização deste trabalho são:

- A necessidade de melhor atender os usuários de um ambiente de Grade;
- Melhor utilização dos recursos e serviços de uma Grade Computacional;
- Definição de SLAs (Acordos em Nível de Serviço) para realizar cobranças distintas para serviços diferenciados oferecidos por uma Grade que tenha suporte a *Grid–QoS*;
- Permitir que usuários que não detenham conhecimentos sobre como utilizar um ambiente de grade possa fazê-lo com suporte a QoS de forma transparente.

Estes fatores foram escolhidos devido à sua importância ao incentivar o uso de uma Grade Computacional pelos seus usuários. Além disso, são importantes também para garantir o nível de Qualidade de Serviço que deve ser fornecido para um determinado usuário, realizando a cobrança proporcional a este nível de serviço.

1.2 OBJETIVOS

O planejamento e o desenvolvimento deste trabalho têm como objetivo atingir as seguintes metas:

- Definir planos de serviço que possam ser utilizados para definir os níveis de Qualidade de Serviço que um grupo de usuários necessita na execução de suas aplicações;

- Classificar os recursos disponíveis em uma Grade Computacional, aferindo pesos que variam de acordo com sua capacidade de processamento e com a memória disponível para a execução de tarefas;

- Desenvolver uma fórmula para cobrança dos serviços diferenciados, considerando os tipos de serviço e recurso utilizados. Essa fórmula, por sua vez, visa aperfeiçoar o modelo de Economia utilizado como referência para este trabalho ((SCHULZE, 2006)).

- Desenvolver um sistema que possua uma interface que permita uma interação simples entre os usuários e o ambiente de Grade. Esse sistema deve, ainda, oferecer serviços como agendamento de execução de tarefas e pré–escalonamento de recursos, características estas consideradas essenciais para um sistemas de *Grid–QoS* (AL-ALI et al., 2004);

- Realizar comparações para comprovar se houve, ou não, ganho de desempenho ao custo de um maior consumo da unidade contábil em questão. Estes resultados comparativos têm como objetivo também validar a fórmula de cálculo de consumo;

- Inclusão do sistema proposto em uma Grade Computacional comunitária na forma de um serviço.

1.3 METODOLOGIA

A pesquisa realizada para obter um ponto de partida para o desenvolvimento deste trabalho foi feita a partir de artigos científicos sobre trabalhos que, de alguma forma, propusessem métodos e arquiteturas de provisão de Qualidade de Serviço em Grades Computacionais. O levantamento de características e parâmetros relevantes para o desenvolvimento do trabalho também foi feito a partir do estudo de artigos científicos.

Os testes de validação da fórmula de cálculo de custo de submissão e do sistema provedor de *Grid–QoS* foram feitos utilizando recursos de uma Grade Computacional real pertencentes ao Laboratório ComCiDis (Computação Científica Distribuída), localizado no LNCC (Laboratório Nacional de Computação Científica).

Os resultados obtidos a partir dos testes realizados foram agrupados e apresentados na forma de tabelas e gráficos com o objetivo de facilitar o entendimento e validar o modelo proposto neste trabalho.

O texto deste trabalho se encontra organizado da seguinte forma: o capítulo 2 apresenta as principais características dos Sistemas Distribuídos, com um maior foco em Grades Computacionais; o capítulo 3 apresenta os conceitos de Qualidade de Serviço em Rede de Computadores e como tais conceitos se aplicam em Grades Computacionais; o capítulo 4 apresenta trabalhos encontrados na literatura que propõem métodos e arquiteturas para prover Qualidade de Serviço em Grades Computacionais; o capítulo 5 descreve a implementação do sistema proposto, bem como a definição dos planos de serviço e a classificação de recursos e o novo modelo de Economia; o capítulo 6 apresenta ainda os testes realizados e os resultados obtidos, bem como os comentários sobre os mesmos; por fim, o capítulo 7 apresenta as considerações finais acerca deste trabalho.

2 SISTEMAS DISTRIBUÍDOS

O intenso desenvolvimento tecnológico das últimas décadas permitiu que novas tecnologias para processamento de alto desempenho fossem desenvolvidas com o objetivo de auxiliar diversas áreas, tanto acadêmicas e científicas, como também comerciais. No entanto, a necessidade de desenvolver componentes de *hardware* cada vez menores, aliado às limitações da Física, acaba por limitar a capacidade computacional em um dispositivo de processamento único. Buscando alcançar uma alternativa para este problema, foram desenvolvidas formas de processamento distribuído, podendo este ser dentro do mesmo recurso, ou em ambientes geograficamente distantes uns dos outros.

A larga difusão e utilização de sistemas distribuídos na última década, seja ela através de *clusters* ou de grades computacionais geograficamente distribuídas, acabou gerando a necessidade de definir padronizações das características dos sistemas distribuídos com o objetivo de evitar que a heterogeneidade se tornasse um problema e permitir que houvesse uma coordenação harmônica entre os elementos do sistema (TANENBAUM, 2007).

Um sistema distribuído é definido por um conjunto de processadores, geograficamente distribuídos ou não, que não se utilizam de compartilhamento de memória nem relógio. A comunicação entre processadores acontece através de redes de comunicação de alta velocidade (SILBERSCHATZ et al., 2001).

2.1 CARACTERÍSTICAS DOS SISTEMAS DISTRIBUÍDOS

Um sistema distribuído possui características que lhe são peculiares e que devem ser respeitadas, levando–se em consideração a complexidade de sua estrutura. Tais características variam desde o tipo de rede utilizada, até a forma com que a segurança é implementada.

2.1.1 SEGURANÇA

Devido ao fato de um DFS (*Distributed File System*) permitir o acesso a arquivos compartilhados que são de propriedade de vários usuários diferentes, um problema de segurança é gerado naturalmente ao se comprometer a integridade de informações e arquivos dos

usuários do sistema. Para prevenir que usuários não–confiáveis acessem o sistema e adulterem estas informações, existem técnicas de segurança oferecidas pelos próprios sistemas operacionais como também métodos desenvolvidos com este fim, dentre os quais merecem destaque:

- ***Domínio de proteção***: Um domínio de proteção define, basicamente, quais processos podem executar e se utilizar de um determinado *hardware* (CPU, memória, dispositivos de entrada e saída, etc). Desta forma, um processo deve ter permissão de acionar apenas os recursos aos quais foi autorizado a acessar;

- ***Senhas***: Uma forma clássica de autenticação de usuários em um sistema é o uso da combinação do nome de usuário com sua respectiva senha. O uso deste método de autenticação pode ser utilizado para o usuário efetuar sua entrada no sistema, ou para obter acesso a determinadas áreas privadas de um sistema. Com o objetivo de se obter um resultado favorável no uso de senhas, deve–se recomendar que os usuários definam senhas "fortes", que oferecem maior dificuldade aos algoritmos de criptoanálise;

- ***Permissões***: Após um usuário realizar sua autenticação em um sistema distribuído, é necessário verificar o que este usuário está apto a acessar, quais diretórios e arquivos podem ser escritos, lidos e executados por ele. Este é o objetivo de se estipular permissões, tanto para usuários como para a estrutura de diretórios do sistema distribuído;

- ***Certificação Digital***: A utilização de certificados digitais tem por objetivo reconhecer a validade e idoneidade de um usuário, em outras palavras, certificar–se de que um usuário é quem realmente diz ser e se ele está devidamente reconhecido pela Autoridade Certificadora (AC) responsável pelo sistema.

2.2 COMPUTAÇÃO PARALELA UTILIZANDO *CLUSTERS*

Um *cluster* computacional pode ser definido como sendo a integração de recursos computacionais através de *hardware*, conexões de rede e *software* que se comportam como sendo um único computador (CISCO SYSTEMS, 2004). A idéia de cluster que, a princípio, era diretamente ligada à computação de alto desempenho, está cada vez mais sendo re-

vista de forma a ir além de computação paralela, incorporando também *clusters* com balanceamento de carga e *clusters* de alta disponibilidade (CISCO SYSTEMS, 2004).

Dentre os benefícios oferecidos pela infra–estrutura de um *cluster*, merecem destaque:

- ***Escalabilidade***: O fato de um *cluster* utilizar a capacidade de processamento de vários recursos simultaneamente, ao invés de um único recurso, permite que o aumento desta capacidade computacional seja feito apenas adicionando novos recursos ao *cluster* (CISCO SYSTEMS, 2004);

- ***Disponibilidade***: A disponibilidade em um *cluster* é garantida devido ao fato de que um recurso provê cópia de segurança para os demais e vice–versa no caso de uma falha. Da mesma forma, se um recurso está processando uma tarefa e falha por algum motivo, a tarefa é transferida e processada para um recurso que esteja funcional, de forma transparente para o usuário (CISCO SYSTEMS, 2004);

- ***Desempenho***: O ganho de desempenho de um *cluster* se dá pelo fato que a capacidade total de processamento de um *cluster* é igual à soma de todos os recursos funcionais. Além disso, o fato de um *cluster* ser um sistema distribuído geograficamente centralizado aumenta a velocidade da comunicação entre processos e das trocas de mensagens;

- ***Facilidade de gerenciamento***: A transparência de um *cluster* faz com que o usuário não tenha a necessidade de saber a origem do recurso que está utilizando (figura 2.1) (CISCO SYSTEMS, 2004). Somado a isso, o fato de um cluster ser um sistema homogêneo (*hardware* e sistema operacional similares) facilita a gerência e a solução de falhas.

O uso de um *cluster* computacional também permite que aplicações paralelas, como as que utilizam MPI (*Message Passing Interface*), por exemplo, inicialmente definidas para executarem em computadores multi–processados especialmente projetados, sejam executadas nesse tipo de infra–estrutura. O princípio da utilização de um *cluster* como uma máquina multi–processada é o mesmo, utilizando–se o modelo hierárquico mestre–escravo para a execução e coordenação do processamento, conforme ilustrado na figura 2.2.

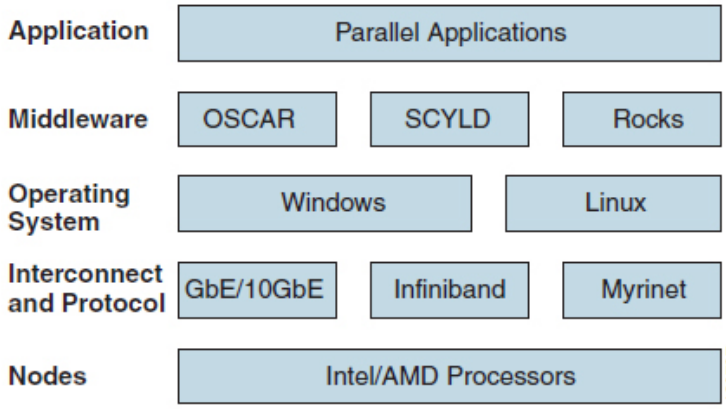

FIG. 2.1: Arquitetura de um *cluster* (CISCO SYSTEMS, 2004).

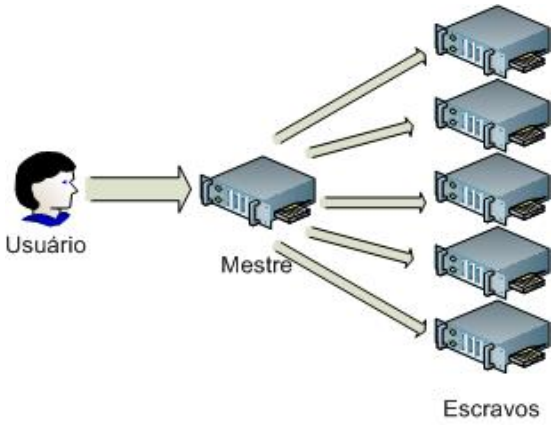

FIG. 2.2: Hierarquia de um *cluster*.

2.3 GRADES COMPUTACIONAIS

Uma das formas de tirar proveito da capacidade computacional proporcionada pelo paradigma de computação distribuída é a utilização da infra–estrutura de grades computacionais. As grades computacionais, assim como a computação paralela vista anteriormente, proporciona um aumento considerável na capacidade computacional. No entanto, possui características peculiares que fazem com que possua um maior foco em compartilhamento de recursos em larga escala (FOSTER et al., 2001). De fato, as grades computacionais diferem em vários aspectos das demais formas de computação de alto desempenho.

Existem diversas referências na literatura que buscam definir o que é uma grade computacional e, de certa forma, conseguem descrever esta infra–estrutura, no entanto, a definição que melhor resume o significado e objetivos das grades computacionais foi feita por Ian Foster e Carl Kesselman e está citada em (FOSTER, 2002).

"*Uma grade computacional é uma infra–estrutura de hardware e software que provê acesso seguro, consistente, de forma dispersa, com baixo custo e potencialidade máxima*".

Para entender melhor o que Foster e Kesselman disseram, é preciso explicitar e esclarecer as características peculiares que fazem com que uma grade computacional seja uma solução que oferece alta capacidade de processamento por um custo relativamente baixo:

- ***Segurança***: Como qualquer sistema distribuído, uma grade computacional se utiliza de compartilhamento de recursos, sejam eles de processamento ou armazenamento. Por isso, uma grade computacional possui mecanismos de controle de admissão e acesso para que não haja invasão nem alteração de informações confidenciais.

- ***Distribuição geográfica***: Uma característica que diferencia as grades computacionais dos outros sistemas distribuídos é a possibilidade de criar um sistema com alta capacidade computacional geograficamente distribuído, utilizando endereçamento IP público, permitindo assim que haja a comunicação e a cooperação entre pessoas e instituições em locais geograficamente distantes;

- ***Capacidade de processamento***: A capacidade total de processamento de uma grade computacional, assim como em um *cluster*, pode ser medida como a soma da capacidade de processamento de todos os recursos que fazem parte da grade.

- ***Heterogeneidade***: O fato de uma grade computacional permitir que recursos com *hardware* e sistemas operacionais distintos possam ser integrados em um mesmo ambiente faz com que não seja necessário o estabelecimento de um padrão para se incluir um novo recurso.

Embora as grades computacionais ofereçam as vantagens acima citadas, algumas dessas vantagens acabam provocando vulnerabilidades que devem ser observadas antes da implantação de uma infra-estrutura de grade computacional e constantemente monitoradas para evitar que estas vulnerabilidades degradem o desempenho do sistema:

- ***Conexões de rede***: Por se tratar de um sistema geograficamente distribuído, mesmo que em locais próximos, uma grade computacional é extremamente dependente das conexões de rede que fazem a ligação entre os seus recursos;

- ***Detecção de inconformidades***: O fato de ser um sistema heterogêneo aumenta a complexidade na detecção de falhas do sistema se comparado a um sistema homogêneo. É necessário que existam mecanismos capazes de prevenir e facilitar a detecção destas falhas para tornar viável a administração do sistema.

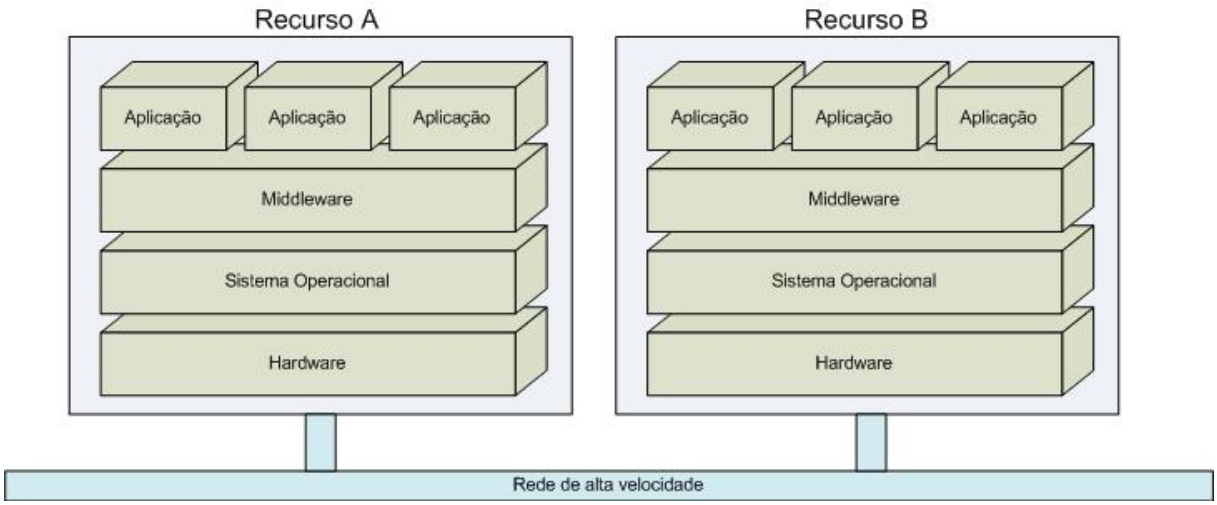

FIG. 2.3: Arquitetura de uma grade computacional.

Como mostrado na figura 2.3, uma grade computacional é formada por camadas de *hardware*, sistema operacional, *middleware* e aplicações que precisam interoperar entre si para que seu funcionamento seja corretamente coordenado.

2.3.1 *MIDDLEWARES* DE GRADE

Com o objetivo de permitir que aplicações de usuários de uma grade computacional consigam interagir com o sistema operacional e com os recursos de *hardware*, é necessário que *softwares* trabalhem para realizar essa comunicação. Estes *softwares* são denominados *middlewares* de grade.

Cada *middleware* de grade possui recursos e características próprias. No entanto, todos eles têm o objetivo principal de fazer com que as aplicações acessem e utilizem os recursos do sistema operacional e de *hardware*, independentemente da arquitetura computacional utilizada pelo recurso, permitindo assim que recursos heterogêneos cooperem entre si.

2.3.1.1 GLOBUS *TOOLKIT*

O Globus *Toolkit*, desenvolvido pela IBM e mantido pela *University of South Carolina*, pelo *Argonne National Laboratory* e pela *University of Chicago*, surgiu em meados da

década de 1990 com o objetivo de fornecer acesso aos recursos de uma grade computacional, bem como fornecer serviços que facilitem o seu uso (FERREIRA et al., 2003).

O fato do Globus *Toolkit* ser capaz de permitir que seus usuários compartilhem processamento e armazenamento de forma segura e sem afetar o desempenho dos recursos locais fez com que este *middleware* se tornasse o mais utilizado entre a comunidade que desenvolve aplicações em grades computacionais (FOSTER, 2005). Além disso, o Globus oferece serviços como:

- ***Grid Resource Allocation Manager***: O GRAM é o principal serviço do Globus Toolkit e é responsável por permitir que usuários localizem, submetam, monitorem e cancelem tarefas em um recurso de uma grade computacional, através do gerenciamento remoto no contexto de cada operação (FOSTER, 2005). O GRAM também é responsável pelo acesso ao armazenamento em memória secundária através do GASS (*Global Access to Secondary Storage*);

- ***Monitoring and Discovery Service***: O MDS é o serviço de monitoramento e descoberta de recursos utilizado pelo Globus *Toolkit*. A descoberta de recursos é feita localmente e então distribuída pelos outros recursos da grade computacional (FERREIRA et al., 2003). As informações podem ser estáticas, como o estado de uma máquina do sistema (ativo/inativo), ou dinâmicas, informando o uso de CPU ou a atividade do disco rígido.

- ***GridFTP***: O GridFTP é o serviço de transferência de dados do Globus. Seu conjunto de ferramentas e bibliotecas torna o GridFTP apropriado para transferir uma grande quantidade de dados de forma mais rápida, segura e confiável do que o Protocolo FTP padrão (FOSTER, 2005). Para aumentar a segurança da transferência de dados, o GridFTP utiliza os mecanismos de segurança do Globus *Toolkit*;

- ***Grid Security Infrastructure***: O GSI é o serviço responsável por fornecer segurança à infra–estrutura de grade computacional. O GSI age de forma a autenticar usuários e serviços, protege a comunicação entre recursos e verifica se um usuário e/ou serviço está, de fato, apto a acessar determinado recurso (FOSTER, 2005). A infra–estrutura de segurança do Globus utiliza–se do conceito de certificação digital e de PKI (*Public Key Infrastructure*).

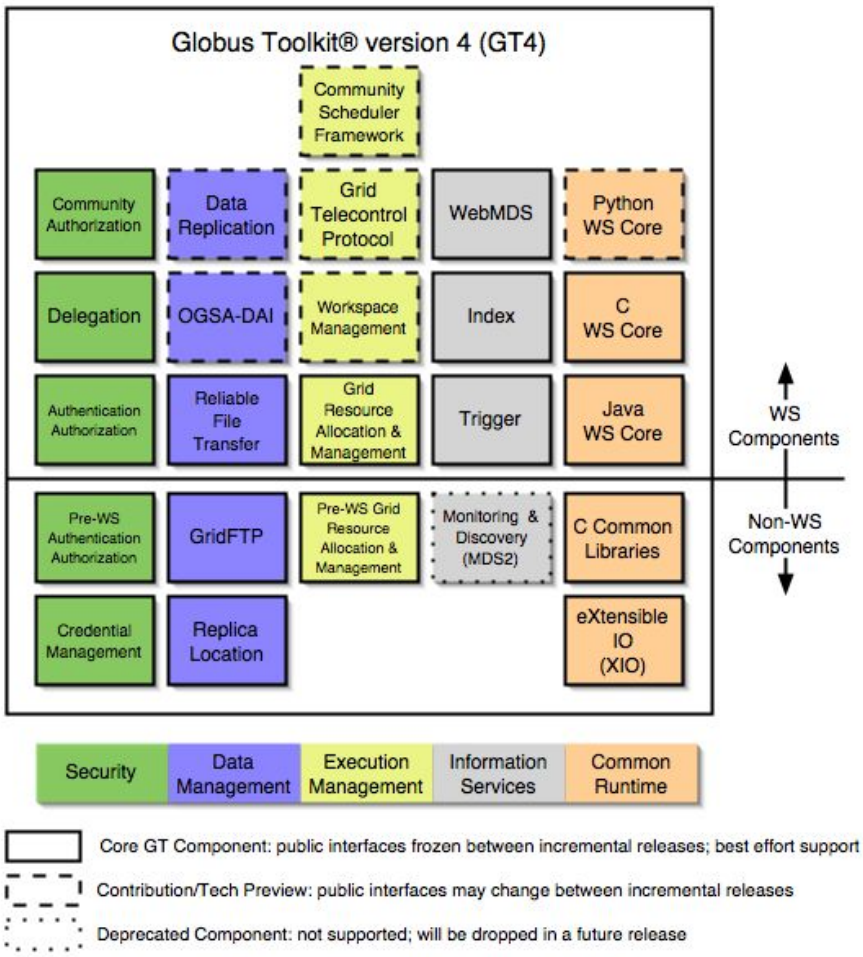

FIG. 2.4: Arquitetura do Globus *Toolkit* 4 (FOSTER, 2005).

Em sua versão mais atual (Globus *Toolkit* 4), representada pela figura 2.4, o Globus utiliza tecnologias responsáveis por otimizar e melhor integrar os serviços de grades computacionais, bem como facilitar o desenvolvimento de novas aplicações. Estas tecnologias são:

- ***Open Grid Service Architecture***: A arquitetura OGSA é baseada em tecnologias e conceitos de *Web Services* e que começou a ser utilizada pelo Globus *Toolkit* a partir da versão 3 para o desenvolvimento de serviços de grade que realizam melhor integração entre si (FOSTER, 2005);

- ***WS–Resource Framework***: O WSRF é um conjunto de especificações de *Web Services* que descrevem como implementar componentes OGSA utilizando *Web Services*.

Com o objetivo de padronizar a forma em que os clientes realizam a submissão de tarefas em recursos de uma grade computacional, o Globus utiliza uma linguagem para especificar as informações de uma tarefa. Estão descritos na RSL (*Resource Specification Language*), o nome da tarefa e de possíveis subtarefas e arquivos executáveis. Através da RSL, também é possível especificar a quantidade de processamento e memória necessários para executar uma tarefa em uma máquina remota (FOSTER, 2005).

Adicionalmente, o Globus permite a execução de aplicações MPI nos recursos da grade que estejam devidamente aptos para tal tarefa. Esta execução é feita através de bibliotecas que permitem a integração Globus/MPI. A mais recente é a implementação MPICH–G2, desenvolvida pelo *Argonne National Laboratory* exclusivamente para o Globus e integrando vários de seus componentes.

2.3.1.2 CONDOR

O Condor surgiu com o principal objetivo de desenvolver, implementar e disponibilizar mecanismos que suportem Computação de Alto Desempenho (HPC) em uma grande quantidade de recursos distribuídos (CONDOR, 2008).

Para permitir a realização de submissão de tarefas de usuários em série ou em paralelo, o Condor provê mecanismos de fila de tarefas, políticas de escalonamento, filas de prioridade e monitoramento e gerência de recursos (CONDOR, 2008). Desta forma, o Condor aloca tarefas em filas, define quando e onde uma tarefa será submetida, monitora sua execução e informa ao usuário quando a tarefa é concluída (CONDOR, 2008).

O Condor pode ser configurado para gerenciar um *cluster* de máquinas dedicadas para processamento. Além disso, é possível integrar um *cluster* Condor com uma grade computacional que utiliza Globus (CONDOR, 2008). Esta integração pode ser feita através de uma derivação do Condor chamada Condor–G. Esta derivação submete tarefas em recursos remotos utilizando uma interface com o Globus *Toolkit*.

Uma característica importante do Condor em sua versão padrão é que ele busca definir uma infra–estrutura de grade computacional em um ambiente de IPs virtuais. Portanto, para que seja possível a integração entre uma grade Condor e uma grade Globus é necessário que haja ao menos um IP público na grade Condor, uma vez que grades Globus utilizam apenas IPs públicos.

Com o objetivo de não prejudicar o desempenho das aplicações locais de um recurso, o Condor utiliza apenas recursos ociosos, que passam um determinado período sem receber

entradas de *mouse* e/ou teclado, criando assim a necessidade de migrar tarefas de um recurso que deixe de estar ocioso para um outro que ainda esteja. Esta migração é feita através da utlização de pontos de verificação e reiniciando a tarefa em outro recurso ocioso a partir deste ponto (CONDOR, 2008).

2.3.1.3 OURGRID

A iniciativa OurGrid surgiu com o objetivo de aumentar a capacidade computacional dos laboratórios de pesquisa ao redor do mundo (CIRNE et al., 2006), permitindo que aplicações sejam distribuídas através de uma grade computacional ou processadas em paralelo em *clusters* utilizando MPI, por exemplo, que façam parte da grade computacional.

Embora seja um *middleware* que forneça acesso aos recursos de uma grade computacional, o OurGrid parte do princípio que um usuário local tem prioridade de uso dos recursos locais. Isto ocorre com o objetivo de evitar que o desempenho do recurso local seja prejudicado pelo uso remoto dos recursos (ANDRADE et al., 2005), (CIRNE et al., 2006).

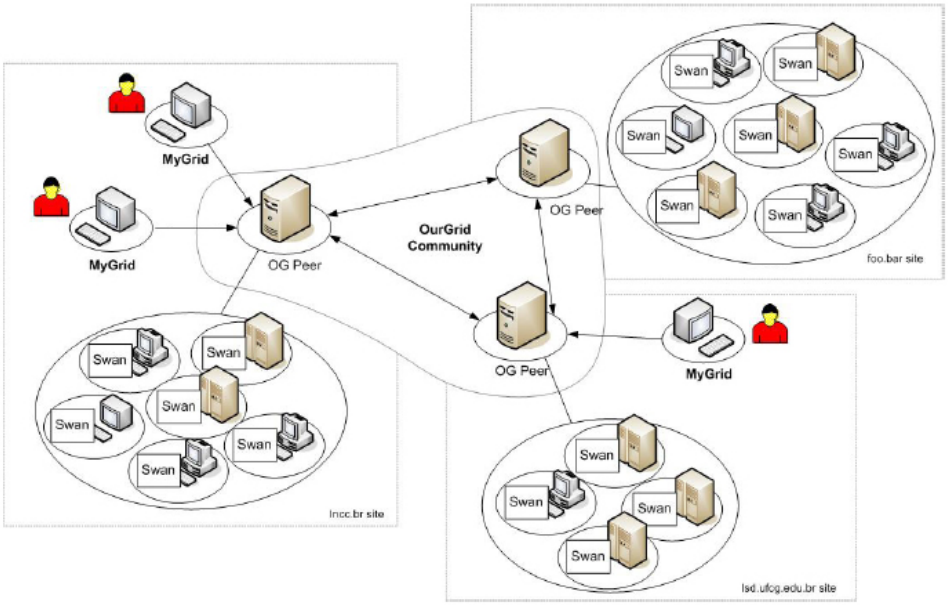

FIG. 2.5: Arquitetura do OurGrid.

O OurGrid foi desenvolvido com o objetivo de ser escalável, podendo suportar a adesão de um número ilimitado de laboratórios que se unem para aumentar a capacidade do sistema. A figura 2.5, retirada de (ANDRADE et al., 2005), demonstra a arquitetura do

OurGrid, exemplificando como é feita a conexão entre os diversos laboratórios através de uma rede P2P (*peer–to–peer*), que é o tipo de rede em que se baseia o OurGrid.

2.3.1.4 LEGION

O middleware de grade Legion surgiu em 1993 desenvolvido pela Universidade de Virginia, sendo assim um dos pioneiros no desenvolvimento de aplicações para grades computacionais (GRIMSHAW et al., 2006).

O Legion possui como principais características, a utilização do paradigma de orientação a objetos, de forma que todos os elementos da grade são vistos de como um objeto (GRIMSHAW et al., 2006). Além disso, oferece suporte a aplicações paralelas desenvolvidas utilizando MPI (*Message Passing Interface*).

Em 2001, os idealizadores do Legion fundaram a empresa Avaki que, atualmente, desenvolve e comercializa produtos que utilizam o Legion e suas tecnologias.

2.4 RESUMO DO CAPÍTULO

Este capítulo apresentou as características dos Sistemas Distribuídos que estão relacionadas com a realização deste trabalho. São exemplificadas também, arquiteturas de processamento distribuído, enfatizando as características das Grades Computacionais que, ao lado de Qualidade de Serviço, são o foco principal deste trabalho.

O capítulo 3 (a seguir) apresenta as características de Qualidade de Serviço em Redes de Computadores e em Grades Computacionais. Estas características foram levadas em consideração para a realização deste trabalho.

3 QUALIDADE DE SERVIÇO

O termo Qualidade de Serviço, atualmente aplicado na área de computação, é na verdade uma preocupação das empresas dos mais diversos segmentos da Economia. O aumento da concorrência obrigou estas empresas a investir em qualidade, tanto para os serviços como para os produtos como uma forma de sobrevivência, permitindo assim uma maior satisfação de seus clientes.

O uso da Qualidade de Serviço em computação surgiu de forma semelhante, com o objetivo de desenvolver sistemas capazes de melhor atender às necessidades dos usuários, sejam elas operacionais, de interface ou financeiras.

Com o avanço tecnológico na área de computação, com novas tecnologias de desenvolvimento, transmissão de dados e processamento, a Qualidade de Serviço passou a ser tratada de forma individual por cada uma das sub–áreas da computação, assim como diferem os requisitos necessários para atender a cada uma delas.

A oferta de serviços diferenciados faz com que seja possível e necessário a cobrança destes serviços de acordo com o que cada um é capaz de fornecer. Ao fazer esta distinção, permite que um serviço de melhor qualidade seja mais oneroso do que serviços de qualidade inferior. Desta forma, pode–se definir um diferenciamento de serviços em função dos custos, sem que haja degradação do sistema devido a utilização demasiada de serviços de alta qualidade que demandam mais recursos computacionais (SAHAI et al., 2002).

Nas seções a seguir serão mostrados alguns padrões e requisitos para que se obtenha Qualidade de Serviço em Redes de Computadores e em Grades Computacionais, uma vez que existem características comuns entre elas, como a especificação de parâmetros chave na tentativa de se estabelecer padrões e protocolos de fornecimento de Qualidade de Serviço.

3.1 QUALIDADE DE SERVIÇO EM REDES DE COMPUTADORES

A idéia de Qualidade de Serviço em Redes busca oferecer melhoria nos serviços oferecidos pela *Internet* que, ainda nos dias de hoje, oferece serviços baseados no “melhor esforço”, uma herança da origem da grande rede na década de 1970. Em uma rede que oferece serviços de “melhor esforço”, não existe nenhuma garantia de que uma determi-

nada aplicação possa reservar a Qualidade de Serviço necessária para melhor lhe atender. Isto acontece porque o protocolo IP (*Internet Protocol*) não oferece suporte à Qualidade de Serviço em sua forma nativa (STARDUST.COM, 1999).

O primeiro passo para se definir um escopo de como a Qualidade de Serviço em Redes pode atender de forma diferente as aplicações que necessitam de tratamentos especiais foi a definição dos tipos de serviços para que, a partir destes, aplicações e parâmetros de desempenho de uma conexão de rede fossem classificados. A primeira definição de planos de serviços diferenciados está descrita em (ISO, 1995) da seguinte forma:

- ***Melhor esforço***: O sistema fornecedor de serviços não apresenta nenhuma garantia de que a qualidade de serviço será mantida dentro dos limites pré–estabelecidos;
- ***Compulsório***: O sistema fornecedor de serviços monitora os recursos e pode atender às necessidades das aplicações, caso os recursos requeridos não se encontrem disponíveis naquele momento, e até mesmo interrompendo o serviço, caso a qualidade de serviço não possa ser mantida dentro dos limites;
- ***Garantido***: O sistema fornecedor de serviço aceita o usuário quando os fatores de QoS forem capazes de atender às necessidades das aplicações, conforme pré–estabelecido pelo nível de serviço associado ao usuário.

Em Qualidade de Serviço em Redes, o nível de serviço, segundo (GOMES, 1999), "expressa o grau de certeza de que a QoS será mantida pelo fornecedor", o que é diretamente dependente das políticas de provisão de QoS definidas pelo provedor de serviços.

3.1.1 PARÂMETROS DE QOS EM REDES

A definição de planos de serviço é importante para que seja possível classificar os serviços quanto à sua qualidade. No entanto, é preciso que estes planos de serviço estejam diretamente vinculados a parâmetros que possam ser quantificados para que, desta forma, seja possível aplicar os valores destes parâmetros, comparando com as necessidades de cada aplicação.

Os parâmetros de medição de desempenho e análise da Qualidade de Serviço relevantes em uma rede, de acordo com (KATCHABAW et al., 1998), são:

- ***Latência da rede***: A latência ou atraso de uma rede (*delay*) é o tempo que pacotes levam para ir de um ponto a outro da rede;

- ***Jitter***: É a variação da latência de uma rede. O *jitter* é influenciado por fatores como colisão de pacotes e pela taxa de utilização da rede;

- ***Capacidade da conexão (throughput)***: É a capacidade real de transmissão de uma rede;

- ***Taxa de perda de pacotes***: É um percentual de quantos pacotes são perdidos em uma transmissão. A taxa de perda de pacotes está diretamente ligada à confiabilidade da rede.

A partir da análise e da combinação dos parâmetros com a definição inicial de tipos de serviço, é possível classificar as aplicações de acordo com as necessidades críticas de cada uma conforme exemplificado na tabela 3.1, retirada de (TANENBAUM, 2003).

TAB. 3.1: Requisitos de Qualidade de Serviço.

Aplicação	**Confiabilidade**	**Retardo**	**Flutuação**	**Largura de banda**
Correio eletrônico	Alta	Baixa	Baixa	Baixa
Transferência de arquivos	Alta	Baixa	Baixa	Média
Acesso à *Web*	Alta	Média	Baixa	Média
Login remoto	Alta	Média	Média	Baixa
Áudio por demanda	Baixa	Baixa	Alta	Média
Vídeo por demanda	Baixa	Baixa	Alta	Alta
Telefonia	Baixa	Alta	Alta	Baixa
Videoconferência	Baixa	Alta	Alta	Alta

3.1.2 PROTOCOLOS E ALGORITMOS DE QOS EM REDES

Como foi visto no início desta Seção, a Qualidade de Serviço em Redes tem o objetivo principal de prover garantia de serviços consistentes para aplicações específicas. Devido ao fato das aplicações possuírem requisitos de QoS distintos, dois tipos básicos de Qualidade de Serviço foram definidos (KATCHABAW et al., 1998), (STARDUST.COM, 1999). São eles:

- ***Reserva de Recursos (IntServ)***: Os recursos de rede são divididos e reservados de acordo com as necessidades de QoS da aplicação, respeitando o uso de banda estipulado nas políticas de uso do provedor de serviço;

- ***Priorização (DiffServ)***: O tráfego da rede é classificado e a divisão de recursos é feita conforme especificado nas políticas de uso de banda estipulada pelo provedor de serviço. A Qualidade de Serviço é garantida quando uma aplicação que possui requisitos de QoS é identificada e tratada de forma a priorizá–la para que suas necessidades sejam atendidas.

Paralelamente, estes tipos de QoS podem ser aplicado em fluxos individuais ou fluxos agregados de aplicações, assim sendo, existem outras duas formas de caracterizar Qualidade de Serviço em Redes:

- ***Fluxo individual***: Um fluxo individual é uma troca de informações, em sentido único, entre duas aplicações (emissor e receptor) identificados por um conjunto único de cinco campos (protocolo de transorte, endereço IP da fonte, porta da fonte, endereço IP do destino e a porta do destino);

- ***Fluxo agregado***: Um fluxo agregado nada mais é do que dois ou mais fluxos que, geralmente, possuem características em comum como, por exemplo, um ou mais dos cinco campos, um rótulo ou número de prioridade, ou ainda alguma informação sobre autenticação.

A decisão de qual tipo de QoS é o mais apropriado para fluxos individuais ou agregados é tomada levando–se em conta as características da aplicação, da rede utilizada e das políticas de uso adotadas. Para melhor acomodar as necessidades dos diferentes tipos de QoS, os seguintes protocolos e algoritmos de QoS foram definidos:

- ***Resource reSerVation Protocol (RSVP)***: Fornece uma sinalização para habilitar a reserva de recursos. O RSVP também é conhecido como Serviços Integrados (*IntServ*). Embora seja utilizado geralmente em aplicações baseadas em fluxo individual, o RSVP também pode ser utilizado para a reserva de recursos para fluxos agregados;

- ***Differentiated Services (DiffServ)***: Fornece métodos simples para classificar e priorizar o tráfego da rede baseado em fluxos agregados;

- ***Multi Protocol Labeling Switching (MPLS)***: Fornece gerenciamento de banda para fluxos agregados através do controle de roteamento da rede de acordo com os rótulos acoplados nos cabeçalhos dos pacotes transmitidos;

- ***Subnet Bandwidth Management (SBM)***: Permite a classificação e o estabelecimento de prioridade na camada de Enlace de dados (nível 2 do modelo OSI) em redes compartilhadas e comutadas baseadas no padrão IEEE 802[1].

TAB. 3.2: Gerenciamento de banda por cada algoritmo e protocolo.

QoS	***Rede***	***Aplicação***	***Descrição***
Maior uso	X		Disponibilização de recursos fim-a-fim (ex: redes privadas e de baixo tráfego)
	X	X	Serviços RSVP (*Resource reSerVation Protocol*) – Fornece troca de informações entre as aplicações
	X	X	Serviço de carga RSVP – Fornece informações para as aplicações
	X		*Multi-Protocol Label Switching* (MPLS)
	X	X	Serviços Diferenciados (*DiffServ*) aplicado no núcleo de redes adequadas para o nível de serviço com reserva de recursos RSVP para determinado fluxo. Priorização utilizando *Subnet Bandwidth Manager* (SBM) aplicado em LANs também se encaixam nesta categoria
	X	X	*Diffserv* ou SBM aplicados nos códigos fontes de aplicações baseadas em fluxo
	X		*Diffserv* aplicado no núcleo de uma rede
	X		Aplicação de sistema de filas "justas" pelos elementos da rede
Menor uso			Serviços de "melhor esforço"

A tabela 3.2, retirada de (STARDUST.COM, 1999), descreve o gerenciamento de banda feito por cada um dos algoritmos e protocolos, seus relativos níveis de utilização de QoS e indica quais são inicializados por mecanismos da rede ou por chamadas das próprias aplicações.

3.2 QUALIDADE DE SERVIÇO EM GRADES COMPUTACIONAIS

As Qualidades de Serviço em Grades Computacionais e em Redes possuem em comum vários aspectos, não apenas nas características que diferem as aplicações que demandam recursos diferentes, como também na própria origem de cada uma delas. Uma vez que os

[1]Norma de padronização para redes locais e metropolitanas, cabeadas ou não, das camadas Física e de Enlace.

objetivos principais em ambos os casos é oferecer serviços de melhor qualidade, oferecendo a opção de escolha do serviço que mais se adequa a cada caso específico. Existe também a preocupação em realizar a cobrança destes serviços de forma justa e que reflita de fato a utilização e a qualidade dos recursos e serviços pelos usuários (SAHAI et al., 2002).

Assim como a Qualidade de Serviço em Redes, o conceito de *Grid–QoS* surgiu como evolução de serviços comuns de Grades Computacionais que, se comparados com os serviços equivalentes na área de Redes, pode também ser chamado de "melhor esforço". Com o aumento do uso de Grades Computacionais por aplicações comerciais e também pelo aumento da complexidade das aplicações, acarretando um aumento da demanda computacional por elas exigidas e na necessidade do estabelecimento de níveis de prioridade (SAHAI et al., 2002), surgiu a necessidade de oferecer serviços de Grades Computacionais com maior qualidade para atender aos novos requisitos de aplicação.

A Qualidade de Serviço em Grades Computacionais no entanto, não pode apenas oferecer serviços de qualidade garantida em um determinado recurso. Deve haver, também, o comprometimento com outros fatores relacionados à confiabilidade, à segurança e à usabilidade do sistema (BURCHARD et al., 2005), uma vez que, ao contrário do que acontece em Redes, os usuários de uma Grade Computacional realizam interações diretas com o sistema fornecedor de serviços.

3.2.1 REQUISITOS DE *GRID–QOS*

De acordo com (AL-ALI et al., 2004), um sistema de fornecimento de Qualidade de Serviço em Grades Computacionais deve manter os esforços concentrados para atender os seguintes requisitos:

- ***Reserva antecipada de recursos***: Um sistema de *Grid–QoS* deve permitir que seja possível realizar reservas de recursos, sejam elas antecipadas, imediatas (aplicações submetidas pela primeira vez em um determinado momento) ou sob demanda (aplicações submetidas anteriormente, que necessitam de renegociação de QoS nos recursos para continuar em execução). Reserva de recursos é importante em situações onde a oferta de recursos é escassa ou quando aplicações necessitam de condições mínimas para o funcionamento favorável;

- ***Política de reservas***: O sistema de *Grid–QoS* deve permitir o estabelecimento de políticas de uso e reservas que definam quais usuários podem acessar os recursos

disponíveis, de qual forma e em qual momento estes recursos podem ser acessados;

- ***Protocolos de acordo***: Os protocolos de acordo tem o objetivo de garantir aos usuários do sistema de *Grid–QoS* que os recursos reservados e a qualidade requerida sejam de fato alcançadas. O controle dos níveis de qualidade para cada serviço pode ser definido e monitorado pelas SLAs (*Service Level Agreements*);

- ***Segurança***: O sistema deve possuir mecanismos de segurança que proíbam usuários mal intencionados de modificar informações sobre reservas de recursos e acordos de serviço. Além disso, o sistema deve ser capaz de realizar a comunicação entre os recursos através de um canal seguro, bem como mecanismos de autenticação e autorização para acesso aos recursos da Grade Computacional;

- ***Simplicidade***: O sistema deve prover uma forma de comunicação simples que não comprometa a infra–estrutura dos recursos existentes. Ao mesmo tempo, deve oferecer uma interface que permita a interação transparente entre os usuários e o sistema;

- ***Escalabilidade***: O sistema deve ser capaz de suportar eventuais aumentos ou diminuições do número de recursos da Grade Computacional, sem que isto afete o funcionamento ou prejudique o desempenho do sistema fornecedor de *Grid–QoS*.

Alguns destes requisitos de *Grid–QoS* já são atendidos naturalmente por propostas já existentes. Como exemplo, pode–se citar o Globus *Toolkit* (FOSTER, 2005), que fornece mecanismos de segurança para autenticação e autorização para a utilização de recursos através do uso de certificação digital e uso de chaves baseado na infra–estrutura PKI (*Public Key Infrastructure*).

3.2.2 PARÂMETROS DE *GRID–QOS*

Para que seja possível classificar os recursos de forma a atender os requisitos especificados na seção 3.2.1, é necessário que parâmetros para a análise de desempenho, confiabilidade e disponibilidade de uma Grade Computacional sejam definidos. A semelhança entre estes parâmetros e os utilizados para Redes é tanta que, na verdade, os parâmetros de análise de rede definidos por (KATCHABAW et al., 1998) e listados na seção 3.1.1 são utilizados, entre outros, como uma parcela na determinação da *Grid–QoS* de um sistema (AL-ALI et al., 2004).

A avaliação de desempenho da Qualidade de Serviço de uma Grade Computacional não pode, no entanto, ser medida levando–se em consideração apenas os fatores de QoS em Redes. Uma vez que recursos de processamento e armazenamento também são utilizados, é necessário obter valores que indiquem a carga de CPU, memória, armazenamento e rede em uma Grade. Isto pode ser feito através dos sistemas de gerenciamento, sejam eles nativos dos próprios *middlewares*, ou desenvolvidos por outras iniciativas.

Outros indicadores também podem ser utilizados para a avaliação da *Grid–QoS* como os de confiabilidade e disponibilidade de recursos em uma Grade propostos por (MURY et al., 2008). O primeiro é definido pelo número de tarefas executadas e bem sucedidas em um recurso dividido pelo número total de tarefas submetidas naquele recurso. O segundo é definido pelo tempo em que um recurso se encontra disponível dividido pelo tempo total de operação da Grade Computacional. Estes dois fatores podem ser considerados para estipular os planos de serviço definidos nas SLAs, assim como distribuir pesos maiores para os recursos mais confiáveis e disponíveis.

De uma forma geral, o conceito de *Grid–QoS* é semelhante ao de QoS em Redes, uma vez que ambos têm o objetivo de garantir que os respectivos serviços sejam oferecidos com a qualidade requerida pelos usuários e realizar a cobrança de forma justa para cada um dos tipos de serviço. A diferença está basicamente na quantidade de variáveis envolvidas na análise de cada caso.

3.3 RESUMO DO CAPÍTULO

Este capítulo retratou os aspectos de Qualidade de Serviço considerados relevantes para a realização deste trabalho. Foram apresentados parâmetros e indicadores que são utilizados para medir e classificar o nível de QoS de um determinado recurso. Alguns dos parâmetros considerados são usados tanto para QoS em redes como para *Grid–QoS*.

O capítulo 4 (a seguir) apresenta propostas encontradas na literatura que objetivam abordar a questão de Qualidade de Serviço em Grades Computacionais.

4 REVISÃO DA LITERATURA

Na literatura, existem propostas de sistemas que tem por objetivo prover Qualidade de Serviço em Grades Computacionais (Grid-QoS). Cada uma delas possui características próprias que são vistas como referência a ser seguida, ou como fatores que devem ser modificados. Dentre eles, os que mais se relacionam com este trabalho e merecem destaque são: GARA (ROY, 2001), G-QoSm (AL-ALI et al., 2004), AppLeS (EPCC, 2003), Nimrod/G (NIMROD/G, 2005).

4.1 GARA

O *framework* GARA (*General-purpose Architecture for Reservation and Allocation*) (ROY, 2001) é capaz de prover suporte à Qualidade de Serviço em grades computacionais (AL-ALI et al., 2004). Basicamente, permite que os desenvolvedores especifiquem parâmetros de *Grid–QoS* fim–a–fim, isto é, permite reservas com tratamento uniforme de recursos tais como conexões de rede, processamento e armazenamento. A reserva de recursos do GARA tem o objetivo de garantir que uma aplicação receberá uma determinada Qualidade de Serviço do gerenciador de recursos. Além disso, o GARA fornece uma interface para gerenciar solicitações de reserva, oferecendo opções como criar, alterar e cancelar (AL-ALI et al., 2004).

Um dos fatores que fizeram com que o GARA se tornasse popular para o fornecimento de Qualidade de Serviço em Grades Computacionais é o fato de ter seu funcionamento intimamente ligado ao *middleware* de grade Globus *Toolkit*. Na verdade, o GARA funciona dentro do Globus na forma de um agente responsável pelo gerenciamento de reserva de recursos, como pode ser visto na figura 4.1, retirada de (ROY, 2001), onde, à esquerda, está a arquitetura do Globus em sua versão padrão, e à direita, a arquitetura do GARA funcionando em conjunto com o Globus.

Embora o GARA tenha se tornado popular na comunidade de grades computacionais, ele possui limitações que envolvem os requisitos para as aplicações e as tecnologias utilizadas:

- Atualmente, a integração entre Grades Computacionais e tecnologias de *Web Ser-*

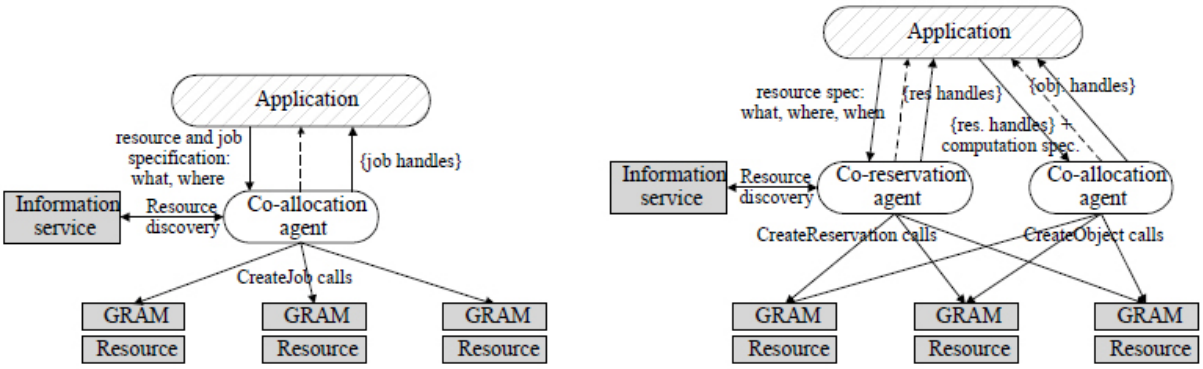

FIG. 4.1: Arquiteturas do Globus *Toolkit*(esq) e do GARA(dir)(ROY, 2001).

vices, bem como o desenvolvimento de *middlewares* de grade utilizando esta tecnologia sugere uma tendência em direção aos padrões de *Web Services*, tais como o WSRF (*Web Services Resource Framework*) e OGSA (*Open Grid service Architecture*). Devido ao fato do GARA não ser compatível com OGSA, aplicações que utilizam esta tecnologia não podem fazer uso dos serviços fornecidos pelo GARA (AL-ALI et al., 2004);

- Por necessitar da alocação de vários recursos simultaneamente, as aplicações de Grades Computacionais necessitam da existência de um protocolo de acordo de serviço que forneça informações sobre os recursos negociados para alocação e o nível de Qualidade de Serviço que uma aplicação demanda. Este tipo de informação geralmente está incluso em Acordos em Nível de Serviço (SLAs). O GARA não fornece o suporte à utilização de um protocolo de acordo de serviço e ao estabelecimento destes protocolos para os recursos (AL-ALI et al., 2004);

- O monitoramento da Qualidade de Serviço durante uma sessão ativa é fundamental para que se possa garantir o fornecimento de *Grid–QoS* para uma aplicação. O GARA não possui mecanismos de cálculo dinâmico do uso dos recursos e por isso, pode disponibilizar informações incorentes e uma conseqüente falha na provisão da Qualidade de Serviço.

4.2 *GRID QOS MANAGEMENT*

Para solucionar algumas inconformidades do GARA, principalmente relacionadas às tecnologias utilizadas, o Grid–QoSm (*Grid QoS Management*) foi desenvolvido no contexto

do OGSA (AL-ALI et al., 2004).

O funcionamento do Grid–QoSm consiste de três fases operacionais (AL-ALI et al., 2004): *estabelecimento*, *atividade* e *conclusão*. Durante a fase de *estabelecimento*, o usuário define quais serviços serão necessários para a execução de sua aplicação, bem como os requisitos de Qualidade de Serviço. A partir destas informações de QoS, o Grid–QoSm consulta o serviço de descoberta de recursos e realiza a negociação de acordo para a aplicação. Durante a fase de *atividade*, o Grid–QoSm pode realizar operações de monitoramento de QoS, adaptação, contabilidade, e renegociação, caso necessário. Durante a fase de *conclusão*, a sessão é terminada por motivos de expiração de reserva, por violação de acordo ou pela finalização do serviço. O Grid–QoSm suporta estas três fases utilizando componentes, conforme especificado na figura 4.2 (AL-ALI et al., 2004).

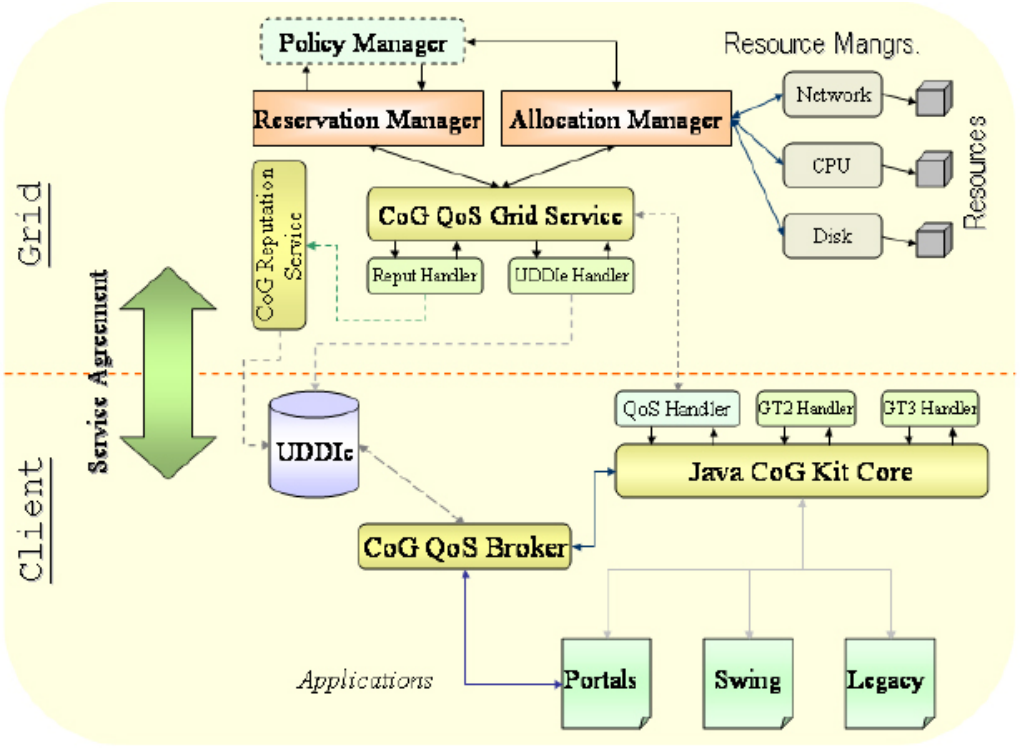

FIG. 4.2: Arquiteturas do Grid–QoSm com suporte A OGSA (AL-ALI et al., 2004).

Para fornecer Qualidade de Serviço em Grades Computacionais, o Grid–QoSm utiliza um componente compatível com OGSA chamado QGS (*QoS Grid Service*). Este componente é capaz de prover funcionalidades de QoS, tais como negociação, reserva e alocação de recursos com uma qualidade considerável (AL-ALI et al., 2004). O QGS publica–se automaticamente de forma que outros serviços compatíveis sejam capazes de identificá–lo

e de consultar suas informações. Adicionalmente, as funcionalidades de QoS do QGS suportam duas estratégias de alocação (AL-ALI et al., 2004). São elas:

- ***Alocação por domínio de recurso***: O usuário pode especificar uma determinada fração da capacidade dos recursos que estejam configurados para fornecer Qualidade de Serviço, como alocação de CPU ou de banda de rede;
- ***Alocação por domínio de tempo***: O usuário pode solicitar que um recurso seja alocado em sua totalidade de forma exclusiva para a execução de sua aplicação, o que significa dizer que a aplicação irá usar toda a capacidade de processamento deste recurso.

Entre as principais limitações encontradas no G–QoSm, merecem destaque o fato de que, assim como o GARA, é necessário que os usuários possuam conhecimentos técnicos sobre a utilização de um ambiente grade para que sejam capazes de otimizar o uso do sistema (BANDINI et al., 2008), além disso, não é utilizada nenhuma forma de Economia nesta proposta, isto significa que não é possível oferecer serviços diferenciados pelo simples fato de que não é possível realizar cobrança distinta para os diferentes tipos de serviços/recursos e pela quantidade utilizada dos mesmos.

4.3 APPLES

O AppLeS (*Application-Level Scheduling*) (EPCC, 2003) é um sistema de escalonamento adaptativo no nível da aplicação que pode ser aplicado a uma grade computacional, onde cada aplicação submetida em uma Grade Computacional pode ter uma instância própria do AppLeS (LI e BAKER, 2005). A idéia principal do AppLeS é que a utilização do sistema e seu desempenho são vistos da perspectiva da aplicação (LI e BAKER, 2005). Para obter o desempenho desejado, o AppLeS realiza a medição do desempenho da aplicação em um recurso e usa estas informações para selecionar e escalonar recursos em submissões futuras (LI e BAKER, 2005).

Para entender melhor como funciona o AppLeS, sua arquitetura está organizada como na figura 4.3 e seus componentes são descritos por (LI e BAKER, 2005) da seguinte maneira:

- ***Network Weather Service (NWS)***: Coletor de informações dinâmicas do estado do sistema e previsão de carga dos recursos;

- ***User specifications***: Informações sobre requisitos especificados por usuários a respeito de desempenho desejado e parâmetros de execução;
- ***Model***: É um repositório de modelos padrões, baseados em classes de aplicações similares que podem ser usados para gerar estimativas de desempenho e seleção de recursos;
- ***Resource selector***: Componente responsável por escolher e filtrar diferentes combinações de recursos;
- ***Planner***: Este componente é utilizado para criar uma descrição de escalonamentos que sejam dependentes de recursos;
- ***Performance estimator***: Este componente é responsável por estabelecer uma estimativa de desempenho para possíveis perfis de escalonamento definidos pelas especificações dos usuários;
- ***Coordinator***: O Coordenador é o componente responsável por escolher a melhor opção de escalonamento;
- ***Actuator***: O componente *Actuator* é o responsável por implementar a melhor opção de escalonamento, definida pelo Coordenador, no sistema de gerenciamento do recurso escolhido.

O AppLeS difere das outras aplicações no sentido de que a seleção de recursos e as decisões relativas ao escalonamento são baseadas em necessidades específicas de desempenho de uma determinada aplicação. Além disso, o AppLeS é voltado para aplicações do tipo mestre–escravo (LI e BAKER, 2005), características típicas de aplicações paralelas, podendo assim ser utilizado em uma infra–estrutura de *cluster*.

O fato de o AppLeS não ser um sistema gerenciador de recursos (LI e BAKER, 2005) faz com que ele não possua funcionalidades fundamentais para um ambiente de *Grid–QoS* como a reserva antecipada de recursos. Da mesma forma, embora realize escalonamento adaptativo, utilizando informações obtidas a partir da execução da própria aplicação, o AppLeS não oferece facilidade aos usuários que não tenham conhecimentos técnicos suficientes para utilizar o sistema de forma plena.

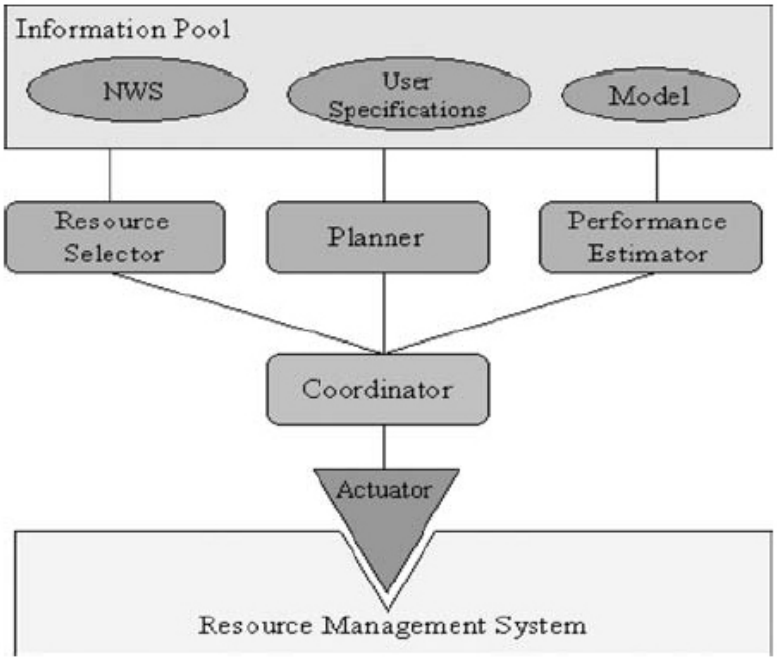

FIG. 4.3: Arquitetura do AppLeS (EPCC, 2003), (LI e BAKER, 2005).

4.4 NIMROD/G

Desenvolvido por pesquisadores da Universidade de Monash na Austrália, o Nimrod/G é uma ferramenta criada com o intuito de auxiliar o desenvolvimento de aplicações que realizam a simulação de modelos paramétricos utilizando Grades Computacionais. Seu objetivo é simular o comportamento de experimentos complexos, que envolvam um número elevado de variáveis, atendendo diversas áreas, tais como engenharia, economia, medicina, etc (NIMROD/G, 2005).

O Nimrod/G utiliza algumas características oferecidas pelo Globus *Toolkit* (FOSTER, 2005) como a descoberta de recursos além de utilizar o conceito de economia computacional (NIMROD/G, 2005). Ele comporta–se como um gerenciador de recursos e também como sistema de escalonamento e faz uso de uma forma de Economia computacional para o cálculo do consumo de recursos, desta forma, fornecendo Qualidade de Serviço diferenciada. A arquitetura do Nimrod/G segue as especificações definidas pela figura 4.4.

Feito com o intuito tanto de aumentar a capacidade computacional, como também de oferecer outras opções de processamento de alto desempenho, o Nimrod/G não está restrito apenas ao uso em Grades Computacionais, em conjunto com o Globus *Toolkit*, ele também permite a integração com outros *middlewares* de grade, tais como o Condor

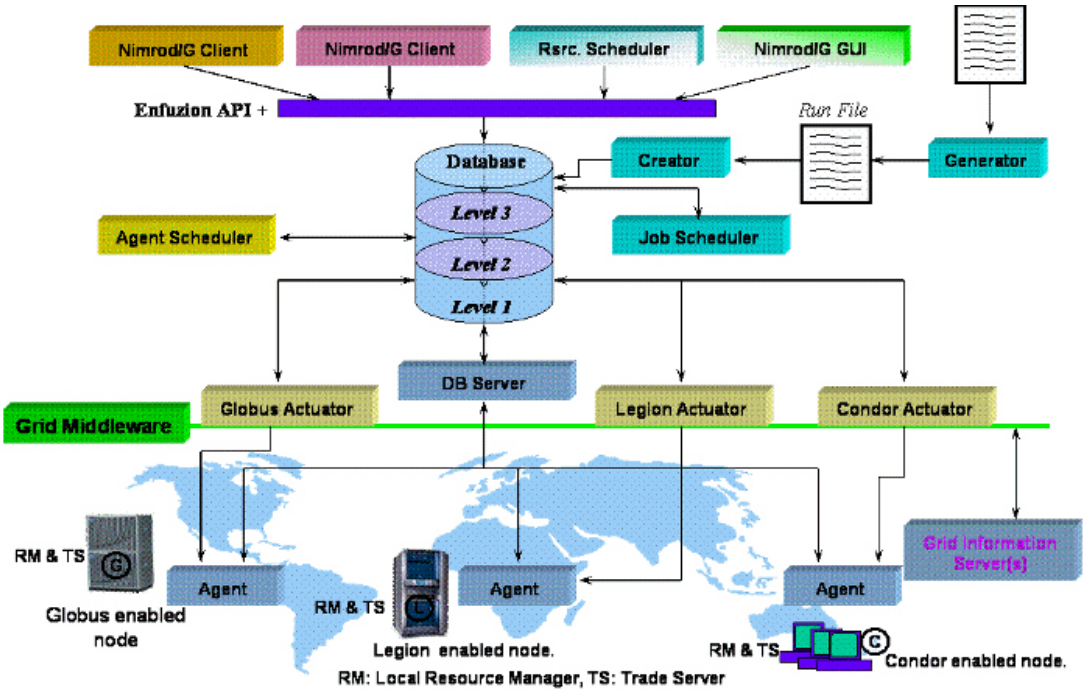

FIG. 4.4: Arquitetura do Nimrod/G (NIMROD/G, 2005).

e o Legion.

Para solucionar uma das principais limitações do Nimrod/G, foi desenvolvido um Portal *web* para facilitar a interação dos usuários com o sistema, uma vez que em sua versão inicial, todas as operações do Nimrod/G eram feitas através de linhas de comando de sistemas Unix. Contudo, o sistema ainda exige que os usuários possuam conhecimentos avançados, não apenas do uso de sistemas de Grades Computacionais, como também da definição dos parâmetros utilizados para cada aplicação. Por fim, o Nimrod/G não trabalha com protocolos de acordo em nível de serviço (SLAs), o que poderia ser uma forma de tornar o uso do ambiente mais transparente para usuários com pouco ou nenhum conhecimento da área.

4.5 *VIRTUAL RESOURCE MANAGER ARCHITECTURE*

O VRM (*Virtual Resource Manager Architecture*) (BURCHARD et al., 2005) é uma arquitetura projetada com o objetivo de fornecer serviços de alto nível combinando diferentes recursos do sistema local e utilizando–se de SLAs para formalizar os requisitos de serviços solicitados pelos usuários. Isto é feito através do estabelecimento de ADs (*Administrative Domains*), que podem ser vistos como um conjunto de recursos sob a regência do mesmo

conjunto de políticas de acesso (BURCHARD et al., 2005). Estas políticas de acesso definem quais recursos são acessíveis e como este acesso pode ser feito.

A Arquitetura VRM é constituída basicamente por três camadas, conforme ilustrado na figura 4.5. São elas:

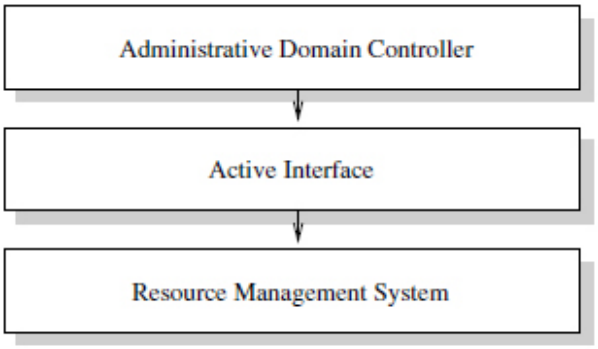

FIG. 4.5: Camadas da Arquitetura VRM (BURCHARD et al., 2005).

- ***Administrative Domain Controller (ADC)***: O objetivo da camada ADC é configurar o domínio administrativo, publicá–lo na Grade e/ou integrá–lo com outros ADs. Desta forma, um ADC pode funcionar como um *gateway* entre os recursos internos e externos, enquanto que o administrador local controla seu próprio sistema (BURCHARD et al., 2005).

- ***Active Interfaces (AI)***: A camada AI funciona na forma de um intermediário entre as demandas solicitadas pelo ADC e a capacidade de atendê–las, definidas por um RMS local específico. Os AIs são capazes de mapear subconjuntos dos mecanismos das SLAs com as capacidades de serviço do RMS (BURCHARD et al., 2005). Isto permite uma compatibilidade entre o que está estipulado nas SLAs e as demandas de aplicações, fazendo assim a negociação entre o AI e o ADC. O AI pode ainda receber mensagens de *feedback* enviadas pelo RMS. Este *feedback* é importante no sentido de garantir que as regras estipuladas pelas SLAs sejam respeitadas;

- ***Resource Management Systems (RMS)***: O RMS permite que o gerenciador local de recursos, quando administrado por um AD, seja capaz de prover acesso a vários tipos de recursos físicos (BURCHARD et al., 2005). Desta forma, o ADC permite acessar o RMS em *clusters* e sistemas de gerenciamento de rede. Ao contrário deste RMS, o sistema de gerenciamento pode se tornar uma outra instância do VRM. Assim sendo, é permitido criar estruturas hierárquicas do VRM.

Além de gerenciar o processo de negociação através do uso das ADCs e AIs, o VRM oferece serviços importantes do ponto de vista da integridade da execução das aplicações de um ambiente de Grade Computacional com Qualidade de Serviço. Estes serviços oferecem tolerância a falhas e controle de tarefas longas. A primeira, de forma a migrar a aplicação que esteja em um recurso que tenha sofrido alguma falha, seja ela de *hardware* ou de conexão de rede, para outro recurso que seja equivalente respeitando o estabelecido pelas SLAs. O segundo é atingido através da execução de pontos de verificação na execução de uma tarefa. Estes pontos de verificação podem ser obtidos em ciclos de tempo fixo ou por um determinado número de passos computacionais realizados.

4.6 RESUMO DO CAPÍTULO

Este capítulo apresentou propostas relacionadas à questão de Qualidade de Serviço em Grades Computacionais. Algumas das características e idéias dos trabalhos relacionados foram consideradas para a realização deste trabalho.

O capítulo 5 (a seguir) apresenta o sistema proposto e os conceitos utilizados para a caracterização de planos de serviço e classificação de recursos, de forma a permitir que o modelo de *Grid–QoS* proposto seja aplicado.

5 DESCRIÇÃO DO SISTEMA PROPOSTO

Este capítulo descreve o desenvolvimento do trabalho proposto, apresentando o modelo geral do sistema, identificando suas dependências, explicitando os módulos componentes, bem como os métodos utilizados para o desenvolvimento de cada um e, por fim, apresentando e analisando os resultados obtidos com testes comparativos entre um ambiente de Grade Computacional utilizando o sistema de *Grid–QoS* e o mesmo ambiente com serviços padrões que não fazem uso de Qualidade de Serviço.

A idéia principal do sistema aqui apresentada é oferecer Qualidade de Serviço para uma Grade Computacional Comunitária que possa ser utilizada, tanto por instituições de ensino, pesquisa e empresas, como por usuários comuns que necessitam de capacidade computacional para executar aplicações complexas.

O princípio de *Grid–QoS* adotado para o desenvolvimento deste sistema é o mesmo encontrado em (SAHAI et al., 2002), onde é definido que um sistema que oferece Qualidade de Serviço deve oferecê–la não apenas no nível dos serviços propriamente ditos, mas também nos meios utilizados pelos usuários para acessar o sistema.

Com o objetivo de atingir as metas necessárias para que o sistema forneça *Grid–QoS*, é sugerida uma nova forma de cobrança pelos serviços diferenciados oferecidos na Grade. Apresenta também, uma submissão onde são definidos os valores de CPU e memória utilizados em cada submissão, além da definição de espaço em disco para os usuários de cada serviço. Desta forma, comparando submissões que utilizam, ou não, Qualidade de Serviço.

Por fim, o Sistema utiliza–se da interface *web* do Portal VCG como uma forma de facilitar o acesso às informações e o uso da Grade por parte dos usuários que não possuam conhecimento específico sobre um ambiente de Grade Computacional.

5.1 DEPENDÊNCIAS DO SISTEMA PROPOSTO

Para que o desenvolvimento de um sistema capaz de oferecer *Grid–QoS* baseado em SLAs seja possível, é necessário utilizar mecanismos de monitoramento de recursos e serviços de uma Grade Computacional para que se possa comparar os serviços disponíveis com os que são solicitados pelos usuários através das definições dos planos de serviço. É necessário

também um sistema de apoio que permita a cobrança pelos serviços utilizados, seja ela em níveis financeiros, ou apenas através da disponibilização e priorização de novos serviços para os usuários que contribuem de alguma forma com o ambiente de Grade.

Embora o sistema proposto se utilize dos sistemas descritos nas seções 5.1.1 e 5.2, estas dependências não sugerem que ele esteja adequado somente a elas, uma vez que a abordagem de definição de planos de serviço e cobrança diferenciada apresentadas neste trabalho podem ser adaptadas para outros sistemas, desde que possuam características semelhantes.

5.1.1 VCG – *VIRTUAL COMMUNITY GRID*

O *Virtual Community Grid* (SCHULZE, 2006) é um projeto realizado pelo Laboratório Nacional de Computação Científica (LNCC) em parceria com a Rede Nacional de Ensino e Pesquisa (RNP). Sua idéia principal é fornecer uma infra–estrutura de Grade Computacional que utiliza um sistema de adesão voluntária. O objetivo deste projeto é disponibilizar e difundir um ambiente de Grade Computacional que ofereça processamento de alto desempenho, com baixo custo, capaz de atender à comunidade científica brasileira.

No contexto do VCG, um projeto é formado por um grupo de usuários e cada usuário fornece seus recursos para a Grade VCG, associando estes recursos ao projeto ao qual está vinculado. Créditos são adicionados à conta de um projeto conforme seus recursos são utilizados por usuários de outros projetos. Da mesma forma, créditos são debitados da conta de um projeto que utiliza recursos de outros projetos para a execução de suas tarefas.

Buscando atender às necessidades de usuários que não possuam conhecimentos técnicos avançados sobre o uso de uma Grade Computacional, o Projeto VCG desenvolveu uma série de serviços próprios para o uso e gerenciamento dos recursos, bem como aplicativos que visam facilitar a instalação do conjunto de *softwares* e *middlewares* necessários para um novo recurso aderir à Grade:

- ***Portal de Informações***: O Portal de Informações do *Virtual Community Grid* é uma interface *web* responsável por permitir a criação e o gerenciamento de contas de usuários e projetos. Provê também gerenciamento de certificados digitais utilizados como método de autenticação para acesso aos recursos da Grade e submissão. Além disso, fornece informações tais como o gerenciamento dos créditos do projeto,

relatórios de uso destes créditos em submissões (créditos e débitos), além de ser o canal de comunicação entre os usuários da Grade e sua administração;

- ***Portal de Submissão***: O Portal de Submissão é uma interface *web* responsável por permitir a submissão de tarefas nos recursos devidamente cadastrados e certificados na Grade VCG. Além disso, oferece um sistema de monitoramento das tarefas submetidas e ressubmissão. Possibilita ainda o *download* e *upload* de arquivos para que as aplicações sejam executadas corretamente e que os resultados possam ser visualizados pelos respectivos usuários;

- ***Portal de Monitoramento***: O Portal de Monitoramento é uma interface *web* responsável por fornecer informações sobre quais os recursos que estão ativos na Grade VCG e estão aptos a receber submissão de tarefas. Além disso, disponibiliza informações individuais de cada recurso como CPU, memória, disco, arquitetura, carga de processamento utilizada, etc. Basicamente, permite que as informações coletadas pelo VMS sejam visualizadas pelos usuários.

- ***VCG Management System (VMS)***: O componente VMS é responsável pela coleta das informações dos recursos que estão ativos na Grade VCG. Além disso, realiza o controle e as operações de débitos e créditos decorrentes da submissão de tarefas. O VMS é o responsável ainda pela comunicação entre os Portais e a Autoridade Certificadora (AC) para a geração e assinatura de certificados digitais de usuários e máquinas. A seção 5.2 descreve a estrutura e o funcionamento do VMS de forma mais detalhada.

- ***Pacotes e Scripts de Instalação***: Os Pacotes de Instalação do VCG foram desenvolvidos com o objetivo de facilitar a instalação e a configuração de programas e serviços necessários para que um recurso possa ser adicionado na Grade VCG.

A figura 5.1 ilustra a estrutura do *Virtual Community Grid* e o fluxograma de operações de associação de projetos e usuários, bem como a descoberta de recursos da Grade.

A importância do Projeto VCG para este trabalho se dá devido ao fato do aproveitamento do Portal de Informações, através de sua interface que será utilizada para a interação entre os usuários e o sistema, que é o responsável por obter as informações dos recursos que serão necessárias para garantir a qualidade dos serviços oferecidos.

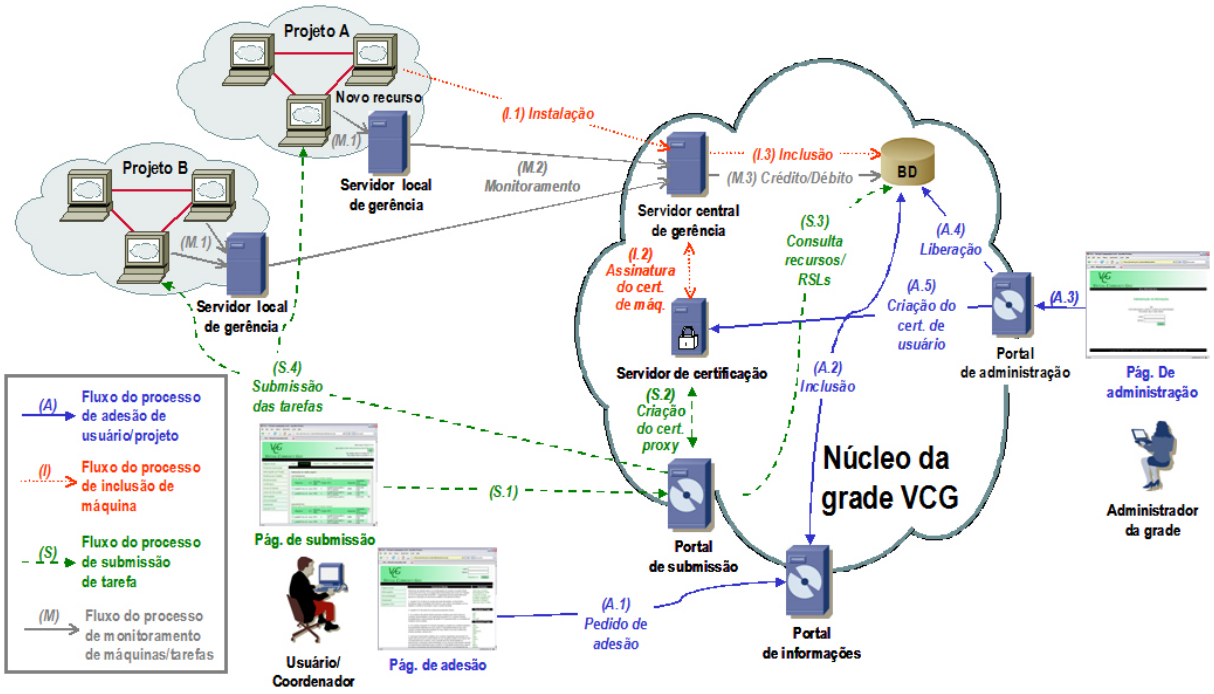

FIG. 5.1: Estrutura do *Virtual Community Grid*.

5.1.2 VMS – *VCG MANAGEMENT SYSTEM*

O VMS (*VCG Management System*) foi desenvolvido para ser capaz de abranger um ou mais domínios, também conhecidos na literatura como Organizações Virtuais (OVs) (FOSTER et al., 2002). O VMS possui uma estrutura hierárquica onde cada domínio possui um Servidor de Dados de Domínio (SDD). O SDD de um domínio recebe informações sobre recursos e submissões de tarefas do respectivo domínio através do Servidor de Dados Locais (SDL), que ficam hospedados em cada um dos recursos alocados na grade. Por fim, o SDD do domínio publica estas informações no SDD global a cada 10 (dez) minutos, caso não haja falhas nas conexões de rede. Em caso de falhas, o SDD do domínio registra as informações em um banco de dados local até que a conexão com o SDD global seja restabelecida. O SDD global possui informações sobre todos os recursos da grade e é o responsável pelas operações que envolvem créditos entre os projetos do VCG. A figura 5.2 ilustra a estrutura hierárquica do VMS.

O uso do VMS como ferramenta de gerência e monitoramento de recursos neste trabalho se deve ao fato de que este obtém as informações relevantes sobre os recursos disponíveis (CPU total e carga utilizada, memória total e disponível, informações de disco e análise de desempenho dos recursos). Estas informações são fundamentais para

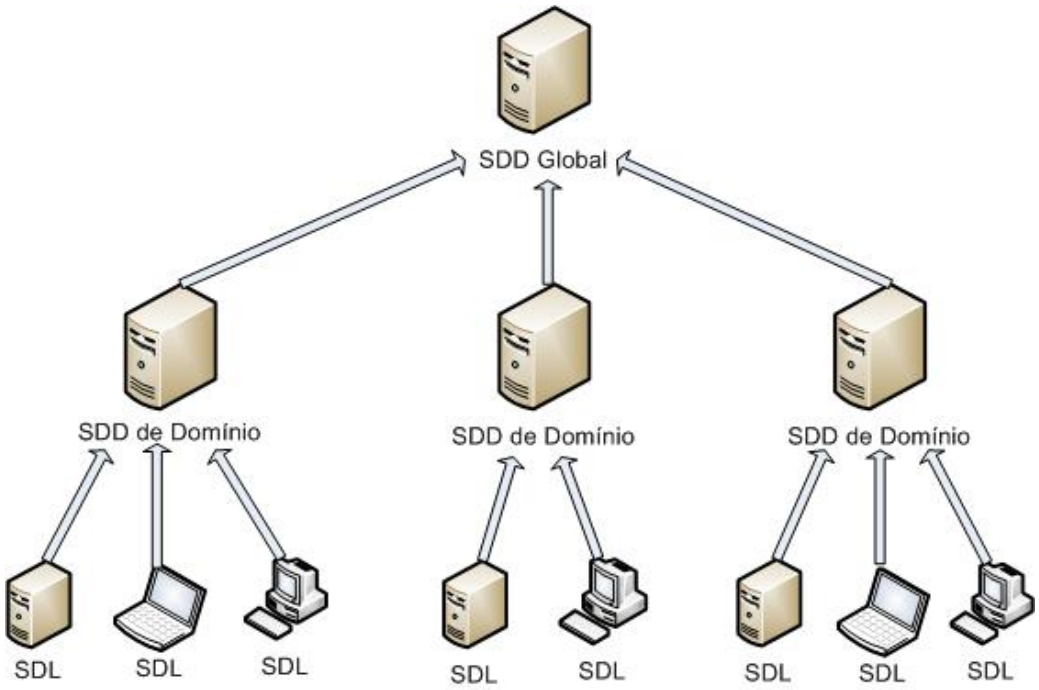

FIG. 5.2: Estrutura hierárquica do VMS.

a definição dos planos de serviço (SLAs), assim como para a ponderação de valores que serão cobrados por cada um deles. Além disso, o VMS também gerencia uma Economia de créditos, o que permite que haja uma cobrança efetiva pelo uso dos serviços.

5.2 ECONOMIA DE GRADE BASEADA EM CRÉDITOS

Para que seja possível aplicar um sistema que forneça Qualidade de Serviço em uma Grade Computacional, é necessário que haja uma forma de cobrança pelos serviços diferenciados oferecidos (SAHAI et al., 2002). Isto significa que algum modelo de Economia, seja ela monetária ou de troca de favores, deve ser utilizada para que o fornecimento de *Grid–QoS* seja viável.

Como referência de um modelo de Economia para este trabalho, será utilizado o do Projeto VCG (SCHULZE, 2006). Este modelo de Economia se caracteriza pela troca de créditos entre os projetos de acordo com o uso dos recursos da grade pertencentes a cada um, conforme visto na seção 5.1.1. O cálculo de créditos consumidos ou creditados para os projetos em cada submissão é feito de acordo com o tempo que uma tarefa necessita para sua execução no(s) recurso(s) selecionado(s).

A figura 5.3 ilustra o processo de aquisição e consumo de créditos gerado a partir

da submissão de uma tarefa de um usuário associado ao Projeto B em três recursos que pertencem a um usuário do Projeto A. O custo total de créditos neste exemplo é calculado através da soma dos custos individuais em cada recurso, medido apenas pelo tempo de execução, não importanto o tipo de recurso utilizado, nem os níveis de prioridade de um determinado projeto. Desta forma, considerando que um crédito corresponde a dez segundos de uso de um recurso por uma tarefa, pode–se dizer que o cálculo de custo de submissões feitas no escopo do *Virtual Community Grid* é feito seguindo a seguinte fórmula:

$$C = S.n \tag{5.1}$$

onde C é o custo total, S é a quantidade de créditos consumidos em um único recurso e n é o número de recursos utilizados em uma submissão. As operações de débito e crédito nas contas dos projetos é realizada pelo VMS.

O objetivo principal deste trabalho é redefinir a forma de cobrança feita pelo VCG. Para isso foram definidos planos de serviços (SLAs) que permitem o fornecimento de serviços diferenciados. Além disso, foi feita também uma classificação de recursos de acordo com as capacidades de processamento e armazenamento de cada um deles. As definições das SLAs e a classificação de recursos encontram–se nas seções 5.2.1 e 5.2.2, respectivamente.

5.2.1 DEFINIÇÃO DOS PLANOS DE SERVIÇO (SLAS)

O primeiro passo para aplicar um sistema que ofereça Qualidade de Serviço baseado em SLAs (*Service Level Agreements*) é definir estas SLAs na forma de planos de serviço que são responsáveis por delimitar os níveis de serviço e garantir que estes níveis sejam atendidos pelo sistema. Para isso é preciso definir quais serão os parâmetros utilizados. No caso do sistema proposto, os parâmetros são os seguintes:

- ***CPU disponível***: Percentual mínimo de CPU disponível que será alocado exclusivamente para a execução de uma determinada tarefa;
- ***Memória disponível***: Percentual mínimo de memória necessário para que uma tarefa execute corretamente;
- ***Disco disponível***: Valor que indica a capacidade máxima de disco que pode ser reservada para armazenar resultados e saídas de tarefas em um determinado recurso.

Este valor é inicialmente considerado para armazenamento temporário, sendo os arquivos armazenados, apagados após um determinado período. O período que um arquivo ocupa determinado espaço em disco pode ser um alvo para a definição de um novo parâmetro de *Grid–QoS*, embora o sistema proposto não considere este fator;

- ***Prioridade***: Estabelece o nível de prioridade que as tarefas de um Projeto possuem;

- ***Rede***: Embora o sistema proposto considere apenas a latência da rede como fator de peso para a definição dos planos de serviço, os demais parâmetros de QoS (variação da latência, capacidade de transmissão e taxa de perda de pacotes) também podem ser considerados.

Com os parâmetros que serão utilizados para a estruturação dos planos de serviço devidamente definidos, é possível caracterizar um ou mais planos de serviço que devem obedecer os parâmetros e seus valores. A tabela 5.1 apresenta os planos de serviço definidos para a realização deste trabalho.

TAB. 5.1: Tabela de planos de serviço (SLAs).

Tipo SLA	CPU (% disponível)	Memória (% disponível)	Disco (% disponível)	Prioridade	Rede (atraso)	Taxa (t) (custo + %)
Básico	Inferior a 30	Inferior a 20	Inferior a 10	Baixa	< 500 ms	+ 0 %
Master	entre 30 e 50	entre 20 e 40	15	Média	< 200 ms	+ 25 %
Premium	entre 50 e 80	entre 40 e 70	25	Alta	< 100 ms	+ 50 %

Além de definir os valores de parâmetros que cada plano de serviço deve garantir, a tabela 5.1 apresenta também uma taxa (t) de acréscimo que será aplicada a cada um dos tipos de serviço oferecidos.

Os parâmetros utilizados pelo sistema proposto, denominados parâmetros quantitativos (MURY et al., 2008), não são os únicos que podem ser considerados para a definição de planos de serviço em um sistema de *Grid–QoS*. Existem outros parâmetros qualitativos, tais como confiabilidade, disponibilidade e segurança, que podem influenciar na qualidade dos recursos e serviços disponíveis na Grade.

5.2.2 CLASSIFICAÇÃO DE RECURSOS

Apesar de ser extremamente necessário definir os tipos de serviço, é preciso também classificar os recursos e definir o custo individual para cada um deles. Isso porque, se dois recursos heterogêneos possuírem o mesmo custo, é natural que aquele que oferecer o melhor desempenho seja sempre o escolhido para uso, o que pode fazer com que este recurso sofra degradação. Para evitar que isso aconteça, foi criada a tabela 5.2 contendo os recursos existentes no ambiente de testes do LNCC com suas características e um fator de peso para cada um, de acordo com suas capacidades de processamento. Estes recursos foram escolhidos para testes por oferecerem alta estabilidade e estarem disponíveis na maior parte do tempo.

TAB. 5.2: Tabela de classificação de recursos.

Tipo Recurso	Qtd núcleos CPU	CPU Total	Memória Total	Fator (*f*)
A	2	Até 1.0GHz	1024 MB	0,2
B	2	Até 1.0GHz	1536 MB	0,4
C	2	Até 1.0GHz	2048 MB	0,6
D	2	Entre 1.0GHz e 1.5 GHz	1024 MB	0,8
E	2	Entre 1.0GHz e 1.5 GHz	1536 MB	1,0
F	2	Entre 1.0GHz e 1.5 GHz	2048 MB	1,2
G	1	Entre 1.5GHz e 2.0 GHz	1024 MB	1,4
H	1	Entre 1.5GHz e 2.0 GHz	1536 MB	1,6
I	1	Entre 1.5GHz e 2.0 GHz	2048 MB	1,8
J	2	Entre 1.5GHz e 2.0 GHz	1024 MB	2,0
K	2	Acima 2.0 GHz	1024 MB	2,2
L	2	Entre 1.5GHz e 2.0 GHz	1536 MB	2,4
M	2	Acima 2.0 GHz	1536 MB	2,6
N	2	Entre 1.5GHz e 2.0 GHz	2048 MB	2,8
O	2	Acima 2.0 GHz	2048 MB	3,0
P	4	Acima 2.0 GHz	4096 MB	3,4
Q	8	Acima 2.0 GHz	4096 MB	3,8

O fato de apenas os recursos existentes no ambiente de teste utilizado estarem listados na tabela 5.2 não impede que novos recursos com diferentes configurações de *hardware* sejam incluídos e devidamente classificados.

Assim como o que acontece na definição dos planos de serviços, a classificação de recursos proposta neste trabalho considera apenas a capacidade de processamento, o número de núcleos do recurso e a quantidade máxima de memória. No entanto, é possível ponderar e reajustar o valor do fator de utilização (f) de cada recurso, conforme proposto em (MURY et al., 2008).

5.3 REDEFINIÇÃO DO MODELO DE ECONOMIA

Após definir os planos de serviço, com os respectivos níveis de prioridade e suas taxas de acréscimo e classificar os recursos de acordo com a capacidade de processamento oferecida, é possível redefinir a fórmula de cálculo de custo de uma submissão, em função dos valores de t e f da seguinte forma:

$$C = S + \alpha(S.t) + \beta(S.f) \tag{5.2}$$

onde:

- C = custo total da submissão no recurso selecionado;
- S = quantidade de créditos utilizados em uma submissão;
- t = taxa que representa o acréscimo relativo ao plano de serviço (tabela 5.1);
- f = fator de utilização do recurso em questão (tabela 5.2);
- $\alpha = 1$; $\beta = 1/8$, Sendo α e β os fatores de equilíbrio da fórmula.[2]

Os valores de equilíbrio de α e β foram obtidos após a realização de testes iniciais e da observação da variação dos tempos de execução das tarefas e dos custos de cada uma destas submissões.

Para se obter o valor do custo total de uma submissão (C_t) é neccessário apenas somar os valores de custo individuais de cada recurso (C) utilizados para a submissão.

$$C_t = C_1 + C_2 + ... + C_n \tag{5.3}$$

Com o novo Modelo de Economia definido, é possível comparar este novo modelo com o usado pela Economia de Créditos do VCG, que também pode ser considerada como sendo o plano de serviço caracterizado pelo uso da Grade sem Qualidade de Serviço, tomando a figura 5.3 como exemplo para realizar uma comparação simples do consumo de créditos. Esta comparação considera que os recursos utilizados são iguais e do tipo "A" definido na tabela 5.2 e com o plano de serviço **Master** especificado na tabela 5.1. Os resultados dos testes reais realizados podem ser vistos no capítulo 6.

[2]Valores de α e β definidos a partir da avaliação do comportamento custo/benefício de submissões de teste.

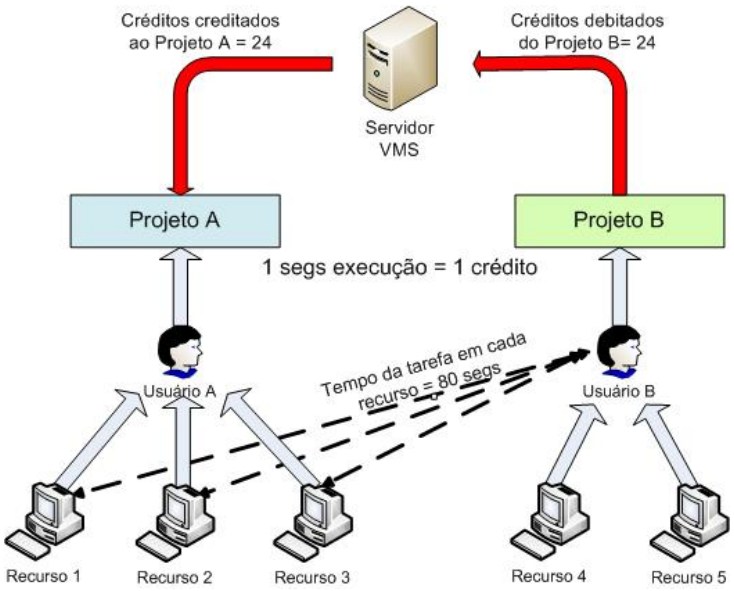

FIG. 5.3: Consumo e aquisição de créditos em uma submissão.

TAB. 5.3: Tabela de comparação entre os Modelos de Economia.

	Tempo utilizado	**Modelo VCG (plano Básico)**	**Modelo proposto (plano Master)**
Custo individual	80 segundos	80 créditos	108 créditos
Custo Total	240 segundos	240 créditos	306 créditos

A tabela comparativa 5.3 comprova que há um consumo maior de créditos para um plano que demanda maior Qualidade de Serviço do que um dos planos com *Grid–QoS* inferior, usando uma tarefa hipotética que consome o mesmo tempo de execução nos mesmos recursos. No entanto, este comparativo apenas não justifica o uso de *Grid–QoS*, uma vez que, não apenas o consumo deve ser diferenciado, como também os serviços devem oferecer vantagens e garantias distintas que favoreçam os planos de custo mais elevados.

5.4 BENEFÍCIOS DO SISTEMA PROPOSTO

A implementação do sistema proposto resultou em algumas características que, agregadas ao Portal de Informações do Projeto VCG, facilita o uso e o acesso a informações, visando atender às necessidades de usuários e suas aplicações no âmbito da Qualidade de Serviço em Grades Computacionais. Estes benefícios, além do novo Modelo de Economia proposto na seção 5.3, são as seguintes: pré–escalonamento de recursos com *Grid–QoS*, reserva

antecipada de recursos e submissão de tarefas com *Grid–QoS*.

5.4.1 PRÉ–ESCALONAMENTO DE RECURSOS COM *GRID–QOS*

Com o estabelecimento de planos de serviço, são definidos os níveis de prioridade necessários para atender às aplicações de usuários, que estejam associados aos projetos que utilizam cada um destes serviços. A partir destas informações, é possível realizar uma seleção de recursos capazes de atender a estas aplicações. Isto é feito comparando as informações de prioridade de um determinado plano de serviço com as informações disponibilizadas no repositório de dados fornecidas pelo VMS, conforme ilustra a figura 5.4.

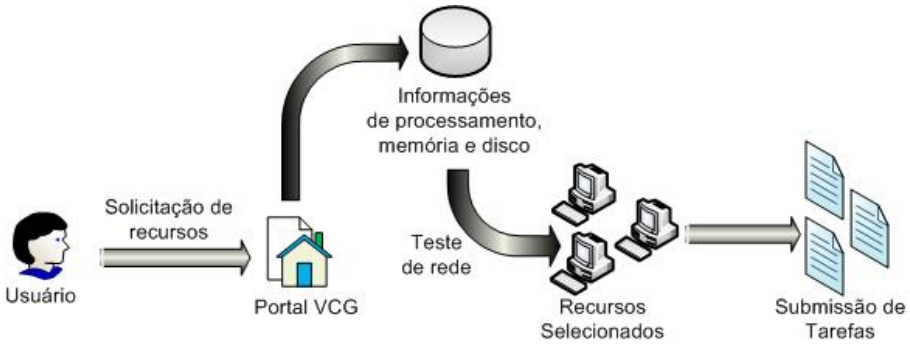

FIG. 5.4: Fluxograma de pré–escalonamento de recursos.

Ao observar a figura 5.4, pode–se concluir que, no contexto do *Virtual Community Grid*, um usuário, ao criar seu projeto, escolhe o plano de serviço que melhor atende às necessidades de processamento, memória, disco e rede das aplicações do projeto. Ainda no contexto do VCG, observando a figura 5.5, conclui–se que existe uma hierarquia no sentido de que um usuário precisa estar associado a um projeto, que, por sua vez, precisa ser cliente de um dos tipos de serviço disponíveis para que possa utilizar os recursos dos demais projetos pertencentes à Grade Computacional com a Qualidade de Serviço requerida.

O processo de pré–escalonamento é feito a partir das informações obtidas pelo VMS sobre o uso dos recursos disponíveis na grade em um determinado momento. Com o cruzamento destas informações com as do plano de serviço adotado, o sistema lista e exibe para o usuário os recursos que atendem os requisitos definidos no plano de serviço. No caso ilustrado pela figura 5.6, o plano **Premium** foi utilizado como exemplo. A interface utilizada para a exibição destas informações é a mesma do Portal VCG e está representada pela figura 5.6.

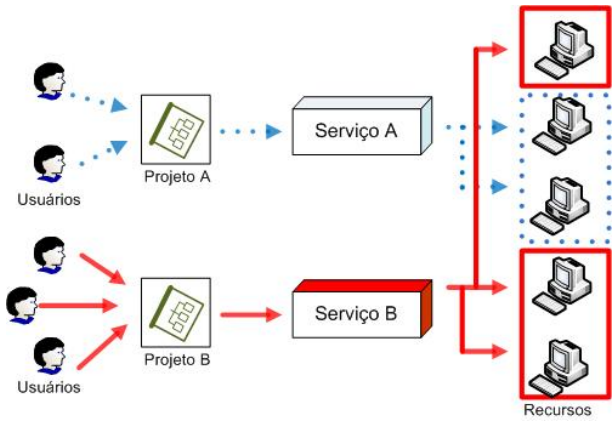

FIG. 5.5: Relacionamento usuários/projetos/serviços/recursos.

Ao analisar a figura 5.6, pode–se observar que os valores de CPU, memória e disco disponíveis estão dentro dos limites estabelecidos para o plano de serviço **Premium**. Isto implica em dizer que novos planos de serviço, com seus próprios requisitos de recursos, podem ser criados, uma vez que o sistema é capaz de identificar qual o plano de serviço utilizado e auxiliar à tomada de decisão, listando apenas os recursos que estejam de acordo com os requisitos do plano de serviço. Além disso, é possível que o usuário visualize os parâmetros de QoS de rede oferecidos por um recurso específico, conforme ilustrado pela figura 5.7.

5.4.2 AGENDAMENTO DE TAREFAS

O Pré–escalonamento de recursos e a sua interface também permitem que os usuários realizem agendamentos de recursos para que tarefas sejam executadas posteriormente. A figura 5.8 ilustra a interface de agendamento de recursos e tarefas.

Adicionalmente, é possível que um usuário apenas visualize as informações de um agendamento específico (figura 5.9), ou gerencie os agendamentos de recursos e tarefas (figura 5.10). No gerenciamento, o usuário pode cancelar um agendamento ou modificar seus dados, como data e hora em que a submissão deve ser feita, os recursos selecionados e os dados da tarefa.

Especificações do Plano de Serviço	
Plano de Serviço	Premium
CPU (mínimo dedicado)	80 %
memória (mínimo dedicado)	70 %
Disco (mínimo dedicado)	25 %
Prioridade (fila de processos)	Alta
Atraso máximo permitido	100 ms
Tarifa cobrada (acréscimo)	50 %

ATENÇÃO: A verificação do atraso da rede é feita a partir da máquina portalvcg.lncc.br.

Recursos que, no momento, atendem aos critérios do plano Premium				
Nome do Recurso	CPU Disponível	Memória Disponível	Disco Disponível	Atraso Rede
grade01.lncc.br	100 %	84.1 %	87 %	0.161 ms
grade02.lncc.br	100 %	80.2 %	88 %	0.057 ms
grade03.lncc.br	100 %	91.0 %	90 %	0.054 ms
grade04.lncc.br	100 %	84.7 %	90 %	0.092 ms
grade05.lncc.br	100 %	81.6 %	90 %	0.097 ms
grade06.lncc.br	100 %	81.1 %	91 %	0.124 ms
grade07.lncc.br	100 %	82.4 %	95 %	0.101 ms
grade08.lncc.br	100 %	84.6 %	95 %	0.077 ms
grade09.lncc.br	100 %	87.7 %	97 %	0.086 ms
grade10.lncc.br	100 %	85.3 %	87 %	0.049 ms
grade11.lncc.br	100 %	81.4 %	86 %	0.049 ms
grade12.lncc.br	100 %	84.4 %	86 %	0.071 ms
grade13.lncc.br	100 %	84.2 %	90 %	0.053 ms
grade14.lncc.br	99 %	83.3 %	87 %	0.172 ms
grade16.lncc.br	100 %	84.3 %	97 %	0.084 ms
grade17.lncc.br	100 %	81.7 %	92 %	0.078 ms
grade18.lncc.br	99 %	84.6 %	97 %	0.134 ms

FIG. 5.6: Pré–escalonamento de recursos aplicado no ambiente VCG.

No caso em que dois usuários selecionam o mesmo recurso para uma submissão em uma mesma data/hora, a preferência é daquele usuário que possui o plano de serviço mais privilegiado, enquanto que as aplicações de um usuário que possui um plano de serviço inferior serão postergadas. Uma vez que usuários que utilizem o mesmo plano de serviço solicitem o mesmo recurso em uma mesma data/hora, o sistema submete as tarefas por ordem de chegada.

A submissão das tarefas agendadas é feita através de um *PHP script* que executa uma vez por minuto, verificando se existe alguma tarefa programada para aquele instante.

Ocasionalmente, é possível que nenhum recurso na grade atenda aos critérios especificados pelo plano de serviço ao qual o usuário está vinculado. Neste caso, duas opções são oferecidas: a primeira é que o usuário tente mais tarde, caso seja realmente necessário que haja algum recurso com os requisitos especificados; a segunda é listar todos os recursos ativos da Grade para que o usuário escolha aqueles que sejam mais apropriados para a sua aplicação. No segundo caso, o custo da submissão é calculado seguindo as especificações do plano "Básico", ou seja, sem Qualidade de Serviço.

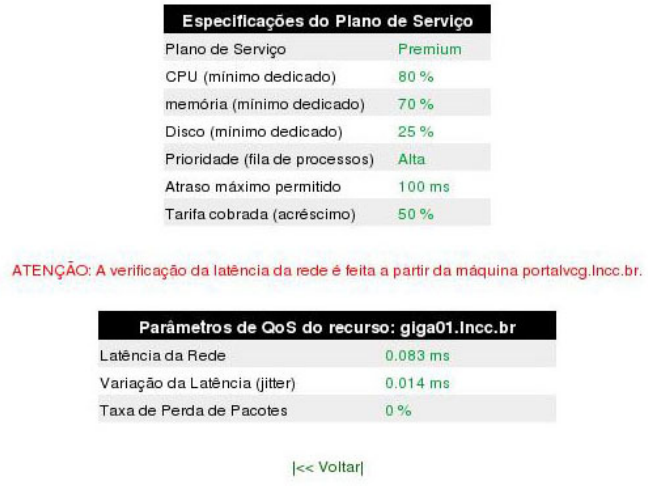

FIG. 5.7: Parâmetros de rede de um recurso da Grade.

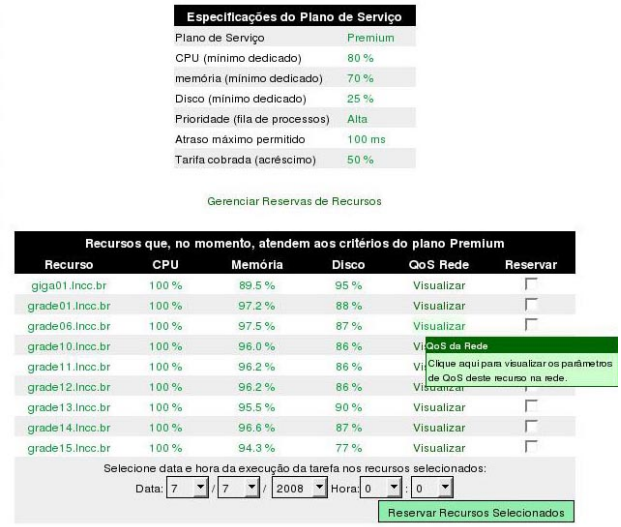

FIG. 5.8: Interface de agendamento de recursos e tarefas.

5.4.3 SUBMISSÃO DE TAREFAS COM *GRID–QOS*

Após o usuário definir quais os recursos que devem ser utilizados para a submissão de uma tarefa baseado no apoio à decisão oferecido, o sistema deve estar pronto para garantir que os parâmetros de *Grid–QoS* sejam, de fato, respeitados. Para que isso fosse possível, foram usados parâmetros que permitem informar ao *Globus Toolkit*, os valores que indicam os percentuais de CPU e memória que devem ser usados em cada submissão.

O cálculo dos valores de CPU e memória a serem utilizados na submissão são feitos a partir dos valores totais destes recursos e, para cada um deles, é calculado o valor real que representa o percentual estipulado pelos planos de serviço.

FIG. 5.9: Visualização das informações de um agendamento.

FIG. 5.10: Interface de gerenciamento de agendamentos.

Os parâmetros do comando *globusrun–ws* utilizados para a submissão com *Grid–QoS* são:

- ***-max-memory*** <***valor***>: Especifica o valor máximo de memória que deve ser usado para a execução de uma tarefa;
- ***-min-memory*** <***valor***>: Especifica o valor mínimo de memória que deve ser usado para que uma tarefa possa ser executada;
- ***-max-cpu*** <***valor***>: Especifica o valor máximo de CPU que deve ser usado para a execução de uma tarefa;
- ***-min-cpu*** <***valor***>: Especifica o valor mínimo de CPU que deve ser usado para que uma tarefa possa ser executada.

Uma submissão de tarefa que utilize no máximo 50% da memória de um recurso que possui um total de 2 GB de memória RAM pode ser feito da seguinte forma:

Exemplo 1: `# globusrun-ws -submit -f matriz.rsl -max-memory 1024`

O exemplo 1 submete uma aplicação especificada no arquivo matriz.rsl (figura 5.11) utilizando um valor máximo de memória RAM igual a 1024 MB do recurso grade10.lncc.br. O arquivo matriz.rsl define qual o executável deve ser utilizado, bem como os arquivos de saída e de erro, caso haja.

```
<job>
    <executable>matriz</executable>
    <directory>${GLOBUS_USER_HOME}</directory>
    <argument></argument>
    <argument></argument>
    <stdout>${GLOBUS_USER_HOME}/saida_matriz.txt</stdout>
    <stderr>${GLOBUS_USER_HOME}/erro_matriz.txt</stderr>
    <fileStageIn>
        <transfer>
            <sourceUrl>gsiftp://grade10.lncc.br:2811/tmp/matriz</sourceUrl>
            <destinationUrl>file:///${GLOBUS_USER_HOME}/matriz</destinationUrl>
        </transfer>
    </fileStageIn>
    <fileCleanUp>
        <deletion>
            <file>file:///${GLOBUS_USER_HOME}/matriz</file>
        </deletion>
    </fileCleanUp>
</job>
```

FIG. 5.11: Conteúdo do arquivo matriz.rsl.

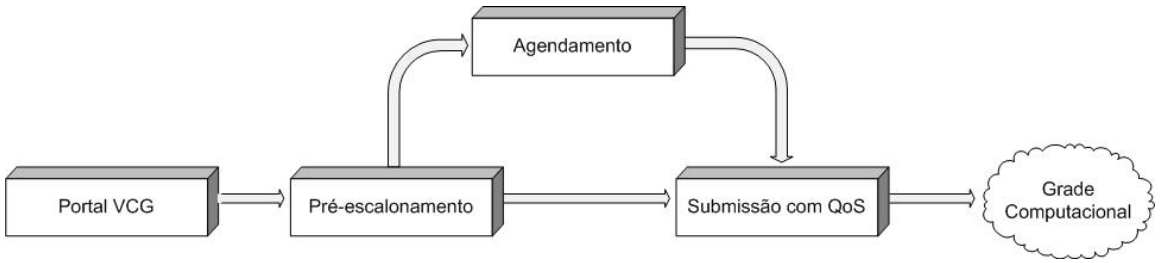

FIG. 5.12: Componentes do Sistema Proposto.

Após passar pelos componentes de pré–escalonamento de recursos, agendamento de tarefas e submissão com *Grid–QoS* (figura 5.12), o usuário obtém os resultados de suas tarefas submetidas.

5.5 DETALHES DO DESENVOLVIMENTO

Quanto ao desenvolvimento deste trabalho, alguns detalhes referentes à implementação da interface do sistema e dos testes realizados devem ser explicitados e esclarecidos:

- ***Interface do Sistema***: O desenvolvimento da interface do Sistema de *Grid–QoS* foi feito utilizando linguagem de programação PHP (versão 5.2.0), com servidor *web* Apache (versão 2) possibilitando assim a integração do Sistema com o Portal de Informações do Projeto VCG. Foi utilizada ainda a base de dados PostgreSQL

(versão 7.4.19) do VCG, alimentada pelo VMS (versão 2), para os processos de pré–escalonamento e agendamento de recursos e tarefas;

- ***Scripts para cálculo de custo de submissão***: Foi desenvolvido um *script* na liguagem PHP (versão 5.2.0) para implementar a fórmula de cálculo de custo de submissão;

- ***Realização de testes***: A realização de testes de submissão sem o uso de *Grid–QoS* foi feita através do Portal de Submissão do Projeto VCG. Os testes que utilizam *Grid–QoS* foram realizados através de comandos do Globus *Toolkit* (versão 4.0.5) em linha de comando.

5.6 RESUMO DO CAPÍTULO

Este capítulo teve como objetivo apresentar o sistema proposto, bem como as suas dependências, além de apresentar a implementação dos componentes, a criação dos planos de serviço e a classificação dos recursos da Grade Computacional utilizada.

O capítulo 6 apresenta os testes realizados com o sistema de *Grid–QoS* proposto e compara os resultados obtidos com os de uma Grade que não faz uso de Qualidade de Serviço.

6 RESULTADOS OBTIDOS

Com o objetivo de verificar o desempenho na execução de aplicações nos recursos da Grade utilizando o sistema de provisão de *Grid–QoS* e comparar com o desempenho de aplicações que não utilizam Qualidade de Serviço, foram executados testes de submissão com e sem o uso do Sistema Proposto. Para que a integridade dos testes não fosse comprometida, foi utilizado o mesmo programa de multiplicação de matrizes. Este programa foi escolhido por ser capaz de realizar operações com números inteiros (multiplicação das matrizes) e números do tipo *float* (cálculo das estatísticas). Para cada teste, a aplicação realizou 13320 iterações, com matrizes de dimensões 100x50 e 50x25. Estes valores foram definidos a fim de que fosse possível coletar dados consistentes o suficiente para se obter resultados estatísticos. Este tipo de teste para a avaliação de desempenho é conhecido como "*Application Benchmark*", uma vez que faz uso de um ambiente real para comparar dois ou mais sistemas.

Os testes foram distribuídos em seis diferentes categorias, cada uma delas com uma quantidade de recursos utilizados, sendo um de cada categoria, conforme indicado na tabela 5.2. Os recursos utilizados nos testes foram das categorias Q, O, N, M, I, H e G, distribuídos da seguintes forma: Teste 1: recursos O, N e I; Teste 2: recursos Q, M, H e G; Teste 3: recursos O e G; Teste 4: recursos N, M, I e H; Teste 5: recurso G; Teste 6: recurso Q. Foram feitos testes com os três planos de serviço especificados em cada um destes recursos, além de testes sem utilizar *Grid–QoS*. Os valores dos tempos consumidos foram obtidos através da média aritmética dos valores de três submissões realizadas em cada recurso. As tabelas e gráficos representam os tempos, em segundos, e os custos, em créditos, dos seis testes, comparando os três planos de serviço com o uso da Grade sem *Grid–QoS*.

É possível observar uma diferença no tempo de execução de, no máximo 10%, entre recursos que possuem configurações semelhantes e que diferem apenas no número de núcleos de processamento. Esta diferença ocorre porque não foram utilizadas aplicações que realizam processamento paralelo, devido ao fato de que os testes em série realizados levaram entre vinte e trinta segundos para serem concluídos. Cálculos nesta faixa de tempo não justificam o uso de aplicações MPI se considerarmos o atraso entre a comunicação

através da rede para a troca de mensagens, embora outras aplicações de teste possam ser utilizadas para avaliar o desempenho com o uso de processamento paralelo. Isto, no entanto, não impede que aplicações MPI de grande porte utilizem o Sistema Proposto de *Grid–QoS*.

Os recursos utilizados na realização dos testes apresentados fazem parte do núcleo da Grade Computacional do Laboratório de Computação Científica Distribuída (ComCiDis), situado no Laboratório Nacional de Computação Científica (LNCC).

- Teste comparativo 1: Este teste expressa os resultados obtidos em relação aos tempos e custos de submissões e compara os benefícios e restrições entre o Plano **Básico** e o uso da Grade sem *Grid–QoS* (tabelas 6.1 e 6.2 e figuras 6.1 e 6.2);

- Teste comparativo 2: Este teste exibe os resultados obtidos em relação aos tempos e custos de submissões e compara os benefícios e restrições entre o Plano **Master** e o uso da Grade sem *Grid–QoS* (tabelas 6.3 e 6.4 e figuras 6.3 e 6.4).

- Teste comparativo 3: Este conjunto de testes exibe os resultados obtidos em relação aos tempos e custos de submissões e compara os benefícios e restrições entre o Plano **Premium** e o uso da Grade sem *Grid–QoS* (tabelas 6.5 e 6.6 e figuras 6.5 e 6.6).

Após a comparação entre cada um dos planos de serviço e os mesmos testes sem o uso de *Grid–QoS*, foi feita uma comparação entre os tempos e custos totais entre os planos de serviço com o objetivo de apontar o ganho de desempenho adquirido pelos planos de serviço em troca de um maior custo na submissão das tarefas. Os valores totais de tempo e custo estão nas tabelas 6.7 e 6.8, respectivamente. As figuras 6.7 e 6.8 traduzem estes valores graficamente.

6.1 AVALIAÇÃO DOS RESULTADOS DOS TESTES DE DESEMPENHO

As tabelas e os gráficos dos resultados obtidos a partir dos testes realizados comprovam que o uso de uma Grade Computacional com Qualidade de Serviço beneficia os usuários que possuem um plano de serviço que oferece maiores vantagens do que outros planos de serviço inferiores, em troca de um custo mais elevado para as submissões.

O uso da Grade Computacional sem o uso de QoS se mostrou um pouco mais eficaz do que os Sistema com Grid–QoS apenas quando comparado com o plano **Básico**. Este resultado era esperado, uma vez que o plano **Básico**, que é o mais restrito dos planos

TAB. 6.1: Tabela comparativa de tempos: Plano Básico X Grade sem QoS.

	Tempo de Submissão nos Recursos Utilizados (segundos)							
	Q	O	N	M	I	H	G	Total
Teste 01 (sem Grid-QoS)		23,448	26,287		26,778			76,513
Teste 01 (com Grid-QoS)		23,894	25,747		27,428			77,069
Teste 02 (sem Grid-QoS)	21,620			27,198		29,578	31,896	110,292
Teste 02 (com Grid-QoS)	21,601			27,705		30,009	33,042	112,357
Teste 03 (sem Grid-QoS)		23,224					31,758	54,982
Teste 03 (com Grid-QoS)		22,893					32,427	55,320
Teste 04 (sem Grid-QoS)			25,950	26,092	26,087	28,703		106,832
Teste 04 (com Grid-QoS)			26,241	25,473	26,667	29,095		107,476
Teste 05 (sem Grid-QoS)							32,402	32,402
Teste 05 (com Grid-QoS)							33,813	33,813
Teste 06 (sem Grid-QoS)	21,458							21,458
Teste 06 (com Grid-QoS)	21,527							21,527

TAB. 6.2: Tabela comparativa de custos: Plano Básico X Grade sem QoS.

	Custo de Submissão							
	Q (3,8)	O (3,0)	N (2,8)	M (2,6)	I (1,8)	H (1,6)	G (1,4)	Total
Teste 01 (sem Grid-QoS)		23,448	26,287		26,778			76,513
Teste 01 (com Grid-QoS)		32,854	34,758		33,599			101,211
Teste 02 (sem Grid-QoS)	21,620			27,198		29,578	31,896	110,292
Teste 02 (com Grid-QoS)	31,861			36,709		36,010	33,824	138,404
Teste 03 (sem Grid-QoS)		23,224					31,723	54,947
Teste 03 (com Grid-QoS)		31,477					38,101	69,578
Teste 04 (sem Grid-QoS)			25,950	26,092	26,087	28,703		106,832
Teste 04 (com Grid-QoS)			35,425	33,751	32,667	34,914		136,757
Teste 05 (sem Grid-QoS)							32,402	32,402
Teste 05 (com Grid-QoS)							39,730	39,730
Teste 06 (sem Grid-QoS)	21,458							21,458
Teste 06 (com Grid-QoS)	31,752							31,752

propostos neste trabalho, limita o uso dos recursos computacionais disponíveis, enquanto que a Grade sem Qualidade de Serviço pode, eventualmente, utilizar toda a capacidade dos recursos, caso estejam disponíveis.

Com a análise dos resultados e recursos que possuem mais de um núcleo de processamento, foi possível observar também que o ganho de desempenho nestes recursos foi devido apenas à velocidade do *clock* dos seus processadores e à quantidade de memória disponível. Isto ocorreu porque os testes realizados não foram feitos utilizando processamento paralelo de forma a aproveitar os núcleos de processamento. No entanto, houve ganho significativo de desempenho nestes recursos quando existiu concorrência de processos, conforme os testes apresentados na seção 6.2.

A análise dos gráficos 6.1 e e 6.2 das tabelas 6.1 e 6.2 permitiu concluir que o ganho de desempenho do plano **Básico** foi baixo ou, algumas vezes, inexistente, se comparado com o uso da Grade sem QoS com a mesma aplicação. Isto ocorreu devido ao fato de

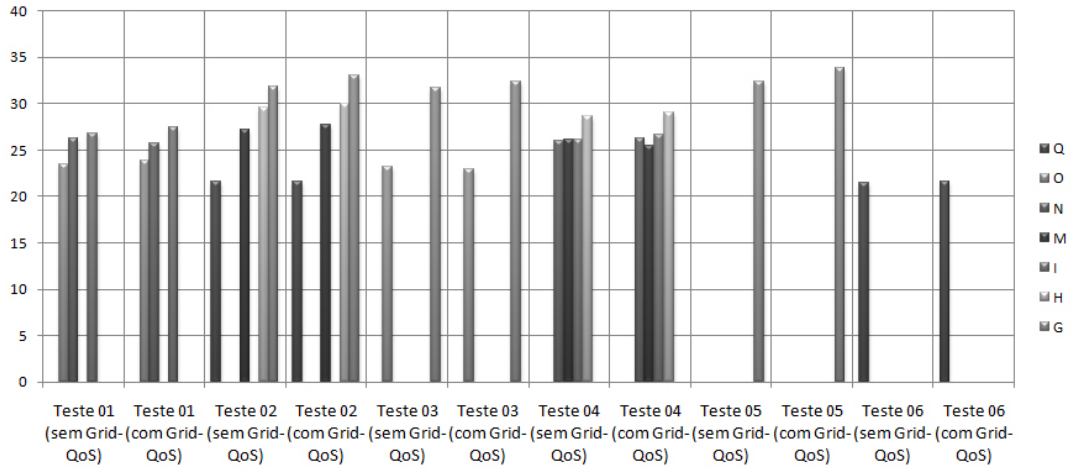

FIG. 6.1: Gráfico comparativo de tempos: Plano Básico X Grade sem QoS.

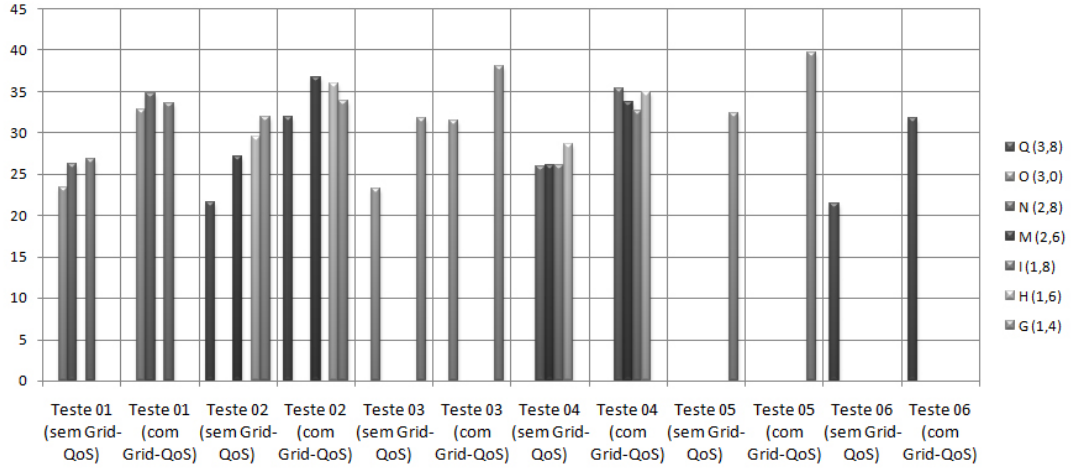

FIG. 6.2: Gráfico comparativo de custos: Plano Básico X Grade sem QoS.

que o plano **Básico** restringe o uso dos recursos conforme a especificação do plano de serviço mesmo que não existam outras aplicações concorrendo por aquele recurso. Por outro lado, em uma Grade sem QoS, é possível que o recurso esteja disponível em um percentual maior do que o especificado para o plano **Básico**.

Em contrapartida, os resultados obtidos com o uso dos planos de serviço **Master** (gráficos 6.3 e 6.4 e tabelas 6.3 e 6.4) e **Premium** (gráficos 6.5 e 6.6 e tabelas 6.5 e 6.6) demonstram que o tempo de submissão de aplicações que utilizaram um destes planos foram 8% e 6%, respectivamente, menores do que o tempo de submissão na Grade sem Qualidade de Serviço.

Ao observar o conjunto de informações obtidos a partir dos testes realizados, foi

TAB. 6.3: Tabela comparativa de tempos: Plano Master X Grade sem QoS.

	Tempo de Submissão nos Recursos Utilizados (segundos)							
	Q	O	N	M	I	H	G	Total
Teste 01 (sem Grid-QoS)		23,448	26,287		26,778			76,513
Teste 01 (com Grid-QoS)		19,576	24,329		24,528			68,433
Teste 02 (sem Grid-QoS)	21,620			27,198		29,578	31,896	110,292
Teste 02 (com Grid-QoS)	20,976			25,891		27,901	29,257	104,025
Teste 03 (sem Grid-QoS)		23,224					31,758	54,982
Teste 03 (com Grid-QoS)		20,012					28,723	48,735
Teste 04 (sem Grid-QoS)			25,950	26,092	26,087	28,703		106,832
Teste 04 (com Grid-QoS)			24,571	24,894	24,318	27,173		100,956
Teste 05 (sem Grid-QoS)							32,402	32,402
Teste 05 (com Grid-QoS)							29,193	29,193
Teste 06 (sem Grid-QoS)	21,458							21,458
Teste 06 (com Grid-QoS)	20,894							20,894

TAB. 6.4: Tabela comparativa de custos: Plano Master X Grade sem QoS.

	Custo de Submissão							
	Q (3,8)	O (3,0)	N (2,8)	M (2,6)	I (1,8)	H (1,6)	G (1,4)	Total
Teste 01 (sem Grid-QoS)		23,448	26,287		26,778			76,513
Teste 01 (com Grid-QoS)		35,061	39,244		36,178			110,483
Teste 02 (sem Grid-QoS)	21,620			27,198		29,578	31,896	110,292
Teste 02 (com Grid-QoS)	36,183			42,836		42,888	45,451	167,358
Teste 03 (sem Grid-QoS)		23,224					31,723	54,947
Teste 03 (com Grid-QoS)		32,519					40,930	73,449
Teste 04 (sem Grid-QoS)			25,950	26,092	26,087	28,703		106,832
Teste 04 (com Grid-QoS)			39,313	41,095	38,478	41,619		160,505
Teste 05 (sem Grid-QoS)							32,402	32,402
Teste 05 (com Grid-QoS)							41,606	41,606
Teste 06 (sem Grid-QoS)	21,458							21,458
Teste 06 (com Grid-QoS)	36,042							36,042

possível avaliar que, embora os usuários dos planos de serviço **Master** e **Premium** paguem mais caro por suas submissões, os mesmos recebem em troca um serviço de melhor qualidade, que é traduzido em um menor tempo de execução de suas tarefas.

6.2 AVALIAÇÃO DOS RESULTADOS DOS TESTES DE CONCORRÊNCIA

Os testes de desempenho usando o programa de multiplicação de matrizes permitiram a análise comparativa entre o ganho de velocidade de processamento entre os planos de serviço propostos e os respectivos aumentos de custo para cada submissão utilizando *Grid–QoS*.

Estes testes, no entanto, foram realizados em momentos em que os recursos estavam ociosos, o que não permitiu avaliar o desempenho destes planos de serviço concorrendo com outras aplicações.

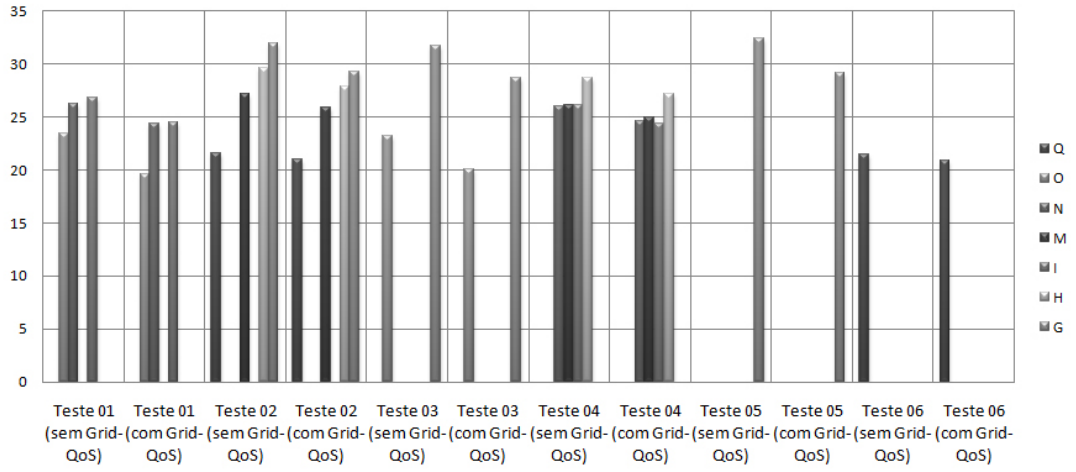

FIG. 6.3: Gráfico comparativo de tempos: Plano Master X Grade sem QoS.

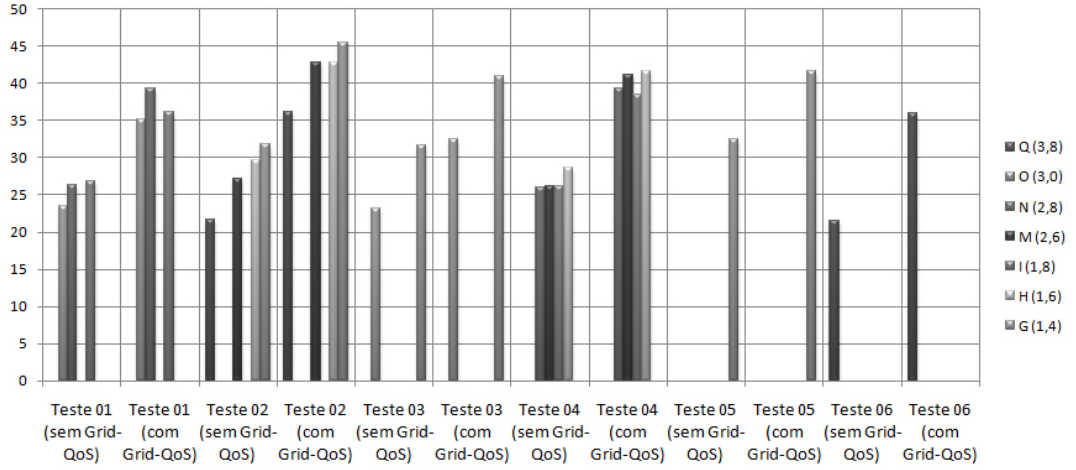

FIG. 6.4: Gráfico comparativo de custos: Plano Master X Grade sem QoS.

Com o objetivo de realizar a análise de desempenho do programa de multiplicação de matrizes concorrendo com outra aplicação, foi utilizado um exemplo de aplicação de Inteligência Artificial que tem por objetivo realizar o treinamento de redes neurais (RUSSEL e NORVIG, 1995) chamado *8–Puzzle*. Esta aplicação simula um jogo onde o principal objetivo é organizar as peças de forma a atingir um estado final pré–definido, conforme exemplificado pela figura 6.9.

O programa que simula o jogo *8–Puzzle* faz o cálculo de possíveis movimentos e os executa, armazenando cada movimento na forma de um nó em uma árvore. A figura 6.9 ilustra os estados, inicial e final, que foram considerados para testes.

Para a realização dos testes de concorrência, a aplicação *8–Puzzle* foi iniciada momen-

TAB. 6.5: Tabela comparativa de tempos: Plano Premium X Grade sem QoS.

	Tempo de Submissão nos Recursos Utilizados (segundos)							
	Q	O	N	M	I	H	G	Total
Teste 01 (sem Grid-QoS)		23,448	26,287		26,778			76,513
Teste 01 (com Grid-QoS)		17,846	22,292		21,278			61,416
Teste 02 (sem Grid-QoS)	21,620			27,198		29,578	31,896	110,292
Teste 02 (com Grid-QoS)	20,821			24,324		26,512	27,961	99,618
Teste 03 (sem Grid-QoS)		23,224					31,758	54,982
Teste 03 (com Grid-QoS)		18,883					27,431	46,314
Teste 04 (sem Grid-QoS)			25,950	26,092	26,087	28,703		106,832
Teste 04 (com Grid-QoS)			22,975	23,874	22,996	26,104		95,949
Teste 05 (sem Grid-QoS)							32,402	32,402
Teste 05 (com Grid-QoS)							28,690	28,690
Teste 06 (sem Grid-QoS)	21,458							21,458
Teste 06 (com Grid-QoS)	20,594							20,594

TAB. 6.6: Tabela comparativa de custos: Plano Premium X Grade sem QoS.

	Custo de Submissão							
	Q (3,8)	O (3,0)	N (2,8)	M (2,6)	I (1,8)	H (1,6)	G (1,4)	Total
Teste 01 (sem Grid-QoS)		23,448	26,287		26,778			76,513
Teste 01 (com Grid-QoS)		33,495	41,240		36,704			111,439
Teste 02 (sem Grid-QoS)	21,620			27,198		29,578	31,896	110,292
Teste 02 (com Grid-QoS)	41,121			44,391		45,070	46,834	177,416
Teste 03 (sem Grid-QoS)		23,224					31,723	54,947
Teste 03 (com Grid-QoS)		35,405					45,946	81,351
Teste 04 (sem Grid-QoS)			25,950	26,092	26,087	28,703		106,832
Teste 04 (com Grid-QoS)			42,503	43,570	39,668	44,376		170,117
Teste 05 (sem Grid-QoS)							32,402	32,402
Teste 05 (com Grid-QoS)							48,055	48,055
Teste 06 (sem Grid-QoS)	21,458							21,458
Teste 06 (com Grid-QoS)	40,673							40,673

tos antes da aplicação de multiplicação de matrizes e, desta forma, o teste obrigou que existisse a concorrência de processos nos recursos selecionados para os testes. Os testes de concorrência foram realizados seguindo a mesma distribuição de recursos especificada para os testes de desempenho.

As tabelas 6.9 e 6.10 representam, respectivamente, o tempo, em segundos, que foi necessário para a realização de cada um dos testes e a quantidade de créditos (custo total) que foi consumido em cada um dos testes. Os gráficos de tempo (figura 6.10) e de custo (figura 6.11) foram feitos a partir dos valores demonstrados nestas tabelas.

A análise dos resultados dos testes de concorrência representados pelas tabelas 6.9 e 6.10 e pelos gráficos 6.10) e 6.11, e a comparação destes resultados com os das tabelas 6.9 e 6.10 e gráficos 6.10) e 6.11, que exibem os resultados dos testes de desempenho sem concorrência de processos permitiu concluir que, em todos os planos de serviço, ocorreu aumento, tanto do tempo de processamento das tarefas nos recursos selecionados, como

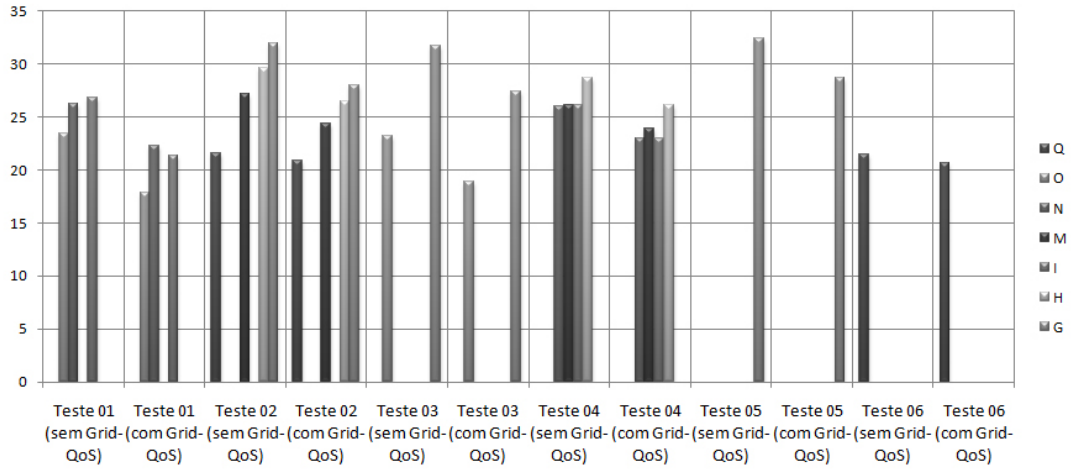

FIG. 6.5: Gráfico comparativo de tempos: Plano Premium X Grade sem QoS.

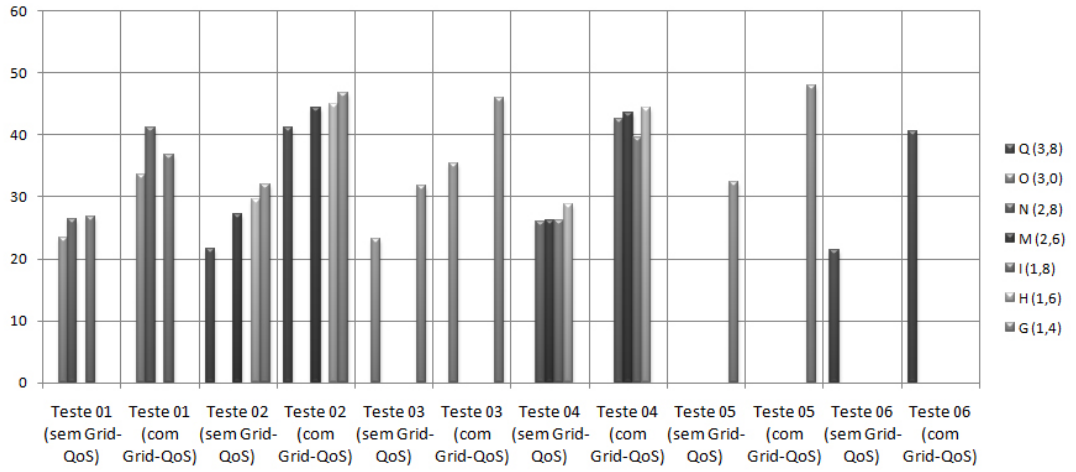

FIG. 6.6: Gráfico comparativo de custos: Plano Premium X Grade sem QoS.

do custo de submissão destas tarefas.

A avaliação dos resultados possibilitou observar ainda que, embora todos os planos de serviço tenham sofrido acréscimos no tempo de execução e nos custos, os planos **Master** e **Premium** tiveram menor degradação de desempenho, cerca de 5% e 4% em média, respectivamente, com o acréscimo do custo em cerca de 9% e 11%, respectivamente para estes planos.

TAB. 6.7: Tabela de tempos de execução.

	Tempo utilizado pelos planos de serviço em cada teste					
	Teste1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6
Sem QoS	76,513	110,292	54,982	106,832	32,402	21,458
QoS Básico	77,069	112,357	55,320	107,476	33,813	21,527
QoS Master	68,433	104,025	48,735	100,956	29,193	20,894
QoS Premium	61,416	99,618	46,314	95,949	28,690	20,594

TAB. 6.8: Tabela de consumo de créditos.

	Custo de submissão dos planos de serviço em cada teste					
	Teste1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6
Sem QoS	76,513	110,292	54,947	106,832	32,402	21,458
QoS Básico	101,211	138,404	69,578	136,757	39,730	31,752
QoS Master	110,483	167,358	73,449	160,505	41,606	36,042
QoS Premium	111,439	177,416	81,351	170,117	48,055	40,673

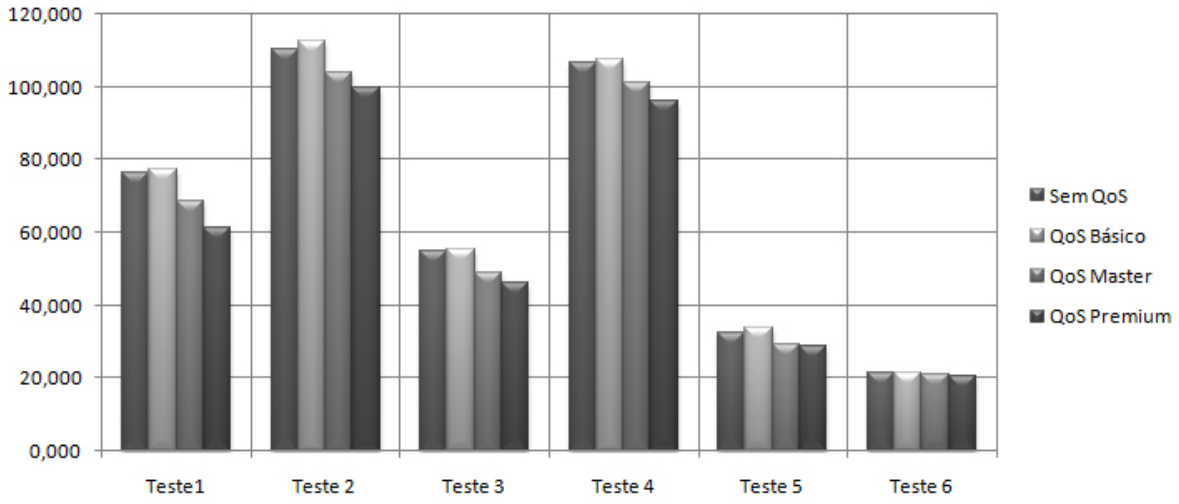

FIG. 6.7: Gráfico de tempos de execução.

TAB. 6.9: Tabela dos tempos utilizados pelos testes de concorrência.

	Tempo utilizado pelos planos de serviço em cada teste					
	Teste1	Teste2	Teste3	Teste4	Teste5	Teste6
Sem QoS	88,762	125,329	62,147	117,892	43,263	31,230
QoS Básico	87,186	125,940	61,884	118,245	43,032	31,194
QoS Master	72,398	109,588	54,020	106,771	33,275	25,126
QoS Premium	65,329	103,105	49,856	100,093	30,903	23,745

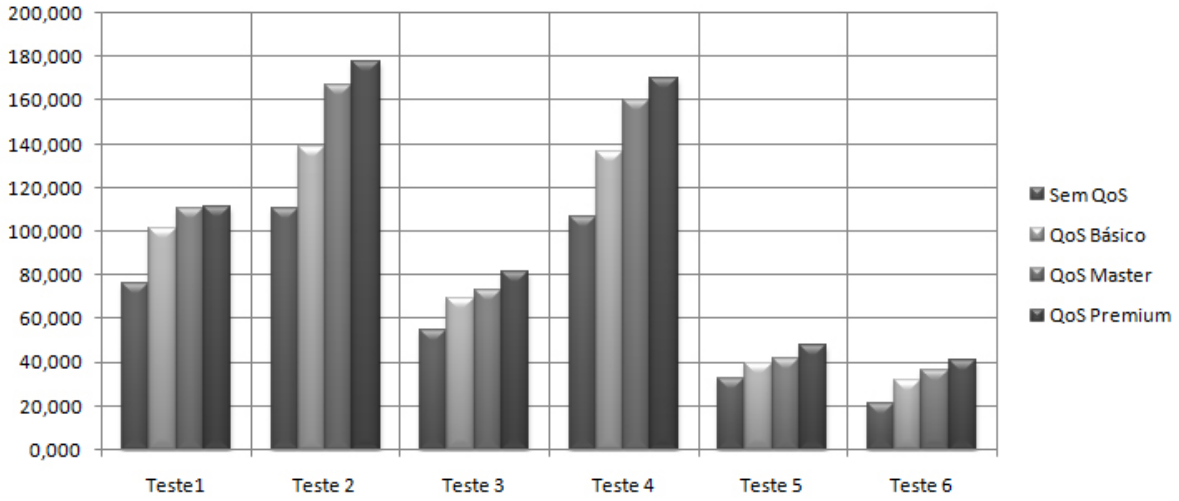

FIG. 6.8: Gráfico de consumo de créditos.

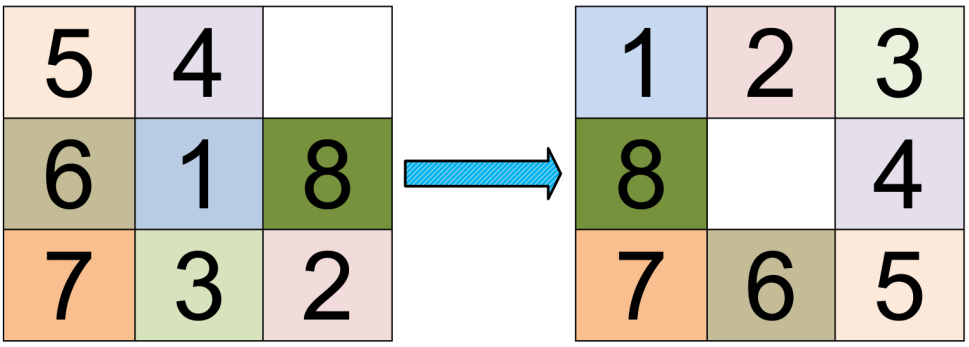

FIG. 6.9: Exemplo do quebra–cabeça *8–Puzzle*.

TAB. 6.10: Tabela de consumo de créditos nos testes de concorrência.

	Custo de submissão dos planos de serviço em cada teste					
	Teste1	Teste2	Teste3	Teste4	Teste5	Teste6
Sem QoS	88,762	125,329	62,147	117,892	43,263	31,230
QoS Básico	110,489	150,905	75,948	146,903	50,416	36,853
QoS Master	115,694	179,286	78,089	172,311	45,333	40,045
QoS Premium	117,247	184,382	85,930	181,852	51,307	44,327

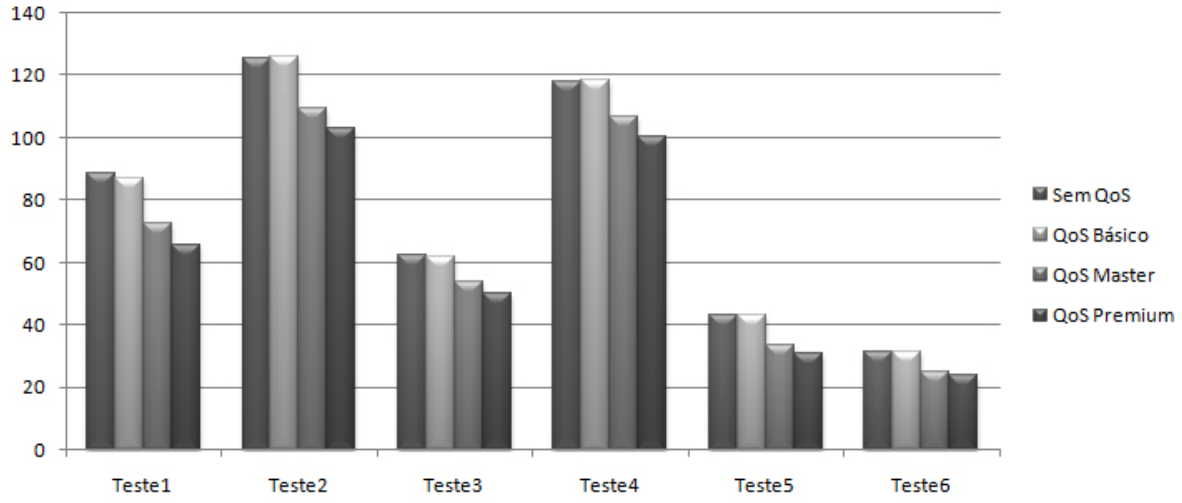

FIG. 6.10: Gráfico de tempos de execução dos testes de concorrência.

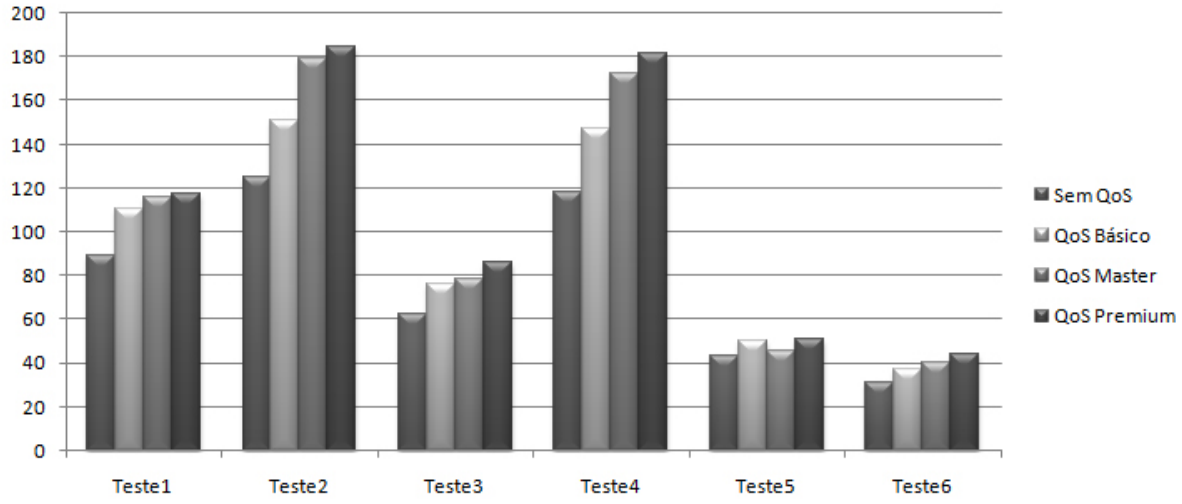

FIG. 6.11: Gráfico de consumo de créditos nos testes de concorrência.

7 CONSIDERAÇÕES FINAIS

Este capítulo tem o objetivo de explicitar as conclusões obtidas com a realização deste trabalho, assim como apresentar as contribuições alcançadas e os possíveis trabalhos futuros identificados que possam vir a ser feitos tomando o trabalho realizado como referência.

7.1 CONCLUSÕES

A principal conclusão obtida com a realização deste trabalho é que, ao observar o comportamento de submissões de tarefas que utilizam os três planos de serviço propostos, foi possível constatar que os usuários dos planos de serviço **Master** e **Premium** tiveram benefícios na execução de suas tarefas ocasionados pela maior prioridade de suas aplicações e pela possibilidade de executar aplicações com um elevado percentual da capacidade computacional dos recursos selecionados. Todavia, essas vantagens são concedidas ao cobrar uma taxa maior pelos serviços e recursos utilizados.

A realização deste trabalho tornou possível ainda, concluir que não é possível prover melhora de serviço, seja ela de qualquer espécie ou em qualquer área do conhecimento, sem que haja uma cobrança efetiva e coerente para os diferentes serviços prestados. Por este motivo, foi necessário modificar o modelo de Economia do Projeto VCG, de forma a adequá–lo para que pudesse atender a esse requisito de Qualidade de Serviço.

Para o cálculo do custo de submissão nos recursos da Grade foi desenvolvida uma fórmula de cobrança em função dos planos de serviço e dos tipos de recurso utilizados. Esta fórmula é aplicada em uma Grade comunitária que funciona de forma a incentivar a inserção de novos recursos na Grade, bem como na utilização de recursos dos parceiros associados.

Com a realização dos testes de submissão, observou–se que recursos com mais de um núcleo só oferecem vantagens consideráveis em termos de melhoria de desempenho, se as aplicações utilizadas nestes recursos forem programadas de forma a explorar técnicas de multiprocessamento; caso contrário, o ganho de desempenho pode ser desprezível, ou até mesmo não existir.

De maneira geral, os objetivos principais do trabalho foram alcançados, uma vez que

foi desenvolvido um sistema capaz de prover Qualidade de Serviço em Grades Computacionais baseado em Acordos em Nível de Serviço, com uma interface *web* transparente para os usuários. O sistema desenvolvido encontra–se parcialmente integrado como uma funcionalidade do Projeto *Virtual Community Grid* e encontra–se em fase de testes.

7.2 CONTRIBUIÇÕES ALCANÇADAS

Com a conclusão deste trabalho e com a realização de testes e obtenção de resultados, foram alcançadas as seguintes contribuições:

- Criação de um sistema capaz de oferecer Qualidade de Serviço em Grades Computacionais baseado em Acordos em Nível de Serviço, garantindo que os requisitos especificados nestes acordos sejam oferecidos e utilizados, bem como quanto às questões de interface entre os usuários e o sistema;
- Categorização de três planos de serviço e classificação dos recursos disponíveis na Grade, de forma a permitir a inclusão futura de novas categorias, caso necessário;
- Aperfeiçoamento de um modelo de Economia baseado na troca de créditos que realiza a cobrança de forma justa de acordo com o tipo de serviço e dos recursos utilizados, onde o uso de recursos com maior capacidade de processamento e uma maior prioridade na execução de aplicações refletem em um custo de submissão mais elevado e na redução do tempo de execução de tarefas;
- Artigo "*Pré–escalonamento com QoS em Grids Computacionais utilizando Economia de Créditos e Acordos em Nível de Serviço*" submetido e aceito no ***Workshop* de Computação *Grid* e Aplicações** (WCGA 08) em 31 de maio de 2008;
- Integração parcial e testes do Sistema de *Grid–QoS* desenvolvido na forma de uma funcionalidade do Projeto *Virtual Community Grid*.

7.3 TRABALHOS FUTUROS

O fato de que os objetivos iniciais estabelecidos para este trabalho tenham sido alcançados e apresentados na forma de contribuições para a comunidade científica não significa que o modelo apresentado aqui não possa ser aperfeiçoado, ou que novas propostas neste

contexto não possam ser apresentadas. Seguindo esta lógica, os itens a seguir foram identificados como possíveis trabalhos futuros:

- Implementar o Sistema, em sua totalidade, em um ambiente de Grade Comunitária, estimulando o uso pelos usuários associados e realizar análises de desempenho, eficácia e satisfação, com o objetivo de melhorar os serviços oferecidos;

- Adicionar ao modelo existente, o uso de indicadores de desempenho, confiabilidade e disponibilidade dos recursos da Grade. Alguns resultados utilizando estes indicadores já foram obtidos e apresentados no artigo "*A Grid-QoS Decision Support System using Service Level Agreements*" submetido no **CCGrid 09 – Ninth IEEE International Symposium on Cluster Computing and the Grid**;

- Definir planos de serviço voltados para aplicações específicas, baseando–se nos indicadores de desempenho, confiabilidade, disponibilidade e os parâmetros de *Grid–QoS* apresentados neste trabalho;

- Transformar o Sistema de *Grid–QoS* em um serviço de Grade, para que sua integração com *middlewares* de Grade seja realizada de forma mais simples e efetiva.

8 REFERÊNCIAS BIBLIOGRÁFICAS

AL-ALI, R., VON LASZEWSKI, G., AMIN, K., AMIN, K., HATEGAN, M., RANA, O., WALKER, D., e ZALUZEC, N. **Qos support for high-performance scientific grid applications**. Em *CCGRID '04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid*, págs. 134–143, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7803-8430-X.

AL-ALI, R. J., AMIN, K., LASZEWSKI, G. V., F, O., WALKER, D. W., HATEGAN, M., e ZALUZEC, N. **Analysis and provision of qos for distributed grid applications**. *Journal of Grid Computing*, 2:163–182, 2004.

ANDRADE, N., COSTA, L., GERMÓGLIO, G., e CIRNE, W. **Peer-to-peer grid computing with the ourgrid community**. Em *in 23rd Brazilian Symposium on Computer Networks (SBRC 2005) - 4th Special Tools Session*, 2005.

BANDINI, M., MURY, A. R., SCHULZE, B., e SALLES, R. **Pré–escalonamento com qos em grids computacionais utilizando economia de créditos e acordos em nível de serviço**. Em *VI Workshop on Grid Computing and Applications*, 2008.

BURCHARD, L.-O., LINNERT, B., HEINE, F., HOVESTADT, M., KAO, O., e KELLER, A. **A quality-of-service architecture for future grid computing applications**. 2005. URL `http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/`.

CIRNE, W., BRASILEIRO, F., ANDRADE, N., COSTA, L., ANDRADE, A., NOVAES, R., e MOWBRAY, M. **Labs of the world, unite!!!** *Journal of Grid Computing*, 4 (3):225–246, 2006. URL `http://dx.doi.org/10.1007/s10723-006-9040-x`.

CISCO SYSTEMS, I. **Cluster computing**. 2004. URL `http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns500`.

CONDOR. **Condor high throughput computing**, 2008. URL `http://www.cs.wisc.edu/condor/`.

EPCC. **Apples: Application–level scheduling system**, 2003. URL `http://www.epcc.ed.ac.uk`.

FERREIRA, L., BERSTIS, V., ARMSTRONG, J., KENDZIERSKI, M., NEUKOETTER, A., TAKAGI, M., BING-WO, R., AMIR, A., MURAKAWA, R., HERNANDEZ, O., MAGOWAN, J., e BIEBERSTEIN, N. ***Introduction to Grid Computing with Globus***. ibm.com/redbooks, 2003.

FOSTER, I. **Globus toolkit version 4: Software for service-oriented systems**. 2005. URL `http://www.globus.org/alliance/publications/papers.php#gt4overview`.

FOSTER, I., KESSELMAN, C., NICK, J., e TUECKE, S. **The physiology of the grid: An open grid services architecture for distributed systems integration**, 2002. URL citeseer.ist.psu.edu/foster02physiology.html.

FOSTER, I. **What is the grid? a three point checklist**, 2002. URL www-fp.mcs.anl.gov/ foster/Articles/WhatIsTheGrid.pdf.

FOSTER, I., KESSELMAN, C., e TUECKE, S. **The anatomy of the Grid: Enabling scalable virtual organizations**. *Lecture Notes in Computer Science*, 2150:1–25, 2001. URL citeseer.ist.psu.edu/foster01anatomy.html.

GOMES, A. T. A. **Um framework para provisão de qos em ambientes genéricos de processamento e comunicação.**, 1999.

GRIMSHAW, A. S., NATRAJAN, A., HUMPHREY, M. A., LEWIS, M. J., NGUYEN-TUONG, A., KARPOVICH, J. F., e ANDADAM J. FERRAR, M. M. M. **From legion to avaki: The persistence of vision**, 2006. URL http://www.stormingmedia.us/48/4807/A480744.html.

ISO. **International organization for standardization/international eletrotechnical commitee. "qos: Basic framework". draft international standard (dis) 9309**, 1995.

KATCHABAW, M. J., LUTFIYYA, H. L., e BAUER, M. A. **A quality of service management testbed**. Em *SMW '98: Proceedings of the IEEE Third International Workshop on Systems Management*, pág. 57, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-8476-3.

LI, M. e BAKER, M. ***The grid core technologies***. John Wiley & Sons, 2005. ISBN 0-47009-417-6.

MURY, A. R., SCHULZE, B., e GOMES, A. T. A. **Towards profile–based task management in grids**. Em *LAGRID 08: Proceedings of the 2nd International Latin American Grid Workshop*, págs. 19–24, Campo Grande, MS, Brazil, 2008. LNCC - Laboratório Nacional de Computação Científica. ISBN 978-85-99961-10-0.

NIMROD/G, T. N. P. **Nimrod: Tools for distributed parametric modelling**, 2005. URL http://www.csse.monash.edu.au/ davida/nimrod/.

ROY, A. J. ***End-to-end quality of service for high-end applications***. Tese de Doutorado, 2001. Adviser-Ian Foster.

RUSSEL, S. e NORVIG, P. ***Artificial intelligence: A Modern Approach***. Prentice-Hall, 1995.

SAHAI, A., GRAUPNER, S., MACHIRAJU, V., e VAN MOORSEL, A. **Specifying and monitoring guarantees in commercial grids through sla**. 2002. URL citeseer.ist.psu.edu/sahai02specifying.html.

SCHULZE, B. **Workgroup proposal: (vcg - virtual community grid)**, 2006. URL `https://vcg.lncc.br`.

SILBERSCHATZ, A., GALVIN, P., e GAGNE, G. ***Applied Operating System and Concepts***. John Wiley & Sons, 2001. ISBN 0-471-36508-4.

STARDUST.COM. **Qos protocols and architectures – a white paper**. 1999. URL `www.cs.ucsb.edu/ almeroth/classes/S01.ECI/qos.pdf.gz`.

TANENBAUM, A. S. ***Redes de Computadores***. Elsevier, Rio de Janeiro, trad. 4 edition, 2003.

TANENBAUM, A. S. ***Sistemas Distribuídos: Princípios e Paradigmas***. Prentice-Hall, 2nd edition, 2007.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo