UNILASALLE

CENTRO UNIVERSITÁRIO LA SALLE

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

INTEGRAÇÃO DO SISTEMA APPMAN DE GERENCIAMENTO DE APLICAÇÕES PARA AMBIENTE DE GRADE COM DIFERENTES SISTEMAS DE GERENCIAMENTO DE RECURSOS

TONISMAR RÉGIS BERNARDO

Canoas, maio de 2008

TONISMAR RÉGIS BERNARDO

INTEGRAÇÃO DO SISTEMA APPMAN DE GERENCIAMENTO DE APLICAÇÕES PARA AMBIENTE DE GRADE COM DIFERENTES SISTEMAS DE GERENCIAMENTO DE RECURSOS

Trabalho de conclusão apresentado à banca examinadora do curso de Ciência da Computação do Centro Universitário La Salle - Unilasalle, como exigência parcial para obtenção do grau de Bacharel em Ciência da Computação, sob orientação da Profa. DSC. Patrícia Kayser Vargas Mangan.

Canoas, maio de 2008

TERMO DE APROVAÇÃO

TONISMAR RÉGIS BERNARDO

INTEGRAÇÃO DO SISTEMA APPMAN DE GERENCIAMENTO DE APLICAÇÕES PARA AMBIENTE DE GRADE COM DIFERENTES SISTEMAS DE GERENCIAMENTO DE RECURSOS

Trabalho de conclusão aprovado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Centro Universitário La Salle - Unilasalle, pela seguinte banca examinadora:

Prof. Me. Marcos Ennes Barreto
Centro Universitário La Salle - Unilasalle

Prof. Me. Mozart Lemos de Siqueira
Centro Universitário La Salle - Unilasalle

Prof. Dsc. Patrícia Kayser Vargas Mangan
Centro Universitário La Salle - Unilasalle

Canoas, maio de 2008

AGRADECIMENTOS

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

RMS	*Resource Management System*
GRAND	*Grid Robust Application Deployment*
DRMAA	*Distributed Resource Management Application*
API	*Application Program Interface*
ISAM	Infra-estrutura de Suporte às Aplicações Móveis
EXEHDA	*Execution Environment for Highly Distributed Applications*
LDAP	*Lightweight Directory Access Protocol*
UML	*Unified Modeling Language*
OGF	*Open Grid Forum*
GT	*Globus Toolkit*
PBS	*Portable Batch System*
MDS	*Metacomputing Directory Services*
GRAM	*Globus Resource Allocation Manager*
OGSA	*Open Grid Services Architecture*
WSRF	*Web Services Resource Framework*
AP	*Application Manager*
SM	*Submission Manager*
TM	*Task Manager*

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Grades computacionais (computational grid) é uma das formas mais recentes de ambiente para processamento geograficamente distribuído, que conta com uma grande infra-estrutura de redes e pode ser empregada em troca de programas, dados e serviços. Segundo Dantas (1), pode-se dizer, também, que Computação em Grade (Grid Computing) representa uma forma estendida dos serviços Web permitindo que recursos computacionais possam ser compartilhados. Podemos definir grades como uma plataforma computacional heterogênea distribuída geograficamente fornecendo serviços e recursos às organizações participantes da plataforma (1).

Um sistema de gerenciamento de recursos (*Resource Management System* - RMS) é a parte central de um sistema distribuído fornecendo um mecanismo de enfileiramento de tarefas, políticas de escalonamento, esquemas de prioridades e monitoramento de recursos proporcionando controles adicionais sobre inicialização, escalonamento e execução de tarefas. Também coordena a distribuição dessas tarefas entre as diferentes máquinas em uma rede (2; 3; 4). Devido a heterogeneidade das grades alguns problemas são apresentados, tais como, a alocação dos nós para um grande número de tarefas, gerenciamento de dados e sobrecarga em nós de submissão. O modelo de gerenciamento de aplicações denominado GRAND (*Grid Robust Application Deployment*) (4) visa permitir um particionamento flexível e utilizar uma hierarquia de gerenciadores que realizam a submissão das tarefas. Baseado nesse modelo um protótipo, AppMan (*Application Manager*) (5), foi implementado objetivando garantir o escalonamento das tarefas bem como a autorização e autenticação para tarefas executadas. Ele foi avaliado apresentando bons resultados referente ao gerenciamento de dados e serviços. Esse protótipo consiste em um gerenciador de aplicações que dispara e controla cada aplicação nos nós baseando-se nas informações indicadas pelo gerenciador de submissão que tem como função, além da já citada, criar e monitorar os gerenciadores de tarefas. Os gerenciadores de tarefas são responsáveis pela comunicação com o escalonador de um determinado domínio

garantindo a execução remota e ordem das tarefas de acordo com a dependência de dados (4).

O modelo GRAND permitiria que qualquer sistema gerenciador de recurso fosse usado nos nós, porém o AppMan funciona unicamente com seu próprio sistema de gerenciamento de recursos. Nenhum estudo mais detalhado foi realizado até o momento de como possibilitar que o protótipo suporte a integração com diferentes RMSs.

Uma interface de aplicação (*Application Program Interface* - API) denominada DRMAA (*Distributed Resource Management Application* API) foi desenvolvida com o objetivo de facilitar a integração das aplicações para diferentes RMSs (6).

Este trabalho pretende avaliar se a especificação DRMAA atende as necessidades do AppMan bem como realizar a integração do AppMan ao menos com um RMS.

1.1 Motivação

Gerenciar aplicações consiste em preparar, submeter e monitorar o progresso de execução de todas as tarefas as quais compõem uma aplicação. Cada nó da grade possui seu gerenciador de recursos, os quais podem possuir inúmeros atributos. No modelo estudado foram considerados como atributos o controle exclusivo e a dependência de trabalhos (4). Verificando esses atributos, determinadas ações podem ser agendadas para as aplicações. O GRAND respeita as políticas bem como as especificações administrativas do domínio escolhido considerando que a maioria dos clusters acadêmicos ou redes locais estarão prontos para serem integrados em uma grade, já possuindo usuários locais, submissão local de tarefas e um administrador de sistema, assim sendo o menos intrusivo possível nos diferentes RMSs existentes em uma grade (4).

A especificação DRMAA tem por objetivo facilitar a interface entre aplicações de RMSs com aplicações de diferentes desenvolvedores abstraindo as relações fundamentais da tarefa do RMS provendo um modelo fácil de usar (easy-to-use) para desenvolvedores tanto de aplicações como de RMSs encorajando, desse modo, adoção dos mesmos (6).

Como já dito anteriormente, qualquer RMS pode ser usado nos nós da grade de acordo com o modelo GRAND, porém o protótipo AppMan funciona apenas com seu próprio gerenciador de recursos (4).

Considerando as vantagens do DRMAA (7) e as questões citadas, acredita-se motivador o estudo e desenvolvimento de uma integração com ao menos um RMS. Maiores detalhes serão esclarecidos no capítulo seguinte.

1.2 Objetivo

1.2.1 Gerais

Permitir que o AppMan exporte tarefas para nós da grade gerenciadas por diferentes RMSs e através disso proporcionar a sua integração e conseqüente utilização de recursos em outras instituições científicas.

1.2.2 Específicos

- Verificar se a especificação DRMAA atende todas necessidades esperadas pelo AppMan assim como confirmar a aptidão do AppMan na integração com a DRMAA;
- Integrar AppMan com pelo menos um RMS;
- Avaliar o sobrecusto (overhead) da solução implementada através da verificação do desempenho da integração comparando com o escalonador atual;

1.3 Metodologia de Pesquisa

Com o propósito de verificar a viabilidade da exportação de tarefas do AppMan para RMS diferentes, foi realizado um estudo da especificação DRMAA para o desenvolvimento desta integração. Também foi feito um estudo aprofundado no modelo GRAND onde foi desenvolvido o protótipo, bem como da implementação atual do AppMan.

O desenvolvimento feito na liguagem Java que foi também a linguagem desenvolvida o AppMan facilitando a portabilidade. A métrica de avaliação usada na verificação das vantagens de um RMS diferente comparado com o escalonador do próprio AppMan foi a escalabilidade, tendo como base um estudo teórico aprofundado incluindo os itens citados na bibliografia presente neste trabalho.

Outros trabalhos que proporcionam integração entre diferentes RMSs forão estudados para averiguaração das soluções adotadas.

Também foram executados todos procedimentos de instalacao do ambiente EXEHDA/ISAM e do AppMan. Alguns problemas foram encontrados com o servidor Lightweight Directory Access Protocol (LDAP) necessário para o funcionamento do EXEHDA. Uma engenharia reversa das classes do projeto AppMan gerando o diagrama UML foi feita para facilitar o estudo da integracao com a DRMAA.

Todo acompanhamento foi feito em companhia do professor orientador através de reuniões periódicas previamente estabelecidas conforme o cronograma previsto (colocar o cronograma?).

1.4 Estrutura do Trabalho

A estrutura deste trabalho é dividida em Xs capítulos e esta Introdução. Os capítulos são estruturados da seguinte forma:

CAPÍTULO 2: *Gerenciamento de Aplicações em Grade.* Este capítulo apresenta uma descrição dos conceitos de Grades Computacionais e também de Sistemas Gerenciadores de Recursos.

CAPÍTULO 3: *O Modelo GRAND.* O capítulo 3 explica as características do modelo GRAND e também o protótipo AppMan desenvolvido com base neste modelo.

CAPÍTULO 4: *A Especificação DRMAA.* Neste capítulo será explanado os conceitos da especificação DRMAA, experiências de implementações e suas vantagens.

CAPÍTULO 5: *Implementação.* O capítulo 5 explica a metodologia mais detalhada, os motvios da escolha das tecnologias e como foi efetivamente implementado o trabalho.

CAPÍTULO 6: *Resultados Experimentais.* O presente capítulo explicará de que forma foram feitos os testes e avaliações dos resultados.

CAPÍTULO 7: *Conclusão.* Finalmente serão apresentadas as conclusões e propostos trabalhos futuros.

2 GERENCIAMENTO DE APLICAÇÕES EM GRADE

A idéia de computação em grade para processamento de aplicações em paralelo veio por consequência dos inúmeros avanços no desempenho de redes de computadores.

Atualmente o uso dessas redes tem aumentado exponencialmente. Muitas dessas redes são distribuídas de forma geograficamente separadas precisando de uma complexa infra-estrutura de software e hardware para gerenciá-las e conectá-las. Dentre as diversas soluções existentes a grade computacional (grid computing) possui características que viabiliza essa conexão.

O Open Grid Forum (OGF) uma comunidade fórum com milhares de indivíduos representando mais de 400 organizações em mais de 50 países criou e documentou (8) especificações técnicas e experiências de usuários. O OGF definiu grades computacionais como um ambiente persistente o qual habilita aplicações para integrar instrumentos, disponibilizar informações em locações difusas. Desde lá esta não é a única e precisa definição para o conceito de grades. Foster (9) define um sistema em grade propondo um *checklist* de três pontos.

1. coordenar recursos os quais não são direcionados para um controle central.
2. usar protocolos e interfaces padronizados, abertos para propósitos gerais.
3. oferecer QoS (qualidade de serviço) não triviais tais como: autenticação, escalonamento de tarefas, disponibilidade.

Uma definição formal do que um sistema em grade pode prover foi definido por Foster et al. em (10). Focando na sua semântica, mostrando que grades não são apenas uma modificação de um sistema distribuído convencional. Podem apresentar recursos heterogênios como sensores e detectores e não apenas nós computacionais. Abaixo uma lista de aspectos que evidenciam uma grade computacional (11):

- heterogeneidade
- alta dispersão geográfica
- compartilhamento (não pode ser dedicado a uma única aplicação)
- múltiplos domínios administrativos (recursos de várias instituições)
- controle distribuído

A grade deve estar preparada para lidar com todo o dinamismo e variabilidade, procurando obter a melhor performance possível adaptando-se ao cenário no momento.

2.1 Gerenciamento de Recursos

Devido à grande escala, ampla distribuição e existência de múltiplos domínios administrativos, a construção de um escalonador de recursos para grades é praticamente inviável, até porque, convencer os administradores dos recursos que compõem a grade abrirem mão do controle dos seus recursos não é uma tarefa nada fácil. Escalonadores têm como características receber solicitações de vários usuários, arbitrando, portanto, entre os usuários, o uso dos recursos controlados.

Casavant (12) considera escalonar como um problema de gerenciamento de recursos. Basicamente um mecanismo ou uma política usada para, eficientemente e efetivamente, gerenciar o acesso e uso de um determinado recurso. Porém, de acordo com o OGF's (8), escalonamento é o processo de ordenar tarefas sobre os recursos computacionais e ordenar a comunicação entre as tarefas, assim sendo, ambas aplicações e sistemas devem ser escalonadas.

O gerenciamento de recursos de um sistema centralizado possui informação completa e atualizada do status dos recursos gerenciados. Este difere do sistema distribuído, o qual não tem conhecimento global de recursos dificultando assim, o gerenciamento. O ambiente em grade introduz cinco desafios para o problema de gerenciamento de recursos em ambientes distribuídos (13):

1. autonomia: os recursos são, tipicamente propriedades e operados por diferentes organizações em diferentes domínios administrativos.
2. heterogeneidade: diferentes lugares podem usar diferentes sistemas de gerenciamento de recursos (RMS - *resource management system*).
3. extender as políticas: suporte no desenvolvimento de nova aplicação de mecanismos de gerência num domínio específico, sem necessitar de mudanças no código instalado nos domínios participantes.

4. co-alocação: algumas aplicações tem necessidades de recursos os quais só podem ser satisfeitos apenas usando recursos simultâneos com vários domínios.

5. controle online: RMSs precisam suportar negociações para adaptar necessidades de aplicações para recursos disponíveis.

Sistemas computacionais, na sua grande parte, falham ao tratar dois problemas (4):

- gerenciamento e controle de um grande número de tarefas;
- o balanceamento da carga da máquina de submissão e do tráfego da rede.

Uma distribuição dinâmica de dados e tarefas em uma hierarquia de gerenciadores poderia ajudar o gerenciamento de aplicações. O modelo GRAND baseado na submissão e controle particionados e hierárquicos foi proposto em (4). Uma melhor explanação deste modelo será dada no próximo capítulo.

Grande parte das pesquisas sobre escalonamento de tarefas em grades seguem uma organização hierárquica ou centralizada tais como: Globus (14), Condor (2), ISAM (15) e PBS (16).

Globus (17), um dos projetos mais referenciados na literatura, tem como principal software o Globus Toolkit (GT). Como o nome indica, o GT não é uma solução completa e sim um conjunto de serviços que podem ser combinados para a construção de um *middleware* de grade. O Globus (14) tem seu modelo de escalonamento centralizado. Não fornece suporte nativo as políticas de escalonamento mas permite que gerenciadores externos adicionem esta capacidade. O Globus (GT versão 3) oferece serviços de informação através de uma rede hierárquica chamada *Metacomputing Directory Services* (MDS) (18). O gerenciamento de cada recurso é feito por uma instância do *Globus Resource Allocation Manager* (GRAM) (19). GRAM é o responsável por instanciar, monitorar e reportar o estado das tarefas alocadas para o recurso. A GT4 (20) disponibiliza tais serviços em uma arquitetura baseada em *Web Services* a *Open Grid Services Architecture* (OGSA) junto com *Web Services Resource Framework* (WSRF). A GT4 é focada na qualidade, robustez, facilidade de uso e documentação.

Um dos gerenciadores que podem ser integrados com o Globus é o PBS (16). O PBS é um RMS que tem por propósito prover controle adicionais sobre a execução de tarefas em batch. O sistema permite um domínio definir e implementar políticas tais como os tipos de recursos e como esses recursos podem ser usados por diferentes tarefas.

Outro projeto bastante importante é o Condor (2), cujos trabalhos originais são voltados para redes de computadores, mas atualmente também contemplam *clusters*

e grades. O Condor trabalha com a descoberta de recursos ociosos dentro de uma rede alocando esses recursos para execução das tarefas. Condor possui uma arquitetura de escalonamento centralizado, ou seja, uma máquina especial é responsável pelo escalonamento. Todas as máquinas podem submeter tarefas a máquina central que se responsabiliza de encontrar recursos disponíveis para execução da tarefa. Tanto o Condor quanto o Globus perdem pontos no quesito tolerância a falhas e escalabilidade devido ao fato de terem um controle centralizado onde um problema na máquina central comprometeria o sistema por inteiro. Além disso, para o Globus, são necessárias negociações com os donos de recursos além da necessidade do mapeamento dos clientes para usuários locais.

Já o projeto ISAM (15) possui uma arquitetura organizada na forma de células autônomas cooperativas. Sua proposta é fornecer uma infra-estrutura tanto para a construção quanto execução de aplicações pervasivas (15). Concebida para habilitar as aplicações a obter informações do ambiente onde executam e se adaptam às alterações que ocorrem durante o transcurso da execução. O ISAM, diferente do Globus e do Condor possui um modelo de escalonador de tarefas descentralizado ajudando o sistema alcançar um bom nível de tolerância a falhas e escalabilidade.

3 MODELO GRAND

No modelo GRAND são tratados três aspectos do gerenciamento de dados: transferência automática dos dados de entrada para o local onde o arquivo será necessário; o envio de resultados é controlado evitando congestionamento da rede; priorização de localidade no disparo de tarefas para não haver transferências desnecessárias de dados degradando o desempenho. Através de uma hierarquia de gerenciadores (Figura 1) é feito o disparo e controle das aplicações. o *Application Manager* (AP) recebe uma submissão de aplicação através de um usuário, os APs mandam os *Submission Managers* (SM) descrições de tarefas assim, sob demanda, são instanciados os *Task Managers* (TM) para controlar a submissão de tarefas a escalonadores de domínios específicos da grade, esses escalonadores recebem requisições dos TMs fazendo a execução das tarefas propriamente ditas.

Pelo fato de que, na atualidade, ambientes grades envolvem principalmente instituições de ensino em aplicações usualmente classificadas como aplicações científicas, o escopo do GRAND é limitado as seguintes itens.

1. heterogeneidade, lembrando que isto afeta diretamente a política de escalonamento por necessitar de saber as características distintas de hardware e software;

2. grande número de submissão de tarefas, referindo-se a aplicações que geram centenas ou milhares de processos;

3. ausência de comunicação por troca de mensagens, pelo fato da necessidade de inúmeros aspectos nas fases de agrupamento e mapeamento serem considerados;

4. interdependência de tarefas, devido ao compartilhamento de arquivos;

5. manipulação de grande número de arquivos pelas tarefas;

6. o uso de arquivos grandes, através de técnicas como *staging* e *caching*, minimizando a perda de desempenho em função da latência de transmissão;
7. segurança, assume-se que exista uma conexão segura entre os nós da grade;
8. descoberta dinâmica de recursos;
9. gerenciador de recursos local em cada nó;
10. uma tarefa é executada em um RMS até sua finalização;

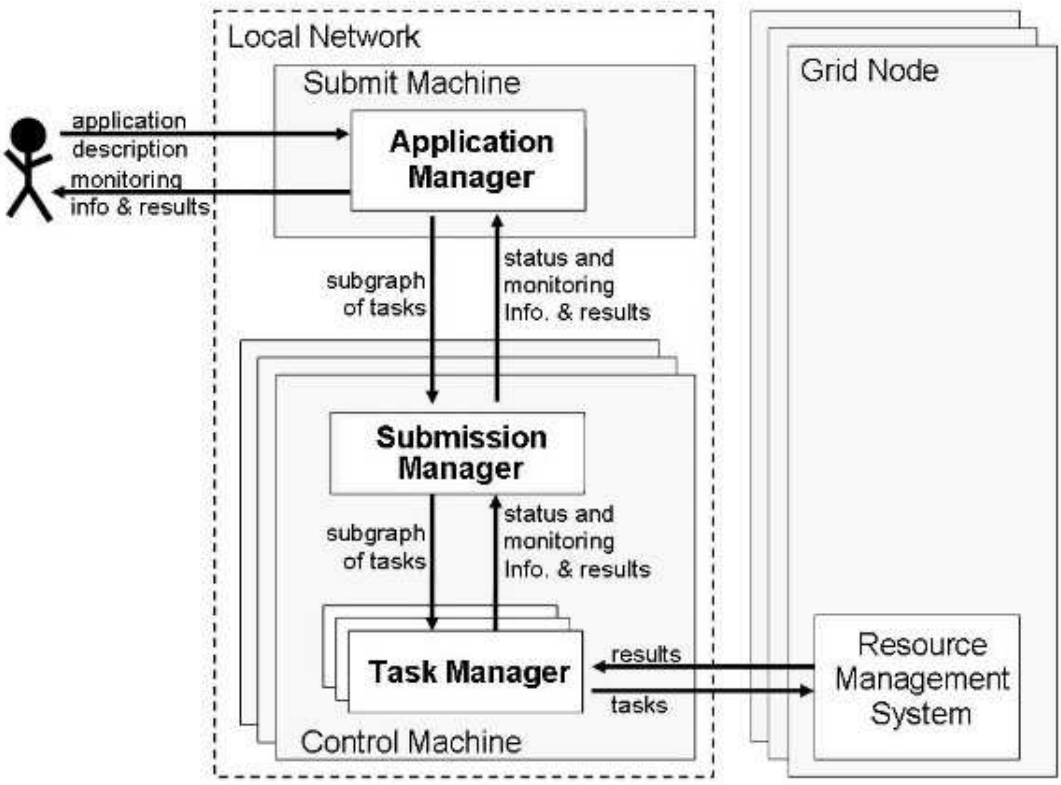

Figura 3.1: Principais componentes do modelo hierárquico de gerenciamento de tarefas

REFERÊNCIAS

[1] M. A. R. Dantas, *Computação Distribuída de Alto Desempenho: Redes, Clusters e Grids Computacionais*. 2005.

[2] A. O. o. t. C. S. High Throughput Computing (CONDOR), "High throughput computing (condor), an overview of the condor system," 2007. acessado em Agosto de 2007.

[3] A. Bayucan, R. L. Henderson, C. Lesiak, B. Mann, T. Proett, and D. Tweten, "Numerical aerospace simulation systems division nasa ames research center," p. 281, 2007.

[4] P. K. V. Mangan, "Grand: Um modelo de gerenciamento hierárquico de aplicações em ambiente de computação em grade," p. 150, 2006.

[5] P. K. Vargas, I. de Castro Dutra, and C. F. R. Geyer, "Hierarchical resource management and application control in grid environments," p. 8, 2003. Relatório Técnico ES-608/03, COPPE/Sistemas UFRJ.

[6] H. Rajic, R. Brobst, W. Chan, F. Ferstl, J. Gardine, A. H. ans Bill Nitzber, and J. Tollefsrud, "Open grid forum documents, distributed resource management application api specification 1.0 (drmaa)," p. 29, Junho 2004.

[7] P. Tröger and B. Gietzel, "Condor drmaa 1.0 implementation - experience report (gfd-103)," tech. rep., Hasso-Plattner-Institute and University of Wisconsin-Madison, Fevereiro 2007.

[8] M. Roehrig, W. Ziegler, and P. Wieder, "Grid scheduling dictionary of terms and keywords," November 2002.

[9] I. Foster, S. Tuecke, and C. Kesselman, "The anatomy of the grid enabling scalable virtual organizations," p. 25, 2001.

[10] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "The physiology of the grid: An open grid services architecture for distributed systems integration," p. 31, Junho 2002.

[11] W. Cirne, "Grids computacionais: Arquiteturas, tecnologias e aplicações," p. 46, 2002.

[12] T. L. Casavant and J. G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," p. 37, 1996.

[13] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A resource management architecture for metacomputing systems," p. 19, 1998.

[14] I. Foster and C. Kesselman, "The globus project: A status report," p. 15, 1998.

[15] ISAM, "Apresentação do isam - http://www.inf.ufrgs.br/ isam/."

[16] A. Bayucan, R. L. Henderson, C. Lesiak, B. Man, T. Proett, and D. Tweten, "Portable batch system - external reference specification," p. 281, Agosto 1998.

[17] T. G. A. http://www.globus.org/alliance/publications/papers.php, "The globus aliance."

[18] L. A. S. Santos, M. T. Rebonatto, P. K. Vargas, and C. F. R. Geyer, "Uma proposta de escalonamento colaborativo de aplicações em um ambiente de computação em grade," p. 8.

[19] N. Andrade, "Acesso em grids computacionais: estado da arte e prespectivas," p. 16, 2002.

[20] J. E. M. León, "Análisis comparativo gt 2.4 - gt 4.0," *Semana de Cómputo Científico - Supercómputo, Visualización y Realidad Virtual*, p. 60, 2006.