Integração do Sistema AppMan de Gerenciamento de Aplicações para Ambiente de Grade com diferentes Sistemas de Gerenciamento de Recursos

Tonismar Régis Bernardo[1],**Patrícia Kayser Vargas Mangan**[1]

[1] Curso de Ciência da Computação - Centro Universitário La Salle (UNILASALLE).
Av. Victor Barreto, 2288, Centro, Canoas - RS - Brasil

tonismar.at.gmail.com

1. Introdução

Grades computacionais (grid computing) é uma das formas mais recentes de ambiente para processamento geograficamente distribuído, que conta com uma grande infraestrutura de redes e pode ser empregada em troca de programas, dados e serviços. Segundo Dantas [1], pode-se dizer, também, que 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].

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

O texto que segue apresenta-se organizado em 5 seções. Na seção 2 é apresentado o estado da arte relacionado ao estudo do gerenciamento das aplicações e do gerenciamento de recursos. A metodologia de pesquisa encontra-se na seção 3 apresentando as características da metodologia até o momento. A seção 4 é sugerido o modelo de solução com ênfase na DRMAA. Por fim, a conclusão do estudo na seção 5.

2. Metodologia

Para verificar a viabilidade da exportação de tarefas do AppMan para RMS diferentes será realizado um estudo aprofundado no modelo GRAND onde o protótipo foi desenvolvido.

O método de pesquisa utilizado está entre um experimento, baseado no fato de que o modelo GRAND espera a integração com qualquer RMS e também em um estudo de caso em função da análise da implementação proposta.

O desenvolvimento proposto na integração será feito na linguagem Java que foi a linguagem de desenvolvimento utilizada no AppMan visando a portabilidade. Além disso algumas comparações através de métricas para avaliar as vantagens e desvantagens com o próprio escalonador do AppMan serão analisadas.

Até o presente momento está sendo feito um estudo da DRMAA e onde ela já foi empregada verificando suas fases bem como o resultado dessas implementações [7], [8] e [9]. Também foi executado todos procedimentos de instalação 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 e estão sendo sanados. Um engenharia reversa das classes do projeto AppMan gerando os diagramas UML foi feita para facilitar o estudo da integração com a DRMAA.

Os testes e avaliações serão baseados nos mesmos ambientes onde foram feitos os experimentos do AppMan [4].

Além do estudo da DRMAA, é pretendido avaliar outras formas de integração entre RMS.

3. Estado da Arte

Atualmente o uso de redes de computadores 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 característica 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 [10] 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 [11] 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 em [12]. Focando na sua semântica, mostrando que grades não são apenas uma modificação de um sistema distribuído convencional. Podem apresentar recursos heterogênicos como sensores e detectores e não apenas nós computacionais. Abaixo uma lista de aspectos que evidenciam uma grade computacional [13]:

- 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.

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 [14] 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 [10], 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 [15]:

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.

3.1. Gerenciamento de Aplicações

Sistemas computacionais, na sua grande parte, falham ao tratar dois problemas [4]: (1) gerenciamento e controle de um grande número de tarefas; (2) 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. Um modelo denominado GRAND (*Grid Robust Application Deployment*) baseado na submissão e controle particionados e hierárquicos foi proposto em [4]

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 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;

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 Manager* (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.

O AppMan usa o serviço provido pelo EXEHDA middleware [16] que libera monitoramento e execução remota. As características básicas do GRAND foram implementadas no AppMan, inclusive a submissão de tarefas e o retorno para o usuário. Em cada nó da grade é necessário estar sendo executada uma instância do ISAM/EXEHDA e do AppMan. Qualquer máquina pode submeter ou executar tarefas. A tarefa do AppMan é garantir o escalonamento das tarefas assim como a autorização e autenticação das tarefas executadas. Ele consiste num gerenciador que dispara e controla cada aplicação nos nós

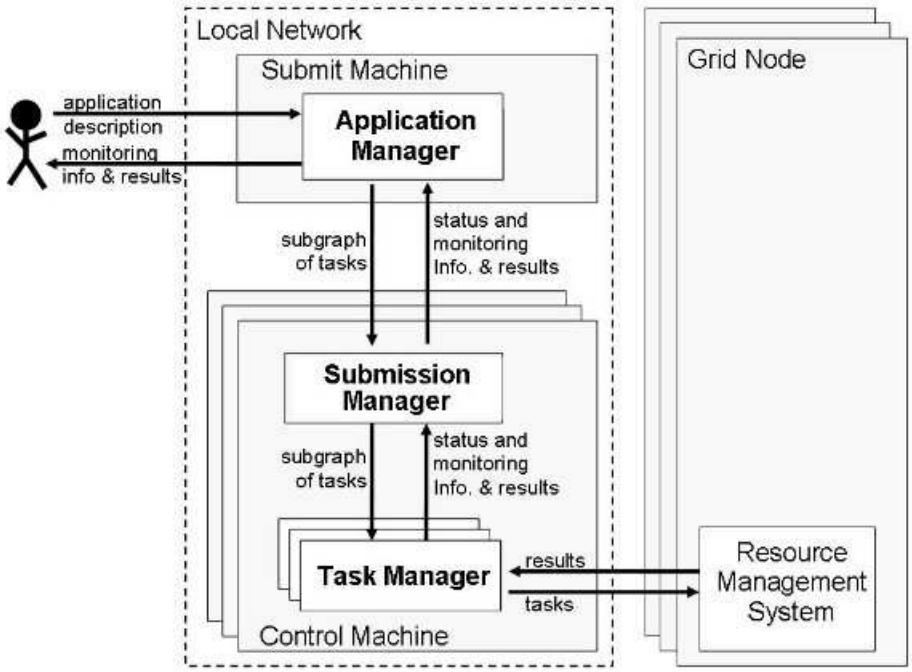

Figura 1. Principais componentes do modelo hierárquico de gerenciamento de tarefas

com base nas informações recebidas do pelo gerenciador de submissão. O gerenciador de submissão, além de enviar informações, cria e monitora os gerenciadores de tarefas. Os gerenciadores de tarefas são responsáveis pela comunicação com o escalonador de um determinado domínio garantindo a execução remota e a ordem das tarefas de acordo com a dependência de dados [4].

Uma proposta de escalonamento colaborativo usando o modelo GRAND foi feita em [17]. Baseada em uma rede *peer-to-peer* (P2P) consiste em analisar o estado de recursos de uma célula (consultando o escalonador local) e as políticas de escalonamento. Através de um algoritmo de escalonamento global, decide quais sub-grafos (conjunto de tarefas) da fila de tarefas globais podem executar nos recursos de sua célula local, e quais sub-grafos serão enviados à outros centros administrativos da rede lógica da grade P2P.

3.2. Gerenciamento de Recursos

Grande parte das pesquisas sobre escalonamento de tarefas em grades seguem uma organização hierárquica ou centralizada tais como: [18, Globus], Condor [2], ISAM [19] e PBS [20].

Um dos projetos mais referenciados na literatura é o Globus [21], que 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 [18] 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 Servi-*

ces (MDS) [17]. O gerenciamento de cada recurso é feito por uma instância do *Globus Resource Allocation Manager* (GRAM) [22]. GRAM é o responsável por instanciar, monitorar e reportar o estado das tarefas alocadas para o recurso. A GT4 [23] 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 foca na qualidade, robustez, facilidade de uso e documentação.

Um dos gerenciadores que podem ser integrados com o Globus é o PBS [20]. 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 [19] possui uma arquitetura organizada na forma de células autônomas cooperativas. Sua proposta e fornecer uma infra-estrutura tanto para a construção quanto execução de aplicações pervasivas [**?**]. Concebida para habilitar as aplicações a obter informações do ambiente onde executam e se adaptar 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.

4. Modelo Proposto

O protótipo AppMan é uma implementação simplificada do GRAND. Ele usa a ferramenta JavaCC para implementar o *parsing* e interpretação da GRID-ADL. A Figura 2 representa um possível cenário do AppMan sendo executado sobre um ambiente de grade.

O passo 1 da figura representa o usuário submetendo um arquivo de descrição na linguagem GRID-ADL. Esse arquivo é analisado e o grafo de aplicação é feito na memória. Um algoritmo é executado e os sub-DAGs do grafo de aplicação é obtido. O *Application Manager* (AM) é inicializado. Então o AM instância *Submission Managers* (SM) no passo 2 e distribui alguns subgrafos para os SMs. Os arquivos de entrada e os executáveis são obtidos através de um *web server* (passo 3). Na sua atual implementação, AppMan necessita que o usuário indique as máquinas onde os SMs irão ser executados. Com essa simplificação um novo SM é instanciado para cada aplicação em cada máquina especificada. Após a criação dos SMs, o AM determina sub-grafos para cada SM. Os SMs informam para o AM o progresso das tarefas. Cada SM, independentemente, verifica a lista das máquinas disponíveis e escolhe aleatoriamente um nó para executar a tarefa. Antes de iniciar a execução de uma tarefa, AppMan transfere todos arquivos de entradas

especificados na descrição GRID-ADL para um diretório temporário no nó remoto onde a tarefa será executada. O arquivo de transferência é executado automaticamente. Então, cada SM recupera a informação atualizada através do serviço de informação do EXEHDA sobre os nós avaliados. Cada SM escolhe onde irá executar estas tarefas aleatoriamente. Imediatamente um *Task Manameger* (TM) é instanciado criando a tarefa remota e monitorando até que a execução da tarefa termine (passo 4).

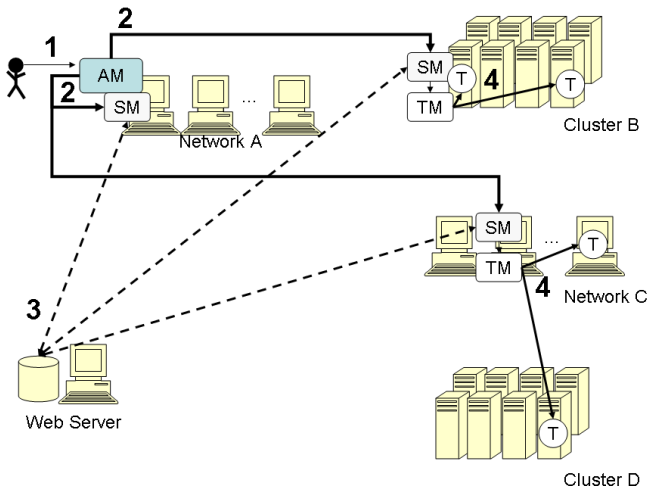

Figura 2. AppMan executando principais passos

Como sugere o modelo GRAND, o AppMan não possibilita o escalonamento através de outros RMS. É necessário que o EXEHDA e o AppMan estejam presentes em todos nós da grade. Integrar o AppMan com outros RMS possibilitará uma maior escalabilidade da grade e também possibilitará uma aproximação do AppMan ao que sugere o modelo GRAND. Os nós principais de cada *cluster* que compõem a base possuem nós abaixo delas. O AppMan espera um serviço de compartilhamento de arquivo entre os nós e base. Isto é, mais uma característica que impossibilita o uso de nós da grade sem NFS, como existente na parceria LNCC/Petrópolis-RJ, pelo AppMan.

Uma especificação *Distributed Resource Management Application* [24] desenvolvida pelo OGF, tem por objetivo abstrair as diferenças dos RMS e fornecer uma API visando facilitar a integração de aplicações. O escopo da DRMAA é limitada a submissão, monitoramento e controle das tarefas além de retornar status de conclusão de uma tarefa. Inicialmente implementada na linguagem C, atualmente possui uma implementação na linguagem Java, apesar de oferecer suporte apenas para o *Sun Grid Engine* [7].

Vários relatórios técnicos N1^{TM} Grid Engine [25], GridWay [26], Condor [27] e PBS [28] existentes da implementação DRMAA, concluem positivamente. Além disso demonstraram uma pequena necessidade de inteferência no código dos RMSs citados.

Em um primeiro estudo da documentação do AppMan pode-se notar duas principais classes, *TaskManager* (Figura 3) e *SubmissionManager*. A classe *SubmissionManager* implementa os métodos da *Interface SubmissionManagerRemote* a qual possui os métodos que recuperam informações das tarefas executadas remotamente e o metódo da geração do grafo da aplicação. A classe *TaskManager* implementa a interface *TaskMana-*

gerRemote contendo os métodos tais como os que adicionam tarefas na lista de execução remota.

Figura 3. Classe TaskManager

Baseado no que foi analisado até o momento da confecção deste artigo, notou-se a necessidade de alteração na *Interface TaskManagerRemote* onde necessitará ser adicionado métodos que implementarão as especificações da DRMAA. Essa seria a melhor forma de implementação com a mínima interferência no código atual. Principalmente por existirem trabalhos nessa área [29].

A presente proposta de pesquisa proporcionará uma maior dispersão geográfica para o AppMan pois, novos domínios adimistrativos (instituições) poderão fazer parte da grade em que se encontra. Contribuiria, também, para o AppMan ficar próximo de atingir o que propõem o modelo GRAND resolvendo o problema da necessidade do EXEHDA/AppMan estarem presentes em todos nós da grade.

A Figura 4 demonstra como deverá ficar o funcionamento do AppMan após a realização do trabalho proposto. Os TMs nos nós locais contatam RMSs remotos despachando as tarefas para esses RMSs.

O modelo proposto para o TCC consiste em avaliar se a especificação DRMAA atende as necessidades do AppMan bem como realizar a integração do AppMan ao menos com um RMS.

5. Conclusão

Teste

Referências

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

[2] C. http://www.cs.wisc.edu/condor/overview, "High throughput computing (condor), an overview of the condor system."

[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.

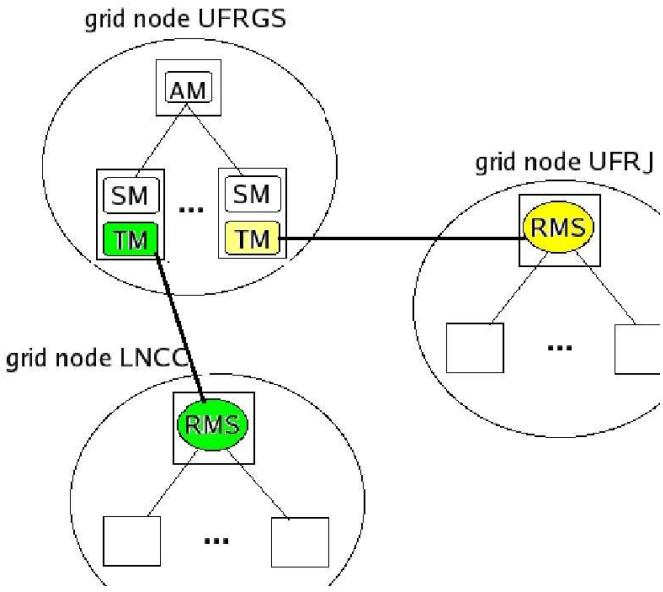

Figura 4. AppMan executando em um cenário com TMs comunicando com RMS

[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] D. Templeton, S. S. Engineer, and S. M. GmbH, "Ggf13: Drmaa tutorial c and java language bindings," tech. rep., Sun Microsystems.

[8] I. M. Llorente, "Drmaa for gridway," tech. rep., Grupo de Arquitetura de Sistema Distribuídos e Segurança and Departamento de Arquitetura de Computadores and Universidade Complutense de Madri and Laboratório de Computação Avançada and Simulação e Aplicação Telemáticas and Centro de Astrobiologia CSIC/INTA associado a NASA Instituto de Astrobiologia, Março 2005.

[9] A. Haas, "Drmaa state of c binding/implementation drmaa implementation compliance test," tech. rep., Sun Microsystems, Março 2004.

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

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

[12] 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.

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

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

[15] 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.

[16] C. P. Nino, "Consciência do contexto e da mobilidade do aprendiz em um ambiente de educação pervasiva," p. 77, 2006.

[17] 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.

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

[19] "Apresentação do isam."

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

[21] http://www.globus.org/alliance/publications/papers.php, "The globus aliance."

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

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

[24] H. Rajic, R. Brobst, C. D. Systems, W. Chan, F. Ferstl, J. Gardiner, A. Haas, B. Nitzberg, and J. Tollefsrud, "Distributed resource management application api specification," p. 20, 2002.

[25] D. Templeton and A. Haas, "N1^{TM} grid engine drmaa 1.0 implementation – experience report (gfd-105)," tech. rep., Sun Microsystems, Inc., Maio 2006.

[26] J. Herrera, E. Huedo, R. S. Montero, and I. M. Llorente, "Gridway drmaa 1.0 implementation – experience report (gfd-104)," tech. rep., Universidad Complutense de Madrid, Fevereiro 2007.

[27] 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.

[28] L. Ciesnik, P. Domagalski, K. Kurowski, and P. Lichocki, "Pbs/torque drmaa 1.0 implementation - experience report (gfd-117)," tech. rep., Poznan Supercomputing and Networking Center and Poland FedStage Systems Inc., Setembro 2007.

[29] W. S. Nobres, "Ánalise do impacto da aplicação de métricas de qualidade de software orientado a objetos no desempenho de grids computacionais," *Ciência da Computação/UNILASALLE. TCC em andamento*, 2007.