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 a metodologia de pesquisa utilizada até o momento bem como suas características. O estado da arte encontra-se na seção 3 relacionado ao estudo do gerenciamento das aplicações, do gerenciamento de recursos. A seção 4 é sugerido o modelo de solução com ênfase na DRMAA. Por fim, a conclusão do estudo encontra-se na seção 5.

2. Metodologia

Para verificar a viabilidade da exportação de tarefas do AppMan para RMS diferentes, um estudo da especificação DRMAA será realizado visando o desenvolvimento desta integração. Será realizado, também, um estudo aprofundado no modelo GRAND onde o protótipo foi desenvolvido e um minucioso estudo da implementação atual do AppMan.

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.

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.

Até o presente momento está sendo feito um estudo sobre a DRMAA e onde ela já foi empregada verificando suas fases bem como o resultado dessas implementações [7], [8] e [9].

Estudos detalhados no diagrama de classes do AppMan auxiliarão na escolha da melhor forma de implementação 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, serão avaliadas 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.

Segundo Foster [10], pode-se dizer, também, que representa uma forma estendida dos serviços web permitindo que recursos computacionais possam ser compartilhados.

Define-se grades como uma plataforma computacional heterogênia distribuída geograficamente fornecendo serviços e recursos às organizações participantes da plataforma.

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 [11] 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 [10] 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 [11], 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 1.1) 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.

Baseado no modelo GRAND criou-se o protótipo AppMan (*Application Manager*). Implementado em Java visando portabilidade. O AppMan usa o serviço provido

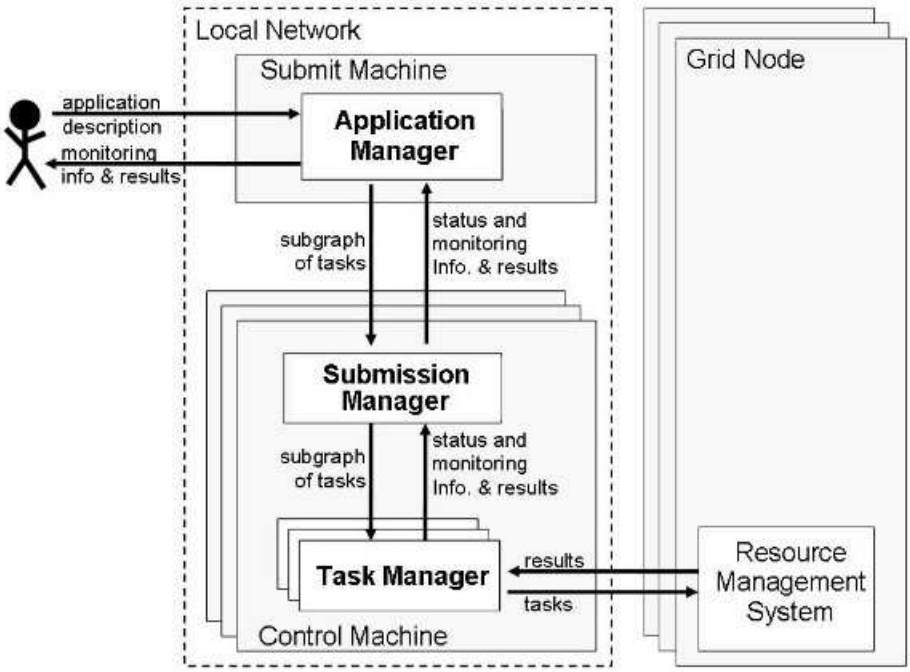

Figure 1. Principais componentes do modelo hierárquico de gerenciamento de tarefas

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 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].

Por definição o modelo GRAND permite que qualquer sistema de gerenciamento de recurso seja usado no nós, porém o protótipo AppMan não contempla essa característica, funcionando apenas com seu próprio gerenciador.

Uma interface de aplicação (*Application Program Interface* API) denominada DRMAA (*Distruited Resource Management Application API*) foi desenvolvida para facilitar a integração de diferentes RMSs [6]. Sendo assim, este artigo sugere verificar se a DRMAA atende as necessidades do AppMan bem com sua utilização com o AppMan na integração com demais RMSs.

References

[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.

[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, "Ggf12: 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] I. Foster, S. Tuecke, and C. Kesselman, "The anatomy of the grid enabling scalable virtual organizations," p. 25, 2001.

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

[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.