UNILASALLE

CENTRO UNIVERSITÁRIO LA SALLE

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

INTEGRAÇÃO DO SISTEMA APPMAN DE GERENCIAMENTO

TONISMAR RÉGIS BERNARDO

Canoas, abril de 2008

TONISMAR RÉGIS BERNARDO

INTEGRAÇÃO DO SISTEMA APPMAN DE GERENCIAMENTO

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, abril de 2008

TERMO DE APROVAÇÃO

TONISMAR RÉGIS BERNARDO

INTEGRAÇÃO DO SISTEMA APPMAN DE GERENCIAMENTO

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, abril de 2008

SUMÁRIO

LISTA DE SÍMBOLOS E ABREVIATURAS

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Grades computacionais (grid computing) 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 (7), 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 (7).

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 (11; 3; 13). 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*) (13) 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*) (16), 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 determi-

nado domínio garantindo a execução remota e ordem das tarefas de acordo com a dependência de dados (13).

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 (14).

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.

Colocar um parágrafo que explicará o restanto da monografia.

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 (15) 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 (9). 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 (5):

- 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 (4) 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 (15), 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 (6):

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 (13): (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 (13)

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;

Grande parte das pesquisas sobre escalonamento de tarefas em grades seguem uma organização hierárquica ou centralizada tais como: Globus (8), Condor (11), ISAM (http://www.cs.wisc.edu/condor/overview) e PBS (2).

Globus (http://www.globus.org/alliance/publications/papers.php), 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 (8) 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) (Santos et al.). O gerenciamento de cada recurso é feito por uma instância do *Globus Resource Allocation Manager* (GRAM) (1). GRAM é o responsável por

instanciar, monitorar e reportar o estado das tarefas alocadas para o recurso. A GT4 (12) 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 (2). 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 (11), 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 (http://www.cs.wisc.edu/condor/overview) 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 (http://www.cs.wisc.edu/condor/overview). 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.

2.2 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.

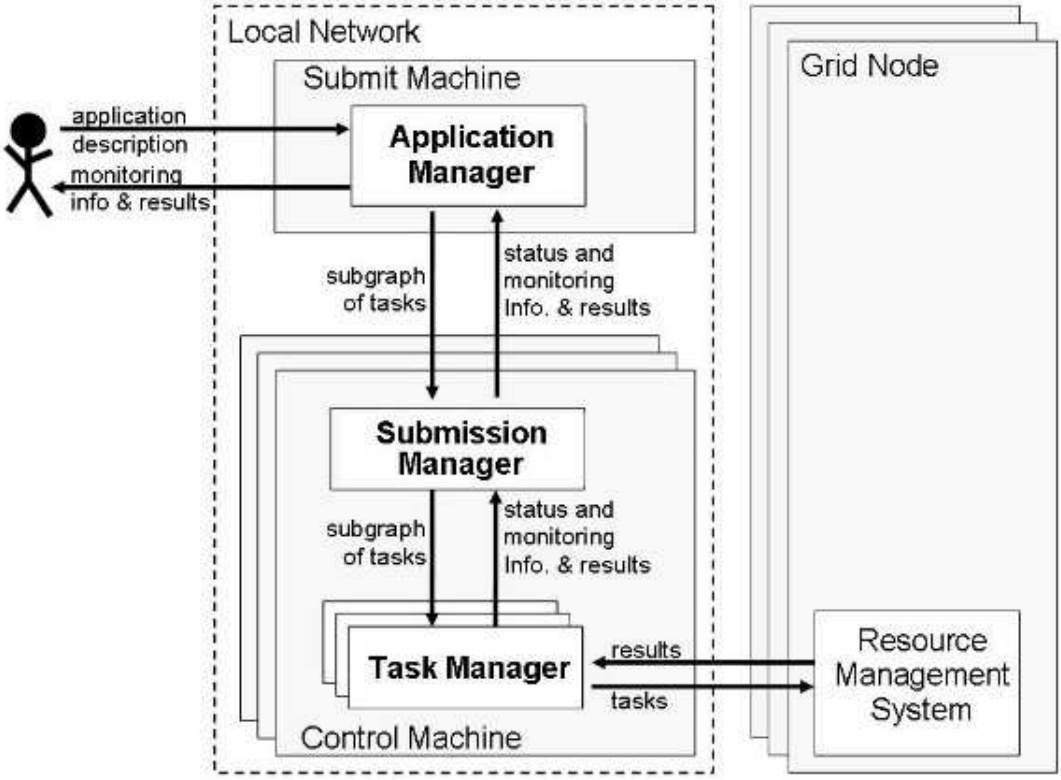

Figura 2.1: Principais componentes do modelo hierárquico de gerenciamento de tarefas

REFERÊNCIAS

[1] Andrade, N. (2002). Acesso em grids computacionais: estado da arte e prespectivas. page 16.

[2] Bayucan, A., Henderson, R. L., Lesiak, C., Man, B., Proett, T., and Tweten, D. (1998). Portable batch system - external reference specification. page 281.

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

[4] Casavant, T. L. and Kuhl, J. G. (1996). A taxonomy of scheduling in general-purpose distributed computing systems. page 37.

[5] Cirne, W. (2002). Grids computacionais: Arquiteturas, tecnologias e aplicações. page 46.

[6] Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., and Tuecke, S. (1998). A resource management architecture for metacomputing systems. page 19.

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

[8] Foster, I. and Kesselman, C. (1998). The globus project: A status report. page 15.

[9] Foster, I., Kesselman, C., Nick, J. M., and Tuecke, S. (2002). The physiology of the grid: An open grid services architecture for distributed systems integration. page 31.

[10] Foster, I., Tuecke, S., and Kesselman, C. (2001). The anatomy of the grid enabling scalable virtual organizations. page 25.

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

[http://www.cs.wisc.edu/condor/overview] http://www.cs.wisc.edu/condor/overview. Apresentação do isam.

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

[12] León, J. E. M. (2006). Análisis comparativo gt 2.4 - gt 4.0. *Semana de Cómputo Científico - Supercómputo, Visualización y Realidad Virtual*, page 60.

[13] Mangan, P. K. V. (2006). Grand: Um modelo de gerenciamento hierárquico de aplicações em ambiente de computação em grade. page 150.

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

[15] Roehrig, M., Ziegler, W., and Wieder, P. (2002). Grid scheduling dictionary of terms and keywords.

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

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