

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

Tonismar Régis Bernardo¹, Patrícia Kayser Vargas Mangan¹

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

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 Dantas [1, apud], pode-se dizer, também, que representa uma forma estendida dos serviços web permitindo que recursos computacionais possam ser compartilhados.

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

[1, apud] O Global Grid Forum (GGF) uma comunidade fórum com milhares de indivíduos representando mais de 400 organizações em mais de 50 países criou e documentou especificações técnicas e experiências de usuários. O GGF 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 [2] 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 [1, apud]. 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 [3]:

- 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 a 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.

[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 GGF's [5], 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. estender 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 [1]: (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 [1]

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.

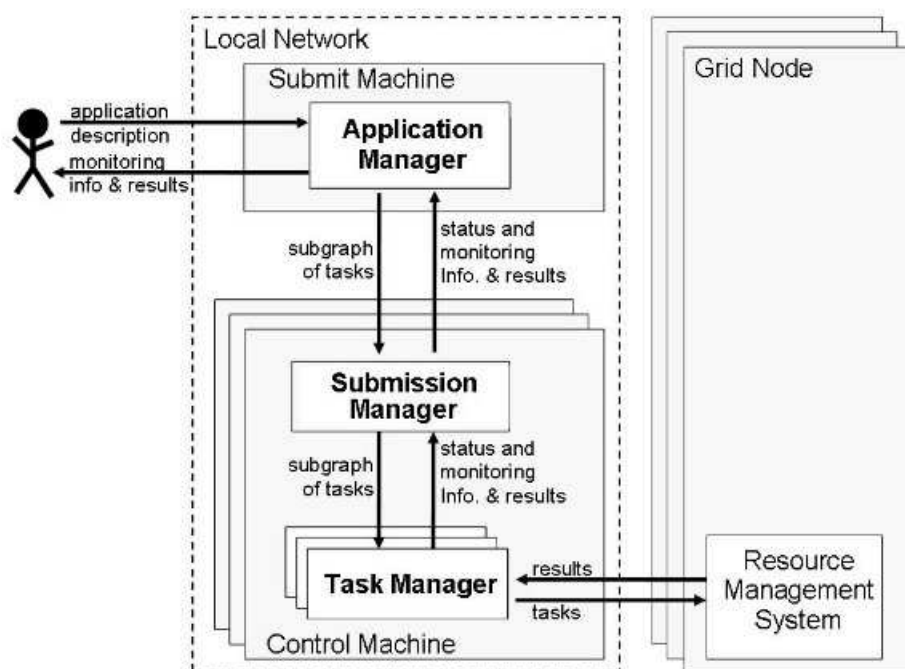


Figure 1. Principais componentes do modelo hierarquico de gerenciamento de tarefas

References

- [1] P. K. V. Mangan, “Grand: Um modelo de gerenciamento hierárquico de aplicações em ambiente de computação em grade,” p. 150, 2006.

- [2] I. Foster, S. Tuecke, and C. Kesselman, "The anatomy of the grid enabling scalable virtual organizations," p. 25, 2001. <http://www.globus.org/alliance/publications/papers/anatomy.pdf>.
- [3] W. Cirne, "Grids computacionais: Arquiteturas, tecnologias e aplicações," p. 46, 2002. <http://walfredo.dsc.ufcg.edu.br/papers/Grids>
- [4] T. L. Casavant and J. G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems," p. 37, 1996. <http://www.eng.uiowa.edu/tomc/papers/taxo.ps.gz>.
- [5] M. Roehrig, W. Ziegler, and P. Wieder, "Grid scheduling dictionary of terms and keywords," November 2002. <http://www.ggf.org/documents/GFD.11.pdf>.
- [6] 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. <ftp://ftp.globus.org/pub/globus/papers/gram97.pdf>.