Gerenciamento Hierárquico de Aplicações em Ambientes de Computação em Grade*

Patrícia Kayser Vargas[1], Inês de Castro Dutra[1], Cláudio F. R. Geyer[2]

[1]COPPE/Sistemas – Universidade Federal do Rio de Janeiro
Caixa postal 68511 CEP 21941-972 – Rio de Janeiro, RJ
[2]PPGCC – Universidade Federal do Rio Grande do Sul
Caixa postal 15064 CEP 91501-970 – Porto Alegre, RS
{kayser,ines}@cos.ufrj.br, geyer@inf.ufrgs.br

Resumo

O termo *Grid computing* ou Computação em Grade [FOS 98], estabelecido no final dos anos 90, denota uma proposta de infra-estrutura computacional distribuída. Uma infra-estrutura de grade deve prover de forma global e transparente os recursos heterogêneos e geograficamente distribuídos requisitados por aplicações de grande demanda computacional (CPU e/ou E/S), como, por exemplo, aplicações em bioestatística [DUT 03]. Vários trabalhos vêm sendo propostos para tratar o gerenciamento de recursos e aplicações no ambiente de grade [ROU 03, BER 03, THA 03]. No entanto, escalonar e controlar a execução de aplicações compostas por centenas ou milhares de tarefas ainda apresenta-se como um desafio técnico e científico: o grande número de tarefas pode causar uma sobrecarga na máquina de submissão, e o controle manual é proibitivo uma vez que (a) estas aplicações utilizam uma grande quantidade de recursos e (b) elas podem levar vários dias ou meses para serem concluídas com sucesso. Este trabalho propõe um esquema hierárquico de submissão e controle de tarefas para tratar aplicações compostas por uma grande quantidade de tarefas, na ordem de milhares, em um ambiente de computação em grade. Pressupõe-se que as aplicações submetidas são formadas por tarefas que podem ter dependências na sua ordem de execução, mas não realizam comunicação por troca de mensagens. Estas dependências são determinadas pelos dados de entrada e saída (normalmente arquivos de dados), conforme explicitado no arquivo de descrição da aplicação.

O modelo trata três aspectos do gerenciamento de dados: (a) os dados de entrada são transferidos automaticamente para o local onde o arquivo será necessário como ocorre em outros trabalhos (e.g. [THA 03]); (b) como o volume de dados a ser tratado é potencialmente muito grande, o modelo contempla o envio dos resultados ao usuário de forma controlada para evitar congestionamento da rede; (c) o escalonamento prioriza a localidade no disparo de tarefas para evitar transferências desnecessárias de dados transientes e consequente degradação do desempenho. O disparo e controle das aplicações é feito através de uma hierarquia de gerenciadores: (nível 0) um usuário submete uma aplicação em uma máquina através do *Application Submit*; (nível 1) os *App Submits* enviam aos *Submission Managers* (*SM*) descrições de tarefas; (nível 2) os *Task Managers* (*TM*) são instanciados sob demanda pelos *SM* a fim de controlar a submissão de tarefas a escalonadores de domínios específicos da grade; (nível 3) escalonadores nos domínios específicos recebem requisições dos *TM* e fazer a execução de fato das tarefas.

*Pesquisa parcialmente financiada pelo Centro Universitário La Salle/Canoas-RS e pelo CNPq.

O *App Submit* é encarregado de: (a) receber um arquivo de entrada descrevendo as tarefas; (b) particionar as tarefas em subgrafos que são enviados para diferentes *SM*, buscando manter a localidade de dados e minimizar a comunicação entre os *SM*; (c) mostrar ao usuário, de forma amigável, informações do estado da execução da aplicação.

As principais funções do *SM* são: (a) decidir a alocação dos subgrafos com base em informações dinâmicas sobre os recursos computacionais; (b) indicar o estado da execução e falhas através de comunicação periódica com o *App Submit*; (c) acompanhar o andamento da aplicação e recuperar de falhas através de um registro persistente (*log*); (d) criar e monitorar os *TMs*; (e) avaliar, durante a execução, se deve ou não continuar a execução de tarefas em máquinas que forem liberadas ou retornarem após uma falha.

Cada *TM* é responsável por: (a) se comunicar com o escalonador de um determinado domínio a fim de garantir a execução remota de tarefas; (b) garantir a ordem de execução de acordo com as dependências de dados; (c) controlar a transferência de dados garantindo a disponibilidade dos dados de entrada e o recebimento dos dados de saída.

O uso de uma hierarquia de escalonadores é considerado na literatura com uma boa alternativa para o ambiente de grade [VAD 02, KRA 02]. O principal diferencial do modelo proposto é a hierarquia de gerenciadores de submissão acima do meta escalonador. Deste modo, pode-se notar que o modelo proposto realiza um balanceamento da carga computacional necessário para controlar a execução das tarefas, evitando que a máquina de submissão fique sobrecarregada e que ocorra perda de dados por congestionamento da rede. Além disso, permite que escalonadores já existentes sejam integrados em um único ambiente de escalonamento. O modelo proposto está atualmente sendo implementado no ambiente de simulação MONARC2 [LEG 00], a fim de demonstrar a viabilidade da idéia.

Palavras chaves

Computação em Grade, Escalonamento de Tarefas, Gerenciamento Hierárquico.

Referências

[BER 03] BERMAN, F.; FOX, G.; HEY, T. **Grid computing**: making the global infrastructure a reality. 1.ed. New York, USA: John Wiley, 2003. 1060p.

[DUT 03] DUTRA, I. C. et al. Toward automatic management of embarrassingly parallel applications. In: INTL CONF. ON PARALLEL AND DISTRIBUTED COMPUTING (Euro-par 2003), 26., 2003, Klagenfurt, Austria. **Proceedings...** [S.l.: s.n.], 2003. p.509–516.

[FOS 98] FOSTER, I.; KESSELMAN, C. **The grid**: blueprint for a new computing infrastructure. 1.ed. San Francisco, California, USA: Morgan Kaufmann, 1998. 701p.

[KRA 02] KRAUTER, K.; BUYYA, R.; MAHESWARAN, M. A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. **Software – Practice and Experience**, v.32, n.2, p.135–164, 2002.

[LEG 00] LEGRAND, I. et al. The Monarc toolset for simulating large network-distributed processing systems. In: WINTER SIMULATION CONF. **Proceedings...** ACM, 2000. p.1794–1801.

[ROU 03] ROURE, D. D. et al. The evolution of the grid. In: BERMAN, F. e. (Ed.). **Grid computing**: making the global infrastructure a reality. New York, USA: John Wiley, 2003. p.65–100.

[THA 03] THAIN, D. et al. Condor and the grid. In: BERMAN, F. et al (Eds.). **Grid computing**: making the global infrastructure a reality. New York, USA: John Wiley, 2003.

[VAD 02] VADHIYAR, S. S.; DONGARRA, J. J. A Metascheduler for the Grid. In: IEEE INTL. SYMP. ON HIGH PERFORMANCE DISTRIBUTED COMPUTING (HPDC-11), 11., 2002, Edinburgh, Scotland. **Proceedings...** IEEE Computer Society, 2002. p.343–354.