Suporte em Java para Alocação Dinâmica de Processadores

Márcia Cristina Cera, Marcelo Pasin

Informática/CT - UFSM Campus - 97105-900, Santa Maria, RS
{cera, pasin}@inf.ufsm.br

Resumo

Aliadas as pesquisas e estudos referentes a sistemas distribuídos [TAN 2002] ocorre o desenvolvimento de ferramentas capazes de proporcionar um melhor aproveitamento das potencialidades destes sistemas. Dentre estas ferramentas, destacam-se as bibliotecas de comunicação, estas possuem definidas rotinas e operações que tornam viável a comunicação entre os processos que integram uma aplicação distribuída. É comum o uso atual de sistemas distribuídos como meio de suporte à execução de aplicações que demandam alto desempenho, tradicionalmente executadas em computadores paralelos. As bibliotecas de comunicação mais difundidas para o processamento de alto desempenho são MPI - *Message Passing Interface* [GRO 94] e PVM - *Parallel Virtual Machine* [SUN 90].

Java tem se mostrado uma boa ferramenta de programação paralela por ser uma linguagem orientada a objetos muito popular, que apresenta mecanismos para operação com processos concorrentes e distribuídos. Ela disponibiliza mecanismos de comunicação dentro do seu próprio ambiente, dispensando o uso de bibliotecas suplementares como é necessário em outras linguagens, como por exemplo em C e Fortran [KIE 2001]. Para a comunicação em sistemas com memória distribuída, Java oferece o RMI[1], ou a invocação de métodos remotos. O RMI possibilita a invocação de métodos em máquinas remotas como se fossem métodos da mesma máquina.

Nosso trabalho visa aproveitar as facilidades oferecidas pela linguagem Java e construir uma estrutura para a alocação dinâmica de computadores em sistemas distribuídos, chamado CADEO (Controle de Alocação Dinâmica de Estações Ociosas). O CADEO é um mecanismo de execução remota de tarefas elementares de programas paralelos. Este mecanismo tira proveito de sistemas distribuídos (computadores pessoais em rede, aglomerados de computadores[2] ou grades [FOS 99]) que se encontram inativos ou com baixa utilização, disponibilizando-os para a execução de tarefas elementares. O termo "ocioso" será usado no sentido de designar computadores que se encontram disponíveis para a execução de tarefas paralelas.

Em linhas gerais, o CADEO é idealizado da forma descrita a seguir. Todas as máquinas disponíveis no sistema executarão um módulo Servidor de Execução Remota (SER). Após detectada a ociosidade de uma máquina, o módulo SER informa ao módulo Alocador de Estações Ociosas (AEO) sobre a disponibilidade da mesma. O AEO é o módulo do sistema que controla a alocação, e almeja-se que este módulo seja implementado de forma distribuída. Nele, serão mantidos os registros de todas as máquinas que encontram-se aptas a receber uma tarefa.

[1] *Remote Method Invocation*
[2] clusters

Completando o CADEO, existe o módulo Lançador de Tarefas Elementares (LTE), o qual será executado por quem possui tarefas a serem lançadas. Este módulo consulta o AEO para descobrir em qual máquina ociosa pode realizar o lançamento de sua tarefa. Em posse da identificação da máquina ociosa, o módulo LTE comunica-se diretamente com o SER desta máquina, solicitando a execução da sua tarefa. Ao final da execução de uma tarefa elementar, o módulo SER responde ao LTE com os resultados da execução e informa, novamente, ao módulo AEO a sua condição de ociosidade. Pode-se visualizar um esquema básico de funcionamento do CADEO na figura 1.

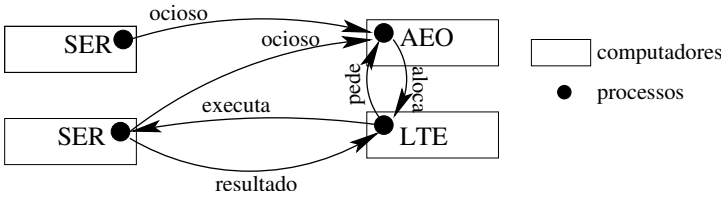

Figura 1: Esquema básico do sistema CADEO

Para se atender aos objetivos do CADEO e torná-lo funcional, almeja-se que o lançamento das tarefas elementares seja de forma assíncrona. Usando uma linguagem de programação orientada a objetos. A escolha mais natural parece ser Java, pela sua facilidade em tratar com sistemas concorrentes distribuídos e desta forma, tarefas paralelas elementares seriam implementadas por chamadas remotas de métodos (RMI). O RMI tradicinal é síncrono, porém já existem implementações que possibilitam sua execução assíncrona [RAJ 97]. Prevê-se que máquinas que deixam de ser ociosas (por exemplo, estações de trabalho que passam a atender um usuário local, ou nós de um aglomerado cujo tempo de disponibilidade ao usuário expirou) possam interromper a execução de suas tarefas, informando o AEO que não está mais disponível. Uma implementação usando RMI assíncrono permitirá o relançamento transparente de tarefas interrompidas.

Palavras-chave: ***Java, RMI Assíncrono, Alocação Dinâmica.***

Referências

[FOS 99] FOSTER, I.; KESSELMAN, C. (Eds.). **The grid**: blueprint for a future computing infrastructure. San Francisco, California: MORGAN-KAUFMANN, 1999.

[GRO 94] GROPP, W.; LUSK, E.; SKJELLUM, A. **Using MPI**: Portable Parallel Programming with the Message-Passing Interface. Cambridge, MA: MIT Press, 1994.

[KIE 2001] KIELMANN, T. et al. Enabling Java for high-performance computing. **Communications of the ACM**, v.44, n.10, p.110–117, Oct. 2001.

[RAJ 97] RAJE, R. R.; WILLIAMS, J. I.; BOYLES, M. Asynchronous Remote Method Invocation (ARMI) mechanism for Java. **Concurrency: Practice and Experience**, v.9, n.11, p.1207–1211, Nov. 1997.

[SUN 90] SUNDERAM, V. S. PVM: A framework for parallel distributed computing. **Concurrency: practice and experience**, v.2, n.4, p.315–339, Dec. 1990.

[TAN 2002] TANENBAUM, A. S.; STEEN, M. van. **Distributed systems**: principles and paradigms. Upper Saddle River, NJ: Prentice Hall, 2002.