# Alocação Dinâmica de Computadores Ociosos em Java

Márcia Cristina Cera e Marcelo Pasin

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

## Resumo

Existem muitas iniciativas, atualmente, no sentido de tirar proveito da capacidade de processamento de computadores interligados em rede como ferramenta de computação de alto desempenho. Computadores pessoais ligados por uma rede local [POL 96], aglomerados de computadores [BAK 99] e, e uma escala maior, grades de computadores [FOS 98] foram e ainda são foco de pesquisa como plataformas de programação paralela. Para cada uma delas foram propostos inúmeras e diferentes ferramentas de programação, como por exemplo MPI [POL 96, BAK 99] ou Globus [FOS 98]. Embora muitas delas preconizem tirar proveito de maneira eficiente dos recursos computacionais, poucas delas permitem a sua programação de maneira simples e elegante.

Buscando uma maior produtividade e facilidade na programação, as linguagens de programação orientadas a objetos vem se destacando como uma boa opção. Isso porque possibilitam a construção de tipos abstratos e a reusabilidade de código. Dentre essas linguagens, Java é uma das mais populares e se destaca por ser simples, portável e por dar suporte a programação concorrente. Porém, tais vantagens acarretam em sobrecarga no desempenho de aplicações. Essa sobrecarga vem sendo amenizada através de iniciativas como por exemplo a invocação de métodos nativos e compiladores JIT (*Just In Time*).

Visando a programação paralela, uma forma intuitiva de simplificar a implementação de aplicações paralela é proporcionar a execução concorrente de procedimentos sequênciais. Para que essa execução concorrente seja correta, não devem existir depedências de dados entre os procedimentos. Dessa forma, é possível conseguir uma paralelização automática de programas sequenciais já existentes. A biblioteca ProActive [CAR 98] oferece isso através de invocação assíncrona de métodos, com objetos futuros e espera por necessidade. Não obstante, a localização dos objetos não é transparente.

Com o objetivo de melhor aproveitar os recursos de sistemas distribuídos, as tarefas de um programa paralelo podem ser lançadas em máquinas que estejam disponíveis. Em aglomerados de computadores, a disponibilidade pode ser dada pelo seu sistema de alocação de recursos, como por exemplo gerentes de filas ou escalonadores (PBS, etc.). Em um sistema distribuído mais genérico, tais como redes locais ou grades, pode-se associar a disponibilidade de computadores ao seu estado atual de utilização (ocupado ou ocioso).

O sistema Cadeo (Controle e Alocação Dinâmica de Estações Ociosas) objetiva a programação de sistemas distribuídos de forma simples e para isso concilia as características citadas nos parágrafos anteriores. Ele disponibiliza uma plataforma dinâmica

para a execução de aplicações paralelas e distribuídas, a qual é chamada de aglomerado dinâmico. Tal plataforma é composta por computadores de um sistema distribuído qualquer (aglomerados, redes locais ou grades de computadores). Um computador é dito ocioso quando estiver disponível para a execução de aplicações paralelas do Cadeo.

A principal característica do Cadeo é proporcionar transparência, tanto da localização quanto da dinamicidade, para o aglomerado dinâmico. Ao esconder a localização dos computadores que executam tarefas, tem-se uma maior facilidade na utilização do sistema. Dessa forma, o programador apenas lança tarefas de uma aplicação e a determinação de onde ocorrerá a sua execução fica por conta do Cadeo. Os computadores que compõem os aglomerados dinâmicos podem estar ociosos apenas por determinados períodos. Tal característica configura um cenário dinâmico, com frequente inclusão e exclusão de computadores. O Cadeo é responsável por controlar tal dinamicidade e fazer com que ela não interfira na execução da aplicação.

O paralelismo é obtido através da execução concorrente de métodos sequencialmente invocados, desde que não haja dependência de dados. Assim, uma chamada assíncrona de método do ProActive é o que se considera uma tarefa paralela no Cadeo. Também com o ProActive é possível que objetos de computadores que deixam de estar disponíveis sejam migrados. Assim, o sistema garante que todos os resultados das tarefas que foram lançadas serão obtidos corretamente.

O Cadeo proporciona ao programador uma interface bastante simples. Um programa paralelo que executará no Cadeo seguirá o modelo de programação padrão do Java. A interação deste com o Cadeo se dará através da herança de métodos e propriedades. O programador elabora seu programa nos mesmos moldes de um programa sequencial e tanto a distribuição quanto a localização dos procedimentos será realizada pelo Cadeo.

# Referências

[BAK 99] BAKER, M.; BUYYA, R. Cluster computing at a glance. In: BUYYA, R. (Ed.). **High performance cluster computing**. Upper Saddle River, NJ: Prentice Hall PTR, 1999. v.1, Architectures and Systems, p.3–47. Chap. 1.

[CAR 98] CAROMEL, D.; KLAUSER, W.; VAYSSIERE, J. Towards seamless computing and metacomputing in java. In: CONCURRENCY PRACTICE AND EXPERIENCE, 1998. **Anais...** Wiley & Sons: Ltd., 1998. v.10, n.11–13, p.1043–1061. http://www-sop.inria.fr/oasis/proactive/.

[FOS 98] FOSTER, I.; KESSELMAN, C. Globus: A toolkit-based grid architecture. In: **The grid**: blueprint for a future computing infrastructure. [S.l.]: MORGAN-KAUFMANN, 1998. p.259–278.

[POL 96] POLLATOS, S.; CANDLIN, R. Parallelism on a network of workstations. In: TDP96: TELECOMMUNICATION - DISTRIBUTION - PARALLELISM, 1996, Agelonde, La Londe Les Maures, France. **Proceedings...** [S.l.: s.n.], 1996. p.439–454? Parallelisme sur un Reseau de Stations de Travail.