Escalonamento Dinâmico de Programas MPI com Migração de Processos

Marcelo Veiga Neves, Tiarajú Asmuz Diverio, Nicolas Maillard

Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil
{mvneves, diverio, nmaillard}@inf.ufrgs.br

Resumo

A biblioteca de comunicação MPI tornou-se, nos últimos anos, um padrão para programação em *clusters*. No entanto, a norma MPI não prevê nenhuma especificação quanto ao escalonamento de processos. Este fica a cargo do programador, que deve proporcioná-lo a fim de obter maior eficiência na execução das aplicações.

Quanto à classificação, o escalonamento pode ser dividido em dois grandes grupos: estático ou dinâmico [CAS 88]. No escalonamento estático, as decisões sobre a distribuição dos processos são tomadas antes da execução da aplicação e colocadas em prática no lançamento. Neste caso, para que o escalonamento tenha um bom resultado, é necessário conhecer previamente as características do ambiente e do comportamento da execução da aplicação. Uma solução para isso é executar a aplicação uma vez, obter informações sobre seu comportamento e, então, executá-la novamente utilizando estas informações para guiar o escalonamento [SIL 2006]. No entanto, esta abordagem impõem o sobrecusto de uma execução prévia e restringe o escopo de aplicações que podem ser escalonadas àquelas que mantém o mesmo comportamento após execuções sucessivas.

No escalonamento dinâmico, por outro lado, as decisões são tomadas em tempo de execução, isto é, os processos podem ser redistribuídos durante a execução da aplicação. Se for necessário alterar a localização dos processos para um nó distinto, faz-se necessário a utilização de algum mecanismo de migração de processos. Apesar da norma MPI não especificar nada sobre migração, existem algumas iniciativas que se propõem adicionar essa funcionalidade ao MPI.

As informações necessárias para a tomada de decisão do escalonamento podem ser obtidas a partir de um grafo que represente a aplicação. O uso de grafos para representar aplicações paralelas é uma abordagem que já foi explorada por uma série de ambientes, tais como Cilk [BEN 2000], Athapascan [CAV 99] e Satin [NIE 2004]. A partir de um grafo de fluxo de dados (DFG), por exemplo, é possível extrair o volume de comunicação entre os processos. Um escalonador, neste caso, poderia agrupar processos que trocam muitas informações em uma mesma máquina e, desta forma, reduzir o volume de dados trafegados na rede.

A proposta deste trabalho é permitir escalonamento dinâmico e automático em aplicações MPI utilizando, para isso, migração de processos e análise de grafos. Com isso, espera-se aumentar o desempenho de aplicações MPI através da redução de comunicação.

A primeira parte do trabalho consistiu de um levantamento e analise de mecanis-

mos para migração e *checkpoint* de processos MPI. A maior parte destes mecanismos apresentaram restrições técnicas que limitam a utilização prática ou foram projetados para alguma implementação MPI pouco utilizada, ficando restritas ao uso acadêmico. LAM/MPI, por outro lado, é uma implementação da norma MPI que possui um grande número de usuários e apresenta um bom desempenho quando comparada com outras implementações. Além disso, LAM/MPI possui um suporte a *checkpoint/restart* funcional que foi utilizado com sucesso para realizar migração de processos. A aplicação de teste utilizada foi o *benchmark* High-Performance Linpack (HPL), na qual foi possível redistribuir os processos entre os nós do *cluster* durante sua execução.

O DFG de uma aplicação pode ser construído através da análise das primitivas de troca de dados do MPI. Isto pode ser feito de forma manual pelo programador durante a codificação da aplicação. Neste trabalho, optou-se por utilizar uma biblioteca de geração automática de DFGs em programas MPI, que foi projetada em um outro trabalho [SIL 2005]. Esta biblioteca também realiza o particionamento do grafo de acordo com os objetivos do escalonamento, porém de forma estática. O próximo passo do trabalho será, então, adaptar essa biblioteca para funcionar também de forma dinâmica a fim de permitir o particionamento do grafo e posterior redistribuição dos processos, através de migração, durante a execução da aplicação.

Referências

[BEN 2000] BENDER, M. A.; RABIN, M. O. Scheduling Cilk multithreaded computations on processors of different speeds. In: ACM SPAA'2000, 2000. **Proceedings...** [S.l.: s.n.], 2000. p.13.

[CAS 88] CASAVANT, T.; KUHL, J. A taxonomy of scheduling in general-purpose distributed computing systems. **IEEE Transactions on Software Engineering**, Los Alamitos, CA, USA, v.14, n.2, p.141–154, 1988.

[CAV 99] CAVALHEIRO, G. G. H. **Athapascan-1** : interface générique pour l'ordonnancement dans un environnement d'exécution parallèle. PhD Thesis, INPG - IMAG, Grenoble, December 1999.

[NIE 2004] NIEUWPOORT, R. V. van et al. Satin: simple and efficient java-based grid programming. In: JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING PRACTICE, 2004. **Anais...** [S.l.: s.n.], 2004.

[SIL 2005] SILVA, R. E. et al. Automatic data-flow graph generation of mpi programs. In: SBAC-PAD '05: PROCEEDINGS OF THE 17TH INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE ON HIGH PERFORMANCE COMPUTING, 2005, Washington, DC, USA. **Anais...** IEEE Computer Society, 2005. p.93–100.

[SIL 2006] SILVA, R. E. **Escalonamento Estático de Programas-MPI**. 2006. Master Thesis — Instituto de Informática, UFRGS, Porto Alegre.