Escalonamento Dinâmico de Processos MPI*

Márcia C. Cera, Nicolas Maillard e Philippe A. O. Navaux

Universidade Federal do Rio Grande do Sul
Instituto de Informática - UFRGS Campus do Vale - Porto Alegre, RS
{mccera, nmaillard, navaux}@inf.ufrgs.br

Resumo

A biblioteca de comunicação MPI [GRO 94] é um padrão para programação paralela com troca de mensagens e é a mais utilizada em sua categoria. O padrão MPI-1.2 permite apenas a criação estática de processos (ao disparar a aplicação). Nesse padrão, um conjunto de processos é definido no início da execução e se mantém o mesmo até seu término. A estaticidade na criação de processos, particularmente bem adaptado à programação em agregados de computadores [BAK 99], dificulta o uso de MPI em ambientes dinâmicos, onde o conjunto de computadores varia constantemente. Buscando preencher essa lacuna, foi definida a norma MPI-2 que suporta a criação dinâmica de processos (em tempo de execução). As versões mais recentes da LAM-MPI implementam grande parte das primitivas definidas por MPI-2. Recentemente (janeiro/2005), a MPI-CH também passou a implementar primitivas MPI-2. Esse comportamento reflete o crescente interesse pela programação dinâmica, incentivado pelo destaque que as grades computacionais [FOS 2003] vêm recebendo nos últimos anos.

Assim como as normas anteriores, MPI-2 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 paralelas. Nesse contexto, foi planejada a biblioteca βMPI-2 [CER 2005] com o objetivo de possibilitar escalonamento dinâmico de processos MPI-2 automaticamente. Ela atua através da sobrecarga de algumas primitivas da biblioteca MPI. Assim, a lógica de programação das aplicações MPI não é alterada uma vez que a interface das primitivas se mantém a mesma.

A biblioteca βMPI-2 é composta por uma interface de primitivas e um escalonador. A interface de primitivas é responsável pela sobrecarga de funções MPI. Durante a etapa de ligação de um programa, todas as primitivas MPI redefinidas em βMPI-2, são sobrecarregadas. A sobrecarga possibilita a coleta de informações dos processos em execução e estabelece a interação deles com o escalonador da biblioteca. As informações coletadas irão compor um grafo de fluxo de dados que será atualizado dinamicamente. O escalonador de βMPI-2 é responsável por, dinamicamente, distribuir os processos entre os computadores disponíveis e por liberar a execução destes conforme suas prioridades.

A βMPI-2 emprega um algoritmo de escalonamento em lista (*list-scheduling*) [ELR 94] para escalonar os processos da aplicação baseado em uma ordem de prioridades. A fim de obter eficiência, é fundamental que a prioridade dos processos defina uma ordem de execução compatível com suas dependências causais. Nas bibliotecas de *threads* ou em Cilk [BLU 95], geralmente a dependência é do tipo "os filhos dependem do pai". Em Athapascan-1 [GAL 98], a dependência está descrita nos dados passados

*Este trabalho faz parte do projeto Clumssy que é patrocinado pela HP Brasil.

como parâmetro aos processos. No caso de βMPI-2 o modo natural de definir a relação de causalidade é troca de uma mensagem, onde um processo A depende de B se ele tem que receber (MPI_Recv) dados de B.

Um protótipo da βMPI-2 vem sendo implementado para a validação da biblioteca. Primeiramente, foram definidas as primitivas MPI a serem sobrecarregadas, são elas: MPI_Comm_spawn, MPI_Comm_spawn_multiple e MPI_Finalize. A sobrecarga das primitivas MPI_Comm_spawn e MPI_Finalize e a interação dos processos com o escalonador estão representadas nas figuras 1 e 2, respectivamente. Na primeira especificação do escalonador de βMPI-2, a lista de processos é centralizada para simplificar a implementação. Futuramente, por questões de escalabilidade, pretende-se oferecer uma implementação distribuída dessa lista. O gerenciamento da fila de processos foi implementada, inicialmente, seguindo o algoritmo de *Round-Robin* onde as tarefas são escalonadas conforme sua ordem de inclusão. Buscando um algoritmo mais eficiente no contexto da βMPI-2 uma implementação de *Heap* com prioridades foi adaptada ao escalonador.

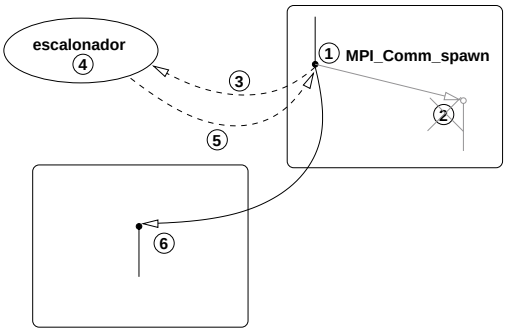

Figura 1: Sobrecarga de MPI_Comm_spawn

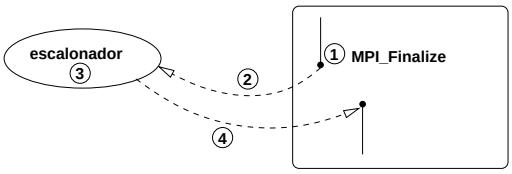

Figura 2: Sobrecarga de MPI_Finalize

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.

[BLU 95] BLUMOFE, R. D. et al. Cilk: an efficient multithreaded runtime system. **ACM SIGPLAN Notices**, v.30, n.8, p.207–216, Aug. 1995.

[CER 2005] CERA, M. C. et al. Betampi-2: uma biblioteca para o escalonamento dinâmico de processos mpi. In: III WORKSHOP PPD/UFRGS - WSPPD'2005, 2005, Porto Alegre - RS. **Anais...** [S.l.: s.n.], 2005.

[ELR 94] EL-REWINI, H.; LEWIS, T. G.; ALI, H. A. **Task scheduling in parallel and distributed systems**. USA: Prentice Hall Series in Innovative Technology, 1994.

[FOS 2003] FOSTER, I.; KESSELMAN, C. **The grid**: blueprint for a new computing infrastructure. 2.ed. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2003. 800p.

[GAL 98] GALILÉE, F. et al. **Athapascan-1: on-line building data flow graph in a parallel language**. Paris, France: [s.n.], 1998. 88–95p.

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