Aplicações Dinâmicas MPI-2 com *threads*[*]

João Vicente Lima[1], **Nicolas Maillard**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS

{joao.lima,nicolas}@inf.ufrgs.br

Introdução

As aplicações dinâmicas apresentam características próprias quanto ao particionamento de dados e quantidade de trabalho a ser realizado. O particionamento de dados por decomposição recursiva é um exemplo onde cada divisão pode ser atribuída a um novo processo como ocorre no modelo Divisão-e-Conquista (D&C). Uma distribuição de dados com diferentes cargas de trabalho caracteriza aplicações não uniformes [Grama et al. 2003] nas quais os processos são criados conforme a demanda por processamento. A implementação de aplicações desse tipo está relacionada a ferramenta de programação em questão e entre elas deve-se mencionar o Cilk [Blumofe et al. 1995] para máquinas SMP e outras voltadas para *clusters* SMP (*Symmetric multiprocessor*) como KAAPI [Gautier et al. 2007] e AMPI [Huang et al. 2004]. Estes ambientes provêem primitivas para execução dinâmica e escalonamento com balanceamento de carga mas não seguem ou propõem uma interface padrão. Este trabalho utiliza para seus propósitos uma interface padrão de programação largamente aceita que é o MPI.

O MPI é uma interface de programação padrão e amplamente utilizada em aplicações com troca de mensagens. A versão MPI-2 [Geist et al. 1996] incorpora a criação dinâmica de processos e a utilização de várias *threads* concorrentemente. A união destas duas características habilita programas MPI-2 a explorar o dinamismo e o compartilhamento de memória em máquinas SMP *multicore*, onde este trabalho se insere. O objetivo deste trabalho é avaliar alterações no estilo de programação e benefícios ao lançar *threads* ao invés de processos dinamicamente com o padrão MPI-2 em diferentes aplicações definidas conforme as características citadas anteriormente.

Metodologia e Resultados Esperados

A primeira fase do trabalho consistiu na avaliação do suporte a múltiplas *threads* entre as implementações MPI-2 disponíveis. Como poucos trabalhos abordam este aspecto, um conjunto de testes foi desenvolvido com o objetivo de avaliar a estabilidade e o desempenho das implementações MPI-2 [Lima 2008] com relação as principais funções da interface. Cada teste executa chamadas MPI-2 concorrentes entre as várias *threads* e verifica possíveis erros. A maior parte dos testes de estabilidade estão implementados e o MPICH2 é a implementação que apresenta mais estabilidade nas chamadas MPI-2.

O próximo passo é a escolha de aplicações MPI-2 em função das características de particionamento de dados e carga de trabalho em aplicações dinâmicas. A partir destas escolhas duas versões dos programas serão implementadas em que uma lança processos

[*]Trabalho financiado pela CAPES

e a outra cria *threads* dinamicamente, onde o número de processadores em cada nó definirá o limite de *threads* que um processo conterá. A especificação MPI-2 não descreve mecanismos de gerenciamento para as *threads*, dessa forma o programa identifica processos mas não o faz com *threads*. A substituição de processos (`MPI_Comm_spawn`) por *threads* (`pthread_create`) altera as chamadas comunicantes para prever se o destino é um processo ou um de seus fluxos de execução. No caso deste trabalho, o compartilhamento ou cópia de memória substituirá as trocas de mensagens entre *threads* de um mesmo processo de acordo com o tipo de acesso ao endereço de memória em questão após a comunicação.

As implementações e o estudo do estado da arte proporcionarão descrever as principais diferenças ao implementar programas com processos e depois com *threads*. Uma das alterações esperadas é a extinção de variáveis globais a fim de evitar acessos indevidos entre os fluxos de execução. Outra modificação prevista é o uso específico de variáveis globais e eliminação das transferências de memória no caso de endereços sem alteração de conteúdo entre as *threads* após a comunicação. A contribuição futura desta fase permitirá esconder as alterações do programa e atribuir esta tarefa para a implementação MPI-2 conforme o tipo de aplicação.

A avaliação dos resultados da execução em um *cluster* SMP *multicore* mostrará os ganhos no uso de várias *threads* ao invés de processos na criação dinâmica de tarefas com o padrão MPI-2. Dessa maneira a versão com processos terá por nó vários processos ao passo que a outra versão terá um processo com várias *threads*. A expectativa com relação aos vários fluxos por processo é que além do menor custo ao criar *threads* se obtenha ganhos de desempenho significantes em comunicação com o uso de memória compartilhada.

Referências

Blumofe, R. D., Joerg, C. F., Kuszmaul, B. C., Leiserson, C. E., Randall, K. H., and Zhou, Y. (1995). Cilk: An Efficient Multithreaded Runtime System. In *PPOPP '95*, pages 207–216, New York, NY, USA. ACM Press.

Gautier, T., Besseron, X., and Pigeon, L. (2007). KAAPI: A thread scheduling runtime system for data flow computations on cluster of multi-processors. In *PASCO '07*. ACM.

Geist, A., Gropp, W., Huss-Lederman, S., Lumsdaine, A., Lusk, E. L., Saphir, W., Skjellum, T., and Snir, M. (1996). MPI-2: Extending the Message-Passing Interface. In Bougé, L., Fraigniaud, P., Mignotte, A., and Robert, Y., editors, *Euro-Par, Vol. I*, volume 1123, Lyon, France. Springer.

Grama, A., Gupta, A., Karypis, G., and Kumar, V. (2003). *Introduction to Parallel Computing*. Addison-Wesley, 2th edition.

Huang, C., Lawlor, O., and Kalé, L. V. (2004). Adaptive MPI. In *LCPC 2003*, volume 2958/2004 of *Lecture Notes in Computer Science*, pages 306–322, College Station, Texas. Springer Berlin / Heidelberg.

Lima, J. V. (2008). MPI Thread Tests Suite. Disponível em: `http://mpi-thread-test.googlecode.com`.