Criação Dinâmica de Tarefas MPI.NET em Ambiente Heterogêneo

Fernando A. Afonso, Ismael Stangherlini, Nicolas Maillard

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

{faafonso,istangherlini,nicolas}@inf.ufrgs.br

1. Introdução

MPI *(Message Passing Interface)* é a especificação padrão da indústria para o desenvolvimento de aplicações de alto-desempenho com comunicação via troca de mensagens principalmente em ambientes com memória distribuída. Por se tratar de uma especificação de biblioteca existem diferentes implementações, porém a norma somente especifica interfaces para as linguagens de programação C, Fortran e C++ fazendo com que somente essas linguagens possuam implementações de bibliotecas MPI fortemente suportadas [Gropp 1995].

Por um lado as linguagens de programação evoluíram para linguagens de alto-nível orientadas a objetos como Java e C# permitindo a execução de programas em ambientes heterogêneos. Por outro lado a especificação MPI evoluiu para MPI-2 oferecendo suporte a criação dinâmica de tarefas, E/S paralela e operações remotas de memória [Gropp et al. 1999].

Tendo em vista que MPI é um padrão muito popular no contexto de HPC *(High Performance Computing)* logo surgiram projetos visando integrar MPI com as novas linguagens de programação. Entre esses diversos projetos podemos destacar Java (JavaMPI, mpiJava, PJMPI, MPJava) [Getov et al. 1999] [WenSheng 2000] [Pugh and Spacco 2003], C# (MPI.NET, Pure MPI.NET) [Gregor and Lumsdaine 2008] [PMP], Python (pympi), Ruby (ruby-mpi).

A falta de um ambiente de programação de alto-nível baseado em MPI que ofereça criação dinâmica de processos e alto desempenho motiva a pesquisa sobre projetos que permitam a utilização de MPI nesses ambientes alto-nível, sendo que o projeto MPI.NET demonstrou bons resultados de desempenho [Gregor and Lumsdaine 2008], bem como uma boa interface de programação, deixando a desejar no quesito de criação dinâmica de tarefas.

Portanto, a forte motivação deste trabalho está em explorar as possibilidades de criação dinâmica de tarefas MPI dentro do ambiente de programação alto-nível provido pela biblioteca MPI.NET.

2. Metodologia e Resultados Esperados

Durante o estudo da biblioteca MPI.NET foi constatado que para implementar as funções de criação dinâmica de tarefas será primeiramente necessário substituir a biblioteca MPI-C a qual a biblioteca MPI.NET utiliza em sua camada mais baixa para realizar as chamadas MPI devido a essa biblioteca não oferecer suporte a criação dinâmica de tarefas.

O ciclo de desenvolvimento deste trabalho será baseado primeiramente no estudo das possibilidades de tornar a biblioteca MPI.NET capaz de executar em ambientes heterogêneos.

Após a biblioteca permitir execução de processos MPI em ambientes heterogêneos, serão então estudadas as possibilidades de adicionar funções para criação dinâmica de tarefas e funções para gerenciamento desses processos criados dinâmicamente.

A biblioteca será validada através da utilização de aplicações que criem processos dinâmicamente, como por exemplo aplicações baseadas no modelo de Divisão e Conquista.

Por fim a biblioteca terá seu desempenho testado ao lado de uma biblioteca MPI-C para verificar se o *overhead* de estar utilizando uma linguagem de programação com maiores níveis de abstração não irá ser muito impactante nos resultados de desempenho. Segundo [Gregor and Lumsdaine 2008] o desempenho da biblioteca se mantém muito similar ao da linguagem C.

Como resultado final espera-se uma biblioteca que possua uma interface de programação com alto-nível de abstração a qual possua bom desempenho e permita a programação intuitiva de programas MPI baseados em criação dinâmica de tarefas pelos programadores C#, e que esses programas possam ser executados sobre ambientes heterogêneos. A biblioteca terá seu código livre permitindo que outras funcionalidades possam ser agregadas.

Referências

Pure mpi.net. Disponível em http://www.purempi.net/. Acesso em 14 de nov. de 2008.

Getov, V., Gray, P., and Sunderam, V. (1999). Mpi and java-mpi: contrasts and comparisons of low-level communication performance. In *Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*, page 21, New York, NY, USA. ACM.

Gregor, D. and Lumsdaine, A. (2008). Design and implementation of a high-performance mpi for c# and the common language infrastructure. In *PPoPP '08: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pages 133–142, New York, NY, USA. ACM.

Gropp, W. (1995). *Using mpi : portable parallel programming with the message-passing interface*. Cambridge : Mit.

Gropp, W., Thakur, R., and Lusk, E. (1999). *Using MPI-2: Advanced Features of the Message Passing Interface*. MIT Press, Cambridge, MA, USA.

Pugh, B. and Spacco, J. (2003). Mpjava: High-performance message passing in java using java.nio.

WenSheng, T. W. Y. H. Y. (2000). Pjmpi: pure java implementation of mpi. In *High Performance Computing in the Asia-Pacific Region*, volume 1, pages 533 – 535.