Avaliação de Desempenho da Criação Dinâmica de Processos MPI.NET

Fernando A. Afonso, 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,nicolas}@inf.ufrgs.br

1. Introdução

Esse artigo trata do desenvolvimento e avaliação de desempenho da criação dinâmica de processos na biblioteca MPI.NET [Gregor and Lumsdaine 2008]. Existem diversos projetos de bibliotecas MPI para linguagens de programação não suportadas pela norma como Java e C#. Essas linguagens permitem a simplificação das chamadas MPI bem como o suporte a comunicação de objetos devido a sua capacidade de introspecção e serialização. No entanto esses projetos suportam somente a norma MPI-1. Certos algoritmos, por exemplo Divisão e Conquista, obtêm vantagem na utilização da criação dinâmica de processos, a qual está presente na norma MPI-2 [Gropp et al. 1999], motivando a implementação dessas funcionalidades na biblioteca MPI.NET.

2. Criação Dinâmica de Processos MPI.NET

Dentre as diversas bibliotecas MPI estudadas, a biblioteca MPI.NET apresentou a melhor combinação de API/desempenho. Essa biblioteca escrita em C# pode ser utilizada pelas linguagens que fazem parte da plataforma .Net. Ela faz chamadas as bibliotecas MPI nativas, podendo executar sobre qualquer biblioteca MPI.

As funções `MPI_Comm_spawn`, `MPI_Comm_spawn_multiple` e `MPI_Comm_get_parent` da API MPI-C são responsáveis pela criação dinâmica de processos. Para a implementação dessas funções na biblioteca MPI.NET, foi tomada como base a API MPI-2 para. Foram criadas as interfaces para permitir as chamadas nativas à biblioteca MPI-C. Para cada parâmetro utilizado pelas funções, foi especificado o tipo de dado C# mais compatível em relação ao parâmetro da chamada nativa em C, a qual é chamada pela biblioteca MPI.NET.

Os métodos foram inseridos na classe `Communicator`, sendo que as interfaces desses métodos são mais simples que a interface oferecida pelo C e C++ e ao mesmo tempo permitem que o usuário utilize todos os parâmetros permitidos por essas linguagens. Para isso foram escritas cinco sobrecargas de método as quais permitem que o usuário especifique somente o nome do programa a ser criado dinamicamente ou faça uso completo dos parâmetros.

3. Resultados Obtidos

Para avaliar o desempenho foram executados alguns *benchmarks* sintéticos comparando a biblioteca MPI.NET em relação a biblioteca MPI-C++. Primeiramente foi executado um *benchmark* o qual mede o tempo de execução de n chamadas ao método `Spawn`. Esse *benchmark* revelou que a criação de um processo é diversas vezes mais lenta em C# do

que em C++, levando respectivamente 0.1507s e 0.0209s para serem executadas. Essa grande diferença se deve ao fato de que para cada novo processo C# uma nova máquina virtual tem de ser inicializada.

Para avaliar o impacto da diferença de tempo de criação de processos foi executado um *benchmark* que calcula o i-ésimo número da sequência de Fibonacci (figura 1 gráfico A). Esse *benchmark* foi programado para executar de maneira recursiva criando dois novos processos para cada n. Dessa forma são criados diversas tarefas de baixa granularidade permitindo a avaliação do impacto da criação dos processos no tempo de execução. Os resultados demonstram um impacto negativo no tempo final de execução.

Foi executado também o cálculo do fractal de Mandelbrot (figura 1 gráfico B), um *benchmark* que cria menos processos com maior granularidade. Esse *benchmark* demonstrou que quando as tarefas possuem maior granularidade o tempo de criação dos processos possui menor impacto no tempo de execução. Maiores detalhes sobre os *benchmarks* realizados podem ser vistos em [Afonso and Maillard 2009].

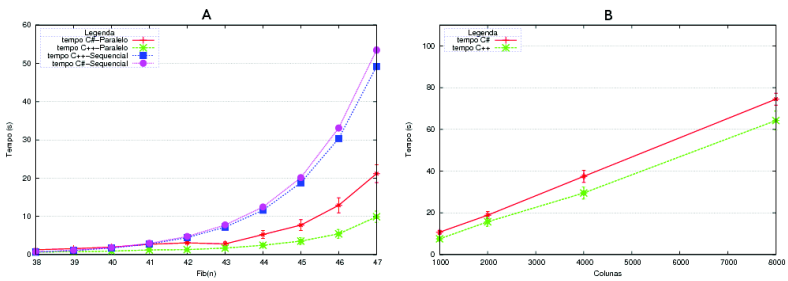

Figura 1. Execução dos benchmarks Fibonacci e Mandelbrot

4. Trabalhos em Andamento

Uma vez que os testes de desempenho demonstraram que o tempo de criação dos processos impacta de forma significativa no tempo de execução das aplicações, foi desenvolvida uma versão da biblioteca a qual cria threads e processos de maneira intercalada, uma vez que a criação de threads é mais rápida. Em testes preliminares essa técnica demonstrou bons resultados.

Referências

Afonso, F. and Maillard, N. (2009). Implementação da criação dinâmica de tarefas na biblioteca mpi.net. *Scientia (Unisinos)*.

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