MPI-Rastro: Rastreamento e Análise de Aplicações Paralelas MPI

Lucas Mello Schnorr*, Philippe O. Alexandre Navaux

Universidade Federal do Rio Grande do Sul – Instituto de Informática
Av. Bento Gonçalves, 9500 – Porto Alegre – RS
{lmschnorr,navaux}@inf.ufrgs.br

Resumo

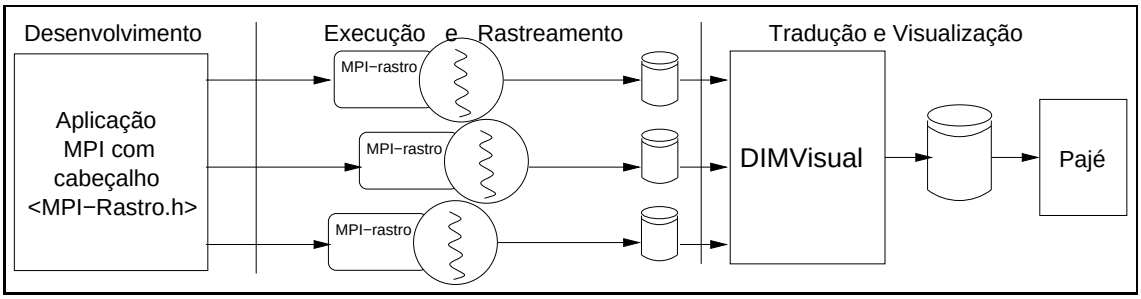
Aplicações paralelas são normalmente desenvolvidas em três etapas: programação, correção de erros e depuração de desempenho. As duas primeiras etapas dizem respeito a programação utilizando alguma biblioteca de programação, como MPI, e a correção de erros através da utilização de um depurador seqüencial. A última etapa, de depuração de desempenho, é vital, pois é através dela que a aplicação paralela pode ser otimizada para se obter um alto desempenho na sua execução. Nesta etapa, o programador faz uso de ferramentas e instrumentações em seu programa com o intuito de observar o comportamento da aplicação. Em geral, o resultado dessa observação deve fornecer dados e informações que são úteis ao programador para que ele possa melhorar o desempenho de sua aplicação.

Assim, este trabalho tem como foco a etapa de depuração de desempenho. Existem várias caminhos possíveis, como por exemplo, a análise estatística dos tempos de execuções de regiões do código, verificação da comunicação da aplicação paralela e visualização do comportamento dos programas após ou durante a execução. Mesmo assim, ainda não há um consenso de qual é o melhor método para ser aplicado. Em geral, isto ocorre porque as diferentes formas de construir aplicações paralelas nas diferentes arquiteturas e topologias de rede obrigam o uso de técnicas específicas a cada caso.

Ferramentas de instrumentação foram criadas no sentido de tentar padronizar a forma de se analisar o desempenho de aplicações paralelas. Atualmente, a implementação MPICH [1] trás consigo uma biblioteca de rastreamento chamada MPE - *Multi-processing Environment*. Ela tem como objetivo fornecer ao programador uma forma de se registrar o comportamento da aplicação paralela durante a sua execução. Esse registro é descrito por eventos que são analisados após o término da aplicação, por uma ferramenta específica. Os problemas desta abordagem são o formato do rastro de execução e a a falta do rastreamento de determinado conjunto de operações que podem ser usadas nas aplicações MPI. Isto porque nem todas as operações MPI são monitoradas pela MPE.

Neste contexto, foi construída a biblioteca MPI-Rastro. O trabalho teve como objetivo suprir as lacunas deixadas pela MPE, sem forçar o programador a alterar sua aplicação, utilizando métodos que resultem em uma baixa intrusão na aplicação. O trabalho também consistiu na construção de um módulo MPI-Rastro para o modelo de integração de dados DIMVisual [3], sendo possível então analisar os rastros juntamente com dados de ferramentas como o Ganglia [2]. A análise é feita através da ferramenta genérica de visualização Pajé [4]. A figura a seguir ilustra as três etapas do monitoramento quando se usa a biblioteca MPI-Rastro, e sua relação com o DIMVisual e o Pajé.

*Trabalho realizado com financiamento CAPES



A biblioteca MPI-Rastro tem como objetivo o rastreamento de aplicações desenvolvidas com qualquer implementação MPI. Além disso, a construção da biblioteca pretende ter as seguintes características: extensibilidade, permitindo a adição de novos tipos de eventos de acordo com a necessidade; configurabilidade, permitindo especificar o que quer se monitorar; baixa intrusão, alterando o mínimo o comportamento natural da aplicação; transparência, podendo ser utilizada de forma simples pelo programador.

Com essas características, MPI-Rastro pode se adaptar a diferentes situações e necessidades. Por exemplo, o programador pode ativar o monitoramento apenas em regiões específicas de seu programa ou então deixar que a biblioteca monitore toda a execução da aplicação. Essa ativação pode ser programada na própria aplicação através do uso das funções *MPIRastro_Enable*, *MPIRastro_Disable* e *MPIRastro_Update_Mask (int mask)*. Com essas funções, o programador tem controle do que é monitorado, e quando.

Outra situação acontece quando já se tem uma aplicação MPI e deseja monitorá-la: usando a MPI-Rastro, ele não precisa alterar muito o código de sua aplicação para obter informações de rastreamento. Caso o programador queira rastrear toda a aplicação, basta ele incluir um cabeçalho em cada arquivo com código fonte e compilar a aplicação com a biblioteca. Assim, o programador poderá depois analisar visualmente a aplicação.

Por causa das características da MPI-Rastro, acredita-se que a MPI-Rastro seja uma solução mais completa que as outras ferramentas de rastreamento de aplicações MPI. Sua flexibilidade e facilidade de uso em conjunto com as ferramentas DIMVisual e Pajé formam um sistema completo para o monitoramento e análise das aplicações paralelas desenvolvidas dentro da especificação MPI. Como trabalhos futuros, pode-se trabalhar na análise da sobrecarga e no uso em aplicações MPI de produção.

Referências

[1] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: portable parallel programming with the message-passing interface*. MIT Press, Cambridge, MA, USA, 1994.

[2] Matthew L. Massie, Brent N. Chun, and David E. Culler. The ganglia distributed monitoring system: Design, implementation, and experience. *Parallel Computing*, 30(7):817–840, 2004.

[3] Lucas Mello Schnorr, Benhur de Oliveira Stein, and Philippe Olivier Alexandre Navaux. Dimvisual: Data integration model for visualization of parallel programs behavior. In *CCGRID '06: Proceedings of the 6th International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2006.

[4] Benhur Stein, Jacques Chassin de Kergommeaux, and Pierre-Eric Bernard. Pajé, an interactive visualization tool for tuning multi-threaded parallel applications. *Parallel Computing*, 26:1253–1274, 2000.