Escalonamento dinâmico em nível aplicativo sensível à arquitetura e às dependências de dados entre as tarefas*

Rodolfo Migon Favaretto†, Guilherme P. B. Cousin, Deives M. Kist, Gerson Geraldo H. Cavalheiro, Maurício Lima Pilla

Programa de Pós-Graduação em Computação
Centro de Desenvolvimento Tecnológico – Universidade Federal de Pelotas (UFPel)
Campus Porto – Rua Gomes Carneiro, 1 – 96010-610 – Pelotas – RS – Brazil

{rmfavaretto,gpbcousin,dmkist,gerson.cavalheiro,pilla}@inf.ufpel.edu.br

Resumo. *Neste trabalho busca-se modelar e implementar uma estratégia de escalonamento para programas multithread em arquiteturas NUMA. Esta estratégia considera, em nível aplicativo, a localidade física dos dados e as dependências entre as tarefas em tempo de execução para realizar o escalonamento dos threads do programa. O objetivo é minimizar o impacto das diferentes latências no acesso aos módulos de memória pelos processadores.*

1. Introdução

O aumento do número de processadores nas arquiteturas SMPs (*Symmetric Multi-Processor*) atuais acaba gerando um problema em função da disputa entre os processadores por acesso ao barramento, quando do acesso à memória. Como alternativa, têm-se as arquiteturas NUMA (*Non-Uniform Memory Access*). O interesse no uso destas arquiteturas é ainda maior quando considera-se que estas oferecem uma relação mais atraente entre custo e desempenho quando comparadas às arquiteturas SMPs [Kumar et al. 2004].

Porém, para que se consiga obter bons índices de desempenho em seus programas, é desejável que o programador conheça os detalhes da arquitetura disponível e inclua no código da aplicação instruções específicas para o mapeamento de suas atividades sobre os recursos de processamento disponíveis. As arquiteturas NUMA têm como característica diferentes tempos de acesso à memória para diferentes combinações de núcleos de processamento e endereços de memória, uma vez que o espaço de endereçamento encontra-se divido em diferentes módulos de memória e que cada um destes módulos encontra-se fisicamente próximo a um determinado subconjunto de processadores.

Trabalhos como [Olivier et al. 2012] trazem estratégias de escalonamento para arquiteturas NUMA. Neste trabalho, busca-se aumentar o desempenho das aplicações paralelas pela gestão, em nível aplicativo, dos acessos à memória via escalonamento considerando, em um primeiro nível, as entradas e saídas produzidas pelas diferentes atividades paralelas (tarefas) criadas pelo programa em execução com suas dependências e, num segundo nível de escalonamento, considerando a topologia física da máquina, com as assimetrias de acesso provenientes das estruturas de memória.

*FAPERGS/PqG (11/1065-1), PRONEX/FAPERGS/CNPq (10/0042-8)
†Bolsista de Mestrado FAPERGS

O objetivo principal deste trabalho, então, consiste na modelagem de uma estratégia de escalonamento que contemple esses dois níveis. Essa estratégia foi inserida no núcleo de escalonamento do Anahy [Cavalheiro et al. 2007], um ambiente de programação e execução *multithread* dinâmico projetado para arquiteturas multiprocessadas, onde seu núcleo de escalonamento foi concebido para explorar, no nível aplicativo, informações sobre a estrutura do programa paralelo.

2. Metodologia de desenvolvimento

Após realizada a especificação e parametrização da estratégia e das heurísticas utilizadas pelo escalonador para realizar a distribuição e o balanceamento de carga, a estratégia foi implementada na forma de um protótipo para avaliação. Para tal, foi utilizada a ferramenta Charm++, pois oferece uma estrutura para o balanceamento de carga com estatísticas da aplicação em tempo de execução.

Para extrair as características da arquitetura consideradas pelo escalonador no momento da decisão, foi utilizada a ferramenta HwLoc (abreviação para *Hardware Locality*). Trata-se de uma ferramenta para coletar as informações da topologia da máquina e também informações sobre memória e interconexões, foi estendida por [Pilla et al. 2012] para fornecer informações de latências de acessos e largura de banda na transmissão dos dados. Após a validação da estratégia em Charm++, a estratégia foi inserida no núcleo de execução do Anahy, a fim de contribuir com o aumento de desempenho desta ferramenta.

3. Experimentos e resultados

Atualmente, o trabalho encontram-se em fase de experimentação, onde diversos testes estão sendo realizados para se aferir o desempenho da estratégia desenvolvida. Estes testes consistem em aplicações sintéticas, capazes de explorar as características deste tipo de escalonamento, considerando diferentes estruturas de programas paralelos.

Os testes serão compostos por três aplicações e serão executados em 3 arquiteturas distintas, sendo duas NUMA e uma SMP, para fins de comparação. Os resultados obtidos com Anahy serão comparados com os dados obtidos pelas principais ferramentas *multithread* disponíveis, são elas: Cilk Plus, OpenMP e Threading Building Blocks.

Referências

Cavalheiro, G., Gaspary, L., Cardozo, M., and Cordeiro, O. (2007). Anahy: A programming environment for cluster computing. In *High Performance Computing for Computational Science - VECPAR 2006*, volume 4395, pages 198–211.

Kumar, R., Tullsen, D. M., Ranganathan, P., Jouppi, N. P., and Farkas, K. I. (2004). Single-isa heterogeneous multi-core architectures for multithreaded workload performance. In *Proceedings. 31st AISCA, 2004.*, pages 64 – 75.

Olivier, S. L., Porter, A. K., Wheeler, K. B., Spiegel, M., and Prins, J. F. (2012). Openmp task scheduling strategies for multicore NUMA systems. *HPCA*, pages 110–124.

Pilla, L. L., Navaux, P. O., Ribeiro, C. P., Coucheney, P., Broquedis, F., Gaujal, B., and Mehaut, J.-F. (2012). Asymptotically optimal load balancing for hierarchical multicore systems. In *18th ICPADS, 2012 IEEE*, pages 236 –243.