Implementação da PLASMA para Arquiteturas Heterogêneas Multi-CPU e Multi-GPU em XKaapi

João V. F. Lima, Nicolas Maillard

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

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

Resumo. *Este trabalho objetiva ampliar os benchmarks usados no XKaapi através da implementação de tarefas em CUDA para a biblioteca PLASMA. Dessa forma, os algoritmos descritos e otimizados em PLASMA são usados para avaliação de desempenho de novas funcionalidades no XKaapi. Além disso, este trabalho possibilita a extensão do PLASMA para arquiteturas heterogêneas.*

1. Introdução

Hoje em dia, arquiteturas multi-CPU e multi-GPU têm sido amplamente utilizadas como uma alternativa eficiente e viável na busca por maior poder de processamento. Essas arquiteturas têm processadores heterogêneos em termos de poder de processamento e modelo de programação. Em muitos casos, algoritmos amplamente conhecidos precisam ser repensados e reescritos de forma que tenham proveito dos aceleradores.

PLASMA [Buttari et al. 2009] é uma biblioteca de algebra linear que consiste na reimplementação do LAPACK e ScaLAPACK para arquiteturas *multi-core*. Ela descreve os algoritmos em blocos ou *tiles* com o propósito de prover paralelismo de grão fino. Esses algoritmos são representados na forma de grafos onde os nós são tarefas e as arestas são dependências, semelhante a aplicações de *data-flow*. PLASMA oferece duas formas de escalonar suas tarefas: estática ou dinâmica. O escalonamento dinâmico usa a ABI QUARK [YarKhan et al. 2011] que é uma biblioteca interna ao PLASMA, mas que pode ser ligada e implementada por outras ferramentas de programação paralela.

XKaapi é um ambiente de programação que oferece paralelismo de tarefas com dependencias em arquiteturas multi-CPU e multi-GPU [Lima et al. 2012]. O XKaapi é uma reimplementação do KAAPI com suporte ao paralelismo de tarefas de grão fino, e inclui um conjunto de ABIs (QUARK e libGOMP) e APIs (C, Fortran e C++). Sua interface C++ suporta a definição de diferentes implementações de uma tarefa através de uma *Task Signature*. Ela permite descrever os parâmetros de uma tarefa e seus modos de acesso (leitura, escrita, etc.), assim como suas especializações para cada processador, CPU ou GPU. O modelo de programação abstrai o cálculo de dependências, o mapeamento de tarefas, assim como o gerenciamento de memória.

2. Objetivo e Resultados

Os resultados obtidos com a decomposição de Cholesky demonstram ganhos significativos com o roubo de tarefas em arquiteturas heterogêneas [Lima et al. 2012]. Todavia, a implementação de tais algoritmos não é trivial. Este trabalho objetiva ampliar o número de *benchmarks* usados no XKaapi através da implementação de tarefas em CUDA para

a biblioteca PLASMA. Dessa forma, os algoritmos descritos e otimizados em PLASMA são usados para avaliação de desempenho de novas funcionalidades no XKaapi. Além disso, este trabalho possibilita a extensão da PLASMA para arquiteturas heterogêneas.

PLASMA possui uma API interna chamada *core_blas* que implementa as tarefas CPU. A utilização de uma versão CUDA é possível através de uma extensão da ABI QUARK com a finalizade de registrar para cada tarefa CPU uma versão GPU em CUDA. A biblioteca *core_cuda* implementa as tarefas da *core_blas* em CUDA e registra seus métodos junto ao XKaapi. Dessa forma, o escalonador verifica se uma determinada tarefa possui uma versão GPU em tempo de execução e possibilita a execução de tarefas PLASMA em GPUs. Essa estratégia permite reaproveitar os algoritmos da PLASMA, e reduz drasticamente a quantia de adaptações necessárias para testes com XKaapi. A única alteração necessária é a inicialização das funções GPU (*core_cuda*) no código fonte do programa que utiliza um método PLASMA. Salienta-se que chamadas à API XKaapi não são necessárias devido ao uso do QUARK em XKaapi.

A Figura 1 ilustra os resultados preliminares com multiplicação de matriz, Cholesky, LU, e QR em uma arquitetura com quatro CPUs e oito GPUs. Os resultados demostram que os dois primeiros escalam conforme o tamanho da matriz. Porém, no caso das decomposições LU e QR, ainda há questões de desempenho a serem estudadas.

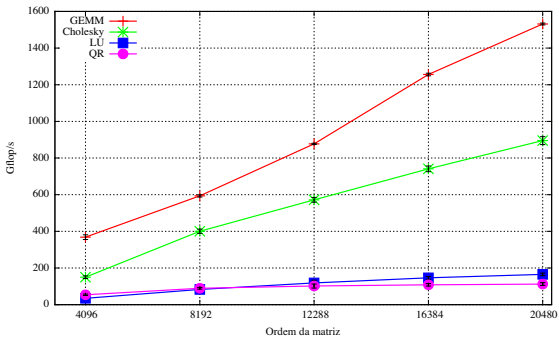

Figura 1. PLASMA com XKaapi em quatro CPUs e oito GPUs.

Este trabalho faz parte da colaboração entre o GPPD/UFRGS e a equipe francesa MOAIS (INRIA Rhône-Alpes) no desenvolvimento da ferramenta XKaapi.

Referências

[Buttari et al. 2009] Buttari, A., Langou, J., Kurzak, J., and Dongarra, J. (2009). A class of parallel tiled linear algebra algorithms for multicore architectures. *Parallel Computing*, 35(1):38–53.

[Lima et al. 2012] Lima, J. V. F., Gautier, T., Maillard, N., and Danjean, V. (2012). Exploiting Concurrent GPU Operations for Efficient Work Stealing on Multi-GPUs. In *Proc. of the 24th SBAC-PAD*, New York, USA. IEEE.

[YarKhan et al. 2011] YarKhan, A., Kurzak, J., and Dongarra, J. (2011). QUARK Users' Guide: QUeueing And Runtime for Kernels. Technical Report ICL-UT-11-02, University of Tennessee.