Análise de Aplicação baseada em Tarefas em Arquitetura Híbrida CPU/GPU

Vinícius Garcia Pinto[1,2], Lucas Mello Schnorr[1], Arnaud Legrand[2]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre – Brasil

[2]CNRS - Univ. Grenoble Alpes – France

{vgpinto, schnorr}@inf.ufrgs.br, arnaud.legrand@inria.fr

Resumo. *Este trabalho apresenta uma abordagem modular e incremental para análise de desempenho de aplicações baseadas em tarefas em arquiteturas híbridas. A estratégia proposta é construída sobre ferramentas atuais para análise de dados como R, pjdump, ggplot e plotly. A abordagem é validada por meio de um estudo de caso da fatoração de Cholesky.*

1. Introdução

Os atuais sistemas de Processamento de Alto Desempenho têm sido construídos com nós computacionais híbridos visando atender a crescente demanda por poder computacional. Entretanto, a exploração eficiente e escalável destas arquiteturas têm se tornado desafiadora. Uma solução em potencial para esta questão é a programação da aplicação utilizando um grafo de tarefas. No momento da execução, estas tarefas são mapeadas para o hardware da plataforma por meio de uma camada de software intermediária chamada *runtime*. Esta abordagem permite remover sincronizações artificiais, implementar politicas de escalonamento complexas bem como automatizar as transferências de dados.

A execução de aplicações neste modelo de *hardware* e *software* é inerentemente estocástica. O mapeamento das tarefas pode mudar drasticamente de uma execução a outra. No contexto da análise de desempenho, a natureza dinâmica das execuções torna ineficiente o uso de ferramentas clássicas de análise de desempenho pois estas foram projetadas majoritariamente para análise de aplicações bem estruturadas, tais como BSP ou MPI. Neste trabalho, nós apresentamos uma abordagem construída sobre ferramentas modernas e genéricas para análise de dados (R, ggplot, plotly, paje_dump). Construída sobre *scripts*, esta abordagem permite criar visualizações ricas e adaptáveis.

2. Metodologia

A abordagem proposta permite representar execuções de maneira visual utilizando rastros de execução obtidos previamente. Estes rastros são carregados a partir de arquivos texto (CSV) para um ambiente R onde os estados representados são limpos, filtrados e sintetizados estatisticamente. Após esta etapa, pode-se obter um diagrama espaço/tempo representando a execução. Por meio de uma estratégia modular e incremental, é possível enriquecer a visualização básica adicionando informações relevantes como arestas de dependências entre as tarefas, estimativas de *makespan* e porcentagem de tempo ocioso. Além disso, por meio de gráficos auxiliares, pode-se visualizar outras informações inferidas do rastro tais como a quantidade de tarefas disponível para execução em cada instante, a repartição de trabalho entre os recursos de processamento de diferentes tipos e a progressão da computação ao longo do tempo de execução [Pinto et al. 2016].

3. Estudo de Caso: Fatoração de Cholesky

A Figura 1 apresenta um diagrama espaço/tempo obtido com o *framework* proposto a partir do rastro da execução de uma fatoração de Cholesky proveniente do pacote de álgebra linear Chameleon [MORSE 2016]. Neste exemplo, a visualização básica foi enriquecida com informações sobre o tempo ocioso de cada recurso, estimativas do *makespan* (CPE e ABE), realçamento de estados com duração consideravelmente maior e arestas de caminho crítico das tarefas `dpotrf`. Estas informações adicionais permitem identificar problemas cujo tratamento pode levar a ganhos de desempenho. O percentual de tempo ocioso (a direita do gráfico) permite identificar um desbalanceamento da carga de trabalho entre as CPUs (ociosidade varia entre 75% e 34%) e GPUs (de 7.5% a 10.4%). Pode-se identificar também algumas tarefas do tipo `dsyrk` com duração anormal. As arestas do caminho crítico permitem identificar alguns erros de priorização entre as tarefas (p.ex em CUDA2, entre 565ms e 572ms, tarefas `dsyrk` e `dtrsm`). Os valores CPE (353ms) e ABE (418ms) são limites inferiores para o *makespan* , e ilustram que, teoricamente, há espaço para redução do valor observado atualmente (765ms). Trabalhos futuros terão foco no incremento das técnicas de visualização e na avaliação de outras aplicações.

Os dados deste experimento, o código para processamento do rastro e geração da Figura 1, bem como uma versão interativa (com plotly) da mesma estão disponíveis na versão reprodutível deste trabalho em `http://perf-ev-runtime.gforge.inria.fr/erad2017`.

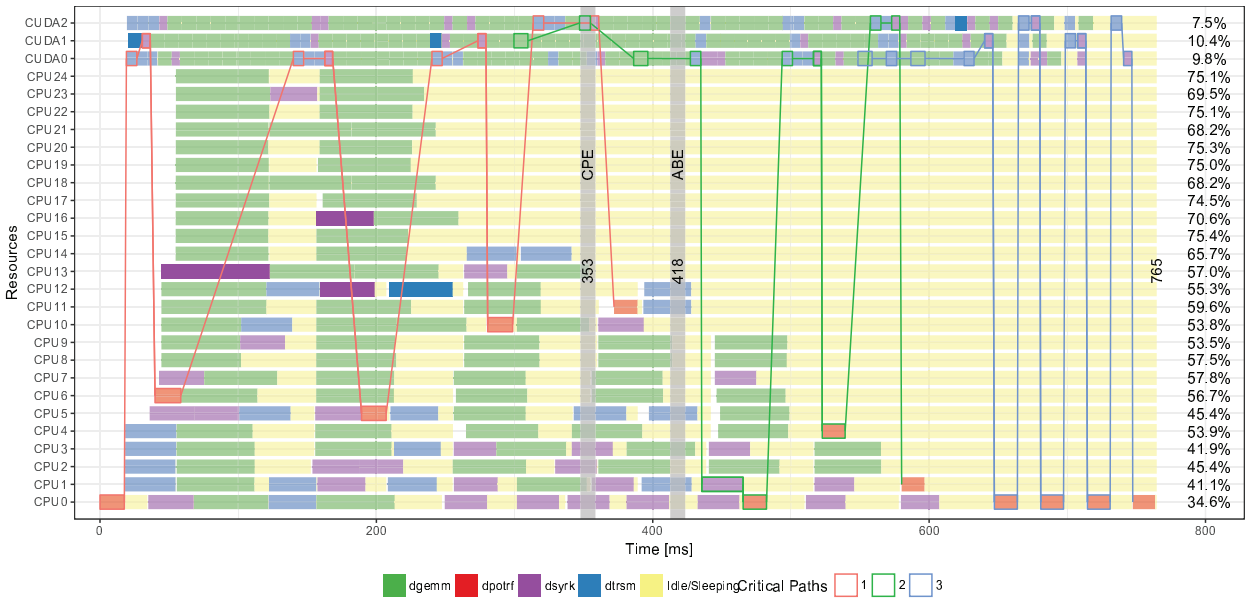

Figura 1. Representação visual da execução de uma fatoração de Cholesky ($12x12$ blocos de 960) em nó com 28 núcleos (2x Intel Xeon E5-2697v3) e 3 GPUs (NVIDIA Titan X). Execução com Chameleon+StarPU e escalonador DMDA.

Referências

MORSE (2016). Chameleon: A dense linear algebra software for heterogeneous architectures. https://project.inria.fr/chameleon.

Pinto, V. G., Stanisic, L., Legrand, A., Schnorr, L. M., Thibault, S., and Danjean, V. (2016). Analyzing dynamic task-based applications on hybrid platforms: An agile scripting approach. In *Proceedings of the 3rd International Workshop on Visual Performance Analysis*, VPA '16, pages 17–24, Piscataway, NJ, USA. IEEE Press.