Estendendo o OpenACC para Geração e Execução Eficiente de Códigos Estêncil por Esqueletos Paralelos

Alyson D. Pereira[1], **Rodrigo C.O. Rocha**[3],
Márcio Castro[1], **Luís F. W. Góes**[2], **Mário A. R. Dantas**[1]

[1] Laboratório de Pesquisa em Sistemas Distribuídos (LaPeSD)
Universidade Federal de Santa Catarina (UFSC) – SC, Brasil

[2] Grupo de Computação Criativa e Paralela (CreaPar)
Pontifícia Universidade Católica de Minas Gerais (PUC Minas) – MG, Brasil

[3] Institute for Computing Systems Architecture (ICSA)
University of Edinburgh – Escócia, Reino Unido

alyson.pereira@posgrad.ufsc.br, r.rocha@ed.ac.uk,
{marcio.castro,mario.dantas}@ufsc.br, lfwgoes@pucminas.br

Resumo. *O modelo de programação OpenACC simplica a programação para GPUs, porém seu modelo de abstração não permite explorar otimizações específicas de arquitetura. Este trabalho propõe uma extensão ao OpenACC para geração e execução eficiente de código estêncil através de frameworks baseados em esqueletos paralelos. Os resultados experimentais mostram que a abordagem proposta melhora o desempenho em até 22% em GPU e 82% em CPU.*

1. Introdução

Unidades de processamento gráfico (GPUs) e unidades centrais de processamento (CPUs), apesar de serem encontradas na mesma plataforma computacional, possuem diferentes interfaces de programação. OpenMP, OpenACC, CUDA e OpenCL são algumas opções disponíveis para o desenvolvimento de aplicações. Enquanto as duas últimas fornecem abstrações de baixo nível, sendo desafiadora e propensa a erros até pelos mais experientes, as duas primeiras fornecem abstrações de alto nível baseadas em diretivas de compilação utilizadas para anotar trechos de códigos paralelizáveis. No entanto, os modelos de programação para GPUs baseados em diretivas podem não gerar códigos otimizados para certas classes de aplicações, que possuem padrões previsíveis de computação e comunicação [Lashgar and Baniasadi 2016]. O conhecimento prévio sobre estes padrões poderia ser utilizado para alcançar um maior desempenho, minimizando a contenção de recursos limitados como o uso eficiente da memória *cache*.

Uma outra abordagem para tratar a complexidade no desenvolvimento de aplicações para sistemas heterogêneos é fazer uso de uma programação baseada em esqueletos paralelos. Esqueletos modelam padrões de computação e coordenação que ocorrem frequentemente em aplicações paralelas e provêem abstrações de alto nível com o uso de interfaces genéricas. Escrever aplicações com esqueletos é vantajoso, uma vez que os detalhes sobre paralelismo, sincronização e otimizações de código são abstraídos. Em um trabalho recente foi proposto o *framework* PSkel [Pereira et al. 2015], para programação de aplicações do padrão estêncil em C++ com suporte para CPU e GPU. Nele, o usuário define o *kernel* da computação estêncil enquanto o *framework* se encarrega de executar a computação, auxiliando no gerenciamento de memória e transferência de dados.

Neste trabalho propomos uma extensão ao OpenACC composta por diretivas estêncil e um compilador *source-to-source* para permitir uma geração e execução eficiente de código pelo *framework* PSkel. Esta extensão expõe o padrão estêncil ao compilador e sistema de execução e permite otimizações específicas para esta classe de aplicação.

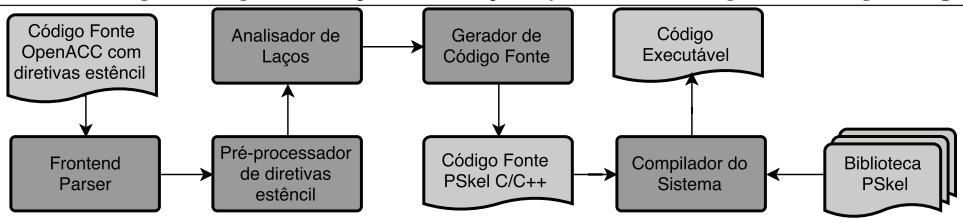

Figura 1. Visão geral do processo de compilação.

2. Proposta

As diretivas estêncil propostas são utilizadas conforme exemplo abaixo. Neste caso, a computação estêncil será executada 100% na GPU por t iterações, tendo como entrada e saída os vetores A e B, de tamanho $M \times N$. As variáveis t, A, B, M e N devem corresponder as variavéis utilizadas nos laços da computação estêncil do código original. Uma característica do *framework* PSkel é de particionar os dados de entrada e realizar uma execução simultânea de cada partição pela CPU e GPU, que pode ser realizada indicando um valor entre 0.0 (somente CPU) e 1.0 (somente GPU) na diretiva $device$. Por exemplo, um valor de 0.6 indica que 60% da entrada será processada pela GPU e o restante pela CPU.

#pragma acc stencil device(GPU, 1.0) iterations(t) input(A[M,N]) output(B[M,N])

Um compilador *source-to-source* é responsável por transformar um código OpenACC anotado com as diretivas estêncil em um código do *framework* PSkel. A Figura 1 exibe a visão geral deste processo. O **Frontend Parser** realiza um *parsing* sobre o código-fonte e produz uma árvore sintática abstrata (AST) anotada. O **Pré-processador de diretivas estêncil** percorre as construções estêncil anotadas e extrai informações das diretivas. O **Analisador de Laços** valida a semântica dos laços aninhados que realizam a computação estêncil, anotando os nodos da AST para a geração de código. O **Gerador de Código Fonte** percorre a AST produzindo código C/C++, transformando os laços anotados com as diretivas estêncil em códigos equivalentes do *framework* PSkel, gerando a implementação da função correspondente ao *kernel* da computação estêncil e chamadas de métodos que instanciam e configuram a execução do *kernel* em CPU e/ou GPU. No estágio final o **Compilador do Sistema** (por exemplo o NVCC para execução em GPU ou GCC para execução em CPU) é invocado para gerar o código executável.

3. Resultados e Trabalhos Futuros

O benefício da extensão proposta foi avaliado através de três aplicações estêncil distintas, comparando o desempenho dos códigos utilizando PSkel em relação as versões OpenACC originais, em CPU e GPU. Os experimentos foram realizados em uma máquina NUMA com 2 processadores Intel Xeon E2620 e uma GPU Nvidia Tesla K20, com os compiladores NVidia CUDA 7.5, GCC 4.9.2 e PGI Accelerator 16.5. Os resultados mostraram que as aplicações executando com o uso do *framework* PSkel tiveram uma melhora de desempenho de até 82% quando executadas em CPU e de até 22% em GPU. Dados do *profiling* das aplicações indicaram que o *framework* PSkel faz um melhor uso da memória *cache* nas execuções em CPU e é capaz de fazer uso da memória compartilhada da GPU, uma memória *cache* de alta velocidade gerenciada por software.

Em trabalhos futuros pretende-se incluir a extensão proposta em um compilador OpenACC de código aberto e suportar outros esqueletos paralelos, como *reduce* e *scan*.

Referências

Lashgar, A. and Baniasadi, A. (2016). Employing software-managed caches in openacc: Opportunities and benefits. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 1(1):2:1–2:34.

Pereira, A. D., Ramos, L., and Góes, L. F. W. (2015). PSkel: A Stencil Programming Framework for CPU-GPU Systems. *Concurrency and Computation: Practice and Experience*, 27(17):4938–4953.