StencilBench: Um Benchmark Sintético para Avaliação de Frameworks do Padrão Estêncil

Alyson D. Pereira[1], **Sérgio V. Silva**[1], **Rodrigo C. O. Rocha**[1], **Márcio Castro**[2], **Luís F. W. Góes**[1]

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

[2]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – SC, Brasil

{alyson.pereira,sergio.vitarelli}@sga.pucminas.br,
{rcor,lfwgoes}@pucminas.br, marcio.castro@ufsc.br

Abstract. *The stencil pattern computes elements as a function of its neighborhood. Currently, several frameworks support the stencil pattern for different parallel architectures. Despite the diversity of real applications implemented for these frameworks, they represent only a part of the space of stencil applications. This paper proposes a synthetic benchmark called StencilBench for evaluation of the stencil pattern frameworks for heterogeneous architectures. StencilBench was implemented for the PSkel framework, but can be easily adapted to other frameworks. The experimental results show that in addition to replicate the behavior of real applications, it explores a large space of stencil applications.*

Resumo. *O padrão estêncil permite a computação paralela de elementos em função da sua vizinhança. Atualmente, diversos frameworks suportam este padrão para diferentes arquiteturas paralelas. Apesar da diversidade de aplicações reais implementadas para estes frameworks, elas representam apenas uma parte das aplicações estêncil. Neste artigo é proposto um benchmark sintético, chamado StencilBench, para avaliação de frameworks do padrão estêncil para arquiteturas heterogêneas. O StencilBench foi implementado para o framework PSkel, mas pode ser facilmente adaptado para outros frameworks. Os resultados experimentais mostram que além de replicar o comportamento de aplicações reais, ele explora um amplo espaço de aplicações estêncil.*

1. Introdução

Padrões de programação paralela capturam o comportamento de comunicação e computação de aplicações paralelas, permitindo que o programador concentre-se apenas na programação dos detalhes do contexto específico da aplicação. Estes padrões são implementados geralmente em *frameworks* que podem ser executados em diferentes arquiteturas paralelas [Lutz et al. 2013, Enmyren and Kessler 2010].

Dentre os padrões de programação paralela mais utilizados, pode-se destacar o padrão estêncil, que permite a computação paralela de elementos em função da sua vizinhança. Atualmente, diversos *frameworks* suportam o padrão estêncil para diferentes arquiteturas paralelas, tais como PSkel, PARTANS, Physis [Pereira et al. 2015, Lutz et al. 2013, Maruyama et al. 2011]. Aplicações no padrão estêncil são importantes por serem amplamente utilizadas na indústria e academia, em aplicações como simulação

de física de partículas, previsão meteorológica, termodinâmica, resolução de funções diferenciais, manipulação de imagens, entre outras [Rahman et al. 2011].

Para a avaliação de desempenho destes *frameworks* em diversas arquiteturas faz-se necessário o uso de *benchmarks* [Bailey et al. 1991, Danalis et al. 2010, Pennycook et al. 2011]. Apesar da diversidade de aplicações reais implementadas para o padrão estêncil, elas não permitem a exploração de todo o espaço de aplicações do padrão, variando-se parâmetros tais como o tamanho da vizinhança, balanceamento da carga, variedade e intensidade computacional. Além do mais, avaliar o desempenho de diferentes aspectos de arquiteturas paralelas é uma tarefa complexa, sendo difícil para qualquer conjunto de *benchmarks* serem capazes de avaliar por completo todas as características do *hardware*.

Neste artigo é proposto um *benchmark* sintético, chamado StencilBench, para avaliação de *frameworks* do padrão estêncil para arquiteturas heterogêneas. *Benchmarks* sintéticos parametrizáveis permitem realizar experimentos simulando diversos aspectos de aplicações reais [Bailey et al. 1991], apresentando grande flexibilidade na configuração da intensidade da computação realizada. O StencilBench foi implementado para o *framework* PSkel, mas pode ser facilmente adaptado para outros *frameworks* existentes. Os resultados experimentais mostram que além de replicar o comportamento de aplicações reais, ele explora um amplo espaço de aplicações estêncil.

Este texto está estruturado da seguinte forma. Na Seção 2 é apresentado o *benchmark* StencilBench. Na Seção 3 é apresentada a metodologia de avaliação do *benchmark* e os resultados obtidos. A Seção 4 apresenta os trabalhos relacionados e a Seção 5 a conclusão e trabalhos futuros.

2. StencilBench

O StencilBench é um *benchmark* sintético do padrão estêncil desenvolvido utilizando-se o *framework* PSkel [Pereira et al. 2015]. O *benchmark* é parametrizável dinâmica e estaticamente, podendo configurar características da computação que será realizada pelo *kernel* estêncil. A principal configuração estática realizada está relacionada à escolha do tipo de dados da matriz de entrada, isto é, inteiro ou de ponto flutuante. Os parâmetros de configuração dinâmica são: tipo e tamanho da máscara de vizinhança; grau de balanceamento da carga e quantidade de operações por elemento. As operações podem ser dos seguintes tipos: adição, subtração, multiplicação, divisão, potenciação e radiciação.

O PSkel é um *framework* de programação do padrão estêncil, escrito em C++ com suporte para geração de código executável para plataformas paralelas, incluindo CPU e GPU. Utilizando uma abstração de alto nível, o usuário define o *kernel* da computação estêncil, enquanto o *framework* se encarrega de executar a computação em CPU ou GPU, auxiliando no gerenciamento de memória e transferência de dados entre dispositivos [Pereira et al. 2015].

O *kernel* estêncil da aplicação sintética recebe matrizes 2D de entrada e saída, a máscara de vizinhança e o conjunto de operações aritméticas a serem realizadas. Para cada tipo de operação, o conjunto de vizinhos é percorrido e a operação é realizada com cada vizinho de acordo com a configuração utilizada. A vizinhança percorrida, definida pela máscara de vizinhança, e a quantidade que cada operação será realizada sobre os vizinhos são especificadas pelos parâmetros de configuração da aplicação sintética. Ao

final da execução do *kernel*, um resultado é produzido e armazenado na matriz de saída. O Algoritmo 1 apresenta um trecho de código que computa as operações de adição na vizinhança a partir de um determinado ponto da matriz de entrada. As demais operações seguem o mesmo formato como exemplificado no código a seguir.

```
1  int loopCtrl = args.nAdd > 0 ? (args.nAdd-1)/mask.size + 1 : 0;
2  int opCtrl = args.nAdd > mask.size ? mask.size : args.nAdd;
3  for(int i = 0; i < loopCtrl; i++)
4    for(int j = 0; j < opCtrl; j++)
5      returnValue = returnValue + mask.getWeight(j);
```

Algoritmo 1: Trecho do *kernel* do StencilBench que realiza as operações de adição.

Inicialmente, o StencilBench particiona a quantidade de cada operação em blocos no mesmo tamanho da vizinhança (linhas 1 e 2). O laço mais externo (linha 3) controla o número de blocos de execução enquanto o mais interno (linha 4) percorre os elementos da vizinhança do elemento atual, utilizando o peso de cada vizinho na máscara como termo da operação (linha 5).

3. Avaliação Experimental

O StencilBench foi avaliado através do *profiling* de 3 execuções parametrizadas em comparação com o *profiling* de três aplicações no padrão estêncil iterativo: CloudSim, Game of Life e o método de Jacobi. A plataforma computacional utilizada consiste de dois processadores Intel Xeon E5-2620 de 6 núcleos físicos e 12 núcleos lógicos com *clock* de 2.10 GHz e memória cache de 15MiB, 64 GiB de memória RAM e processador gráfico NVidia Tesla K20 de 2496 núcleos com *clock* de 706 MHz. O *profiling* das aplicações foi realizado utilizando a ferramenta *nvprof* da NVidia.

A aplicação CloudSim [da Silva and Gouvêa 2010] simula a dinâmica de nuvens através de autômatos celulares. O modelo matemático utiliza a vizinhança de Von Neumann de cinco células, cada uma com propriedades climáticas como: partículas condensadas de água, temperatura e vento. Ele é um estêncil iterativo onde a cada iteração o valor de temperatura de todas as células são atualizadas segundo princípios de termodinâmica.

O Game of Life (GoL) [Gardner 1970] é um autômato celular implementado em uma matriz onde cada célula representa um indivíduo vivo ou morto. Ao decorrer de um número definido de iterações (ou gerações) cada indivíduo analisa o estado de seus vizinhos para definir o seu próprio estado para a próxima iteração.

O método de Jacobi [Demmel 1997] é utilizado na resolução de sistemas de equações lineares. A conversão do método iterativo para a solução é garantida se a matriz de entrada é estritamente ou irredutivelmente dominante em diagonal.

3.1. Representatividade do *benchmark* sintético

Nesta seção, a representatividade do StencilBench é medida através da capacidade do *benchmark* sintético em simular aspectos computacionais de aplicações estêncil reais. Para simular estas aplicações, os parâmetros do *benchmark* referentes ao tipo de vizinhança e quantidade de vizinhos foram extraídos das características de cada aplicação real. Para os parâmetros referentes à quantidade de operações aritméticas, seus valores foram obtidos a partir do *profilling* da execução das aplicações reais em GPU.

Os valores utilizados para simular cada aplicação são apresentados na Tabela 1. Além disso, a tabela mostra uma comparação entre o *benchmark* sintético e suas respectivas aplicações reais. A comparação é feita com base no erro percentual entre as métricas do *profiling* da execução em GPU das aplicações reais e sua equivalência pelo StencilBench. Foram utilizadas entradas de 1000 x 1000 em 10 iterações estêncil em cada aplicação.

Parâmetros	**Aplicações**		
	CloudSim	**Jacobi**	**GoL**
Tipo de vizinhança	Von Neumann	Von Neumann	Moore
Raio da máscara	1	1	1
Quantidade de vizinhos	4	4	8
Operações de adição	11	4	5
Operações de multiplicação	3	2	0
Operações de divisão	1	0	0
Métricas	**CloudSim**	**Jacobi**	**GoL**
Controle de fluxos	21,13%	170,27%	3,45%
FLOPs	3,33%	14,29%	0%
Load/Store	9,07%	38,44%	45,80%
Operações inteiras	4,47%	87,61%	16,15%
Médias	**18,26%**	**62,31%**	**13,41%**

Tabela 1. Parâmetros do *benchmark* sintético para cada aplicação real (cima) e erro percentual entre o *benchmark* sintético e aplicações reais (baixo).

Como descrito anteriormente, apesar das aplicações CloudSim e GoL serem consideravelmente distintas, com uma configuração adequada, o StencilBench é capaz de simular o padrão computacional de ambas as aplicações com um baixo erro percentual, 18,26% e 13,41% respectivamente. Destacamos a precisão do StencilBench em aproximar as operações inteiras e de ponto flutuante (FLOP) de ambas as aplicações, pois enquanto a aplicação CloudSim é intensa em operações de ponto flutuante, a aplicação do GoL possui apenas operações inteiras.

A aplicação Jacobi possui um *kernel* estêncil consideravelmente simples, resultando em um código com poucas instruções de controle de fluxo. Por outro lado, o *kernel* do StencilBench depende de condicionais para controlar o número de execuções para cada operação aritmética, resultando em um alto número de instruções de controle de fluxo. Além disso, o controle da quantidade de execução de cada operação aritmética, por meio da contagem de iterações, provoca uma grande diferença também no número de operações inteiras entre as duas aplicações. Essa diferença inerente de ambas as aplicações faz com que o StencilBench apresente um alto nível de erro percentual em relação ao Jacobi. Apesar disso, o StencilBech apresentou bons resultados em relação às aplicações reais GoL e CloudSim, que possuem características bem distintas quanto ao tipo do dado e quantidade de operações, possibilitados pelo grau de parametrização disponível no *kernel*.

3.2. Exploração do espaço de aplicações estêncil

A Figura 1 representa os parâmetros configuráveis no *kernel* do StencilBench e seu respectivo impacto no tempo de execução. Como medida do impacto de cada parâmetro, consideramos o desvio padrão entre as médias dos tempos de execução com cada valor do parâmetro fixado. O gráfico foi gerado considerando a utilização de operações de adição e multiplicação em ponto flutuante.

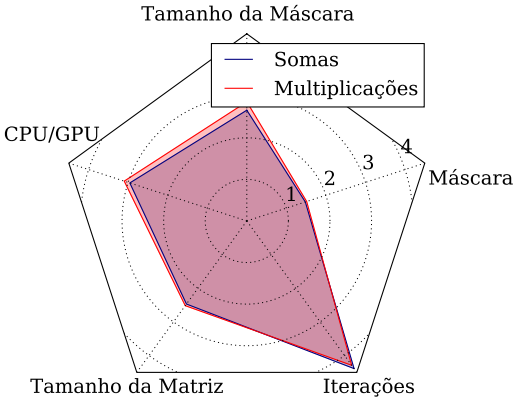

Figura 1. Avaliação do impacto de cada parâmetro no tempo de execução.

Como esperado, o número de iterações apresenta grande variação no tempo de execução do *benchmark* estêncil. Vale observar que, para os valores utilizados, o tamanho da máscara e o tamanho da matriz apresentaram impactos semelhantes no tempo de execução do *benchmark*, enquanto o tipo da máscara apresentou a menor variação, embora significativa. Contudo, todos os parâmetros variados, apresentaram pelo menos desvio padrão maior que 1, ou seja, a variação de cada parâmetro é importante para a simulação de um amplo espaço de aplicações estêncil.

4. Trabalhos Relacionados

Recentemente, com o advento da utilização de placas gráficas (GPUs) em computações de propósito geral, *benchmark*s dedicados à essas arquiteturas massivamente paralelas têm sido amplamente desenvolvidos [Che et al. 2009, Danalis et al. 2010, Pennycook et al. 2011, Wang et al. 2011]. Ambos os *benchmark*s Rodinia e SHOC apresentam uma aplicação que utiliza o padrão estêncil com entradas 2D.

*Benchmark*s sintéticos específicos têm sido utilizados em pesquisas com o padrão estêncil. Em [Strzodka et al. 2011] os autores utilizaram um *benchmark* sintético para avaliar o desempenho de um mecanismo de otimização em *cache* para computações de estêncil iterativas. O *benchmark* sintético realiza diretamente em registradores todas as operações do estêncil iterativo, simulando um caso ótimo da computação utilizando *cache*. Em seus resultados, verificou-se que o mecanismo desenvolvido resulta em 52% do desempenho obtido pelo *benchmark* sintético.

Em [Lutz et al. 2013], os autores apresentaram um *framework* que distribui entre múltiplas GPUs, de maneira otimizada, aplicações que seguem padrão de computação estêncil. Para avaliação dos resultados, eles utilizaram uma variedade de aplicações reais bem como uma aplicação sintética que permite maior controle sobre a granularidade de processamento realizada pelas GPUs. Com base em uma avaliação exaustiva do espaço de otimização, [Lutz et al. 2013] propuseram uma heurística de *autotuning* para múltiplas GPUs. A exploração exaustiva do espaço de otimização foi realizada com auxílio de uma aplicação sintética de um *kernel* altamente parametrizável.

5. Conclusão

Este trabalho apresentou o StencilBench, um *benchmark* sintético desenvolvido para aplicações paralelas no padrão estêncil. Os resultados mostram que com a parametrização

adequada do *kernel*, o *benchmark* consegue simular com baixa margem de erro aplicações reais com diferentes características de computação.

A principal contribuição da pesquisa foi desenvolver uma aplicação adaptável a outros *frameworks* além do PSkel, com grau de confiança satisfatório em relação à capacidade de simulação de aplicações estêncil distintas. O StencilBench poderá ser utilizado em trabalhos futuros para comparação entre diferentes *frameworks* estêncil.

Referências

Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Dagum, L., Fatoohi, R. A., Frederickson, P. O., Lasinski, T. A., Schreiber, R. S., Simon, H. D., Venkatakrishnan, V., and Weeratunga, S. K. (1991). The nas parallel benchmarks: Summary and preliminary results. In *ACM/IEEE Supercomputing*, pages 158–165.

Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., Lee, S.-H., and Skadron, K. (2009). Rodinia: A benchmark suite for heterogeneous computing. In *IISWC*, pages 44–54.

da Silva, A. R. and Gouvêa, Jr., M. M. (2010). Cloud Dynamics Simulation with Cellular Automata. In *SCSC*, pages 278–283.

Danalis, A., Marin, G., McCurdy, C., Meredith, J. S., Roth, P. C., Spafford, K., Tipparaju, V., and Vetter, J. S. (2010). The scalable heterogeneous computing (shoc) benchmark suite. In *GPGPU-3*, pages 63–74.

Demmel, J. W. (1997). *Applied numerical linear algebra*. Soc. for Industrial and Applied Math., Philadelphia, PA, USA.

Enmyren, J. and Kessler, C. W. (2010). Skepu: A multi-backend skeleton programming library for multi-gpu systems. In *HLPP*, pages 5–14.

Gardner, M. (1970). Mathematical Games - The Fantastic Combinations of John Conway's New Solitaire Game 'Life'. *Scientific American*, 223(3):120–123.

Lutz, T., Fensch, C., and Cole, M. (2013). PARTANS: An Autotuning Framework for Stencil Computation on Multi-GPU Systems. *ACM Trans. Archit. Code Optim.*, 9(4):59:1–59:24.

Maruyama, N., Nomura, T., Sato, K., and Matsuoka, S. (2011). Physis: An Implicitly Parallel Programming Model for Stencil Computations on Large-Scale GPU-Accelerated Supercomputers. In *ACM/IEEE SC*, pages 11:1–11:12.

Pennycook, S. J., Hammond, S. D., Jarvis, S. A., and Mudalige, G. R. (2011). Performance analysis of a hybrid mpi/cuda implementation of the naslu benchmark. *SIGMETRICS Perform. Eval. Rev.*, 38(4):23–29.

Pereira, A. D., Ramos, L., and Góes, L. F. W. (2015). PSkel: A Stencil Programming Framework for CPU-GPU Systems. *Concurrency Computat. Pract. Exper.*

Rahman, S. M. F., Yi, Q., and Qasem, A. (2011). Understanding stencil code performance on multicore architectures. In *CF*, pages 30:1–30:10.

Strzodka, R., Shaheen, M., Pajak, D., and Seidel, H.-P. (2011). Cache accurate time skewing in iterative stencil computations. In *ICPP*, pages 571–581.

Wang, F., Yang, C.-O., Du, Y.-F., Chen, J., Yi, H.-Z., and Xu, W.-X. (2011). Optimizing linpack benchmark on gpu-accelerated petascale supercomputer. *J. Comput. Sci. Technol.*, 26(5):854–865.