O IMPACTO DO TAMANHO DO BUFFER DE BUSCA DE INSTRUÇÕES EM RELAÇÃO À LARGURA DO ESTÁGIO DE BUSCA EM UMA GPU

THE IMPACT OF THE INSTRUCTION SEARCH BUFFER SIZE REGARDING THE SEARCH STAGE WIDTH OF A GPU

JOSÉ CARLOS SANCHES AMENDOA **TISSEI**[1], JEFERSON APARECIDO SILVA **BELGAMAZZI**[2], LUCAS DE OLIVEIRA **TEIXEIRA**[3], CLAUDIA HEIDEMANN DE **SANTANA**[4]

1. Aluno do curso de graduação em Ciências da Computação da Faculdade Ingá; **2.** Aluno do curso de graduação em Ciências da Computação da Faculdade Ingá; **3.** Mestre pela Universidade Estadual de Maringá, docente no curso de Ciências da Computação na Faculdade Ingá; **4.** Mestre em Engenharia de Produção pela Universidade Federal de Santa Catarina, docente no curso de Ciências da Computação na Faculdade Ingá.

Rua Lápis Lazuli, 211, Jardim Santa Helena, Maringá, Paraná, Brasil. CEP: 87083-310 chsantana@gmail.com

Recebido em 04/10/2013. **Aceito** para publicação em 07/11/2013

RESUMO

Unidades de processamento gráfico (*GPUs*) possuem uma grande capacidade de processamento *multithread* e por esse motivo se tornaram uma plataforma muito popular para a execução de aplicações paralelas. O modelo de programação utilizado permite a criação de centenas de *threads* executando o mesmo *kernel*, isso e possível graças à quantidade de unidades de processamento existentes em uma GPU. Um dos estágios mais importantes para o aumento do desempenho das aplicações é a busca de instruções. Nesse artigo, investigamos o efeito do tamanho do *buffer* do estágio de busca de instruções em relação à largura desse mesmo estágio. Os dados utilizados na análise foram obtidos por meio de simulações heterogêneas CPU-GPU. Através da análise dos resultados é possível notar que esse *buffer* exerce um papel fundamental no desempenho de aplicações em geral, mas quando este é maior ou igual a largura desse estágio não ocorrem grandes variações.

PALAVRAS-CHAVE: Simulação CPU-GPU, Desempenho CPU-GPU, Busca de instruções GPU, Consumo de energia, Processamento paralelo.

ABSTRACT

Graphical processing units (GPUs) have a great capacity for *multithread* processing and therefore have become a very popular platform for running parallel applications. The programming model used allows the creation of hundreds of threads running the same kernel, this is made possible thanks to the amount of existing GPU processing units. One of the most important stages in increasing application performance is the instruction fetch. In this paper, we investigate the effect of the instruction search buffer size of the instruction fetch stage regarding the width of such stage. The data used in the analysis were obtained by heterogeneous CPU-GPU simulations. Analyzing the results it is possible to realize that this buffer plays a key role in the performance of applications in general, but when it is greater than or equal than the width of the instruction fetch stage there is no big variations.

KEYWORDS: CPU-GPU simulation, CPU-GPU performance, GPU instruction fetch, power consumption, parallel processing.

1. INTRODUÇÃO

Prever Usualmente GPUs (*GraphicsProcessing Unit*) são projetadas para um processamento gráfico de alto desempenho. Devido ao alto investimento no mercado de jogos, as GPUs se tornaram um hardware poderoso em relação ao custo monetário. Nas arquiteturas modernas, onde temos GPUs sofisticadas de alto desempenho, a CPU pode preocupar-se mais com os outros tipos de processamento, visto que o processamento gráfico consome muitos recursos de um processador. Além de renderização de imagens, as GPUs vêm sendo usadas também em programas mais genéricos. Para programadores, a GPU tem sido uma boa alternativa em varias aplicações, principalmente as que fazem um grande uso de paralelismo.

Estes tipos de componentes aproveitam o desprendimento dos dados disponível em aplicações gráficas, executando as instruções em vários fluxos paralelos. Devido ao seu grande poder de processamento paralelo,

GPUs também são comumente usadas no processamento de outros tipos de aplicações, e nestes casos são chamadas de GPGPUs (*General-purpose graphics processing unit*).

As GPUs possuem processadores *multithreaded* chamados de *Streaming Multiprocessors* (SMs). Os SMs executam as *threads* de maneira SIMT (*Single Instruction Multiple Thread*). SIMT e uma forma de execução SIMD (*Single Instruction Multiple Data*) onde a mesma instrução é buscada e executada por um grupo de threads ao mesmo tempo, onde cada thread possui seus próprios conjuntos de dados. Em um SM, *threads* são agrupadas em unidades chamadas *Warps* e são executadas simultaneamente. Devido ao fato das *threads* serem executadas em conjunto, a alocação de recursos, principalmente o estágio de busca de instruções, em uma GPU é uma importante questão a ser tratada.

O estágio de busca é responsável por trazer instruções da memória para a sua execução. Inicialmente, as instruções buscadas são mantidas no *buffer* de instruções de busca e, em seguida, enviadas para o estágio de decodificação. Assim, esse estágio age como uma porta de entrada para a execução e possui um grande papel no desempenho do *pipeline*, principalmente em arquiteturas paralelas que necessitam de um constante fluxo de instruções, como as GPUs. Com isso, esse trabalho visa analisar o impacto da variação do tamanho do *buffer* de busca de instruções em relação à largura do estágio de busca em uma GPU através da análise de simulações em um sistema heterogêneo CPU-GPU. Assim, o principal objetivo é tentar realizar a correlação entre o tamanho do *buffer* e a largura do estágio de busca.

2. MATERIAL E MÉTODOS

O presente estudo foi realizado com base na consulta de bibliografia específica da área e da temática em questão. Foram consultados periódicos nacionais e internacionais disponíveis no portal EBSCO host da Faculdade Ingá. Foram selecionados 13 artigos entre os anos de 1996 e 2014, pela relevância de informações em face do objetivo do presente levantamento.

3. RESULTADOS E DISCUSSÃO

A análise dos resultados foi dividida em três pontos: desempenho geral, no qual é discutida a implicação da alteração proposta na variação do número de instruções executadas por ciclo de *clock* (IPC) na GPU e CPU; desempenho das *caches*, no qual é analisada a implicação da alteração proposta na taxa de acerto das caches L1, L2 e L3; e desempenho de energia, em que se analisa a variação no consumo de energia da CPU e GPU implicada pela alteração proposta.

Na maioria das simulações, as alterações no tamanho do *buffer* de busca de instruções da GPU não tiveram grande influência no desempenho da CPU. Esse era um resultado esperado, uma vez que, foram alterados somente configurações da GPU. O único caso especial foi o programa GESUMMV (multiplicação escalar, vetorial e de matrizes) que as alterações do *Baseline-2* e *Baseline-3* proporcionaram um leve *speedup* no IPC da CPU. No geral, as alterações negativas no *Baseline* causaram um leve *slowdown* no IPC da CPU. Com isso, existem indícios de que o tamanho do *buffer* de instruções da GPU impacta o desempenho da CPU.

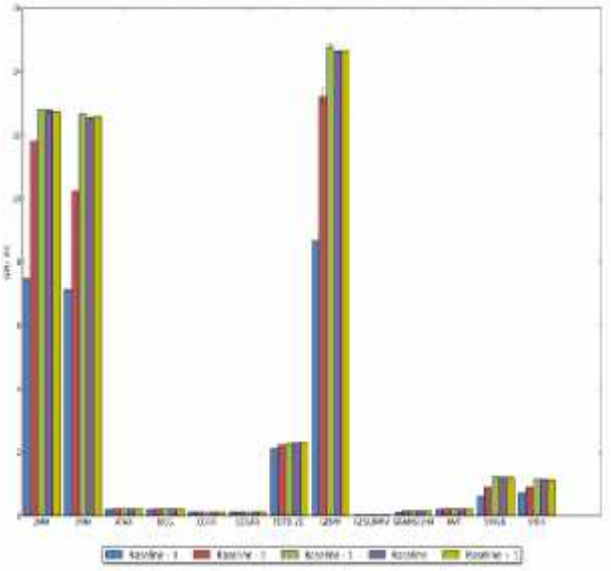

Figura 1. GPU IPC

Analisando o impacto do tamanho do *buffer* de instruções da GPU na quantidade de instruções por ciclo da GPU, apresentada na Figura 1, é possível perceber que a redução do tamanho do *buffer* impacta negativamente no desempenho da GPU. As alterações do *Baseline*-3 e *Baseline*-2 causaram um grande *slowdown* para grande parte dos programas do Polybench/GPU. No caso dos programas 2MM, 3MM e GEMM, os quais implementam variações de multiplicação de matrizes, o *slowdown* do *Baseline*-3 chega a ser de quase metade se compararmos ao *Baseline*. Isto pode indicar que diminuindo o tamanho do *buffer* de busca de instruções da GPU o desempenho é fortemente afetado.

O desempenho das memórias cache L1, L2 e L3 da CPU não foram afetados positivamente ou negativamente pelas alterações no tamanho do *buffer* de busca de instruções da GPU. Novamente, esse era um resultado esperado, uma vez que, as GPU atuais possuem memória própria integrada à placa. Isso significa que os dados utilizados pela GPU ficam na própria memória da GPU. Assim, não existe necessidade de utilizar memórias cache ou RAM para armazenar tais dados.

Finalmente, a última característica analisa é o con-

sumo de energia da CPU e da GPU. Em relação ao consumo de energia da CPU, novamente a alteração do tamanho do *buffer* de busca de instruções da GPU não influenciou bastante o consumo de energia. Da mesma forma, que alguns programas obtiveram *speedupe slowdown*, o consumo de energia se manteve proporcional a essa variação. Isso significa que quanto maior o desempenho da CPU mais energia ela gasta. Em relação ao consumo de energia da GPU, apresentado na Figura 2, é possível observar que o aumento e a diminuição do consumo de energia é proporcional ao aumento e a diminuição do desempenho. Novamente, quanto maior o desempenho da GPU, mais energia ela gasta.

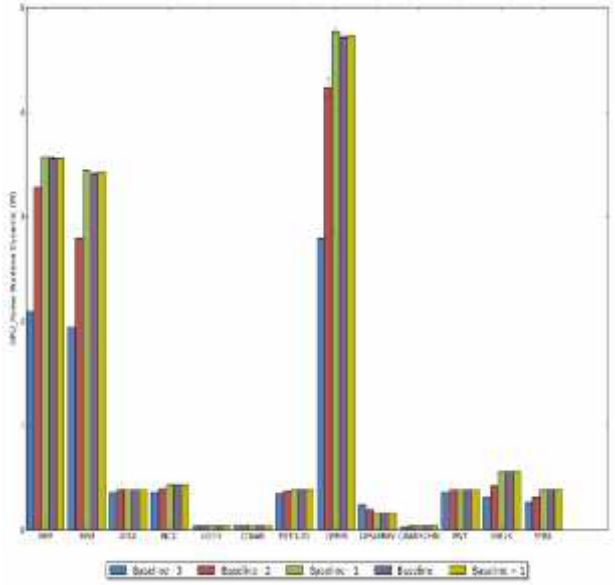

Figura 2. Consumo de energia da GPU.

4. CONCLUSÃO.

De acordo com os resultados apresentados e analisados neste trabalho, é possível concluir que o tamanho do *buffer* de instruções exerce um papel fundamental no desempenho de algumas aplicações executadas na GPU. Por meio de uma análise superficial, é possível notar que existem indícios de que as aplicações com o maior número de instruções vetoriais da ULA são as mais afetadas pela mudança no tamanho do *buffer*, no entanto é necessária uma análise maior e mais aprofundada para se obter tal conclusão. Além disso, o consumo de energia aumenta proporcionalmente com o aumento do desempenho, isso ocorreu em todos os aplicativos analisados.

Em seguida, é possível verificar que não houve grandes variações no desempenho nem no consumo energético da CPU, e isso era um resultado esperado, pois não houve nenhuma alteração em sua arquitetura nessa simulação.

Finalmente, esse trabalho realizou a análise de desempenho e energia através da alteração de apenas uma característica. Nesse caso, apenas o tamanho do *buffer* de instruções de busca. No entanto, a alteração de uma característica pode impactar em diversas outras, com isso a análise deve ser feita variando diversas configurações relacionadas que podem alterar o desempenho do sistema como um todo. Assim, um trabalho futuro é realizar à variação da largura do estágio de busca juntamente com o tamanho do buffer de instruções de busca na tentativa de encontrar um conjunto de valores ótimo que aumente o desempenho geral do sistema.

REFERÊNCIAS

[1] Jiao Y, Lin H, Balaji P, Feng W. Power and performance characterization of computational kernels on the gpu. Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom). IEEE, 2010.

[2] Lakshminarayana NB, Kim H. Effect of instruction fetch and memory scheduling on gpu performance. Workshop on Language, Compiler, and Architecture Support for GPGPU. 2010.

[3] Tullsen DM, Eggers SJ, Emer JS, Levy HM, Lo JL, Stamm RL. CM. Exploiting choice: Instructionfetchandissueonanimplementablesimultaneousmultithreading processor. ACM SIGARCH Computer Architecture News. 1996; 24(2).

[4] Narasiman V, Shebanow M, Lee CJ, Miftakhutdinov R, Mutlu O, Patt YN. Improving gpu performance via large warps and two-level warp scheduling. Proceeding softhe 44th Annual IEEE/ACM International SymposiumonMicroarchitecture. ACM. 2011.

[5] Xiang P, Yang Y, Mantor M, Rubin N, Hsu LR, Huiyang Z. Exploiting uniform vector instructions for GPGPU performance, energy efficiency, and opportunistic reliability enhancement. Proceeding softhe 27th international ACM conference on International conference on supercomputing. ACM. 2013.

[6] Amdradeonhd 7990 graphics, 2013. http://www.amd.com/us/products/desktop/graphics/7000/7990/Pages/radeon-7990.aspx. Accessado em Outubro de 2014.

[7] Craig DC. Extensible Hierarchical Object-Oriented Logic Simulation with an Adaptable Graphical User Interface. PhD thesis, Memorial University of Newfoundland, 1996.

[8] Intel core i7 3820, 2013. http://ark.intel.com/pt-br/products/63698/Intel-Core-i7-3820-Processor-10M-Cache-up-to-380-GHz. Accessado em Outubro de 2014.

[9] Ubal R, Jang B, Mistry P, Schaa D, Kaeli D. Multi2sim: a simulation framework for CPU-GPU computing. Proceedingsofthe 21st international conference on Parallel architectures and compilation techniques. ACM. 2012.

[10] Grauer-Gray S, Xu L, Searles R, Ayalasomayajula S, Cavazos J. Auto-tuning a high-level language targeted to gpu codes. Innovative Parallel Computing (InPar). IEEE. 2012.

[11]Li S, Ahn JH, Strong RD, Brockman JB, Tullsen DM, Jouppi NP. Mcpat: an integrated power, area, and timing

modeling framework for multicore and manycore architectures. Microarchitecture, MICRO-42. 42nd Annual IEEE/ACM International Symposiumon. IEEE. 2009.
[12]Calborean H, Vintan L. Towardan efficient automatic design space exploration frame for multicore optimization. ACACES 2010 poster Abstracts. 2010; 135-8.
[13]Hunter JD. Matplotlib: A 2d graphics environment. Computing in Science & Engineering. 2007; 9(3):0090-95.