

ESTRATÉGIAS DE OTIMIZAÇÃO DE CÓDIGO EM OPENCL

Claudio André da SILVA JUNIOR¹

RESUMO

Avanços notáveis na tecnologia de processadores multinúcleo estão ocorrendo, especialmente nas placas gráficas de propósito geral que oferecem enorme paralelismo a baixo custo. Tal evolução tem sido bem empregada na computação científica. Contudo, por se tratar de uma área nova, as técnicas a ela relacionadas ainda não estão estabelecidas. Neste contexto, este trabalho avaliará algumas técnicas úteis para a programação paralela que não são difundidas e aplicadas adequadamente.

INTRODUÇÃO

Os avanços na tecnologia dos processadores têm oferecido ao usuário doméstico acesso a um poder de processamento equivalente ao disponível nos supercomputadores de poucos anos atrás (TAVARES, 2012). De fato, as atuais placas gráficas de propósito geral ou GPGPUs (*General Purpose Graphics Processing Unit*), embora sejam para uso doméstico e de baixo custo, contém milhares de núcleos de processamento, são capazes de realizar milhões de operações por segundo e podem ser usadas para executar programas de propósito geral. Ademais, a literatura tem demonstrado que quando se compara o desempenho de uma GPGPU ao de uma CPU de custo equivalente, a GPGPU é de 10 (dez) a 100 (cem) vezes mais rápida para resolver o mesmo problema (TAVARES, 2012).

Neste ambiente de evolução tecnológica surgiu o OpenCL. O OpenCL (KHRONOS, 2012) é um padrão aberto, *royalty-free* para programação paralela em processadores multinúcleo. Suportam OpenCL, entre outros, as CPUs Intel e AMD,

¹ Universidade Federal de Alfenas – Câmpus Sede Alfenas. Alfenas/MG, email: claudioandre.br@gmail.com;

as GPGPUs AMD, NVIDIA e Intel. Ainda, processadores de sinal, unidades FPGA, entre outros. Qualquer fabricante pode implementar em seus produtos o suporte ao OpenCL.

Isto posto, este trabalho avaliará a aplicabilidade de algumas técnicas de otimização em programação paralela². A relevância desta empreitada está em buscar formas para resolver os problemas em menos tempo. O resultado obtido usando-se um *software* otimizado é a economia em dias e KWh que hoje são desperdiçados, uma vez que para a resolução de problemas computacionalmente muito difíceis e demorados quaisquer ganhos de desempenho implicam economia de recursos.

MATERIAL E MÉTODOS

Este trabalho realizou uma série de experimentos com o famoso *software* livre de auditorias de senhas *John the Ripper* (OPENWALL, 2014). Para tanto, o código fonte do programa foi alterado utilizando as abordagens listadas a seguir:

- Fracionamento do *kernel*: como os compiladores OpenCL não lidam muito bem com *kernels* grandes, uma possível otimização é quebrar as tarefas em rotinas pequenas, que uma vez compiladas, resultam em um código executável mais eficiente. Tal técnica mostrou-se proveitosa no *hash* iterado sha512crypt (SILVA JUNIOR, 2012).
- Transferência de dados em lotes: o programa OpenCL executa em *hardware* paralelo especializado. A transferência de dados da memória principal do computador para a memória do dispositivo demanda tempo. Nesta técnica os dados são transferidos da memória do computador para a da GPGPU em pequenos lotes, de forma que o processamento possa iniciar-se tão logo existam dados disponíveis na memória do dispositivo.

Os testes foram executados com o auxílio de três GPGPUs com especificações distintas: uma placa AMD Radeon 6770, uma NVIDIA GTX 570 e

² Na área de computação científica o termo comumente empregado é “programação massivamente paralela”.

uma AMD Radeon 7970. Ou seja, uma placa de vídeo básica, uma intermediária e uma *top* de linha, respectivamente, nesta ordem.

O *hash* iterado md5crypt foi selecionado para os experimentos devido à sua importância histórica no contexto da segurança da informação e, principalmente, por ser um dos primeiros esquemas de *hash* otimizado e adaptado para OpenCL (MITRE, 2012). Mas apesar de sua maturidade, ainda se beneficiou das técnicas propostas por este trabalho, fato que confirma a tese em foco de que os métodos propostos não são conhecidos ou aplicados a contento.

Por fim, em virtude do alto grau de otimização já existente no código OpenCL do md5crypt que é executado pela GPGPU, apenas o código fonte de chamadas do *kernel* (código que executa no *host*, ou seja, no processador principal da máquina) foi alterado.

RESULTADOS E DISCUSSÃO

O alto grau de otimização presente no código fonte OpenCL executado na GPGPU inviabilizou o uso da técnica de fracionamento do kernel. Por outro lado, houve um ganho de performance mensurável com o uso da técnica de transferência de dados em lotes. Os resultados dos experimentos podem ser vistos na tabela abaixo.

Tabela 1: perda ou ganho obtido com o uso das técnicas de otimização propostas.

	Radeon HD 6770	NVIDIA 570 GTX	Radeon HD 7970
Kernel fracionado	Perda de $\pm 25\%$	Perda de $\pm 11\%$	Perda de $\pm 9\%$
Transferência otimizada	Ganho de 5,3%	Ganho de 1,5%	Ganho de 1%

Como o *hash* em foco é muito rápido, a técnica de kernel fracionado falhou pois o custo extra para salvar os dados entre cada uma das chamadas das sub-rotinas é muito relevante, mesmo se o novo código executável gerado for mais eficiente que o original “não fracionado”. Portanto, ao adicionar-se este tempo extra a duração total para execução se torna desfavorável e o resultado final é a perda de

desempenho. Este resultado é o oposto do obtido para o sha512crypt em que fracionar o *kernel* em tarefas menores e mais simples permitiu um ganho de performance significativo (SILVA JUNIOR, 2012).

Por outro lado, na transferência otimizada de dados houve um ganho real, embora pequeno. Uma vez que o md5crypt demanda muito mais processamento que transferência de dados, é fácil verificar que o fator limitante não é a taxa de transferência entre a memória principal e a GPGPU. Portanto, não surpreende que os ganhos sejam singelos. Tal resultado seria muito diferente caso a otimização fosse aplicada a *hashes* não iterados que demandam muita transferência de dados e pouco processamento, como, aliás, já foi comprovado (SILVA JUNIOR, 2014).

CONCLUSÕES

Os resultados obtidos neste trabalho comprovam que existem algumas abordagens para otimização de código OpenCL que não são aplicadas de maneira corriqueira pelos desenvolvedores de *software* paralelo. Em realidade, até no famoso *software John the Ripper*, projetado e mantido por alguns programadores de renome, as técnicas não foram aplicadas até o momento.

Naturalmente, alguns métodos aqui estudados não se aplicam indistintamente a quaisquer problemas. De fato, algumas abordagens que aqui não puderam ser aplicadas, se mostraram imprescindíveis para a solução de outros problemas. Da mesma forma, os resultados encontrados por este trabalho podem não se repetir em problemas de outra natureza. O ponto central proposto nesta discussão é que as técnicas precisam ser conhecidas e devem ser testadas sempre que possam oferecer algum benefício.

Por fim, é forçoso ressaltar que apesar dos ganhos obtidos serem modestos, a técnica de transferência otimizada é necessária uma vez que na área de auditoria de senhas os programas podem ser executados por semanas ou meses. Assim, qualquer ganho pode representar vários dias a menos de processamento, fato que justifica qualquer tentativa de otimização.

REFERÊNCIAS BIBLIOGRÁFICAS

KHRONOS. **OpenCL - The open standard for parallel programming of heterogeneous systems**. Oregon: Khronos Group, 2012. Disponível em: <<http://www.khronos.org/opencv/>>. Acesso em: 04 abr. 2012.

MITRE CORPORATION. **CVE-2012-3287**. 2012. Disponível em: <<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-3287>>. Acesso em: 15 jul. 2014.

OPENWALL. **John the Ripper password cracker**. 2014. Disponível em: <<http://www.openwall.com/john/>>. Acesso em: 15 jul. 2014.

SILVA JUNIOR, C. A. **OpenCL SHA-2**. 2012. John the Ripper GitHub Repository. Disponível em: <<https://github.com/magnumripper/JohnTheRipper/pull/78>>. Acesso em: 15 jul. 2014.

_____. Os riscos de segurança inerentes ao uso dos modernos equipamentos médicos informatizados. In: Congresso Científico-Cultural da Unifal-MG, 2014, Alfenas. **Anais do Congresso**. Alfenas: Unifal, 2014.

TAVARES, A. A revolução das GPUs no Brasil e no mundo. In: Semana sobre Programação Massivamente Paralela, 2012, Petrópolis. 2012. **Anais**. Petrópolis: LNCC, 2012. Disponível em: <http://www.lncc.br/eventoSeminario/eventoconsultar.php?vMenu=&idt_evento=946> Acesso em: 15 jul. 2014.