CUDA x OpenMP x Pthreads: Implicações no custo total de uma solução de distribuição segura de vídeos

Leandro A. S. Gomes, Bruno S. Neves e Leonardo B. Pinho

Engenharia de Computação – Universidade Federal do Pampa (UNIPAMPA) - Bagé
Caixa Postal 07 – 96.413-170 – Bagé – RS – Brasil

heco@unipampa.edu.br

Resumo. *A distribuição segura de vídeos para grandes audiências envolve alta capacidade de processamento paralelo em módulos de recriptografia capazes de individualizar os fluxos de vídeo aos clientes, evitando a necessidade de compartilhamento de chaves. A natureza paralela do problema faz dele candidato potencial para implementações paralelas que explorem o hardware paralelo disponível nos sistemas computacionais atuais. Com a disseminação das GPUs, surge a necessidade de avaliar empiricamente não apenas o desempenho neste hardware, mas também outros requisitos fundamentais como consumo energético e custo total de implementação. Neste trabalho são comparadas alternativas de implementação do módulo de recriptografia com CUDA, Pthreads e OpenMP e coletados resultados empíricos para diferentes quantidades de clientes concorrentes, sugerindo que o uso das GPUs pode diminuir o custo total de implementação.*

1. Introdução

Com o advento da disseminação de arquiteturas propícias a exploração do paralelismo, cresce a necessidade de avaliar novas aplicações neste novo contexto. Em particular, o uso, para processamento de propósito geral, das chamadas Unidades Gráficas de Processamento (*Graphics Processing Units* - GPUs) vem chamando a atenção de vários pesquisadores como alternativa para aumentar o desempenho de aplicações e possibilitar alto desempenho com poucos recursos de *hardware* [Accelerators 2009]. Resumidamente, a ideia básica é aproveitar a crescente capacidade de processamento, muitas vezes ociosa, oferecida por estes dispositivos de *hardware* originalmente introduzidos nos sistemas computacionais para o processamento gráfico, como elemento de co-processamento para execução eficiente de aplicações de propósito geral.

Neste contexto se enquadra o problema da distribuição de fluxos de vídeos (*streaming*) para grandes audiências. As soluções escaláveis que apresentam melhor relação custo-benefício se baseiam em servidores *proxy*. Também existe a necessidade de resolver paralelamente o problema do DRM (*Digital Rights Management*), resolvido com criptografia, o que, para filmes muito populares, implica em múltiplas recriptografias dos mesmos trechos de vídeo a serem enviados de forma individualizada aos clientes.

Em trabalho anterior [Gomes et al. 2011] foi proposto e avaliado empiricamente o emprego de GPUs em *proxies* de VoD como um módulo de recriptografia, usando o algoritmo de criptografia AES (*Advanced Encription Standart*), na sua variante ECB (*Eletrocnic Code Book*), o qual possui um alto potencial para exploração do

paralelismo, pois não possui dependência entre os blocos de entrada. Para tanto, a eficiência da proposta, implementada com CUDA, foi contrastada com uma implementação paralela baseada em Pthreads, capaz de explorar múltiplos núcleos de um processador de propósito geral padrão de mercado, mostrando que a implementação com GPU foi capaz de atender uma maior quantidade de fluxos concorrentes.

Apesar destes resultados promissores, em um cenário ideal a melhoria no desempenho não deve ocorrer em detrimento de outros requisitos fundamentais desta aplicação, tais como consumo energético (CE) e custo total de implementação (CTO). O CE tende a aumentar com a introdução de uma GPU ou outras CPUs, impactando o CTO que será função do custo de aquisição dos componentes de *hardware* e do CE. No presente trabalho, são adotadas metodologias para avaliação do CE de diferentes versões concebidas para o módulo de recriptografia, considerando variadas estratégias de exploração de paralelismo, buscando encontrar a melhor relação de CE e desempenho tal que seja maximizada a diferença obtida entre o CTO do *proxy* baseado em GPU em relação ao que seria obtido com um *proxy* convencional multiprocessado. Além desta mudança de paradigma de avaliação, considerando o CTO, neste trabalho é estendida a análise para uma segunda biblioteca de programação paralela: OpenMP.

Ao focar a aplicação de *streaming*, o presente trabalho se diferencia de trabalhos relacionados onde é avaliada a eficiência da GPU em termos de desempenho (tempo de execução). Como exemplo representativo, em [Seshadrinathan e Dempski 2008] os autores utilizaram a API HLSL (*High Level Shader Language*), que hoje em dia não é a mais adequada para GPGPU (*General Purpose Computing on Graphics Processing Units*), como foi demonstrado pelos autores de [Manavski 2007]. Já em [Di Biagio et al.2009], os autores apresentaram duas estratégias de paralelização do AES, a *fine-grained*, que atua entre os *rounds* do algoritmo e a *coarse-grained*, que possui o foco em um paralelismo de mais alto nível, exposto pelos modos de operação ECB e CTR.

2. Recriptografia assistida por GPU aplicada a Sistemas VoD

Esta proposta passa pela paralelização do AES em sua variante ECB com base no núcleo de criptografia presente na OpenSSL. Essa paralelização gerou três modelos diferentes (*coarse, fine e mix-grained*). No *coarse-grained*, cada *thread* é responsável por cifrar um *frame* inteiro para um cliente onde cada um deles possui uma chave de criptografia diferente; no *fine-grained*, todas as *threads* responsáveis pela criptografia operam em cooperação para cifrar o mesmo frame para um cliente e esse procedimento é repetido até que todos os clientes tenham o *frame* cifrado; no *mix-grained,* ocorre uma mescla entre concorrência da versão *coarse-grained* e a cooperação da versão *fine-grained*, de modo que um grupo de *threads* cifra um *frame* para um cliente em concorrência com outros grupos de *threads*.

Supondo um cenário onde os clientes requisitem o mesmo vídeo ao mesmo tempo, para codificar o vídeo inteiro para cada um dos clientes basta que um *frame* por vez seja enviado para a memória da GPU levando em consideração que as chaves de criptografias são diferentes. Foram codificadas diferentes versões em C/C++ baseadas nestes três modelos, tal que todas elas possuem uma *thread* de leitura responsável pela produção e inserção dos *frames* em um *buffer* de entrada, enquanto uma *thread* de controle tem o trabalho de consumir os *frames* do *buffer* de entrada e passar um *frame*

por vez para as *threads* responsáveis pela criptografia. As versões que fazem uso da GPU foram implementadas com a API CUDA (*Compute Unified Device Architecture*). Já as versões *CPU-only* também têm suas *threads* de criptografia criadas através da APIs Pthreads e/ou OpenMP (dependo da versão), competindo pelo reduzido número de núcleos de processamento quando comparado com as GPUs mais recentes.

3. Análise de desempenho

Os experimentos foram realizados, atualmente, em um ambiente de testes composto por uma GPU nVidia GTS250 utilizando o *driver* (270.41.19), SDK e Toolkit (versão 4.0) do CUDA, uma CPU Intel Core i5 750 (quatro núcleos), RAM de 4GB, Ubuntu Linux 11.04 (*kernel* 2.6.38-10-generic) e GCC 4.4.5.

As primeiras experiências realizadas com as versões portadas para CUDA (GPU) demonstraram que a versão *mix-grained* obteve o melhor desempenho em relação a todas as outras versões experimentadas, tanto na CPU quanto na GPU, como pode ser verificado nas Figuras 1 e 2.

As versões OpenMP foram criadas para uma comparação mais justa entre a API CUDA e APIs paralelas de propósito geral. Os resultados gerados mostram uma pequena vantagem de desempenho da implementação com Pthreads em relação à OpenMP em contraste com uma menor diferença de vazão das versões que usam a GPU devido à atualização do ambiente de testes.

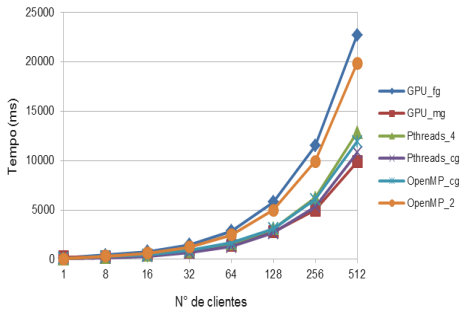

Figura 1. Melhores versões, vídeo de 4,8MB

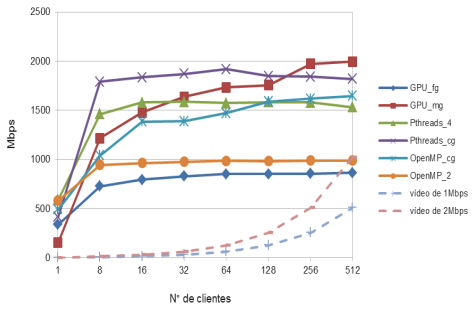

Figura 2. Throughput (gerado em termos do tempo de execução)

Nas Figuras 1 e 2 o número após o rótulo representa a quantidade de *threads* de criptografia que são lançadas nas versões Pthreads e OpenMP (casos de desempenho mais representativo), e as linhas tracejadas são uma estimativa da demanda de processamento necessária para atender a um determinado número de clientes.

4. Mensuração do consumo energético

A mensuração do consumo energético visa encontrar a melhor relação de consumo energético e desempenho tal que seja maximizada a diferença obtida entre o custo total de implementação do *proxy* baseado em GPU em relação ao que seria obtido com um *proxy* convencional multiprocessado. Em particular, são usados como referência dois principais trabalhos relacionados [Collange et al. 2009] e [Hong et al. 2010], onde são propostos modelos matemáticos para estimar o CE das GPUs. Ambos os modelos foram

testados por meio de versões do módulo de recriptografia codificadas e instrumentadas para esse fim. Embora ainda não seja possível apresentar resultados para o primeiro modelo, foco de trabalho em andamento, cabe destacar os resultados preliminares gerados pelo segundo modelo que apontam um CE de pico de aproximadamente 193 W pela GPU (sendo o consumo de pico 184 W para a CPU usada nos experimentos), considerando a carga de trabalho máxima (512 clientes concorrentes).

5. Conclusões e trabalhos em andamento

A partir de uma análise qualitativa e quantitativa sobre a viabilidade do uso, para uma determinada aplicação, das GPUs, ao invés de um ou mais processadores *multicore*, este trabalho visa realizar a interface entre sistemas de distribuição de vídeo e criptografia assistida por GPU. A proposta baseada em GPU (API CUDA) tende a proporcionar uma vazão significativamente superior às versões CPU-*only*, conforme os resultados preliminares obtidos demonstraram, independentemente da API de programação paralela adotada (Pthreads e OpenMP). Cabe lembrar de que a diferença quantitativa expressa o comportamento para o ambiente experimental adotado, tal que, considerando-se a evolução das GPUs e dos processadores *multicore*, especula-se que a diferença de desempenho se torne ainda maior. Embora a atualização do SO tenha diminuído a diferença de desempenho entre as versões que utilizam a GPU e as CPU-*only*, esta aplicação precisa, potencialmente, ficar executando 24 horas por dia para atender a demanda dos clientes, logo cada segundo que for ganho de vantagem é crucial para manter a qualidade de serviço. Dentro do conceito da computação de alta eficiência, atualmente estão sendo realizados estudos para melhor estimar a eficiência energética desta proposta, incluindo a mensuração do consumo energético com o uso de instrumentos de aferição de potência dissipada. Uma vez conhecido os fatores do consumo energético, espera-se que seja possível determinar com precisão qual configuração de *software* + *hardware* possui o menor custo total de implementação.

Referências

Accelerators and GPUs track. (2009) IEEE SBAC-PAD´09, São Paulo, Brazil.

Collange, S., Defour D., and Tisserand, A. (2009) "Power Consumption of GPUs from a Software Perspective", ICCS´09, Louisiana, U.S.A.

Di Biagio, A., Barenghi, A., Agosta G., and Pelosi, G. (2009) "Design of a Parallel AES for Graphics Hardware using the CUDA framework", IEEE IPDPS´09, Rome, Italy.

Gomes, L., Neves, B. S., and Pinho, L. B. (2011) "Proposta e Avaliação de Recriptografia Assistida por GPU Aplicada a Servidores Escaláveis de Distribuição de VoD", ERAD´11, Porto Alegre, Brasil.

Hong, S., and Kim, H. (2010) "An Integrated GPU Power and Performance Model", ISCA '10, Saint Malo, France.

Manavski, S. A. (2007) "CUDA Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography", IEEE ICSPC´07, Dubai, United Arab Emirates.

Seshadrinathan, M., and Dempski, K. L. (2008) "Implementation of Advanced Encryption Standard for Encryption and Decryption of Images and Text on a GPU", IEEE CVPRW´08, Anchorage, AK, USA.