Computação sobre dados cifrados em *GPGPUs*

Pedro Geraldo M. R. Alves , Diego F. Aranha

Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Cidade Universitária Zeferino Vaz – CEP 13083-970 – Campinas – SP – Brazil

{pedro.alves,dfaranha}@ic.unicamp.br

Abstract. *Under the dominant cloud computing paradigm, employing encryption for data storage and transport may not be enough. Security guarantees should also be extended to data processing. Homomorphic encryption schemes are natural candidates for computation over encrypted data since they are able to satisfy the requirements imposed by the cloud environment. This work presents* cuYASHE *as a GPGPU implementation of the leveled fully homomorphic scheme* YASHE*. It employs* CUDA*, the Chinese Remainder Theorem and the Fast Fourier Transform to obtain significant performance improvements. In particular, there was a speedup between* 6 *and* 35 *times for homomorphic multiplication.*

Resumo. *No contexto da computação na nuvem, a aplicação de métodos criptográficos exclusivamente no armazenamento e transporte dos dados não é suficiente, uma vez que precisam ser revelados ao serviço para ocorrer processamento. Esquemas de cifração homomórfica são candidatos naturais para computação sobre dados cifrados, o que os torna capazes de satisfazer esse novo requisito de segurança. Este trabalho apresenta a* cuYASHE*, uma implementação em GPGPUs do criptossistema homomórfico* YASHE*. A* cuYASHE *emprega* CUDA*, o Teorema Chinês do Resto e a Transformada Rápida de Fourier para obter ganho de desempenho sobre o estado da arte. Em especial, destaca-se uma redução de* 6 *até* 35 *vezes no tempo de execução da operação de multiplicação.*

1. Introdução

A computação em nuvem tem sido responsável por uma profunda mudança na comunidade de processamento distribuído. A possibilidade de terceirizar a instalação, manutenção e a escalabilidade de servidores, somada a preços competitivos, faz com que esses serviços se tornem altamente atraentes.

Diversos esquemas criptográficos são utilizados como padrão para o armazenamento e transferência de dados. Contudo, no caso de serviços na nuvem existe a possibilidade de se lidar com um adversário que não apenas tenha acesso aos dados como também ao *hardware* que realiza seu processamento. Desse modo, é necessário que se estabeleça uma trajetória completamente sigilosa para os dados durante o transporte, armazenamento e processamento para que seja preservada a privacidade.

A criptografia homomórfica se mostra promissora para satisfazer esse novo requisito de segurança. Os esquemas dessa classe permitem que o processamento seja feito sobre criptogramas mesmo sem o conhecimento das chaves de cifração ou decifração. Assim, não há razão para que dados sejam revelados no momento do processamento.

O objetivo deste trabalho consistiu em obter ganho de desempenho no estado da arte do criptossistema homomórfico em nível YASHE [Bos et al. 2013]. Para isso, foram aplicadas técnicas de computação paralela em *GPGPUs* por meio da arquitetura CUDA; utilizou-se

o Teorema Chinês do Resto para simplificar a manipulação de inteiros grandes; e a Transformada Rápida de Fourier foi aplicada para reduzir a complexidade computacional das operações de multiplicação polinomial utilizadas por este criptossistema.

2. YASHE - *Yet Another Somewhat Homomorphic Encryption*

A classe de criptossistemas homomórficos é caracterizada por esquemas que permitem a aplicação de operações de adição ou multiplicação sobre seus criptogramas. O resultado dessas operações gera um novo criptograma que, quando decifrado, se apresenta de forma equivalente ao que seria esperado em uma operação sobre texto claro.

YASHE é um criptossistema completamente homomórfico em nível baseado em reticulados ideais [Bos et al. 2013]. Sua classificação significa que suporta uma quantidade limitada de operações de adição e multiplicação. Além disso, é um esquema probabilístico e tem sua segurança baseada nos problemas *RLWE* e *DSPR*. Ainda, o YASHE atinge o padrão de segurança *IND-CPA*, o que significa que seus criptogramas são resistentes a um adversário que seja capaz de cifrar texto claro.

3. Metodologia

Como forma de reduzir o tempo de execução das operações do YASHE, este trabalho se focou em paralelizar a implementação das operações polinomiais nas quais o esquema se baseia. Isso foi feito por meio da plataforma CUDA, escolhida principalmente devido a seu excepcional desempenho na aplicação de *GPGPUs* na resolução de problemas baseados em dados.

O Teorema Chinês do Resto, ou *CRT*, foi utilizado como ferramenta para simplificar a manipulação dos coeficientes polinomiais [Ding et al. 1996]. Sua função é mapear um polinômio com coeficientes inteiros arbitrariamente grandes em diversos polinômios com coeficientes inteiros arbitrariamente pequenos chamados polinômios residuais, ou simplesmente resíduos. Essa substituição permite que os coeficientes sejam manipulados utilizando uma aritmética mais simples e suportada nativamente pela arquitetura.

As operações do YASHE dependem de adições e multiplicações polinomiais. Enquanto aquelas tem complexidade linear e são trivialmente paralelizáveis, estas são consideravelmente mais complexas e requerem uma estratégia para a redução de sua complexidade computacional. Dessa forma, este trabalho estudou a aplicação da Transformada Rápida de Fourier, ou *FFT*, para a implementação dessa operação [Cooley and Tukey 1965]. Em seu domínio, a multiplicação polinomial é aplicada como uma simples multiplicação coeficiente-a-coeficiente. Assim, sua complexidade torna-se $\Theta(n \log n)$ com o custo da aplicação da transformada.

4. Implementação

A biblioteca cuYASHE foi desenvolvida durante a execução deste trabalho. A implementação consolida a metodologia descrita e permite a avaliação dos ganhos com a sua aplicação. Seu código está disponível ao público sob uma licença GNU GPLv3 [Alves and Aranha 2016]. Apesar do foco na execução em *GPUs*, algumas operações sobre inteiros grandes são realizadas pela *CPU*, como por exemplo a formatação e a cópia dos operandos entre as memórias. Essas operações são realizadas por meio da biblioteca *NTL* [Shoup 2003].

A aplicação do *CRT* nos operandos da cuYASHE é feita completamente na *GPU*. Dessa forma, uma vez que os resíduos ainda não foram calculados, o suporte a aritmética de inteiros grandes torna-se necessário. A biblioteca RELIC [Aranha and Gouvêa 2016] foi utilizada como base e as rotinas requeridas (adição, subtração, multiplicação, divisão e resto modular) foram adaptadas para CUDA.

A implementação da *FFT* foi realizada sobre a biblioteca cuFFT [NVIDIA 2015]. Para isso, precisou-se aplicar operações de conversão dos coeficientes entre o conjunto dos inteiros e o corpo dos complexos. Além disso, aritmética de ponto-flutuante utilizada por esta biblioteca gerou erros de precisão que precisaram ser mitigados para o funcionamento correto do algoritmo.

Para maximizar os ganhos com o uso das estratégias descritas, a cuYASHE trabalha com resíduos dentro do domínio da transformada. Ou seja, aplica-se o *CRT* seguido da *FFT* em cada um dos polinômios residuais. Assim, as operações polinomiais de adição e multiplicação tem aplicação coeficiente-a-coeficiente, o que implica em complexidade linear. Além disso, os resíduos são mantidos exclusivamente na memória da *GPU*, dispensando os custos envolvidos com a cópia de dados entre as memórias.

5. Resultados

Com o intuito de avaliar a qualidade da implementação e das estratégias propostas, as operações da cuYASHE foram comparadas com três trabalhos no estado da arte: [Bos et al. 2013], BLLN; [Lepoint and Naehrig 2014], *LN*; [Dowlin et al. 2015], SEAL. Foram utilizados os parâmetros de configuração do YASHE propostos por BLLN [Bos et al. 2013] e *LN* [Lepoint and Naehrig 2014] como padrão para as comparações de tempo de execução [1]. Os autores argumentam que esses parâmetros oferecem segurança de 80 *bits*.

Para as implementações disponíveis à comunidade, *LN* e SEAL, realizou-se medições dos tempos de execução das principais operações do YASHE: cifração, decifração, adição e multiplicação homomórfica. Os resultados apresentados são os tempos médios calculados a partir de 100 execuções isoladas de cada operação. A máquina utilizada possui uma *CPU Intel Xeon E5-2630 @ 2,6GHz* com uma *GPU NVIDIA GeForce GTX TITAN Black @ 0,98 GHz*. Além disso, utiliza-se a versão 7.5 do kit de desenvolvimento CUDA e as versões 4.8.4 dos compiladores *gcc* e *g++*. Por fim, empregou-se a biblioteca *NTL* 9.1.0 compilada sobre a *GMP* 6.0.0. A comparação com a BLLN é feita por meio dos resultados oferecidos pelos autores, uma vez que seu código não foi disponibilizado.

Tabela 1. Comparação entre a cuYASHE e as implementações *LN*, SEAL e BLLN. Os valores para as três primeiras foram medidos na mesma máquina, enquanto a última teve seus tempos fornecidos pelos autores. Os tempos para multiplicação homomórfica incluem o custo de relinearização.

Operação	cuYASHE(ms)	*LN*(ms)	SEAL(ms)	BLLN(ms)
Cifração	1,85	15,4	34,93	27
Decifração	2,01	13,71	34,1	5
Adição Homomórfica	0,02	0,59	0,18	0,02
Multiplicação Homomórfica	5,49	31,07	194,94	31

Como pode ser visto na Tabela 1, as operações de cifração, decifração e multiplicação homomórfica da cuYASHE apresentaram tempos 8, 7 e 9 vezes menores do que o trabalho *LN* e 15, 6 e $2,5$ vezes menores do que o BLLN, respectivamente. Sobre a SEAL os ganhos foram ainda mais expressivos, atingindo 19, 17 e 35 vezes, respectivamente.

A principal motivação para a utilização de um criptossistema homomórfico é poder operar sobre criptogramas. Dessa forma, apesar de ganhos na cifração e decifração serem importantes, operações homomórficas são o alvo principal para otimizações. Assim, os expressivos ganhos

[1] $R = \mathbb{Z}[X] / \left(x^{4096} + 1\right)$, $q = 2^{127} - 1$, $w = 2^{32}$, $t = 2^{10}$.

de velocidade apresentados sobre o estado da arte tem grande importância do ponto de vista de viabilizar aplicações práticas.

Por fim, este documento apresenta aperfeiçoamentos na implementação em relação a resultados preliminares [Alves and Aranha 2015]. A revisão na máquina de estados da CUYASHE permitiu a implementação da aritmética com operandos não apenas mapeados em resíduos pelo *CRT* como também interpolados para o domínio da *FFT*. Assim, a complexidade computacional dessas operações se tornou linear e evitou-se o custo de aplicação dessa transformada. Além disso, toda a aritmética passou a ser computada na *GPU*. Restando à *CPU* apenas o gerenciamento das operações.

6. Conclusão

Este trabalho investigou estratégias para a implementação do criptossistema YASHE em *GPGPUs* por meio da plataforma CUDA. Com os resultados obtidos a biblioteca CUYASHE foi desenvolvida, otimizada e disponibilizada à comunidade [Alves and Aranha 2016].

Aplicou-se o *CRT* para simplificar a manipulação de inteiros grandes na *GPU* e a *FFT* para a redução da complexidade de operações de multiplicação polinomial, nas quais o YASHE é altamente dependente.

A comparação com outros trabalhos da literatura demonstrou uma redução expressiva nos tempos de execução desse criptossistema, o que sugere que a metodologia proposta foi adequada ao contexto. Como trabalho futuro, espera-se a aplicação da CUYASHE em uma implementação de um protocolo que preserve privacidade.

Referências

[Alves and Aranha 2015] Alves, P. and Aranha, D. (2015). cuYASHE: Computação sobre dados cifrados em GPGPUs. In *XV Simpósio Brasileiro de Segurança da Informação e Sistemas Computacionais (SBSeg 2015)*, pages 55–60.

[Alves and Aranha 2016] Alves, P. and Aranha, D. (2016). cuYASHE. https://github.com/cuyashe-library/cuyashe. Acessado pela última vez: 19/05/2016.

[Aranha and Gouvêa 2016] Aranha, D. F. and Gouvêa, C. P. L. (2016). RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.

[Bos et al. 2013] Bos, J., Lauter, K., Loftus, J., and Naehrig, M. (2013). Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. Springer Berlin Heidelberg.

[Cooley and Tukey 1965] Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301.

[Ding et al. 1996] Ding, C., Pei, D., and Salomaa, A. (1996). *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*. World Scientific Publishing Co., Inc.

[Dowlin et al. 2015] Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. (2015). Manual for Using Homomorphic Encryption for Bioinformatics.

[Lepoint and Naehrig 2014] Lepoint, T. and Naehrig, M. (2014). A Comparison of the Homomorphic Encryption Schemes FV and YASHE. Springer International Publishing.

[NVIDIA 2015] NVIDIA (2015). CUDA Toolkit Documentation. http://docs.nvidia.com/cuda/cufft/. Acessado pela última vez: 12/08/2015.

[Shoup 2003] Shoup, V. (2003). NTL: A library for doing number theory. `http://www.shoup.net/ntl`. Acessado pela última vez: 05/03/2016.