

클라우드 환경을 위한 멀티 디바이스 Bootstrapping 최적화

하회리¹, 정헌희¹, 백윤희¹¹서울대학교 전기정보공학부

{wrha, hhjung}@sor.snu.ac.kr, ypaek@snu.ac.kr

Multi-device Bootstrapping Optimization for Cloud Computing

Whoi Ree Ha¹, Heonhui Jung¹, Yunheung Paek¹¹Dept. of Electrical and Computer Engineering and Inter-university Semiconductor Research Center, Seoul National University

요 약

클라우드 제공자들은 이미 여러 가속 기기들을 통해 더 좋은 가속 환경을 하지만 많은 연구들은 단일 기기 최적화에 중점을 두고 있다. 특히 동형암호에서의 연산 병목이 되는 Bootstrapping 은 굉장히 큰 연산 overhead 를 발생시킨다. 따라서 멀티 디바이스 클라우드 환경에 맞춰 Bootstrapping 연산을 모델링하고 최적화하였으며, 실험결과 최대 1.84 배의 speedup 을 달성하였다.

1. Introduction

머신러닝 서비스의 인기 증가는 프라이버시 문제를 야기했습니다. 동형암호는 암호화된 데이터에 대한 연산을 허용함으로써 프라이버시를 보존합니다. 이에 따라 사용자는 민감한 비암호화 데이터를 클라우드에 전송할 필요가 없습니다[1].

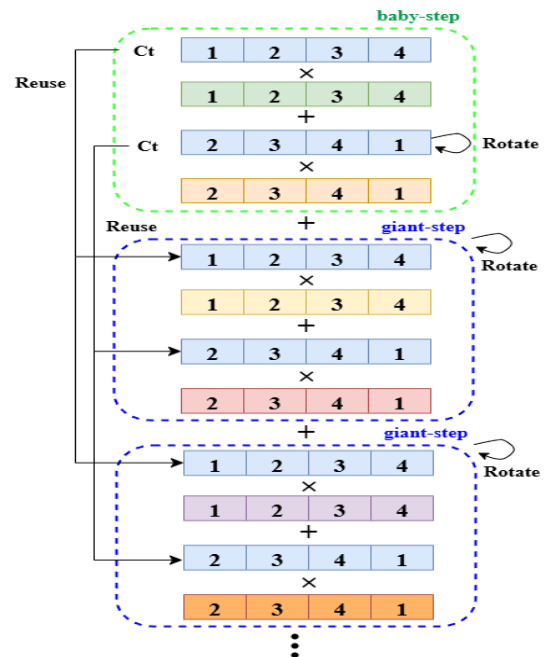
하지만 동형암호는 연산량을 굉장히 증가시키기 때문에 이를 효율적으로 최적화하는 부분이 필수입니다. 특히 Bootstrapping 은 noisy 한 암호화된 데이터를 새로운, noise 적은 암호 데이터로 변환함으로써 더 많은 동형암호 연산을 수행할 수 있게 해주는 핵심 연산입니다. 하지만 Bootstrapping 은 많은 연산량을 요구하며 대부분의 동형암호 application 의 연산 병목점이 됩니다.[2],[3]

기존 연구들은 단일 기기에서의 가속에 집중하지만, 주요 클라우드 제공자들은 더 좋은 가속을 위해 여러 대의 가속기를 제공합니다. [4],[5] 이러한 멀티 디바이스 환경은 아직 충분히 다루어지지 않은 새로운 최적화 과제를 제시합니다.

이 연구에서는 멀티 디바이스 클라우드 환경에서의 가속을 위해 Bootstrapping 연산을 최적화합니다. 기존 연구에서는 최적화를 위해 Bootstrapping 을 Baby-step Giant-step 으로 구분하여 연산을 수행하는데, 이 때 사용하는 configuration 에 따라서 연산의 수와 병렬성이

바뀌게 됩니다. 이 Baby-step Giant-step 을 수학적으로 모델링하여 멀티 디바이스 환경에 최적화합니다. 이를 통해 최대 1.84 배의 speedup 을 달성합니다.

2. Baby-step Giant-step



(그림 1) Baby-step Giant-step 설명

Bootstrapping 에서 여러 연산을 수행하지만 그 중 가장 비중이 높은 건 Discrete Fourier Transform (DFT) 연산입니다. DFT 연산은 다수의 버터플라이 연산을 포함하며, 이는 여러 암호문 곱셈 및 회전을 요구합니다. 특히 암호문의 회전은 많은 연산량을 요구합니다. 따라서 DFT 연산의 암호문 회전을 최소화하기 위해 baby-step giant-step 알고리즘이 도입되었습니다. 이 알고리즘은 (그림 1)과 같이 baby-step 과 giant-step 두 부분으로 나뉩니다. Baby-step 은 원래 행렬 곱셈의 작은 부분을 직접 수행합니다. 이는 데이터 순서를 유지하기 위해 입력 암호문의 회전을 필요로 합니다. Giant-step 은 baby-step 에서 회전된 입력을 재사용하여, 나머지 계산을 baby-step 의 크기로 나뉩니다. 각 입력 암호문이 회전되는 baby-step 과 대조적으로, 각 giant-step 은 마지막에 한 번의 회전만을 요구하여, 계산 비용이 많이 드는 회전 연산을 최소화합니다.

3. Baby-step Giant-step Optimization

기본적으로 위 설명과 같이 baby-step 에서 더 많은 입력을 사용하게 되면 giant-step 의 개수가 줄어들게 됩니다. 전체 연산을 baby-step 에서 사용된 입력 수에 맞춰서 나눠지기 때문입니다. 따라서 baby-step 의 크기에 따라서 전체 동형암호 연산 수가 달라집니다. 이를 $bs \times gs = C$ 라고 모델링 할 수 있습니다. bs 는 baby-step 의 크기이고, gs 는 giant-step 의 수, C 는 constant 입니다. 하지만 멀티 디바이스 환경에서는 연산의 병렬성과 data-dependency 로 인한 기기간의 통신 overhead 가 발생합니다.

단일 기기에서는 baby-step 이 단 한 번 수행되지만 baby-step 의 입력들이 giant-step 에서 재사용되기 때문에 한 기기에서 baby-step 을 수행하여 회전된 암호문들을 다른 기기들에게 broadcasting 해주는것보다 각 기기가 baby-step 을 수행하는 것이 더 효율적입니다. 따라서 각 기기에서 발생하는 연산 수는 $bs + \frac{C}{N \times bs}$ 라고 정의할 수 있습니다. 이 때 N 은 사용 가능한 연산 기기의 수입니다. 또한 giant-step 을 수행한 후 모든 데이터들의 합을 수행해야 되며 이는 기기간의 통신을 필요로 합니다. 이 때 통신은 $\log_2(N)$ 번 발생하게 됩니다. 따라서 최종적으로 이 연산을

$$T_{bs} + \frac{C}{N \times T_{bs}} + T_{comm} \times \log_2(N)$$

과 같이 모델링 할 수 있습니다. 이 때 T_{bs} 는 baby-step 을 수행하는데 걸리는 시간, T_{comm} 은 기기간의 통신 시간입니다. 위 식을 사용하여 각 N 에 따라서 최적의 baby-step 크기를 결정하여 멀티 디바이스 환경에 따라 최적화할 수 있습니다.

4. Evaluation

State-of-the-art 연구인 Hydra[6]에서 사용한 환경을 사용하였습니다. 실험 환경은 x86 cpu 를 가지고 여러 Alveo U280 FPGA 를 가진 클라우드 환경을 사용했습

니다. 이 때, 기기간의 통신 bandwidth 는 100Gb/s 입니다. 동형암호 parameter 는 Polynomial degree= 2^{16} 이며 $\log Q=1260$, $\log PQ=1692$ 입니다. 또한 C 는 32768 입니다.

# of Devices	Naïve (s)	Ours (s)	Speedup
1	0.014297	0.014297	1.00
2	0.008049	0.007447	1.08
4	0.004975	0.004072	1.22
8	0.003488	0.002448	1.42
16	0.002795	0.001522	1.84

<표 1> 실험 결과

<표 1>은 baby-step giant-step 최적화를 하지 않고 기존 단일 기기 가속에서 사용된 baby-step giant-step configuration 을 사용했을 때의 결과 (naïve)와 최적화를 수행한 결과를 비교하였습니다. 이 때 우리 연구는 최대 1.84 배의 speedup 을 보였습니다. 또한 사용할 수 있는 기기가 늘어남에 따라 더 좋은 speedup 을 보여줍니다. 이는 기기가 늘어남에 따라 통신 및 accumulation 에서의 수행 시간이 상대적으로 늘어나기 때문입니다. 이를 우리는 최적화를 통하여 최소화하여 더 많은 가속기를 사용할 수 있는 환경에서 더 좋은 성능을 보였습니다.

5. Conclusion

이 연구는 클라우드가 제공하는 멀티 디바이스 환경에 맞춰서 동형암호 연산의 가장 핵심적인 연산인 Bootstrapping 연산을 최적화하였습니다. 기존 연구들은 단일 기기 환경을 가정하여 baby-step giant-step 을 정하여 사용하지만, 사용하는 기기, 통신 bandwidth 에 따라서 가속효율이 바뀔 수 있는 멀티 디바이스 환경에서는 효율적이지 않습니다. 따라서 우리는 멀티 디바이스 환경에 맞춰 baby-step giant-step 알고리즘을 모델링하고, 이 모델을 사용하여 최적화하였습니다. 실험 결과 최대 1.84 배의 speedup 을 보였습니다.

Acknowledgement

이 논문은 2025 년도 BK21 FOUR 정보기술 미래인재 교육연구단, 정부(과학기술정보통신부)의 재원으로 한국연구재단(RS-2023-00277326), 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.RS-2023-00277060, 개방형 오픈 AI 반도체 설계 및 SW 플랫폼 기술개발). IDEC 에서 EDA Tool 를 지원받아 수행하였습니다.

참고문헌

- [1] Cheon, Jung Hee, et al. "Introduction to homomorphic encryption and schemes." *Protecting Privacy through Homomorphic Encryption*. Cham: Springer International Publishing, 2022. 3-28.
- [2] Kim, Sangpyo, et al. "Bts: An accelerator for bootstrappable fully homomorphic encryption." *Proceedings of the 49th annual international symposium on computer architecture*. 2022.

- [3] Agrawal, Rashmi, Anantha Chandrakasan, and Ajay Joshi. "Heap: A fully homomorphic encryption accelerator with parallelized bootstrapping." *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2024.
- [4] "F2." *Amazon Web Services, Inc.*, 2024, aws.amazon.com/ec2/instance-types/f2/.
- [5] "NP Size Series - Azure Virtual Machines." *Microsoft.com*, 24 Oct. 2024, learn.microsoft.com/en-us/azure/virtual-machines/sizes/fpga-accelerated/np-series?tabs=sizebasic.
- [6] Yang, Yinghao, et al. "Hydra: Scale-out FHE Accelerator Architecture for Secure Deep Learning on FPGA." *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2025.