Cross-scheme 완전동형암호(FHE)가속기를 위한 DRAM-Aware Memory 접근 최적화 기법에 대한 연구

정헌희 [1], 백윤흥 [2]
[1] 서울대학교 전기 정보공학부, 반도체 공동연구소 박사과정
[2] 서울대학교 전기 정보공학부, 반도체 공동연구소 교수

honeyjung@snu.ac.kr ypaek@snu.ac.kr

DRAM-Aware Memory Access Optimization for Cross-scheme Fully Homomorphic Encryption(FHE) Accelerators

Heonhui Jung[1], Yunheung Paek[2]
[1]Dept. of Electrical and Computer Engineering and Inter-University Semiconductor Research Center (ISRC), Seoul National University
[2]Dept. of Electrical and Computer Engineering and Inter-University Semiconductor Research Center (ISRC), Seoul National University

요 약

최근 MLaaS 환경에서의 프라이버시 보호를 위해 동형암호(FHE) 기반 가속기에 대한 수요가 증가하고 있으나, 높은 연산 복잡도와 메모리 병목으로 인해 성능과 에너지 효율 확보가 중요한 도전 과제로 남아 있다. 특히 CKKS 와 TFHE 를 혼합 사용하는 cross-scheme FHE 가속기는 residue polynomial-wise, coefficient-wise, element-wise 및 key access 등 불규칙한 메모리 접근 패턴을 요구하여 DRAM 의 row-miss 가 빈번하게 발생한다. 본 연구에서는 DRAM 의 구조적 제약을 고려하여, 암호문 단위가 아닌 **연산기 단위로 데이터를 분할·저장하는 최적화 기법**을 제안한다. DRAMSim3 을 활용한 HBM2 cycle-accurate 시뮬레이션 결과, 제안 기법은 최적화 전보다 메모리 대역폭을 약 **66.20%** 개선하였으며, READ 에너지 소모를 약 **60.54%** 절감하였다. 이는 DRAM 컨트롤러의 자체적 rescheduling 기능에도 불구하고, FHE 가속기 환경에서는 workload-aware 메모리 접근 최적화가 필요함을 보여주며, 차세대 cross-scheme FHE 가속기의 성능 및 효율 향상에 기여할 수 있음을 입증한다.

1. 서론

인공 신경망(Artificial Neural Network, ANN)의 급속한 발전과 함께, 복잡한 모델을 클라우드 환경에서 제공하는 Machine Learning as a Service (MLaaS)의 수요가 증가하고 있다.[1] 사용자는 고성능 연산 자원을 직접 보유하지 않고도 클라우드에서 신경망 추론 결과를 얻을 수 있지만, 이 과정에서 민감한 입력 데이터를 서비스 제공자에게 전송해야 하므로 개인정보 노출 위험이 따른다. 기존의 암호화 방식은 데이터 전송 구간의 보호에는 효과적이나, 클라우드 서버 내부에서 연산을 수행하기 위해서는 복호화가 필요하므로 근본적인 보안 문제를 해결하지 못한다. 이러한 맥락에서 동형암호(Fully Homomorphic Encryption, FHE) 는 암호화된 상태로 연산을 지원하여 프라이버시 보존형 MLaaS 실현을 위한 핵심 기술로 자리잡고 있으며, 이를 실용화하기 위한 하드웨어 가속 연구가 활발히 진행되어 왔다. [2], [3]

특히 주로 연구되는 FHE scheme 들인 CKKS[4]와 TFHE[5]는 근사 연산 기반의 CKKS 는 대규모 연산에 유리하나 정확도 저하 문제가 존재하며, TFHE 는 높은 정확도를 제공하지만 연산 속도가 상대적으로 느리다. 이에 따라 CKKS 와 TFHE 의 장점을 결합한 cross-scheme 기법이 연구되기 시작하였고, 이를 효과적으로 지원하기 위한 cross-scheme 하드웨어 가속기에 대한 연구가 진행되고 있다. [6], [7], [8], [9]

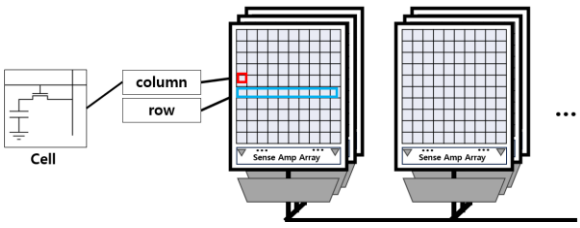
그러나 이러한 가속기 연구의 성능은 연산 유닛뿐 아니라 메모리 시스템에 크게 제약을 받는다. 동형암호 연산은 높은 차수의 다항식

으로 구성되어 있어 대규모 암호문과 대응되는 연산 키를 반복적으로 불러와야 하며, 이로 인해 매우 높은 메모리 대역폭이 요구된다. 기존 연구들은 HBM(High Bandwidth Memory) 채택이나 메모리 재사용(memory reuse) 기법을 통해 이를 완화하고자 하였으나,[10], [11], [12] DRAM 의 구조적 특성–예컨대 주기적 refresh 지연, row/bank 접근에 따른 가변적인 latency–은 충분히 고려되지 않았다.

이 문제는 cross-scheme 환경에서 더욱 심각해진다. CKKS 와 TFHE 는 각기 다른 연산 데이터를 필요로 하므로, DRAM 접근 패턴이 더욱 불규칙해지고 실제 대역폭은 저하된다. 본 연구는 이러한 한계를 실험을 통해 보이고 해결하기 위해 DRAM 의 구조적 특성을 반영한 메모리 접근 기법을 제안하는 것을 목표로 한다.

2. 배경

2.1 DRAM 의 구조와 동작



(그림 1) DRAM 의 Bank 구조

DRAM 은 기본적으로 1 비트를 저장하는 cell 로 구성되며, 각 셀은 하나의 트랜지스터와 커패시터로 이루어진다. 이러한 셀들이 2 차원 배열 형태로 집적되어 각각의 bank 를 형성한다. 이때 array 의 각 column 은 bit line 을 통해 연결되며, 열마다 배치된 sense amplifier 가 미세한 전하 변화를 증폭하여 안정적으로 데이터를 판독할 수 있도록 한다. 이러한 bank 들이 모여 **rank** 를 이루며, 하나의 rank 는 동일한 동작 명령을 동시에 수행하는 병렬적 단위로 동작한다. 여러 rank 와 bank 는 다시 **channel** 단위로 묶여 최종적으로 메모리 컨트롤러와 연결되게 된다.

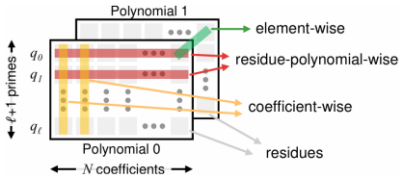
DRAM 에서 읽기/쓰기 연산을 수행하기 위해서는 우선 셀 내부 커패시터에 저장된 전하를 amplifier 로 이동시켜 증폭하는 과정, 즉 activation 이 필요하다. 이 activation 은 비교적 긴 지연시간을 수반하며, 한 번 활성화된 row 내에서는 동일 row 의 다른 column 접근이 빠르게 이루어질 수 있다(이를 row hit 이라 한다). 그러나 만약 다른 row 에 접근할 경우 기존 row 를 precharge 한 뒤 새로운 row 를 다시 활성화해야 하므로(row miss), 접근 지연이 크게 증가한다.

또한 DRAM 은 전력 및 안정성 한계로 인해 Four-Activation Window (FAW) 라는 제약을 가진다. 이는 일정한 시간 윈도우 내에서 동시에 최대 네 개의 bank 만 활성화할 수 있도록 제한하는 규칙으로, 더 많은 bank 접근을 원할 경우 기존 bank 를 precharge 하고 새로운 bank 를 activate 해야 한다. 이러한 제약은 특히 다수의 bank 에 불규칙하게 접근하는 워크로드에서 성능 저하를 야기한다.

이 문제를 완화하기 위해 현대 DRAM 컨트롤러들은 command queue 구조를 도입하고 있다. 컨트롤러는 동일 bank 내에서 발생하는 연속적인 접근 요청을 큐에서 묶어 처리함으로써 row hit 을 최대화하고 불필요한 precharge/activate 동작을 줄인다. 그러나 command queue 는 하드웨어적 버퍼로서 크기에 한계가 있으며, 대규모 병렬 접근이나 불규칙한 접근 패턴에서는 여전히 성능 병목이 발생할 수 있다.[13]

2.2 CKKS 와 TFHE 의 연산 구조

CKKS 스킴에서 암호문은 복수의 고차 다항식(residue polynomials) 으로 구성되며, 이는 거대한 2 차원 배열 형태로 메모리에 저장된다. 연산 과정에서는

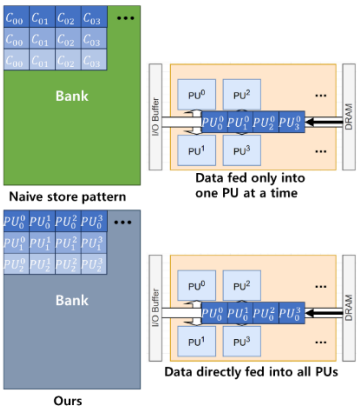


(그림 2) CKKS 연산시 암호문 데이터에 대한 접근구조[3]

서로 다른 차원의 접근 패턴이 요구된다. 예를 들어, NTT(Number Theoretic Transform) 수행 시에는 다항식 단위로(residue polynomial-wise) 데이터를 불러와야 하며, 암호문 간 덧셈이나 NTT 가 완료된 암호문 간 곱셈에서는 element-wise 방식으로 서로 다른 암호문 데이터를 동시에 참조한다. 또한 곱셈 이후의 관리 연산(예: rescaling, modulus switching 등)에서는 다항식의 계수 단위(coefficient-wise) 접근이 요구된다.[3] 즉, CKKS 연산은 residue polynomial-wise, element-wise, coefficient-wise 접근이 교차적으로 발생하여 메모리 접근 패턴이 매우 불규칙적이다.

TFHE 연산으로 전환되는 경우 상황은 더욱 복잡해진다. CKKS 암호문을 TFHE 암호문으로 변환하는 과정에서 rotation key 및 key-switching key 와 같은 부가적인 키 데이터를 참조해야 하며,[14] 이 역시 residue polynomial-wise, element-wise, coefficient-wise 접근이 혼합적으로 발생한다. 이러한 과정은 DRAM 의 row locality 를 해치기 때문에 row-miss 가 빈번히 발생하고, 앞서 언급한 activation/precharge 지연이 누적되면서 전반적인 메모리 병목을 심화시키는 원인이 된다.

3. 제안 기법



(그림 3) 최적화 전(naïve) 후(Ours) 방식의 비교

본 연구의 주된 목적은 row-hit 비율을 극대화하고, 다양한 접근 패턴이 혼합되는 상황에서도 일정 수준 이상의 hit ratio 를 유지하는 것이다. 기존의 단순한 데이터 배치 변경 방식은 한 가지 접근 패턴(residue polynomial-wise, coefficient-wise, element-wise, key access)에 최적화될 경우, 다른 접근 패턴에서는 오히려 접근성이 저하되는 한계가 있다. 따라서 본 연구에서는 데이터 배치 최적화 대신, 동형암호 가속기의 병렬성에 착안한 새로운 접근법을 제안한다.

동형암호 가속기들은 본질적으로 대규모 다항식 연산을 병렬적으로 처리하기 위해 설계되어 있으며, 병렬화가 상대적으로 어렵다고 평가되는 TFHE 가속기조차도 intra-chip communication 을 최소화하기 위해 high-level 연산 단위에서 서로 독립적인 연산기를 배치하여 병렬성을 확보하고 있다.[6], [7], [8], [12] 이러한 특성에 기반하여, 우리는 기존처럼 암호문 단위로 데이터를 저장 · 로드하는 대신, 연산기(Processing Unit, PU) 단위로 데이터를 분할 · 저장하는 방식을 제안한다.

구체적으로, (그림 3)에서와 같이 DRAM 에서 각 read cycle 마다 하나의 column 을 읽을 때, 해당 column 안에는 여러 연산기에 동시에 공급될 데이터가 함께 포함되도록 batch 단위로 암호문을 나누어 저장하는 기법을 제안한다. 이를 통해 각 연산기가 필요한 데이터를 동시에 확보할 수 있으며, 불규칙한 접근 패턴에서 row-miss 빈도를 줄일 수 있다.

4. 성능 평가 방법

제안 기법의 성능을 검증하기 위해, 우리는 cycle-accurate 하게 DRAM 동작을 시뮬레이션 할 수 있는 오픈소스 메모리 시뮬레이터인 DRAMSim3[15]을 활용하였다. DRAMSim3 은 실제 DRAM 타이밍 제약과 명령 스케줄링을 정확히 모델링할 수 있어, 제안한 메모리 접근 기법이 DRAM 의 구조적 특성(row activation, precharge, FAW 제약 등)에 미치는 영향을 정밀하게 분석하는 데 적합하다.

특히 본 연구에서는 cross-scheme FHE 가속기에서 널리 사용되는 HBM2[16]를 주요 대상으로 설정하였으며, DRAMSim3 과 함께 제공되는 8Gb HBM2 메모리와 DRAM 컨트롤러 파라미터를 사용하여 시뮬레이션 하였다.

실험은 (그림 3)과 같이 메모리를 각각 최적화하기 전(naïve store pattern), 최적화 한 후(Ours)로 배열한 상황에서 수행되었으며, CKKS 연산의 다양한 접근 패턴인 polynomial-wise, element-wise, coefficient-wise 및 TFHE 환경으로의 conversion 시를 고려한 rotation key 에 접근하여 순차적으로 READ 하는 상황을 가정하였다. 이러한 시나리오에서 메모리 대역폭과 에너지 소비량을 측정하고 비교하였다.

5. 실험 결과 및 결론

	최적화 전	최적화 후
Bandwidth	3605.51MBs	5992.33MB/s
총 사용 에너지	2.36mJ	1.47mJ

(표 1) 최적화 전 후의 비교

실험 결과, 제안한 batched reading 기법은 기존 대비 메모리 대역폭과 에너지 효율에서 유의미한 개선을 보였다. 구체적으로, 메모리 대역폭은 최적화 이전 3605.51 MB/s 에서 5992.33 MB/s 로 향상되어 **66.20%**의 개선을 달성하였다. 또한 READ 과정에서 소모된 에너지는 2.36 mJ 에서 1.47 mJ 로 감소하여 약 **60.54%**의 절감 효과를 보였다.

이러한 개선은 DRAM 접근 과정에서 불필요한 activation 및 precharge 동작 빈도의 감소 덕분이다. 잦은 row 전환은 sense amplifier 의 반복적인 활성화를 요구하여 단순히 latency 증가뿐 아니라 에너지 소비까지 악화시키는데, 제안된 기법은 row locality 를 강화함으로써 이를 효과적으로 완화하였다.

한편, 현대 DRAM 컨트롤러들은 내부적으로 command queue 기반의 rescheduling 기법을 통해 동일 bank 내 요청을 묶어 처리하거나 row hit 을 최대화하려는 최적화를 수행한다. 그러나 본 연구 결과는 이러한 컨트롤러 차원의 최적화만으로는 cross-scheme FHE 가속기에서 발생하는 불규칙한 접근 패턴을 충분히 보완할 수 없음을 보여준다. 즉, DRAM 구조적

제약과 workload 특성 간의 불일치를 해결하기 위해서는, 메모리 컨트롤러의 스케줄링을 넘어 workload-aware 데이터 배치 및 접근 기법이 반드시 병행되어야 한다.

따라서 본 연구에서 제안한 batched reading 기법은 단순한 컨트롤러 레벨 최적화를 보완하며, FHE 가속기와 같이 대규모 병렬 연산과 불규칙한 메모리 접근이 혼합된 환경에서 성능과 에너지 효율을 동시에 개선할 수 있는 실질적 방안을 제시한다.

ACKNOWLEDGEMENT

이 논문은 2025 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구이며 (RS-2023-00277326). BK21 FOUR 정보기술 미래인재 교육연구단에 의하여 지원되었으며 2025 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2021-0-00528,

하드웨어 중심 신뢰계산기반과 분산 데이터보호박스를

위한 표준 프로토콜 개발) 2023 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원

의 지원을 받아 수행된 연구임 (No.RS-2023-00277060, 개방형 엣지 AI 반도체 설계 및 SW 플랫폼 기술개발)

참고문헌

[1] I. Grigoriadis, E. Vrochidou, I. Tsiatsiou, and G. A. Papakostas, “Machine Learning as a Service (MLaaS)—An Enterprise Perspective,” in *Lecture Notes in Networks and Systems*, Springer Science and Business Media Deutschland GmbH, 2023, pp. 261–273. doi: 10.1007/978-981-19-6634-7_19.

[2] L. W. Folkerts, C. Gouert, and N. G. Tsoutsos, “REDsec: Running Encrypted Discretized Neural Networks in Seconds,” in *30th Annual Network and Distributed System Security Symposium, NDSS 2023*, The Internet Society, 2023. doi: 10.14722/ndss.2023.24034.

[3] S. Kim *et al.*, “BTS: An Accelerator for Bootstrappable Fully Homomorphic Encryption,” in *Proceedings - International Symposium on Computer Architecture*, Institute of Electrical and Electronics Engineers Inc., Jun. 2022, pp. 711–725. doi: 10.1145/3470496.3527415.

[4] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, “A Full RNS Variant of Approximate Homomorphic Encryption,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2019, pp. 347–368. doi: 10.1007/978-3-030-10970-7_16.

[5] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, “TFHE: Fast Fully Homomorphic Encryption Over the Torus,” *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, Jan. 2020, doi: 10.1007/s00145-019-09319-x.

[6] Y. Wei, X. Wang, S. Bian, Y. Huang, W. Zhao, and Y. Jin, “PPGNN: Fast and Accurate Privacy-Preserving Graph Neural Network Inference via Parallel and Pipelined Arithmetic-and-Logic FHE Accelerator,” in *Proceedings - Design Automation Conference*, Institute of Electrical and Electronics Engineers Inc., Nov. 2024. doi: 10.1145/3649329.3656517.

[7] X. Deng *et al.*, “Trinity: A General Purpose FHE Accelerator,” in *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, IEEE Computer Society, 2024, pp. 338–351. doi: 10.1109/MICRO61859.2024.00033.

[8] M. Zhou *et al.*, “UFC: A Unified Accelerator for Fully Homomorphic Encryption,” in *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, IEEE Computer Society, 2024, pp. 352–365. doi: 10.1109/MICRO61859.2024.00034.

[9] J. Mu *et al.*, “Alchemist: A Unified Accelerator Architecture for Cross-Scheme Fully Homomorphic Encryption,” in *Proceedings - Design Automation Conference*, Institute of Electrical and Electronics Engineers Inc., Nov. 2024. doi: 10.1145/3649329.3657331.

[10] N. Samardzic and D. Sanchez, “BitPacker: Enabling High Arithmetic Efficiency in Fully Homomorphic Encryption Accelerators,” in *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*, Association for Computing Machinery, Apr. 2024, pp. 137–150. doi: 10.1145/3620665.3640397.

[11] J. Kim *et al.*, “ARK: Fully Homomorphic Encryption Accelerator with Runtime Data Generation and Inter-Operation Key Reuse,” in *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, IEEE Computer Society, 2022, pp. 1237–1254. doi: 10.1109/MICRO56248.2022.00086.

[12] K. Nam, H. Oh, H. Moon, and Y. Paek, “Accelerating N-bit operations over TFHE on commodity CPU-FPGA,” in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, Institute of Electrical and Electronics Engineers Inc., Oct. 2022. doi: 10.1145/3508352.3549413.

[13] E. Ipek, O. Mutlu, J. F. Martínez, and R. Caruana, “Self-optimizing memory controllers: A reinforcement learning approach,” in *Proceedings - International Symposium on Computer Architecture*, 2008, pp. 39–50. doi: 10.1109/ISCA.2008.21.

[14] H. Chen, W. Dai, M. Kim, and Y. Song, “Efficient Homomorphic Conversion Between (Ring) LWE Ciphertexts.”

[15] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, “DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator,” *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 110–113, Jul. 2020, doi: 10.1109/LCA.2020.2973991.

[16] “JEDEC STANDARD High Bandwidth Memory DRAM (HBM1, HBM2).” [Online]. Available: www.jedec.org