https://doi.org/10.7840/kics.2023.48.10.1211

Look-Up Table을 이용한 부호화 기반 프라이버시 보장 행렬 곱 기법

김 민 철◆, 양 희 철°

Look-Up Table-Based Fully Private Coded Matrix Multiplication

Minchul Kim◆, Heecheol Yang°

요 약

Private Coded Matrix Multiplication (PCMM)은 기존의 coded computation과 Private Information Retrieval (PIR) 기법을 결합한 기법으로, 많은 파생 연구가 진행되고 있다. Fully Private Coded Matrix Multiplication (FPCMM)은 PCMM의 파생 연구 중 하나로서, 하나의 master와 다수의 worker로 구성된 분산 컴퓨팅 시스템에서 master가 자신의 데이터를 가지지 않는 경우를 고려한다. 본 논문에서는 이러한 FPCMM에서 Look-Up Table (LUT)를 이용하여 계산의 효율성을 높이는 방법을 제안한다.

Key Words : coded computation, private information retrieval

ABSTRACT

Private Coded Matrix Multiplication (PCMM) is a technique that combines existing coded computation and Private Information Retrieval (PIR) methods, and it has been the subject of extensive research. Fully Private Coded Matrix Multiplication (FPCMM) is one of the derivative studies of PCMM, which considers the scenario where a master node in a distributed computing system composed of multiple worker nodes does not possess its own data. In this paper, we propose a method to enhance the efficiency of computation in such FPCMM scenarios by utilizing Look-up tables (LUTs).

Ⅰ. 서 론

Private Coded Matrix Multiplication (PCMM)[1]은 coded computation[2]과 Private Information Retrieval (PIR)[3] 기법을 결합한 방법으로, coded computation을 통해 지연 시간을 경감할 수 있고 PIR을 통해 master의 data access privacy를 보장할 수 있다. 최근 이와 관련한 파생 연구들이 제안되었다[4-8]. 그 중, Fully Private Coded Matrix Multiplication (FPCMM)[9]은 두 파일 데이터의 행렬 곱 연산을 수행할 때 두 파일 모두에 대한 data access privacy를 보장하는 시스템 모델이다.

본 논문에서는 Look-Up Table (LUT) 기반의 새로운 FPCMM 모델과 LUT 디자인을 제안한다. FPCMM은 PCMM과 달리 가능한 행렬 곱의 경우의 수가 유한하므로 각 worker들이 LUT를 이용한다고 가정하면 전체 행렬 곱 연산 중에서 LUT에 포함된 부분은 별도의 재계산 없이 바로 전송이 가능하므로 worker의 computational complexity와 runtime 측면에서 LUT를 사용하지 않을 때 대비 이득이 있다. 즉, FPCMM에서의 LUT의 도입은 불필요한 반복 계산으로 인한 컴퓨팅 자원의 낭비를 해결할 수 있는 방법이다.

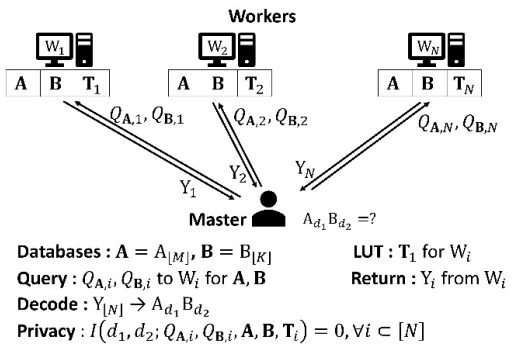

Fig. 1. System model

※ 본 연구는 전북대학교 2022-2023년도 교내연구비, 한국연구재단 이공분야기초연구사업(2022R1C1C1005749, RS-2023-00239524) 지원에 의해 수행되었습니다.

◆ First Author : (ORCID:0000-0002-0987-3911) Jeonbuk National University, Department of Computer Science and Artificial Intelligence, mc.kim@jbnu.ac.kr, 조교수, 정회원

° Corresponding Author : (ORCID:0000-0002-2802-2143) Chungnam National University, Department of Computer Convergence, hcyang@cnu.ac.kr, 조교수, 종신회원

논문번호 : KICS2023-08-033, Received August 4, 2023; Revised August 7, 2023; Accepted August 7, 2023

Ⅱ. 시스템 모델

LUT 기반 FPCMM의 시스템 모델은 그림 1과 같다. 하나의 master와 N개의 worker들로 구성된 시스템이며, 각 worker W_i는 2개의 파일 데이터베이스 **A**, **B**와 LUT $\mathbf{T}_i$를 저장하고 있다. 이 때, 각 파일 데이터베이스에 포함된 파일의 개수는 각각 M, K개이며, 각 파일 데이터베이스 내의 파일들의 크기는 모두 동일하다고 가정한다. 또한 본 논문에서 worker들은 서로 협력하지 않는다고 가정한다. Master는 데이터베이스 **A**에 포함된 특정 파일 $\mathrm{A}_{d_1}\,(d_1 \in [M])$과 데이터베이스 **B**에 포함된 특정 파일 $\mathrm{B}_{d_2}\,(d_2 \in [K])$의 행렬 곱 $\mathrm{C}_{d_1 d_2} = \mathrm{A}_{d_1}\mathrm{B}_{d_2}$을 worker들을 통해 계산하고자 하며, 이를 위해 각 worker W_i에 2종류의 query $Q_{\mathbf{A},i}$, $Q_{\mathbf{B},i}$를 보낸다. 이를 수신한 worker W_i는 자신이 저장하고 있는 2개의 파일 데이터베이스와 LUT $\mathbf{T}_i$의 정보를 종합하여 master에게 전송할 응답 Y_i 를 결정한다. Master는 N개의 worker들로부터 N개의 응답 $\mathrm{Y}_{[N]}$을 수신 후 복호화 과정을 거쳐 목표로 했던 행렬곱 $\mathrm{C}_{d_1 d_2} = \mathrm{A}_{d_1}\mathrm{B}_{d_2}$를 복원한다. Worker W_i에 대한 privacy constraint는 아래 식과 같다:

$$I\left(d_1, d_2; Q_{\mathbf{A},i}, Q_{\mathbf{B},i}, \mathbf{A}, \mathbf{B}, \mathbf{T}_i\right) = 0$$

이러한 LUT는 worker의 저장 용량이 제한적인 일반적인 FPCMM 문제 전반에 적용이 가능하다. 예를 들어, FPCMM을 이용한 추천 시스템에서 데이터베이스 **A**에는 추천 서비스 사용자의 선호도 데이터들의 프리셋 유형별로 저장되어 있고, 데이터베이스 **B**에는 쇼핑, 음악, 게임 등 사용자가 추천받고자 하는 각 카테고리의 데이터가 저장되어 있다고 가정하자. 즉, 사용자가 **A**에서 자신과 가장 비슷한 프리셋 유저의 선호도 데이터를 선택하고, **B**에서 추천받을 카테고리를 선택하면 추천 아이템은 행렬곱 $\mathrm{A}_{d_1}\mathrm{B}_{d_2}$를 기반으로 결정된다. 이 때 각 worker의 저장용량의 최대치를 넘지 않는 한도에서 MK개의 모든 가능한 행렬 곱 $\mathrm{A}_{d_1}\mathrm{B}_{d_2}$ 조합들의 전부 혹은 일부를 저장하여 LUT를 구성하면 계산 속도를 높일 수 있다.

Ⅲ. FPCMM을 위한 LUT 디자인

3.1 Full-size LUT vs Partial LUT

2개의 파일 데이터베이스 **A**, **B**가 각각 M, K개의 파일로 구성되므로 master가 고려할 수 있는 행렬 곱 $\mathrm{A}_{d_1}\mathrm{B}_{d_2}$의 가짓수는 MK개이다. 따라서 각 worker의 저장 용량이 충분하다면, 각 worker W_i가 저장하는 LUT $\mathbf{T}_i$에는 MK개의 행렬 곱 $\mathrm{A}_m\mathrm{B}_k\,(m \in [M], k \in [K])$이 모두 저장될 수 있고, 그 결과 FPCMM 문제는 MK개의 파일(행렬 곱) 중 하나를 data access privacy를 보장하면서 다운로드 받아야 하는 기존의 PIR 문제로 바뀐다. 즉, 각 worker node W_i 별로 모든 가능한 MK개의 행렬 곱들을 미리 계산하여 LUT $\mathbf{T}_i$에 저장하고, 각 행렬 곱을 식별하기 위한 MK개의 식별자를 master에 알려주면, master는 자신이 원하는 행렬 곱에 해당하는 식별자를 PIR 방식으로 은폐하면서 별도의 계산과정 필요없이 바로 다운로드만 하면 되는 것이다. 이렇게 worker node의 저장 용량이 충분히 커서 가능한 계산의 모든 경우가 LUT에 저장되는 경우를 full-size LUT라 하자.

상술한 full-size LUT는 LUT가 없을 때 대비runtime 이득이 있을 수 있으나 full-size LUT를 달성하기 위해 필요한 worker의 저장 용량이 M, K의 값이 큰 경우에는 매우 크고, 이에 따라 전체 MK개의 행렬 곱 중 일부만을 LUT에 저장하는 partial LUT의 사용을 고려할 수 있다.

3.2 Partial LUT 디자인

Partial LUT를 디자인하기 위해 크게 두 가지의 접근법을 생각할 수 있다. 첫 번째는 MK개의 행렬 곱 중 일부인 k개 행렬 곱의 전체 데이터를 LUT에 저장하고 나머지 (MK - k)개의 행렬 곱은 저장하지 않는 것이고, 두 번째는 각 행렬 곱에 대해 일정 비율 α ($0 \le \alpha \le 1$)만을 LUT에 저장하되 누락되는 행렬 곱 없이 MK개의 행렬 곱 전체를 LUT에 저장하는 방법이다. 본 논문에서는 두 번째 방법만을 고려하며, 첫 번째 방법은 후속 연구에서 다룬다.

두 번째 방법인 전체 MK개의 행렬 곱을 일정 부분 α만큼만 LUT에 저장하는 방법은 popularity 정보를 요구하지 않으므로 별도의 정보 처리가 불필요하다. 이 때, master가 원하는 행렬 곱 $\mathrm{A}_{d_1}\mathrm{B}_{d_2}$에 대해 LUT에 저장된 α 부분은 PIR scheme으로 다운로드 받고, 나머지 1-α 부분은 FPCMM scheme으로 계산하게 된다.

구체적으로, A_m이 $\mathrm{s} \times \mathrm{t}$ 행렬, B_k 가 $\mathrm{t} \times \mathrm{r}$ 행렬일 때, $\mathrm{A}_m\mathrm{B}_k$ 는 $\mathrm{s} \times \mathrm{r}$ 행렬이고 이를 계산하기 위해 srt번의 스칼라 곱이 필요하다. 이 때, α는 a/s (a ∈[s])의 형태를 가지며 각 worker W_i는 각 행렬 A_m을 2개의 부행렬 $\mathrm{A}_{m,1}$, $\mathrm{A}_{m,2}$로 나누되 $\mathrm{A}_{m,1}$은 $\mathrm{a} \times \mathrm{t}$ 행렬, $\mathrm{A}_{m,2}$는 (s - a) × t행렬로 구성한 후, 행렬 곱 $\mathrm{A}_{m,1}\mathrm{B}_k$만 LUT $\mathbf{T}_i$에 저장하고 데이터베이스 **A**에는 $\mathrm{A}_{m,2}$만 저장한다. 이에 따라 **A**에 저장된 M개의 파일 $\mathrm{A}_{[M],2}$들은 모두 (s - a)

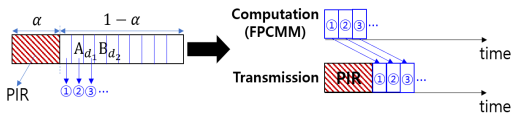

Fig. 2. Pipelining for partial LUT

× t 행렬이 된다. 이후 각 W_i가 $\mathbf{T}_i$에 저장된 각 행렬 곱 $A_{m,1}B_k$ $(m \in [M], k \in [K])$을 식별하기 위한 MK개의 식별자를 master에 알려주면, master는 원하는 행렬 곱 $A_{d_1,1}B_{d_2}$에 해당하는 식별자를 PIR 방식으로 은폐하면서 별도의 계산과정없이 바로 다운로드한다. 그리고 나머지 행렬 곱 $A_{d_1,1}B_{d_2}$는 FPCMM 방식으로 계산한다.

이 때 지연을 고려하여 α 부분에 대한 다운로드가 일어날 때 동시에 1- α 부분에 대한 연산을 수행하여 PIR phase가 끝남과 동시에 1 - α 부분에 대한 계산 결과를 반환할 수 있도록 파이프라이닝(pipelining) 할 수 있으며, 이는 그림 2와 같이 나타낼 수 있다.

3.3 평균 runtime 분석

Full-size LUT와 partial LUT, LUT가 없을 때의 평균 runtime을 비교해본다. 각 worker에서의 지연(straggling) 발생 조건은 세 경우 모두 동일하므로 분석에서 제외한다. 고려하는 PIR scheme[10]과 FPCMM scheme[11]은 세 경우에 공통으로 적용된다. PIR과 FPCMM은 모두 data access privacy를 위해 다운로드 양이나 계산량에 있어 redundancy를 필요로 하고 이는 곧 overhead로 작용한다. 이러한 overhead를 $\beta_1, \beta_2, \beta_3$라는 각각이 1보다 큰 multiplicative factor로 표현하며, 이들은 각각 PIR의 다운로드 양 overhead, FPCMM의 계산량 overhead, FPCMM의 다운로드 양 overhead를 의미한다. 그리고 Partial LUT와 LUT가 없을 때 FPCMM에서의 계산-반환 사이의 파이프라이닝이 최적이라고 가정한다. 이에 따라 세 경우의 전송 속도가 동일하다면 runtime은 전적으로 worker가 master에게 반환하는 communication load에 의존하게 된다. 분석 결과 runtime의 비율(full-size LUT, partial LUT, no LUT 순서)은 다음과 같다.

$$\beta_1 : [\alpha\beta_1 + (1-\alpha)\beta_3] : \beta_3$$

이 때, FPCMM[11]에서 행렬 곱 A_mB_k를 계산하기 위해 A_m를 p개의 부행렬로, B_k를 q개의 부행렬로 분할했다고 가정하면 $\beta_3 = 1 + p^{-1} + q^{-1} + p^{-1}q^{-1}$이 성립하고, $\beta_1 < 1 + (N-1)^{-1} \leq 1 + (pq + p + q)^{-1}$이 성립한다. 즉, β_1의 상한이 β_3보다 작으므로 항상 $\beta_1 < \beta_3$가 성립함을 알 수 있고 이에 따라 LUT가 없을 때 대비 LUT 사용의 이득을 runtime 측면에서 확인할 수 있다. 한편, partial LUT에서 α는 다음 식과 같이 주어진다:

$$\alpha = |\mathbf{T}_i|/MKsr$$

따라서 각 파일 데이터베이스의 행렬의 개수 M, K가 커지거나 각 행렬 A_mB_k의 크기 s × r이 커질수록, 혹은 worker W_i의 저장용량 $|\mathbf{T}_i|$가 작아질수록 α가 감소하여 partial LUT의 runtime 이득이 작아지는 것을 확인할 수 있다.

3.4 Computational complexity 분석

본 논문에서 computational complexity는 master가 원하는 행렬 곱 $A_{d_1}B_{d_2}$을 얻기 위해 각 worker W_i가 수행해야 하는 행렬 곱 연산의 계산 복잡도로 정의하며, LUT를 구성하기 위한 연산은 사전에 계산된다고 가정하여 computational complexity 에서 제외한다.

기존의 FPCMM scheme[11]에서 computational complexity는 $\mathcal{O}(\mathrm{srt/pq})$로 주어지므로 LUT가 없을 때의 computational complexity는 $\mathcal{O}(\mathrm{srt/pq})$가 된다. 반면 partial LUT에서는 $\alpha = a/s, (a \in [s])$에 대해 각 행렬 A_m을 2개의 부행렬 $A_{m,1}$ $A_{m,2}$로 분할하여 $A_{d_1,2}B_{d_2}$만 FPCMM 방식으로 계산하고 $A_{d_1,1}B_{d_2}$은 미리 계산되어 바로 다운로드 하므로 computational complexity가 $\mathcal{O}((1-\alpha)\mathrm{srt/pq})$로 더 작아지고, full-size LUT에서는 $\alpha = 1$이 되어 computational complexity가 0이 된다.

Ⅳ. 결 론

본 논문에서는 FPCMM에 LUT를 도입하여 계산 효율성을 높일 수 있는 방안을 제시하였다. 구체적으로, LUT의 유형을 full-size LUT와 partial LUT로 구분하여 각각의 특성을 알아보았고 partial LUT의 디자인을 popularity를 고려하는 측면과 고려하지 않는 측면에서 접근하여 각각의 방법의 특성에 따른 LUT 디자인을 제안하였다.

References

[1] M. Kim, H. Yang, and J. Lee, “Private coded matrix multiplication,” *IEEE Trans. Inf.*

Forensics Secur., vol. 15, pp. 1434-1443, 2020.

[2] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory,* vol. 64, no. 3, pp. 1514-1529, Mar. 20. (https://doi.org/10.1109/TIT.2017.2736066)

[3] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965-981, 1998. (https://doi.org/10.1145/293347.293350)

[4] M. Aliasgari, O. Simeone, and J. Kliewer, "Private and secure distributed matrix multiplication with flexible communication load," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 2722-2734, 2020. (https://doi.org/10.1109/TIFS.2020.2972166)

[5] M. Kim and J. Lee, "Private secure coded computation," *IEEE Commun. Lett.*, vol. 23, no. 11, pp. 1918-1921, Nov. 2019. (https://doi.org/10.1109/LCOMM.2019.2934436)

[6] Z. Jia and S. A. Jafar, "X-secure T-private information retrieval from MDS coded storage with byzantine and unresponsive servers," *IEEE Trans. Inf. Theory*, vol. 66, no. 12, pp. 7427-7438, Dec. 2020. (https://doi.org/10.1109/TIT.2020.3013152)

[7] J. Zhu and S. Li, "A systematic approach towards efficient private matrix multiplication," *IEEE J. Sel. Areas in Inf. Theory*, vol. 3, no. 2, pp. 257-274, Jun. 2022. (https://doi.org/10.1109/JSAIT.2022.3181144)

[8] H. Yang, S. Hong, and J. Lee, "Coded distributed computing over wireless networks for preserving information-theoretic data privacy," *KICS Fall Conf. 2021*, pp. 285-286, Nov. 2021.

[9] Q. Yu and A. S. Avestimehr, "Entangled polynomial codes for secure, private, and batch distributed matrix multiplication: Breaking the 'cubic' barrier," *IEEE Int. Symp. Inf. Theory (ISIT)*, pp. 245-250, Jun. 2020. (https://doi.org/10.1109/ISIT44484.2020.9174167)

[10] H. Sun and S. A. Jafar, "The capacity of private information retrieval," *IEEE Trans. Inf. Thoery*, vol. 63, no. 7, pp. 4075-4088, 2017.

[11] M. Kim, H. Yang, and J. Lee, "Fully private coded matrix multiplication from colluding workers," *IEEE Commun. Lett.*, vol. 25, no. 3, pp. 730-733, Mar. 2021.