Near-Data Acceleration of Privacy-Preserving Biomarker Search with 3D-Stacked Memory

Alvin Oliver Glova, Itir Akgun, Shuangchen Li, Xing Hu, Yuan Xie
University of California, Santa Barbara, CA, 93106 USA
{aomglova, iakgun, shuangchenli, huxing, yuanxie}@ece.ucsb.edu

***Abstract*—Homomorphic encryption is a promising technology for enabling various privacy-preserving applications such as secure biomarker search. However, current implementations are not practical due to large performance overheads. A homomorphic encryption scheme has recently been proposed that allows bitwise comparison without the computationally-intensive multiplication and bootstrapping operations. Even so, this scheme still suffers from memory-bound performance bottleneck due to large ciphertext expansion. In this work, we propose *HEGA*, a near-data processing architecture that leverages this scheme with 3D-stacked memory to accelerate privacy-preserving biomarker search. We observe that homomorphic encryption-based search, like other emerging applications, can greatly benefit from the large throughput, capacity, and energy savings of 3D-stacked memory-based near-data processing architectures. Our near-data acceleration solution can speed up biomarker search by 6.3× with 5.7× energy savings compared to an 8-core Intel Xeon processor.**

I. Introduction

Technological advances in genomic data sequencing has fueled the increased availability of genetic data information and rise of genetic analysis services. Although the abundance of digitized personal genomic information enables advances in bioinformatics and medical domains, it also brings security and privacy concerns. For the analysis of genetic data, there is a growing drive to use privacy-preserving computation techniques to process sensitive genetic information securely [1]–[4].

One of the emerging bioinformatics applications is biomarker search. Biomarker search applications are used in medical centers to check for genetic diseases and involve searching a biomarker within a reference database. A match within a reference database indicates high probability of having a certain disease. This type of application was recently explored in the recent iDASH (Integrating Data for Analysis, Anonymization and Sharing) National Center secure genome analysis workshop challenge [5]–[7].

Homomorphic encryption (HE) supports operations on encrypted data thus making it possible for data to remain confidential while it is processed in untrusted environments [8], [9]. This property allows for the protection of private data especially in cloud services. HE can be used for secure outsourcing of biomarker search as shown in Figure 1. An encrypted biomarker query is sent to a server where it is matched with a database that is also encrypted. The encrypted result (match or no match) is sent back to user where it is decrypted. None of the query, the database entries, and the result is revealed to the server during the entire processing.

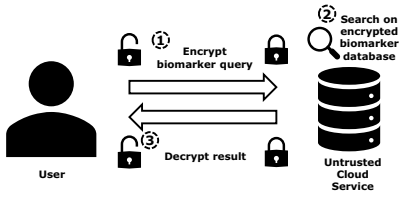

Fig. 1. Privacy-preserving Biomarker Search Overview

However, homomorphic encryption has large computational and storage overheads due to large ciphertext explosion [9], thus limiting its practical usage.

State-of-the-art HE-based privacy-preserving search solutions typically require computationally-expensive homomorphic multiplication and bootstrapping operations [10]. For example, homomorphic multiplication typically runs 10-100× slower compared to homomorphic addition, depending on parameters [11]. *Bian et al.* recently proposed an additive homomorphic encryption scheme for exact match search that removes the computational overhead of multiplication and bootstrapping operations [12].

However, a realistic implementation of this scheme still suffers from the large ciphertext explosion of HE which results in heavy data movement during search operation of moving data from memory to the processing unit to perform the comparison. For example, ciphertext expansion for a 32-bit integer for medium security results in 44,000× explosion in size [12]. Therefore, the performance of this scheme is still limited and also not scalable, especially for growing data sizes that require stronger encryption schemes and even larger resulting ciphertexts.

In this paper, we build on top of this additive HE-based search scheme to propose *HEGA*, a near-data processing (NDP) architecture to accelerate privacy-preserving biomarker search. We adopt a 3D-stacked DRAM to reduce data movement and accelerate basic additive homomorphic operation for this application. [1]

Our contributions in this work are the following:

- We analyze the performance bottleneck of a practical implementation of this homomorphic encryption search scheme
- We propose a 3D-stacked memory-based near-data processing architecture to accelerate search operation based on this homomorphic encryption search scheme

[1]Note that although the specific application we explored here is biomarker search, this architecture can also be used in other privacy-preserving exact search applications.

978-3-9819263-2-3/DATE19/©2019 EDAA

- Using this architecture, we propose the first hardware accelerator for privacy-preserving biomarker search and compare to CPU-based implementation

II. Background

A. Privacy-Preserving Biomarker Search

Biomarker search is one of the key emerging applications in bioinformatics domain [2], as it allows for detection of possible diseases. A specific set of biomarkers are queried from a server that houses a database of these biomarkers. The presence or absence of a specific biomarker or a set of biomarkers indicates a probability of genetic diseases and thus helps medical practitioners to make informed decisions. In dealing with this type of application, however, data is stored in the database and the queries must be encrypted in order to protect privacy.

The biomarkers are stored in Variant Call Format (VCF). These VCF files contain information on biomarkers (genotype information) such as chromosome number and the position of the genome. Furthermore, it contains information for each position such as reference and alternate sequences.

A typical processing flow for HE-based biomarker search is shown in Figure 2. The figure shows two general phases: a preprocessing phase and the query phase. In the preprocessing phase, each entry in the VCF file is first encoded and hashed before performing the actual homomorphic encryption using a generated key. This is to reduce the size of the encrypted entries since the size of the unencrypted entries will affect the size of the data after encryption. In the query phase, the client similarly needs to preprocess the query before it is sent to the cloud service for the exact search operation. An encrypted result of the search is sent back to client where it can be decrypted using the secret key. In this work, we focus on the homomorphic evaluation stage of the search which takes up the majority of the execution time, especially for large number of queries. For this work, we assume a size of 32 bits for the post-hashed unencrypted database entries and queries. Note that this size is realizable as demonstrated by *Cetin et al.* using a cuckoo-based hashing scheme that enables size reduction of the entries to 29 bits [5].

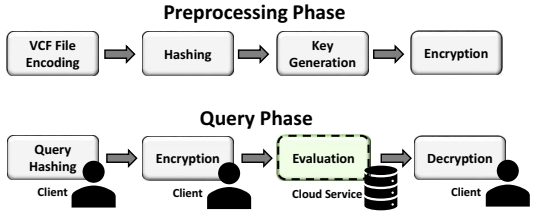

Fig. 2. HE-based Privacy-Preserving Biomarker Search Flow

B. Additive Homomorphic Encryption Scheme for Search

Cryptographic solutions such as homomorphic encryption allow for computations on encrypted data. This makes homomorphic encryption a very promising solution for privacy-preserving applications. Fully homomorphic encryption has received wide attention as it allows computations on arbitrarily deep circuits using an operation called bootstrapping [13]. Bootstrapping is a method to refresh a ciphertext by decrypting and re-encrypting to reduce noise, which is a result of performing many HE computations on encrypted data. However, bootstrapping is a computationally expensive operation and thus most recent work on homomorphic encryption also focus on partial (eg. additive) homomorphic encryption schemes.

Although there have been many studies which contributed to the rapid progress of HE, performance bottlenecks continue to hinder its practical realization. Two of the biggest contributors are *data size explosion* and *slow primitive operations*. Encryption results in ciphertext explosion which translates to computation, storage, and communication overheads. Primitive operations such as polynomial multiplication have slow execution times (often millisecond range) and often requires complex specialized hardware [14]. For privacy-preserving search, these problems become even more prominent since aside from the large computational and storage requirements initially demanded by HE. Furthermore, larger HE parameters are needed to support more entries while maintaining the same security level, which exacerbates the data size explosion problem even more.

Ducas et al. proposed the FHEW scheme that can perform NAND operation with only additive homomorphism which greatly reduces the computational requirements [15]. However, it still needs bootstrapping after each homomorphic gate operation which dominates the runtime. More recently, *Bian et al.* [12] proposed SCAM by modifying the plaintext space of FHEW and introducing an encryption constant to implement a two-stage complex homomorphic Boolean gate which can be used for multi-bit word matching. It is also based on additive homomorphism but does not require bootstrapping or multiplication operations which makes it efficient for use in hardware implementations. Equation 1 shows a bitwise exact search operation using XNOR-AND gates. SCAM scheme achieves exact search in homomorphic encryption domain using only additive homomorphism in homomorphic XOR-OR gate as shown in Equation 2, where c_{x_i} and c_{y_i} are ciphertexts for each bit [12]. In this scheme, each 1-bit plaintext expands to a $(n+1)$ (lg q)-bit ciphertext where q and n are encryption parameters that determined according to the security level. To perform a homomorphic w-bit word matching, $w \cdot (n+1)$ (lg q)-bit integers are added and if the final result decrypts to zero, it means the two words being compared are the same. A non-zero result means the two words do not provide a match.

$$f(x,y) = \prod_{i=1}^{w} \overline{x_i \oplus y_i} \tag{1}$$

$$\widehat{\text{Hom}\overline{\text{XOR}}}\text{-OR}(x,y) = \sum_{i=0}^{w} (c_{x_i} - c_{y_i}) \tag{2}$$

Implementing SCAM for the privacy-preserving search within a database requires performing this search operation in the homomorphic domain through all encrypted database entries and returning the encrypted results of match or no

match, with respect to encrypted query bitstream. The client can later decrypt the matching results that evaluate to zero for match and non-zero for no match in the database search. We also adopt the secure two-round communication protocol of SCAM in this work.

This scheme was proposed with an ASIC design [12] in which all encrypted database entries are stored on-chip to provide large bandwidth. However, due to the data explosion of more than 44k× larger data size after encryption, such design results in unacceptable chip area, making it impractical and also not scalable. For example, even for a database of 100K 32-bit entries using their provided encryption paramaters, their ASIC design would already need more than 21 billion transistors, even without including the large on-chip memory (SRAM) required.

We discuss 3D-stacked memories next and in Section III, we discuss why a 3D-stacked memory NDP-based solution is suited for use with this HE scheme and privacy-preserving biomarker search.

C. 3D-Stacked Memories

Hybrid Memory Cube (HMC) is a type of 3D-stacked memory technology which can be used for near-data accelerator architectures. HMC consists of 4/8 DRAM dies on top of a logic base die, resulting 4/8 GB capacity per device [16]. Each DRAM die is divided into 32 partitions, with each partition consisting of multiple banks. Partitions across dies vertically form a vault. Each vault has an independent vault memory controller within the logic die that manages all memory operations for that vault. The logic base die also includes a crossbar switch that connects the vault memory controllers to the I/O ports. HMC uses SerDes I/O links of up to a total of 320 GB/s peak bandwidth. HMC can also be chained together to increase total memory capacity, which can provide a scalable expansion for applications such as privacy-preserving biomarker search which has large memory requirement.

III. Motivation

In this section, we discuss our motivation for using a near-data processing approach to accelerate the additive homomorphic encryption scheme and its application in privacy-preserving biomarker search. Performing search using the SCAM scheme is very challenging even though the computing is transformed from complex multiplication into a series of simpler homomorphic additions on the encrypted data, as described in Section II-B. For example, by searching a 10k-entry database that is encrypted with SCAM using encryption parameters in [12], we end up with a slowdown of 60k× on CPU compared to unencrypted operation, which becomes worse for larger databases (75k× for a 20k-entry database), shown in Figure 3.

Next, we observe that the application is memory-bound and the challenge lies in performing operations on large data sizes after encryption. Using the same parameters, encrypting the data grows 44k× larger for medium security (80-bit) and 55k× for high security (128-bit) [12]. As a result, a database with 100k entries becomes 16.5GB, which cannot fit on an on-chip cache. At the same time, computation is composed of simple addition operations, making it a memory bandwidth-bound application on all of the available hardware platforms. Using an x86 simulator, we obtain the cycle stack of this application as shown in Figure 4. It shows that 72% of the cycles are DRAM-bound stall cycles, which mainly causes the large slowdown of the application.

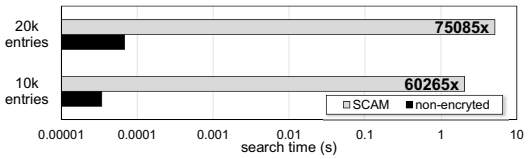

Fig. 3. Slowdown of database search from homomorphic encryption (SCAM)

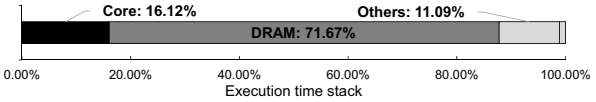

Fig. 4. Cycle breakdown of SCAM running on CPU+HMC

To further understand this application, we build a roofline model. A roofline model is widely used for high performance computing [17]. The y-axis is the performance (in INT32 ADD), thus the peak computation rate forms the flat part of the roofline. The x-axis is the operational intensity, also called operation/byte ratio, which is a measure of operations per DRAM byte accessed. Applications with higher operational intensity would more likely to be compute-bound, i.e., fall to the flat part of the roofline. Applications with lower operational intensity is likely to be memory-bound (the slanted part of the roofline) and cannot achieve the peak performance of the hardware. We model a SCAM-based database query application where we assume the query data (173KB) is stored on-chip while the database (16.5GB) is off-chip. For each operation (INT32 ADD), we need to fetch 4 bytes of data from off-chip memory, making the operation/byte ratio of this application to be 0.25. We draw the roofline model for various hardware in Figure 5, and project the effective performance (indicated by markers) according to the 0.25 operation/byte ratio.

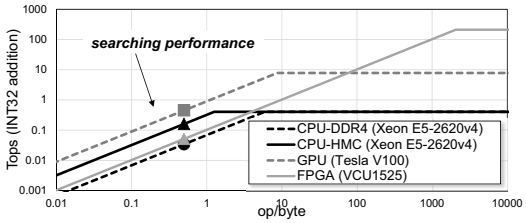

Fig. 5. A roofline model analysis for SCAM on various platforms (FPGA performance estimation with adder in [18])

We observe that this application is memory-bound for all CPU, GPU, and FPGA platforms, since the small operation/byte ratio falls in the slanted part of these rooflines. We conclude that existing hardware solutions are not suitable or

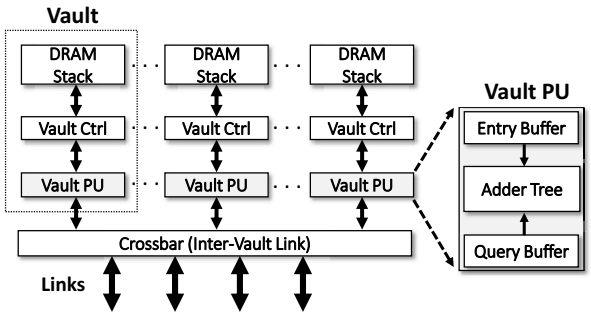

Fig. 6. *HEGA* Architecture Overview

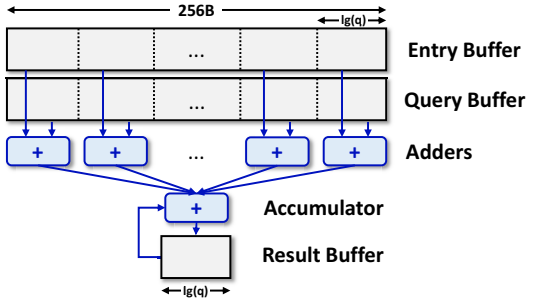

Fig. 7. *HEGA* Vault PU Architecture (for lg q = 42)

efficient for such application. Specifically, if compared to the CPU-DDR4 with FPGA, although the peak performance is improved from 0.4 TOPs to 221 TOPs, the effective performance only improves 1.5× because the memory bandwidth is not significantly improved (68 GB/s vs. 102 GB/s). On the contrary, the effective performance improves 4.7× with the same CPU but changing from DDR4 (68 GB/s) to HMC (320 GB/s). The simple operations required coupled with the associated large data movement overhead makes it an ideal application to accelerate using 3D-stacked memory where a logic die can be used to implement simple operations. Such architecture can provide massive intra-memory bandwidth and hence solve the memory-bound performance bottleneck. Furthermore, since only simple operations are performed on the logic die using this scheme, it is more suited for 3D-stacked memory integration considering its thermal limitations [19] as compared to typical HE schemes that need complex hardware to speedup the computationally-intensive FFT operation needed in large-integer multiplication.

IV. HEGA Architecture

A. HEGA *Overview*

We base the design of our near-memory architecture on Micron's Hybrid Memory Cube [16]. Figure 6 illustrates the high-level architecture of our design. The DRAM layers are composed of multiple independent vertical slices called vaults. Each of the vaults can be accessed in parallel, and thus have independent accelerators and memory controllers associated with them. The accelerators can operate on data residing in their local vault and have direct high-bandwidth access to the DRAM layers via the TSVs. The vault controllers handle requests from accelerators co-located within the vault logic, as well as read and write requests that come from the processor.

B. *Architectural Details*

Vault logic within the logic die consists of vault memory controller along with the vault processing unit (PU). Each vault PU includes the following components for implementing homomorphic addition, as shown in Figure 7. Entry buffer stores the units of a fetched entry from the database. Query buffer stores units of the biomarker query to be searched within the database. Entry and query buffers are 256B in size to match the HMC row buffer size [19]. The adder tree is made of up of adders needed to perform the homomorphic matching as described in Section II-B.

To perform a search operation, a block is first requested to the vault controller and is stored in the entry buffer of the vault PU. Note that the data bus (transfer size) in an HMC vault is 32B and the internal vault bandwidth is defined as 32B/4tCK/vault (10GB/s for tCK = 0.8ns). Once the entry buffers are loaded, the arithmetic units are used to perform (lg q)-bit additions with the partial query data stored in the query buffer. These results are then accumulated and stored in a result buffer. This process continues until all the entry blocks have been processed. The query ciphertext is sent to all vaults to improve efficiency by parallel search. Finally, the search result of size (lg q) bits per entry saved in a result buffer is sent to the user.

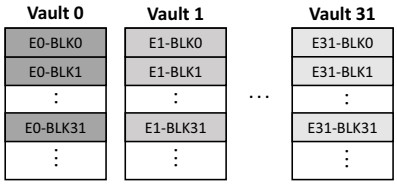

Fig. 8. HEGA Data Mapping

Next, we discuss mapping of database entries to the 3D-stacked DRAM. To map database entries to the HMC, we use the mapping shown in Figure 8. This mapping scheme leverages the vault-level parallelism of the HMC. Each encrypted entry bit composed of $(n+1)$ (lg q)-bit integers is stored in a vault for the w vaults ($w = 32$). The vault PUs perform the corresponding additions for the entry and query units. Finally, the results from all the vault PUs are accumulated as shown in Figure 9. For each entry, $w \cdot (n+1)$ additions of lg(q)-bit entry and query data are computed.

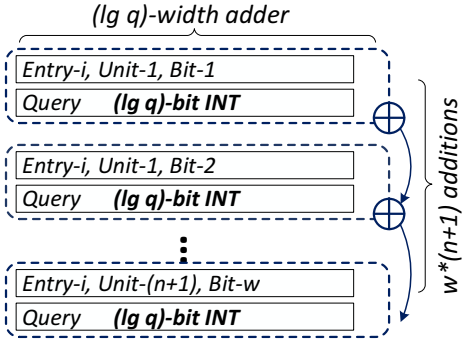

Fig. 9. SCAM Search Operation

Following the data mapping decribed above, Figure 10 shows a sample address mapping from logical binary array address to the HMC physical address using the HE parameters defined in [12].

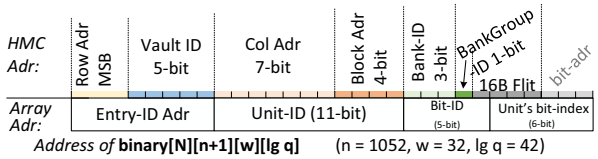

Fig. 10. HEGA HMC Address Mapping

V. Evaluation

A. Methodology

We use Sniper x86 simulator with custom HMC memory model for our baseline CPU+HMC performance evaluation. The power estimates of the x86 cores were obtained from McPAT integrated in Sniper. We used an in-house simulator to perform *HEGA* performance evaluation. The logic components of our design were synthesized with Design Compiler using NanGate 15nm library. To estimate DRAM energy, we assume a DRAM read energy of 3.76 pJ/bit and a logic layer transfer energy of 6.78 pJ/bit from [20], [21]. Table I lists the simulation parameters used.

We analyze the following schemes in our experiments:

- **SCAM**: This baseline scheme performs the SCAM scheme on CPU + HMC
- ***HEGA***: Our proposed near-data acceleration architecture which performs SCAM scheme within the logic die of the HMC

Note that to ensure fair comparison, we evaluate SCAM on a CPU + HMC platform and compare to our proposed NDP + HMC platform. We use 32-bit post-hashed unencrypted database entries and queries ($w = 32$) and use parameters $n = 1052$ and (lg q) $= 42$ as in the instantiation in Table III of [12]. We use a workload consisting of single query on a database of 1k to 16k entries. Note that this small sample range has the advantage of being able to accurately represent the performance of much larger datasets because of the regular workload and at the same time having a feasible simulation speed. Furthermore, this number of entries is big enough to ensure that the size of the encrypted dataset cannot fit into the cache.

TABLE I. Simulation Parameters

Processor	x86, 8-issue width , out-of-order, 64-entry instruction queue, 2.1GHz, 22nm, 8 cores
Cache	L1D/L1I: 32KB, L2: 256KB, shared L3: 20MB, LRU
HMC	4 links, full-lane, 8GB, 32 Vaults, tCK = 0.8ns, tRCD-tCL-tRP = 17-17-17, tCCDS=4, tCCDL=6
HEGA-Logic (NDP)	32 Vault PUs, 1GHz, 15nm node

B. Experimental Results

1) Performance Comparison: Figure 11 shows *HEGA* can already provide up to 6× speedup compared to an 8-core

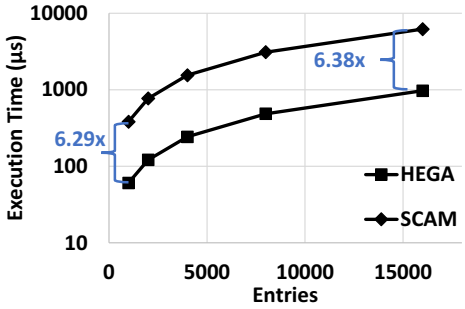

Fig. 11. Execution Time of SCAM and *HEGA*

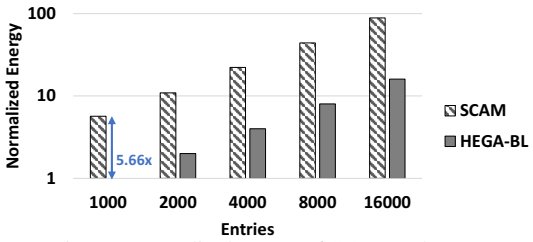

Fig. 12. Normalized Energy of SCAM and *HEGA*

Intel Xeon CPU. This shows the limitation of the CPU in utilizing the large bandwidth available in HMC because of its complex cache hierarchy while in *HEGA*, the NDP units can more efficiently use the internal vault bandwidth. Furthermore, even for the small database sizes explored, we observe that CPU performance becomes worse as the size of the database increases, consistent with our observation from Section III.

Furthermore, *HEGA* performs a single word search in $0.61\mu s$ at 1 GHz. For multi-word comparison, *HEGA* leverages vault-level parallelism and pipelining. Although SCAM leverages the parallel structure for fast multi-word search, the ASIC implementation is not realizable for realistic database sizes, as discussed in Section II-B.

2) Energy Comparison: The normalized energy results are shown in Figure 12. Compared to the CPU-based scheme, *HEGA* can reduce the energy by as much as 5.6×. Lower energy for *HEGA* is achieved due to the proximity of data and computation of NDP compared to the CPU, which further allows excluding energy contributions of power-hungry HMC links and crossbar.

3) Area Overhead: We obtain the area overhead of *HEGA* in the logic die from synthesis results. Since the vault PUs only include a few simple components such as the buffers and an arithmetic unit, the total area across 32 vaults was calculated to be 0.29 mm^2 (15nm node), which represents just 0.4% area of the HMC logic die [21]. Figure 13 shows the area breakdown of main vault PU components implemented in the logic die, namely 256B query and entry buffers, 42-bit adders and accumulator. This evaluation shows that buffers result majority (57%) of area overhead.

VI. Related Work

A. Accelerators on 3D-Stacked Memory

Multiple work have proposed near-data architectures using 3D-stacked DRAM to accelerate data intensive operations

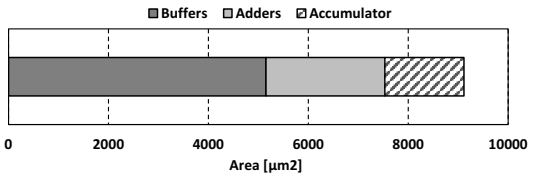

Fig. 13. Area Breakdown of HEGA

[22], [23]. *Alves et al.* proposed HIVE [24], an HMC-based architecture which allows performing common vector operations directly inside the HMC. *Kim et al.* proposed GRIM-Filter [25], a near-data processing architecture within the logic layer of a 3D-stacked memory to accelerate read mapping phase of DNA sequencing application.

Even though these work also propose accelerator architectures on 3D-stacked memories, none of these have focused on accelerating homomorphic encryption and its applications such as privacy-preserving biomarker search.

B. Hardware Acceleration of Privacy-Preserving Search

Few works have presented hardware acceleration schemes for homomorphic encryption-based privacy-preserving search. *Bian et al.* proposed SCAM and an ASIC implementation [12] but has large overheads. *Khedr et al.* introduce a GPU-based approach to homomorphic word searching in their work SHIELD [10]. *Martins et al.* accelerate homomorphic word searching using Intel Xeon Phi [26]. These two implementations still require computationally-expensive homomorphic multiplication. CAMSure [27] allows secure approximate search but biomarker search requires exact search.

Different from prior work on hardware-based secure search, *HEGA* leverages NDP in 3D-stacked memory to handle the large data explosion and the massive data movement due to the streaming search operation. Furthermore, *HEGA*'s use of HMC allows for a scalable solution considering the increasing data expansion rates required for larger databases while maintaining security.

VII. Conclusion

In this work, we propose *HEGA*, a near-data processing architecture that uses 3D-stacked DRAM to accelerate homomorphic encryption-based biomarker search. We observe that emerging applications like homomorphic encryption-based privacy-preserving search can greatly benefit from the throughput, capacity, and energy savings of 3D-stacked DRAM-based NDP architectures. Our NDP-based solution can speed up search by 6.3× with 5.7× energy savings compared to an 8-core Intel Xeon processor. This work represents a step towards achieving practical homomorphic encryption applications through near-data processing.

Acknowledgment

The authors would like to thank Tim Sherwood for insightful discussions. This paper was supported in part by NSF 1730309, 1500848, 1719160, CRISP, one of six centers in JUMP, a SRC program sponsored by DARPA, and NSF grant CCF 1740352 and SRC nCORE NC-2766-A.

References

[1] J.-P. Hubaux *et al.*, "Genomic Data Privacy and Security: Where We Stand and Where We Are Heading," *IEEE Security & Privacy*, vol. 15, no. 5, pp. 10–12, 2017.
[2] M. M. A. Aziz *et al.*, "Privacy-preserving techniques of genomic dataa survey," *Briefings in Bioinformatics*, no. April, pp. 1–9, 2017.
[3] K. Lauter *et al.*, "Private Computation on Encrypted Genomic Data," in *Progress in Cryptology - LATINCRYPT 2014*, vol. 8895. Springer International Publishing, 2015, pp. 3–27.
[4] A. Khedr *et al.*, "SecureMed: Secure Medical Computation using GPU-Accelerated Homomorphic Encryption Scheme," *IEEE Journal of Biomedical and Health Informatics*, pp. 1–1, 2017.
[5] G. S. Çetin *et al.*, "Private queries on encrypted genomic data," *BMC Medical Genomics*, vol. 10, no. S2, p. 45, jul 2017.
[6] M. Kim *et al.*, "Secure searching of biomarkers through hybrid homomorphic encryption scheme," *BMC Medical Genomics*, vol. 10, no. Suppl 2, 2017.
[7] J. S. Sousa *et al.*, "Efficient and secure outsourcing of genomic data storage," *BMC Medical Genomics*, vol. 10, no. S2, p. 46, jul 2017.
[8] C. Gentry, "Computing arbitrary functions of encrypted data," *Communications of the ACM*, vol. 53, no. 3, p. 97, 2010.
[9] P. Martins *et al.*, "A Survey on Fully Homomorphic Encryption," *ACM Computing Surveys*, vol. 50, no. 6, pp. 1–33, Dec 2017.
[10] A. Khedr *et al.*, "SHIELD: Scalable Homomorphic Implementation of Encrypted Data-Classifiers," *IEEE Transactions on Computers*, vol. 65, no. 9, pp. 2848–2858, 2016.
[11] S. Angel *et al.*, "Pir with compressed queries and amortized query processing," in *IEEE SP*, May 2018, pp. 962–979.
[12] S. Bian *et al.*, "SCAM: Secured content addressable memory based on homomorphic encryption," in *DATE*, 2017, pp. 984–989.
[13] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*, 2009, pp. 169–178.
[14] W. Wang *et al.*, "Vlsi design of a large-number multiplier for fully homomorphic encryption," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 1879–1887, Sept 2014.
[15] L. Ducas *et al.*, "FHEW: Bootstrapping homomorphic encryption in less than a second," in *Advances in Cryptology – EUROCRYPT 2015*, vol. 9056, 2015, pp. 617–640.
[16] Hybrid Memory Cube Consortium, "Hybrid Memory Cube Specification 2.1," 2014. [Online]. Available: http://hybridmemorycube.org/
[17] S. Williams *et al.*, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, Apr. 2009. [Online]. Available: http://doi.acm.org/10.1145/1498765.1498785
[18] P. Zicari *et al.*, "A fast carry chain adder for virtex-5 fpgas," in *IEEE Mediterranean Electrotechnical Conference*, April 2010, pp. 304–308.
[19] R. Hadidi *et al.*, "Demystifying the characteristics of 3d-stacked memories: A case study for hybrid memory cube," in *IEEE IISWC*, Oct 2017, pp. 66–75.
[20] S. H. Pugsley *et al.*, "Ndc: Analyzing the impact of 3d-stacked memory+logic devices on mapreduce workloads," in *ISPASS*, March 2014, pp. 190–200.
[21] J. Jeddeloh *et al.*, "Hybrid memory cube new dram architecture increases density and performance," in *VLSI*, June 2012, pp. 87–88.
[22] S. F. Yitbarek *et al.*, "Exploring specialized near-memory processing for data intensive operations," *DATE*, pp. 1449–1452, 2016.
[23] M. Drumond *et al.*, "The Mondrian Data Engine," in *ISCA*, 2017, pp. 639–651.
[24] M. A. Z. Alves *et al.*, "Large Vector Extensions Inside the HMC," in *DATE*, 2016, pp. 1249–1254.
[25] J. S. Kim *et al.*, "GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies," *BMC Genomics*, vol. 19, no. S2, p. 89, May 2018.
[26] P. Martins *et al.*, "HPC on the Intel Xeon Phi: Homomorphic Word Searching," in *High Performance Computing for Computational Science – VECPAR 2016*, 2017, pp. 75–88.
[27] M. Sadegh Riazi *et al.*, "CAMsure: Secure Content-Addressable Memory for Approximate Search," *ACM Trans. Embed. Comput. Syst. Article*, vol. 16, no. 20, pp. 1–20, 2017.