Evaluating Homomorphic Operations on a Real-World Processing-In-Memory System

Harshita Gupta* Mayank Kabra*
Juan Gómez-Luna Konstantinos Kanellopoulos
Onur Mutlu

SAFARI ETH Zürich
Homomorphic Encryption
Homomorphic Encryption

User -> Data

Host System

Encrypts data

Encrypted data
Homomorphic Encryption

- User
  - Data
  - Host System: Encrypts data
  - Encrypted data
  - Central/Cloud Server: Performs operations on the encrypted data without decrypting it
Homomorphic Encryption

User → Encrypts data → Host System → Encrypted result → Central/Cloud Server

Decryption process:
- User inputs data
- Host System encrypts the data
- Encrypted data is sent to Central/Cloud Server
- Central/Cloud Server performs operations on the encrypted data without decrypting it
- Encrypted result is returned
- Host System decrypts the result
- Result is returned to the user
Motivation

Homomorphic operations suffer from large memory capacity and data movement bottlenecks

Acceleration Techniques
These approaches face challenges in resource limitations, data movement, and practical implementation.
Our Goal

Evaluate the suitability of real-world general-purpose processing-in-memory (PIM) architectures to perform homomorphic operations.
Our Goal

Evaluate the suitability of real-world general-purpose processing-in-memory (PIM) architectures to perform homomorphic operations.

UPMEM: First Real World PIM Architecture
Our Goal

Evaluate the suitability of real-world general-purpose processing-in-memory (PIM) architectures to perform homomorphic operations.
Evaluation Methodology

Evaluation of homomorphic addition and multiplication on UPMEM PIM system
Evaluation Methodology

1. Evaluation of homomorphic addition and multiplication on UPMEM PIM system

2. Evaluation with statistical workloads (mean, variance, linear regression)
Evaluation Methodology

1. Evaluation of **homomorphic addition and multiplication** on UPMEM PIM system

2. Evaluation with statistical workloads (**mean, variance, linear regression**)

3. Comparison with custom **CPU and GPU libraries** and an optimized **CPU library (SEAL)**
Evaluation: Homomorphic Addition

- **CPU**
- **PIM**
- **CPU-SEAL**
- **GPU**

The graph compares the execution time (in milliseconds) for different numbers of ciphertexts across various devices. The x-axis represents the number of ciphertexts, and the y-axis represents the execution time.

- **20480**
- **40960**
- **81920**
- **163840**
- **327680**
Evaluation: Homomorphic Addition

50 – 100× speedup provided by PIM over CPU
2 – 15× speedup over GPU
Evaluation: Homomorphic Multiplication

The diagram above depicts the execution time (in milliseconds) for different numbers of ciphertexts processed by various devices:

- **CPU**
- **PIM**
- **CPU-SEAL**
- **GPU**

The x-axis represents the number of ciphertexts, ranging from 5120 to 81920. The y-axis represents the execution time, ranging from $10^0$ to $10^8$. The bars indicate the performance of each device at different ciphertext counts.
Evaluation: Homomorphic Multiplication

PIM lags 10 – 15× behind the GPU due to the lack of native multiplication support
Evaluation: Linear Regression

![Bar chart showing execution time (ms) for different platforms (CPU, PIM, CPU-SEAL, GPU) with 32 and 64 Ciphertexts. The chart indicates CPU has the least execution time compared to other platforms, especially with 64 Ciphertexts.]
Evaluation: Linear Regression

- CPU and GPU are faster than PIM.

**Bar Chart:**
- Execution time (ms) for 32 and 64 Ciphertexts.
- CPU and PIM show significant improvements.

- PIM is 6.4–7.5x faster than the custom CPU implementation.

- CPU-SEAL and GPU are faster than PIM.
UPMEM PIM system natively supports 32-bit integer addition and outperforms CPU and GPU for homomorphic addition.
Key Takeaways

1. UPMEM PIM system natively supports 32-bit integer addition and outperforms CPU and GPU for homomorphic addition.

2. The lack of native support for 32-bit integer multiplication hampers the performance of PIM for homomorphic multiplication.
<table>
<thead>
<tr>
<th>Key Takeaways</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>2</strong></td>
</tr>
<tr>
<td><strong>3</strong></td>
</tr>
</tbody>
</table>
Evaluating Homomorphic Operations on a Real-World Processing-In-Memory System

Harshita Gupta*  Mayank Kabra*
Juan Gómez-Luna  Konstantinos Kanellopoulos
Onur Mutlu

SAFARI  ETH zürich