DaPPA: A Data-Parallel Framework for Processing-in-Memory Architectures

Geraldo F. Oliveira (Ph.D. Candidate) | Onur Mutlu (Ph.D. Advisor)

Data Movement Bottleneck

Data movement is a major bottleneck in modern computer architectures.

Over 60% of the total system energy is spent on data movement.

The Programmability Issue

General-purpose PIM architectures (e.g., UPMEM) often employ a custom programming interface.

To ease programmability for the UPMEM architecture, allowing a programmer to write efficient PIM-friendly code without the need to explicitly manage hardware resources.

Processing-in-Memory: Overview & Landscape

Processing-in-Memory: move computation to where the data resides

1. Processing-near-Memory
2. Processing-using-Memory

DaPPA: A Data-Parallel Processing-in-Memory Architecture

DaPPA Overview & Execution Steps

1. data-parallel pattern APIs
2. Data-Parallel Pattern APIs
3. Dynamic-Template-Based Compilation
4. Putting All Together

President

- Pre-defined functions that implement high-level data-parallel pattern primitives
- DaPPA supports five primary data-parallel pattern primitives, including (i) map, (ii) filter, (iii) reduce, (iv) window, (v) group.
- The user can combine all five data-parallel primitives to describe complex data transformations.

Evaluation

- We evaluate:
  - Performance
  - Programming complexity (in lines of code)

- Workloads
  - 6 workloads from the PrIM benchmark suite
  - Vector addition (VA)
  - Select (SEL)
  - Unique (UNI)
  - Reduce (RED)
  - General matrix-vector multiply (GEMV)
  - Histogram small (HST-S)

- Comparing DaPPA to PrIM

Conclusion

Background: Processing-in-Memory (PIM) alleviates the performance and energy bottlenecks caused by data

- The UPMEM system is an example of a real general-purpose PIM architecture

Problem: Programming the UPMEM system requires non-trivial effort from the programmer

- The programmer needs to have knowledge of the hardware and manage data movement manually

Goal: Ease programmability for the UPMEM architecture, allowing one to write PIM code without prior knowledge of the hardware

DaPPA: A Data-Parallel Processing-in-Memory Architecture that

- Provides a data-parallel pattern-based programming interface that abstracts hardware components
- Automatically distributes input and gathers output data, handles memory management, and parallelizes work across PIM cores

Key Results: Our extensive evaluation shows that DaPPA

- Outperforms hand-tuned PIM implementations by 2.1x
- Reduces programming complexity by 94.4%