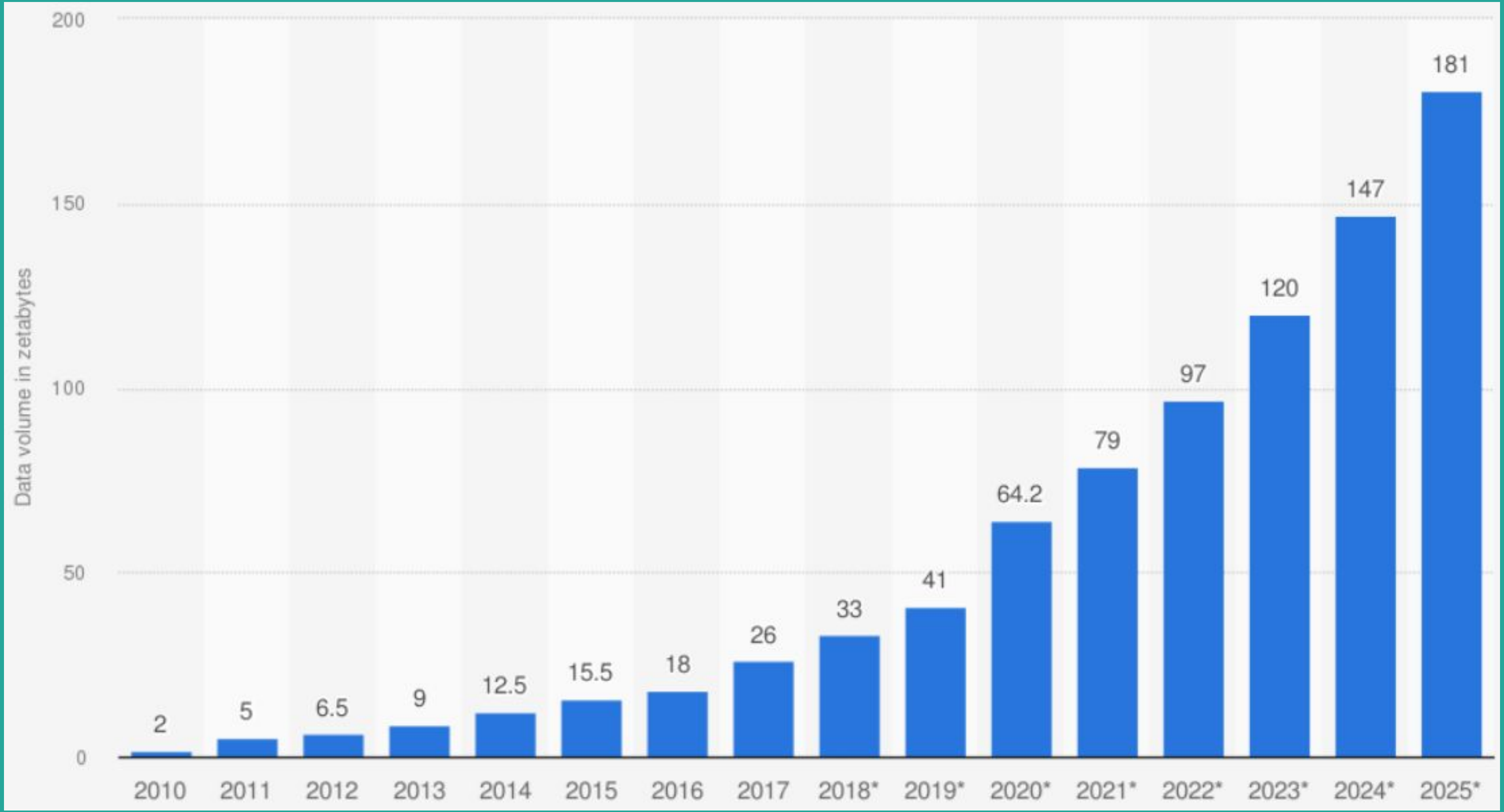
Enabling Multi-threaded Execution and Improved Memory Access in Fine-Grain Near-Data Processing Systems

Sairo Raoní dos Santos
Orientador: Marco Antonio Zanata Alves
Programa de Pós-graduação em Informática - UFPR

Big Data

- We must process all the data!
  - Non-structured;
  - Massive;
  - Fast!

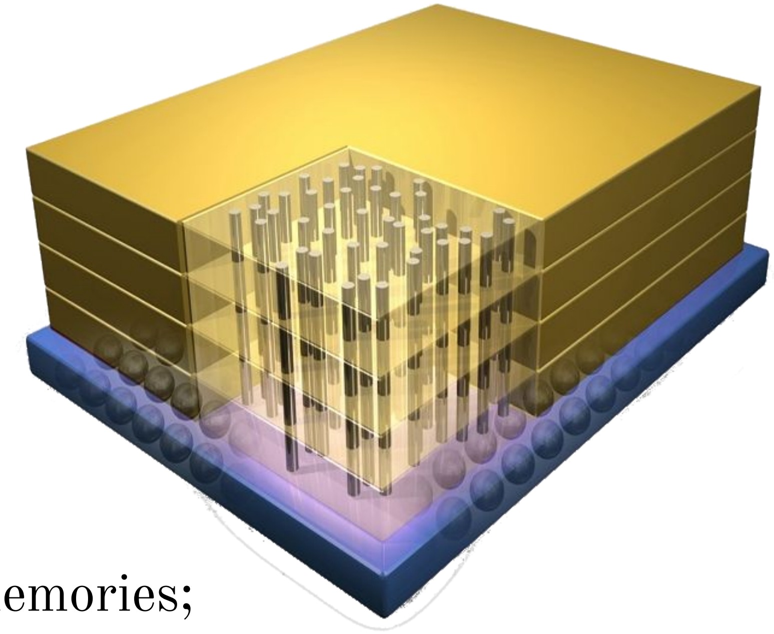


The von Neumann architecture

Can we handle it?

Near-Data Processing

- Proposed in the 1990s;
  - Patterson et al., 1997;
  - Elliott et al., 1999;
  - Impractical!
- Resurgence in the early 2010s;
  - TSV technology → 3D-stacked memories;



Near-Data Processing: Goals

- **Reduce data movement;**
  - Sharply reduces energy consumption;
- **Improve usage of memory bandwidth;**
  - Improve overall performance of data-streaming applications;
- **Improve overall efficiency in resource usage;**
  - Less cache hierarchy pollution;
  - Better resource allocation;

Near-Data Processing: Challenges

- **Programming model;**
  - How to program for NDP?
- **Cache coherence;**
  - Collisions;
- **Virtual memory support;**
  - Virtual address translation;

Near-Data Processing: Challenges

- **Sequential Consistency:**
  - Most assumed model;
  - Assumptions:
    - atomic operations;
    - program order maintenance;
    - any load operation returns the value from last stored to the location;

Near-Data Processing: Challenges

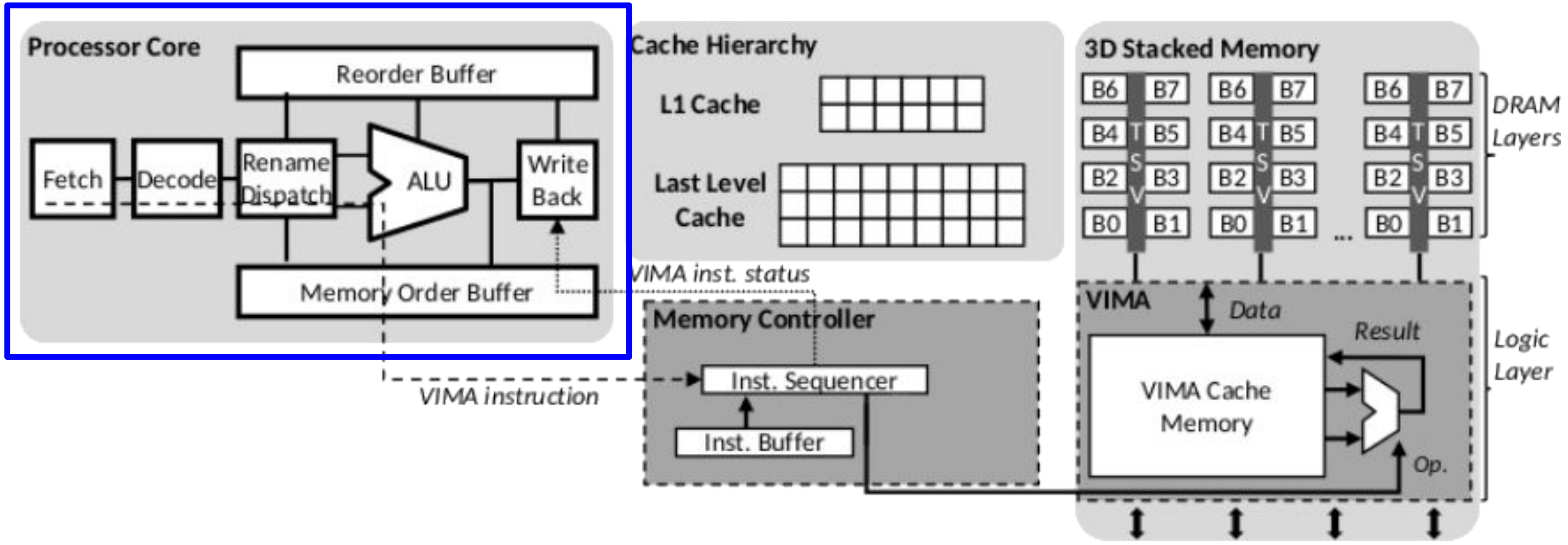
- Sequential Consistency:
  - How do existing NDP proposals deal with this?
    - Fail to discuss it;
    - The programmer must deal;
  - **We propose to provide sequential consistency by implementing precise exceptions near the data.**

It is possible to provide precise exceptions, improved memory access and multi-threading with fine-grain NDP to speedup data streaming applications.

Vector-In-Memory Architecture (VIMA)

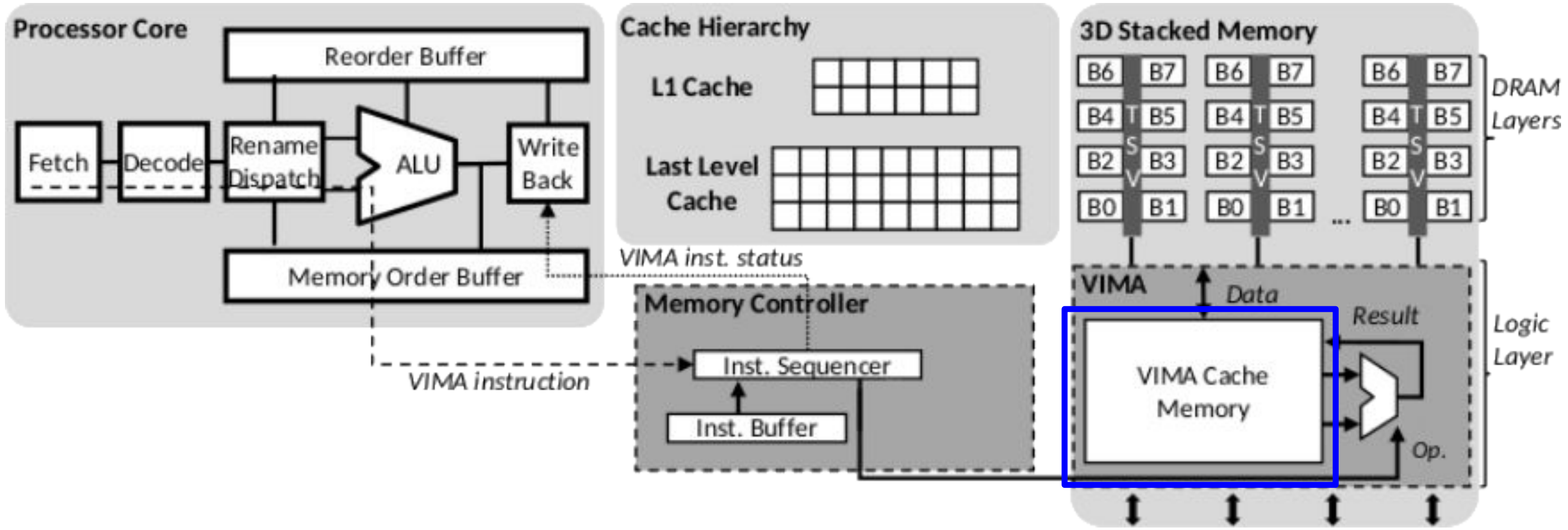
VIMA (Vector-In-Memory Architecture)

- Common fine-grain NDP architecture features:
  - ISA extensions;



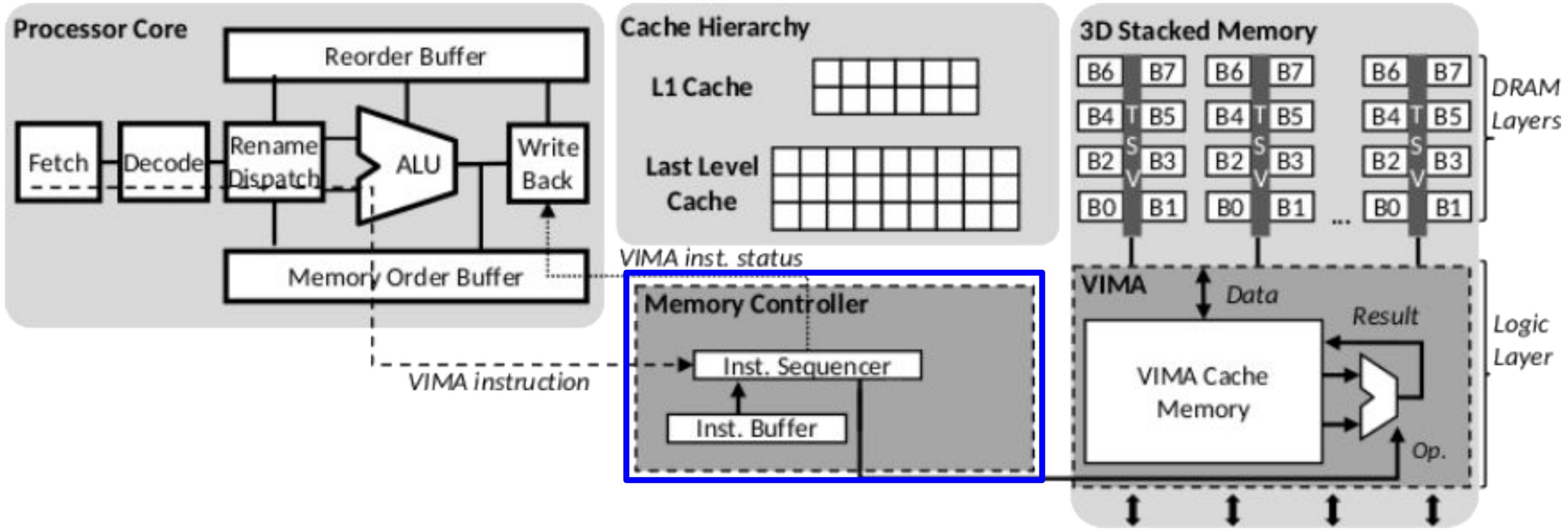
VIMA (Vector-In-Memory Architecture)

- Common fine-grain NDP architecture features:
  - Near-data dedicated storage;



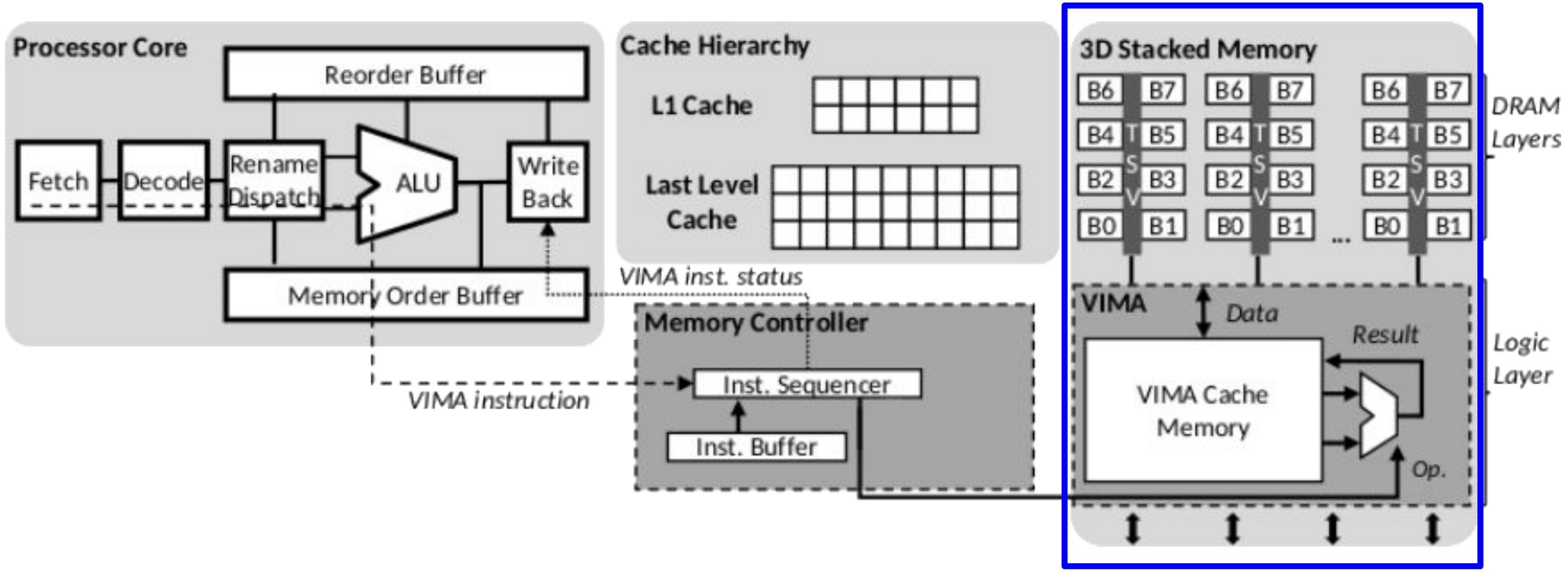
VIMA (Vector-In-Memory Architecture)

- Common fine-grain NDP architecture features:
  - Huge pages in the TLB;



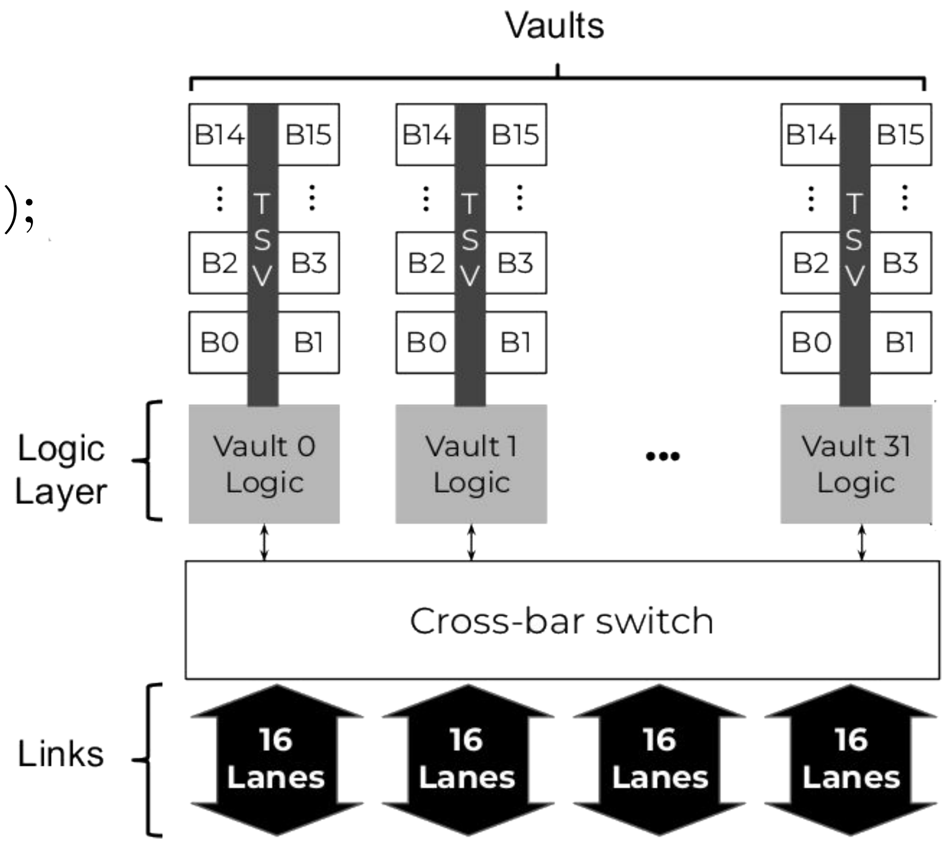
VIMA (Vector-In-Memory Architecture)

- Common fine-grain NDP architecture features:
    - 3D-stacked memory-based in-memory accelerator;



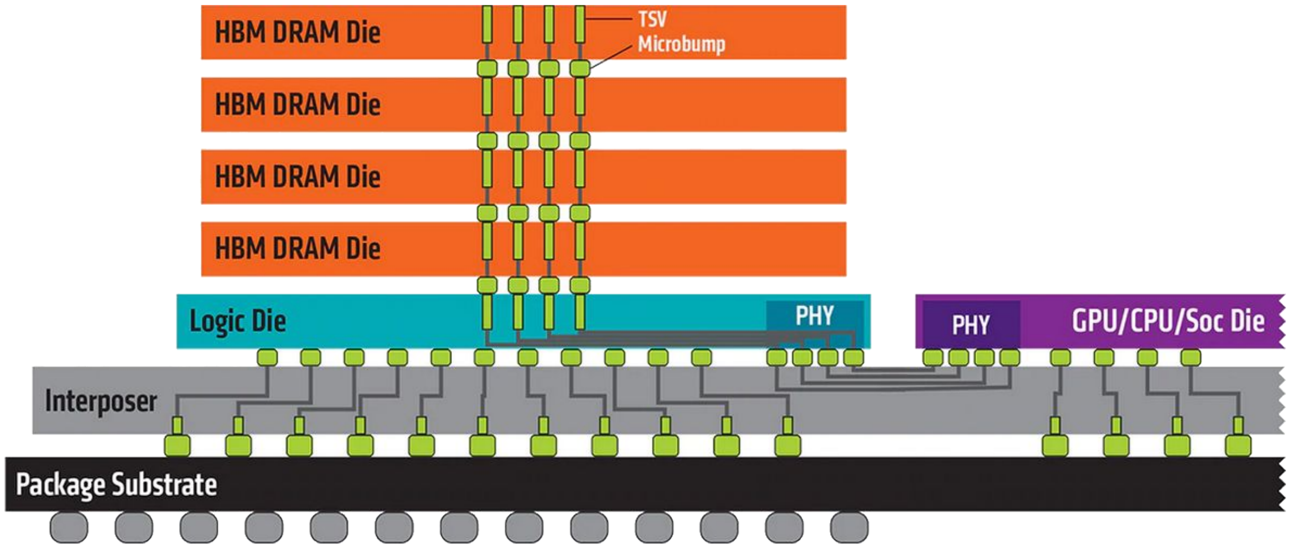
3D-stacked Memories

- Hybrid Memory Cube (HMC);
- High Bandwidth Memory (HBM);



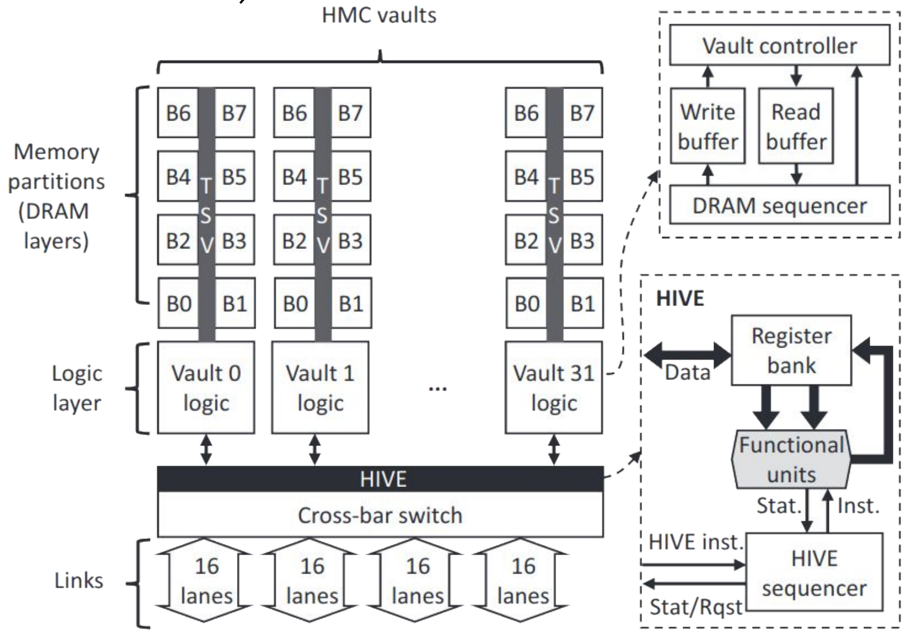
3D-stacked Memories

- Hybrid Memory Cube (HMC);
- High Bandwidth Memory (HBM);



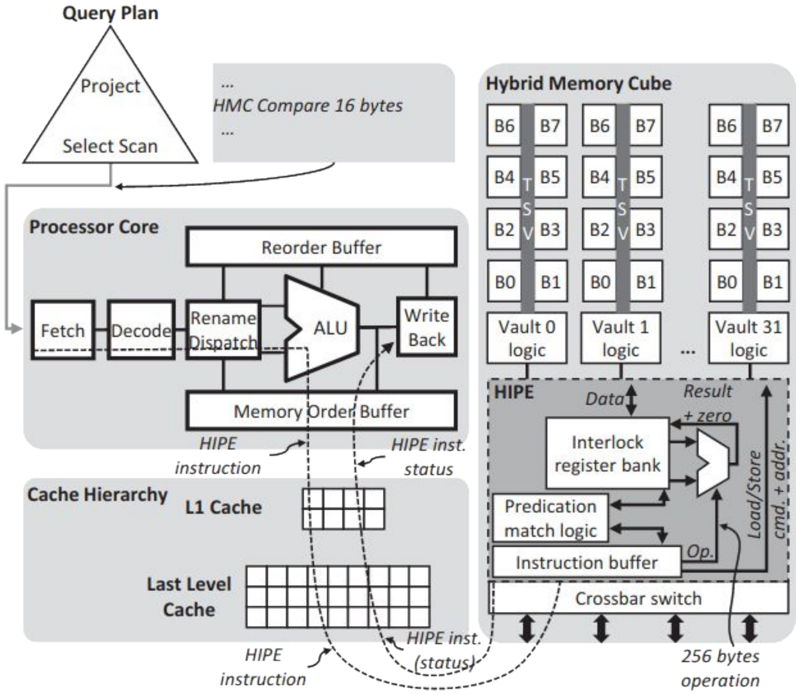
VIMA (Vector-In-Memory Architecture)

- HIVE (HMC Instruction Vector Extensions)
  - Alves et al., 2016;
  - Register bank data storage;
  - Hybrid Memory Cube;
  - Large vector instructions;



VIMA (Vector-In-Memory Architecture)

- HIPE (HMC Instruction Predication Extension)
  - Tomé et al., 2018;
  - Register bank data storage;
  - Hybrid Memory Cube;
  - 256-byte vectors;

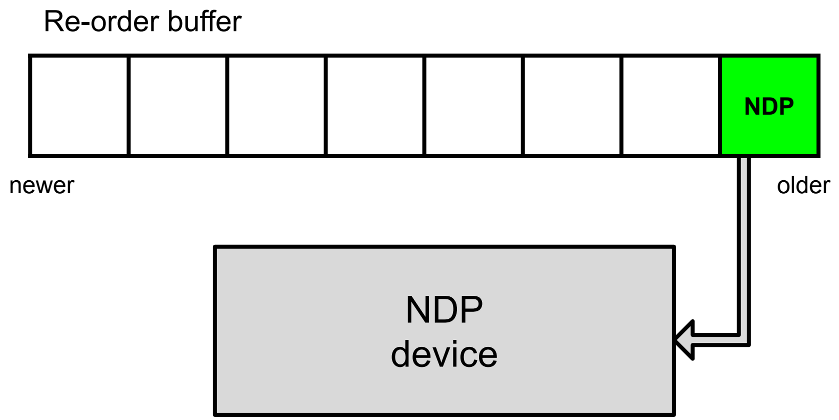


VIMA (Vector-In-Memory Architecture)

- Contributions;
  - 3D-stacked memory-agnostic;
  - Precise exceptions;
  - Multithreading;
  - Maintaining benefits;
    - Reduced execution time;
    - Reduced energy consumption;
    - Increased data throughput;

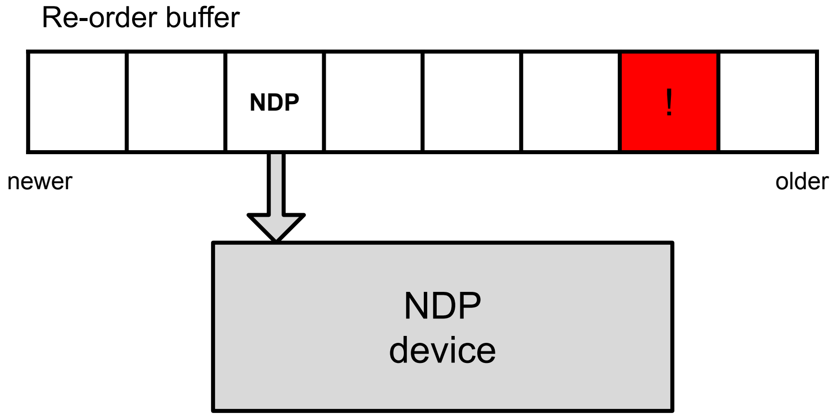
Precise Exceptions

- Adding a accelerator to a system causes it to become multi-processed;
- Programmers still assume a sequentially consistent model;
- How to guarantee precise exceptions with NDP?
  - Strict in-order execution → one instruction at a time;
  - Likely poor performance;
  - State-of-the-art doesn't!



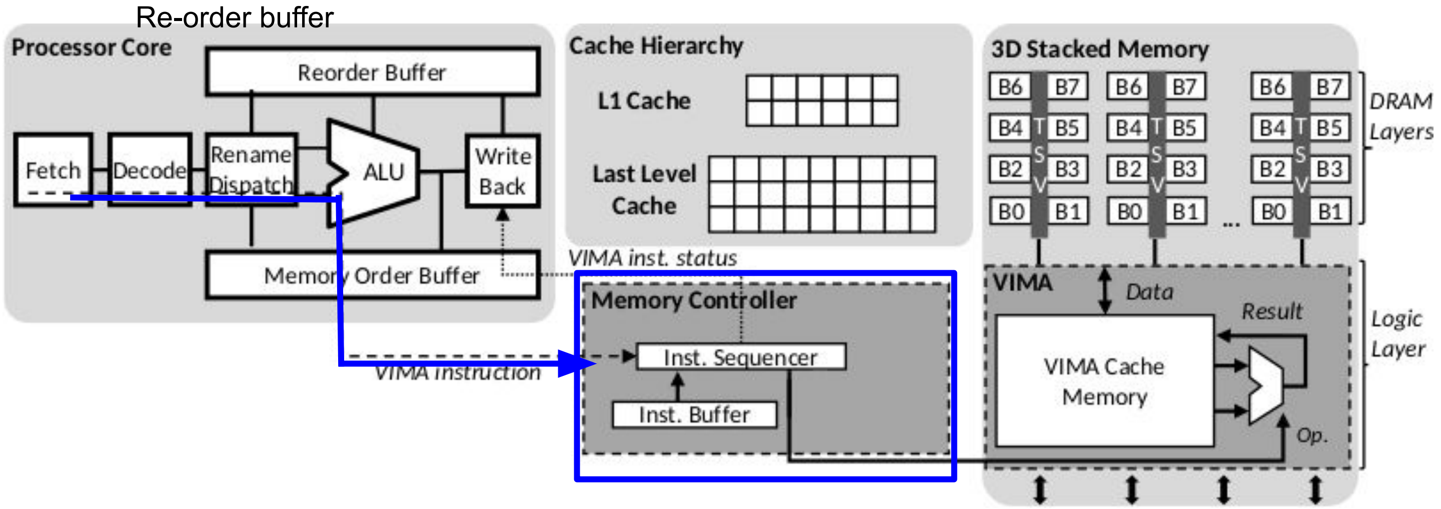
Precise Exceptions

- Adding a accelerator to a system causes it to become multi-processed;
- Programmers still assume a sequentially consistent model;
- How to guarantee precise exceptions with NDP?
  - Strict in-order execution → one instruction at a time;
  - Likely poor performance;
  - State-of-the-art doesn't!



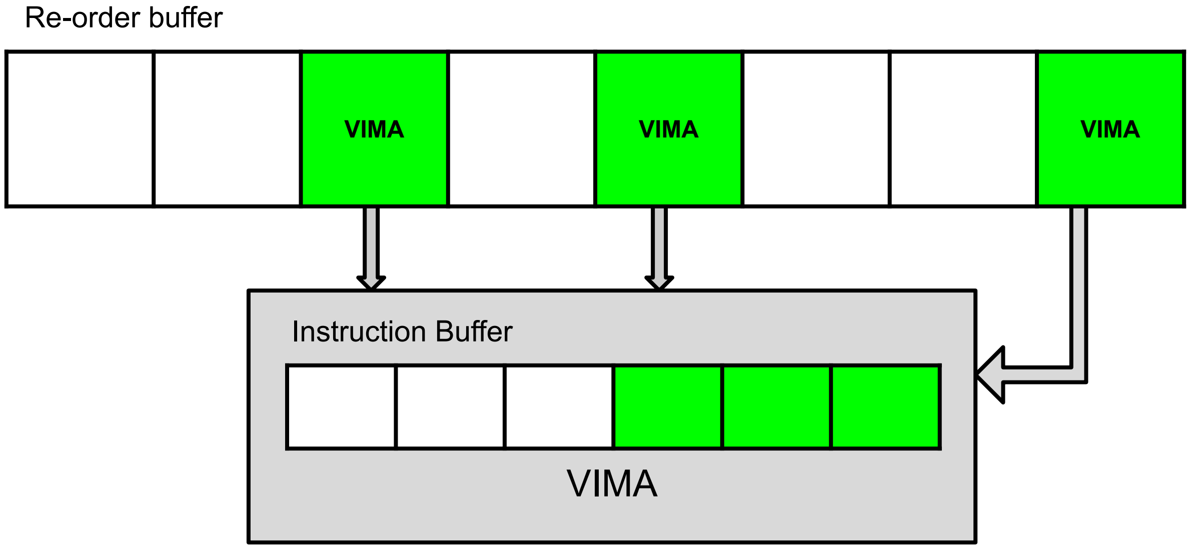
Precise Exceptions

- Providing precise exceptions **and** good performance;
- Instruction pooling;
- Instruction sequencer and buffer;

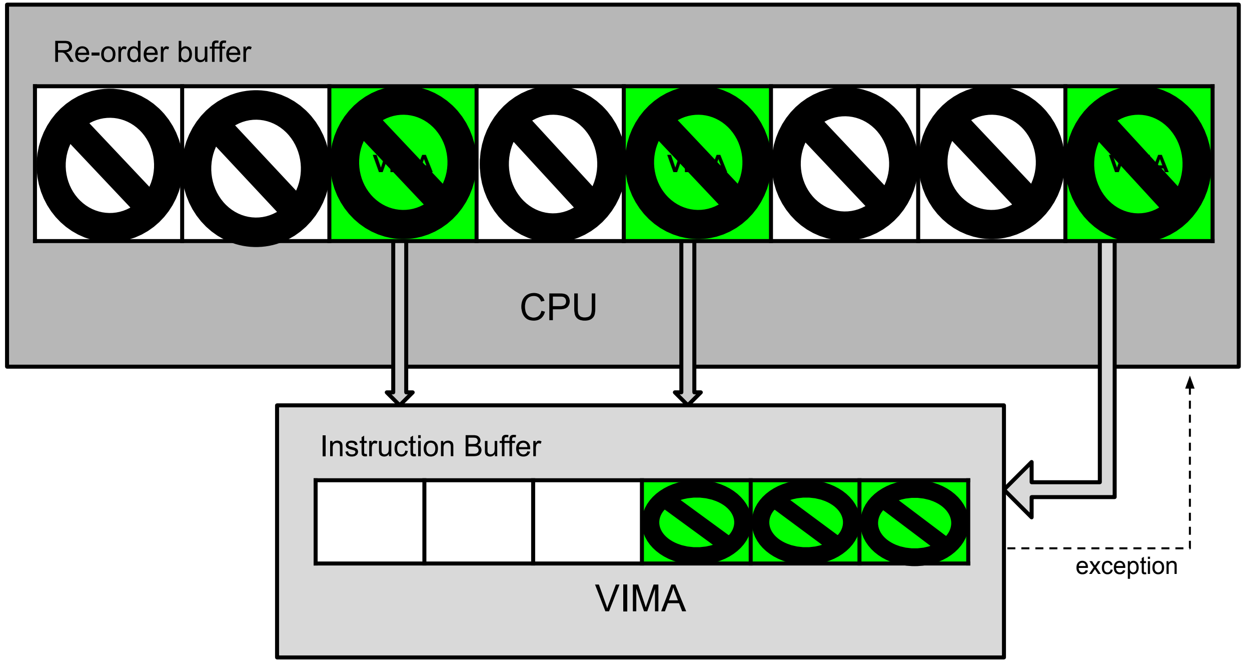


Precise Exceptions

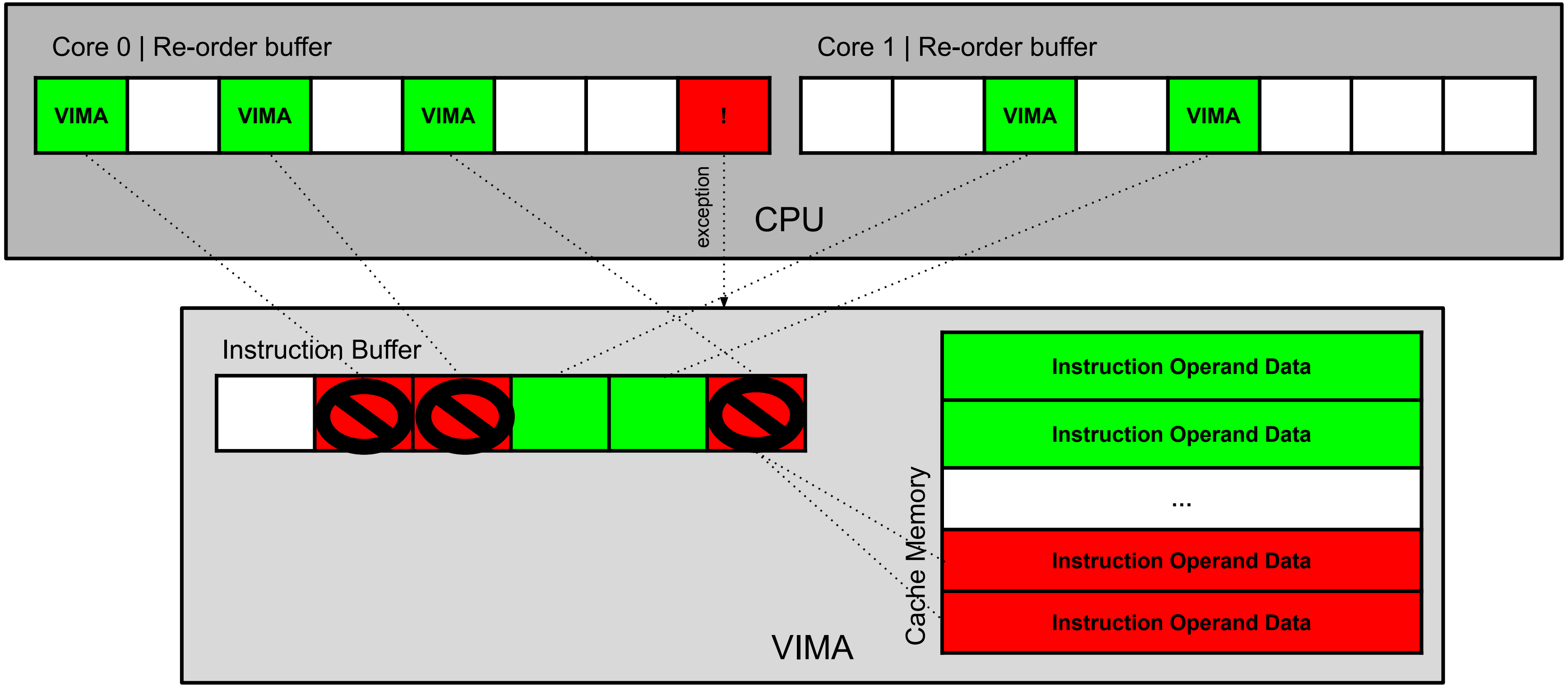
- Providing precise exceptions **and** good performance;
- Instruction pooling;
- Instruction sequencer and buffer;



Precise Exceptions

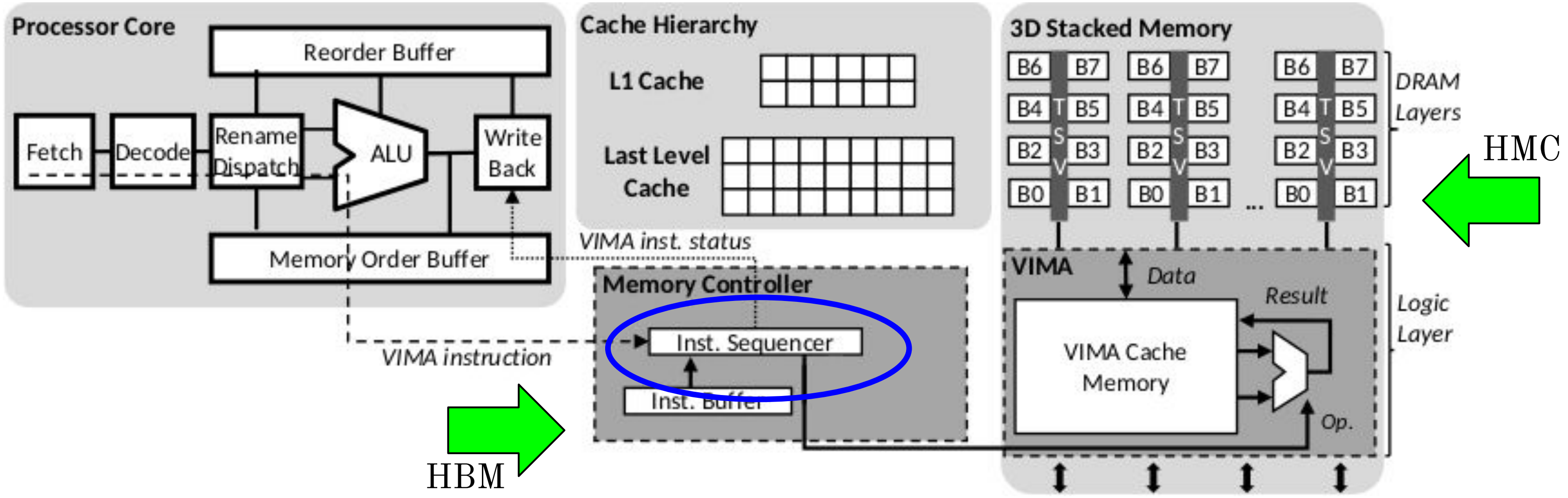


Precise Exceptions



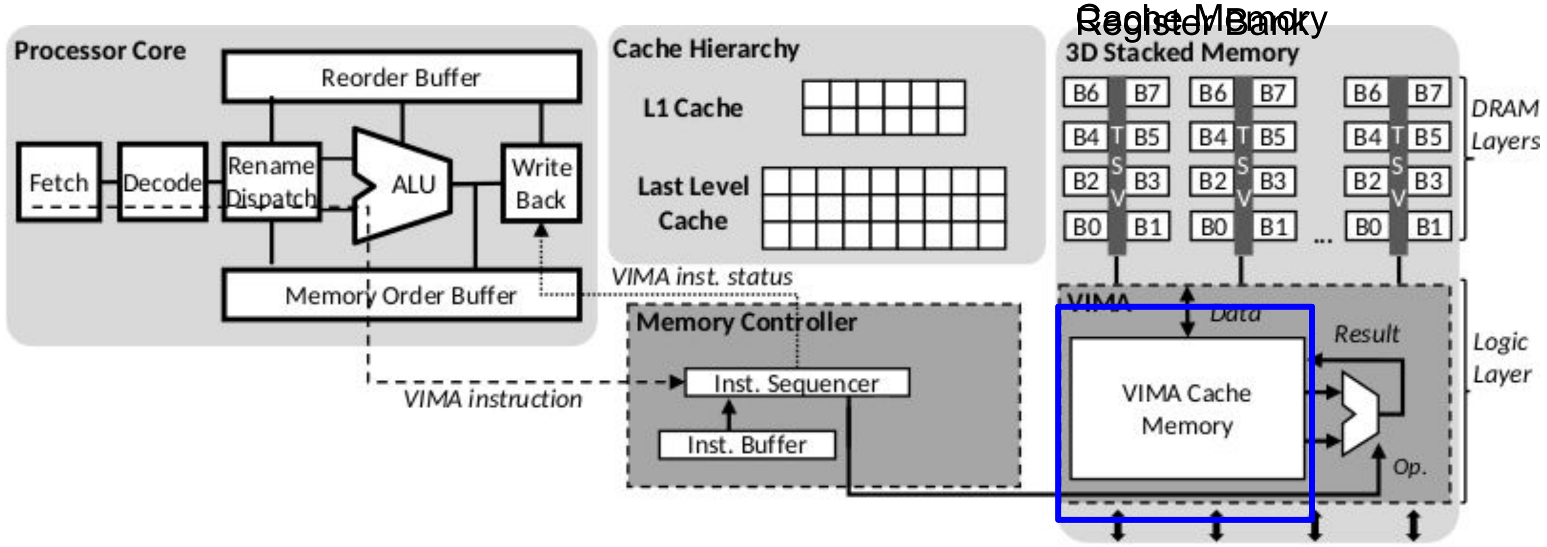
Memory-agnostic

- Instruction sequencer placement;
- Difference in memory control;



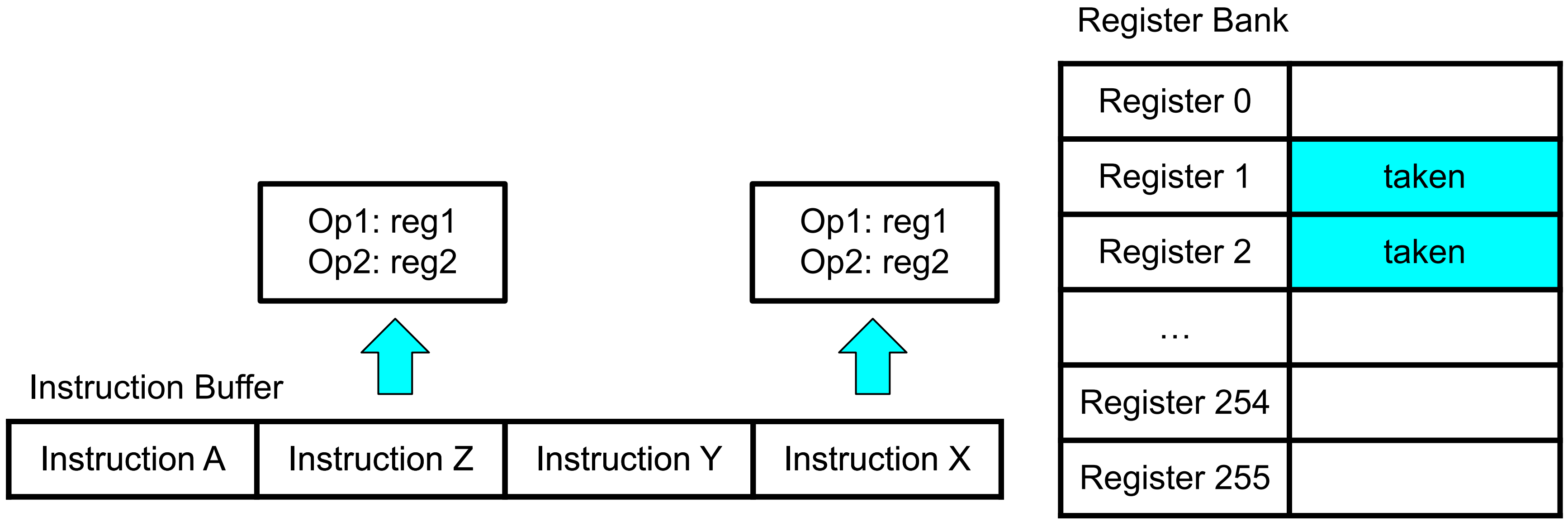
Data Storage

- Dedicated cache memory for operand storage;
- Benefits and drawbacks;
- Transparent allocation of data;



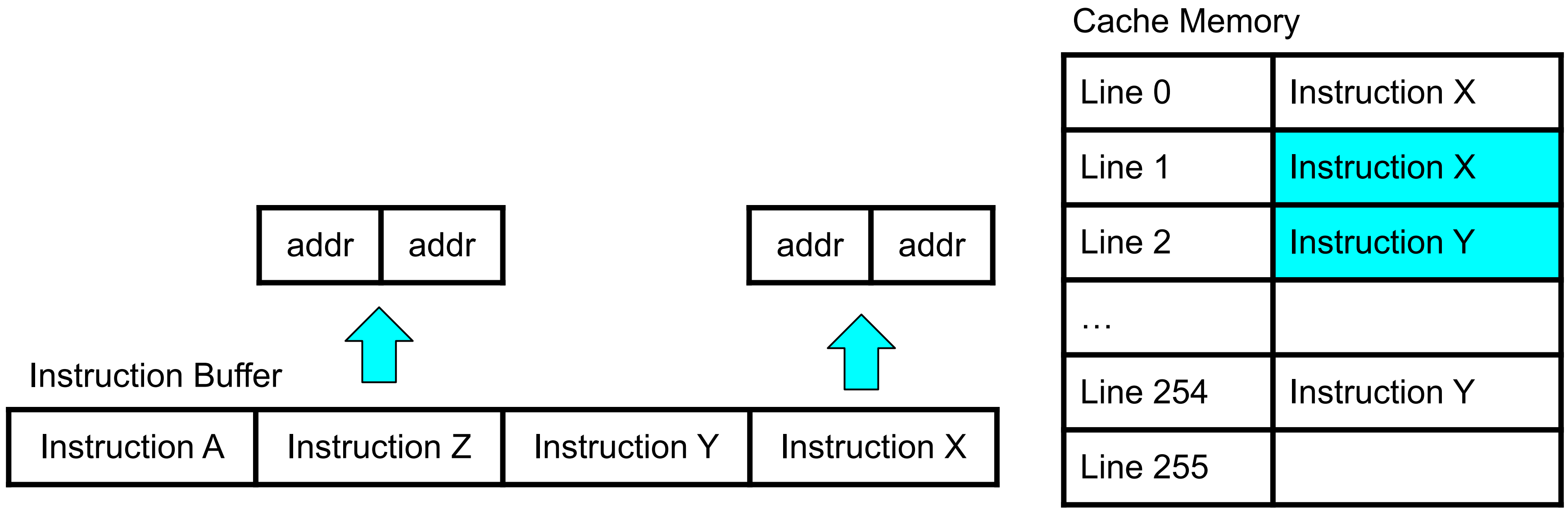
Data Storage

- Dedicated cache memory for operand storage;
- Benefits and drawbacks;
- Transparent allocation of data;



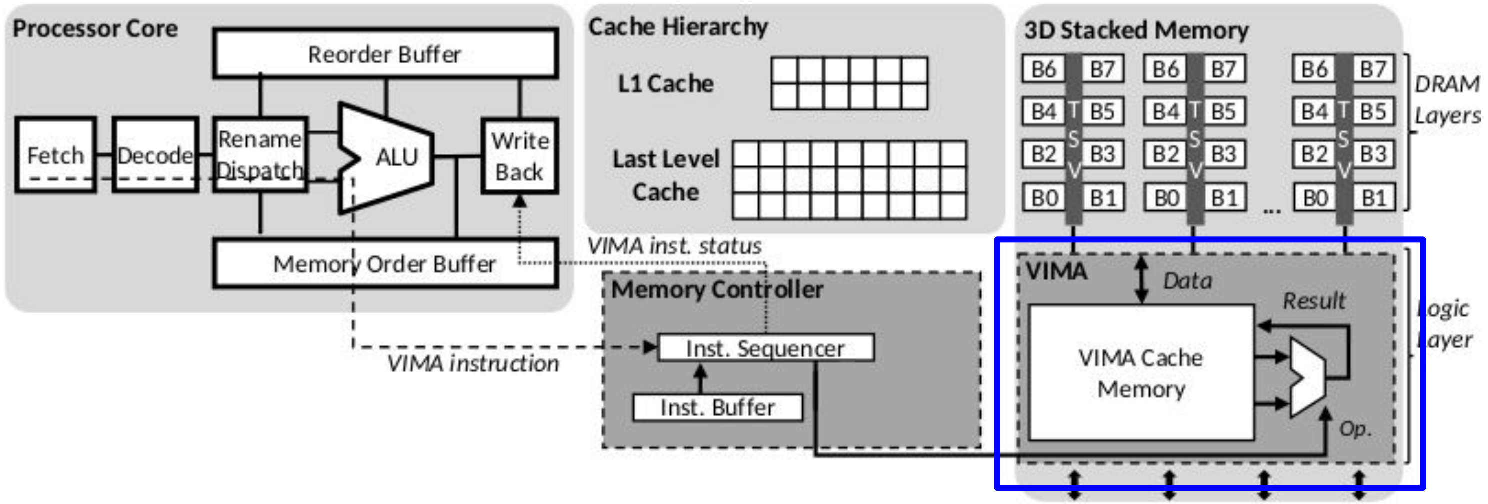
Data Storage

- Dedicated cache memory for operand storage;
- Benefits and drawbacks;
- Transparent allocation of data;

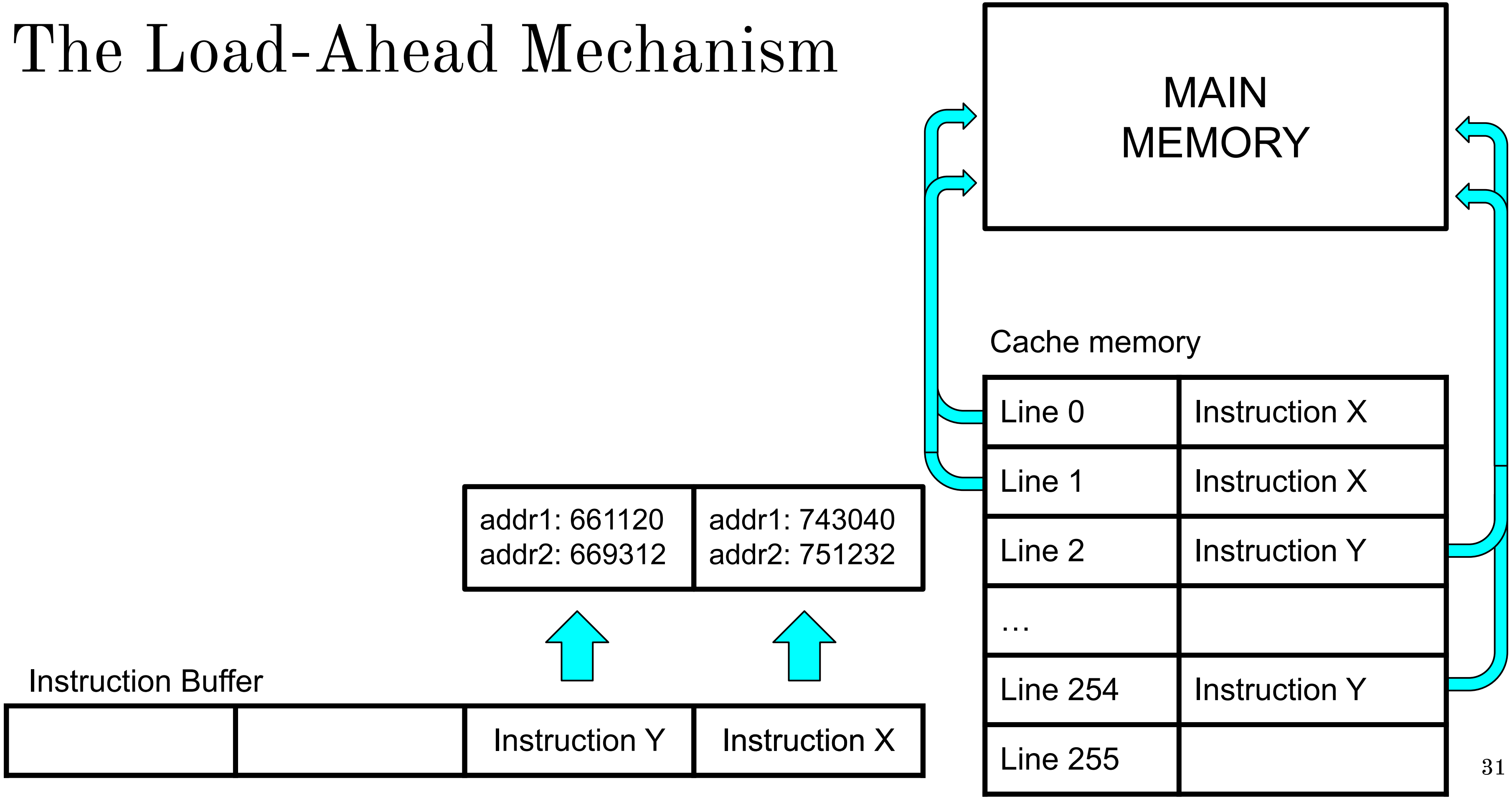


The Load-Ahead Mechanism

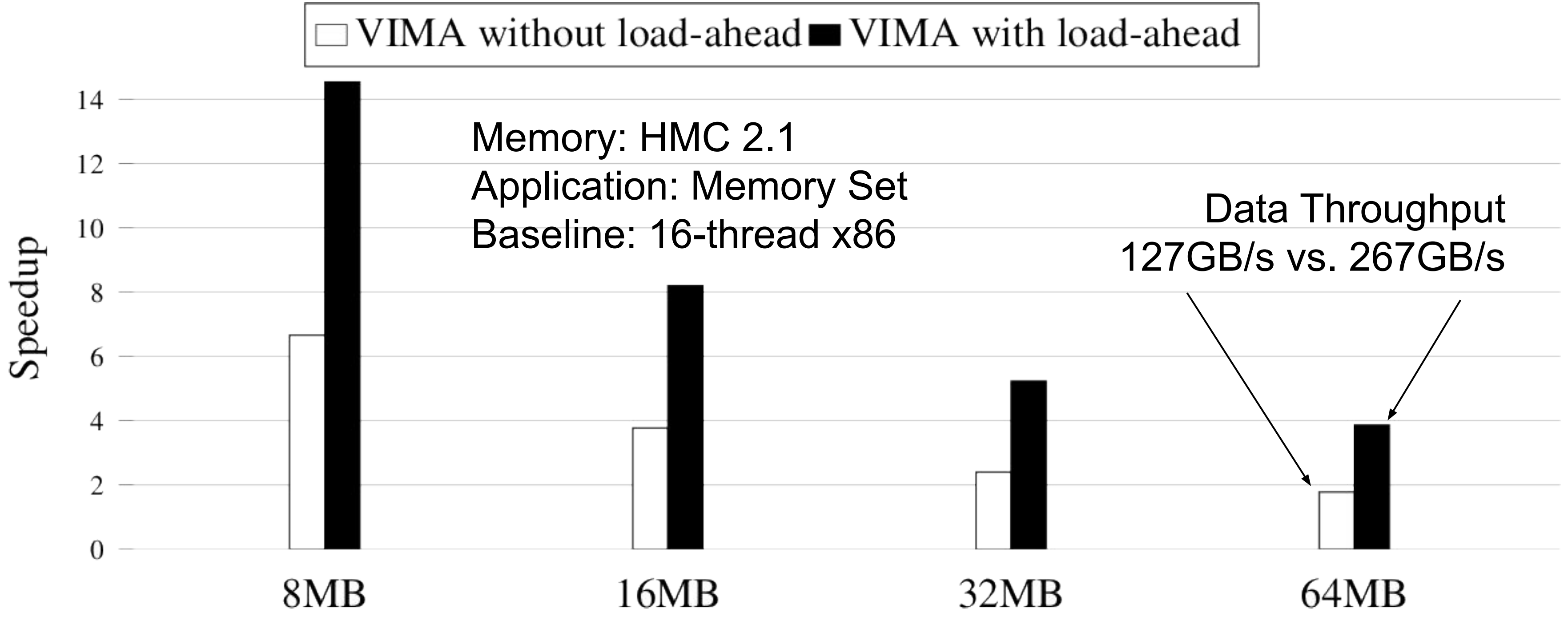
- Further optimizations;
- Buffered instructions + precise exceptions + cache storage;
- Out-of-order operand loading;
- Execution and committing stays in-order;



The Load-Ahead Mechanism



The Load-Ahead Mechanism

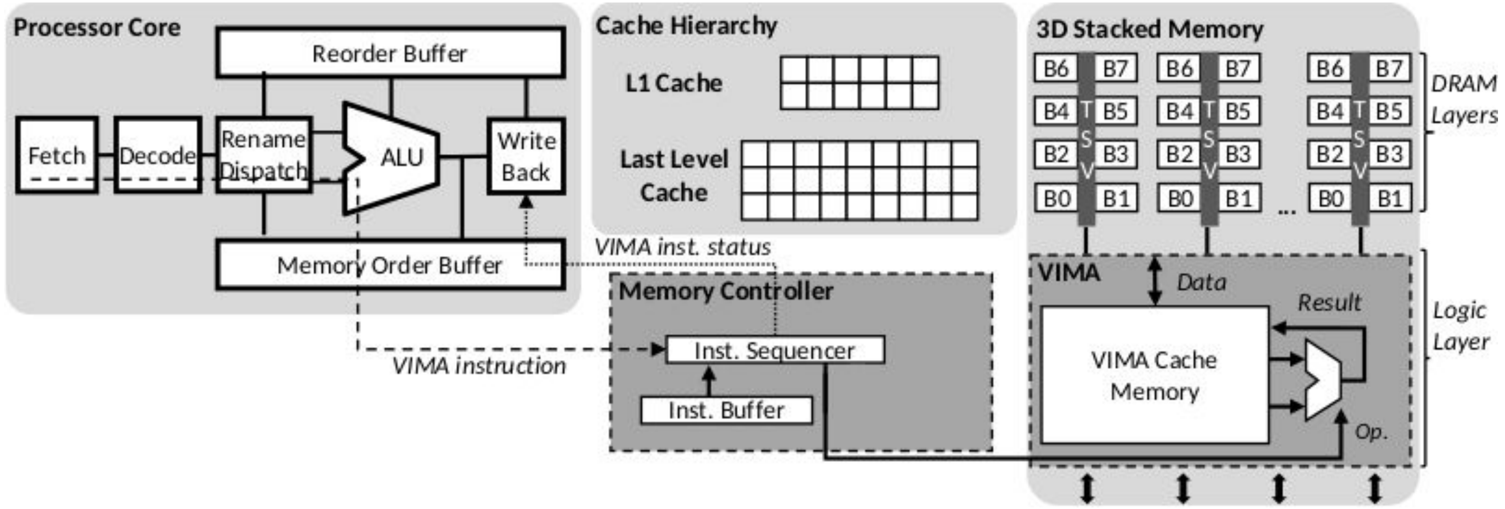


Memory: HMC 2.1
Application: Memory Set
Baseline: 16-thread x86

Data Throughput
127GB/s vs. 267GB/s

Multithreading

- Keeping multiple threads separate;
- Benefit of a cache-based storage;
- Appropriate pressure at smaller vector widths;



Evaluation

Evaluation

- Baseline:
  - Intel Skylake, 16 cores @ 2GHz;
  - AVX-512;
  - 64KB L1;
  - 1MB L2;
  - 16MB LLC;
  - Hybrid Memory Cube 2.1;
- OrCS (Ordinary Computer Simulator);
- VIMA (Vector-In-Memory Architecture):
  - Host single core @ 2GHz;
  - VIMA @ 1GHz;
  - 256KB dedicated cache;
  - Varying main memory;

3D-stacked Memories

- Hybrid Memory Cube 2.1
  - 32 vaults;
  - 8 banks;
  - 256-byte row buffer;

32 x 256 = 8192 bytes

- High Bandwidth Memory 3
  - 16 vaults;
  - 64 banks;
  - 1024-byte row buffer;

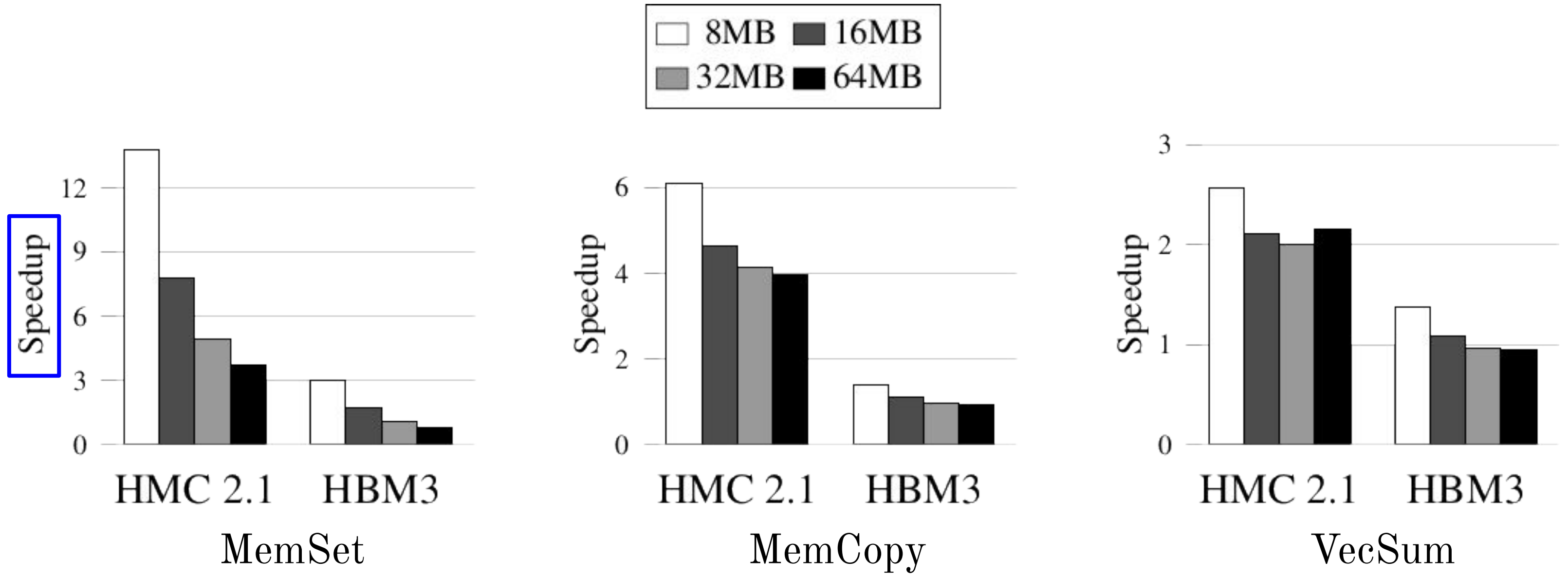
16 x 1024 = 16384 bytes

Workload Characterization

- 8, 16, 32 and 64MB input datasets;
    - Baseline cache;
- Applications:
    - Consumer Workloads
        - Memory Set
        - Memory Copy
        - Vector Sum
    - Database Query Operators
        - Selection
        - Projection
    - Computational Fluid Dynamics
        - 5-point convolution
    - String Matching
        - Bloom Filter

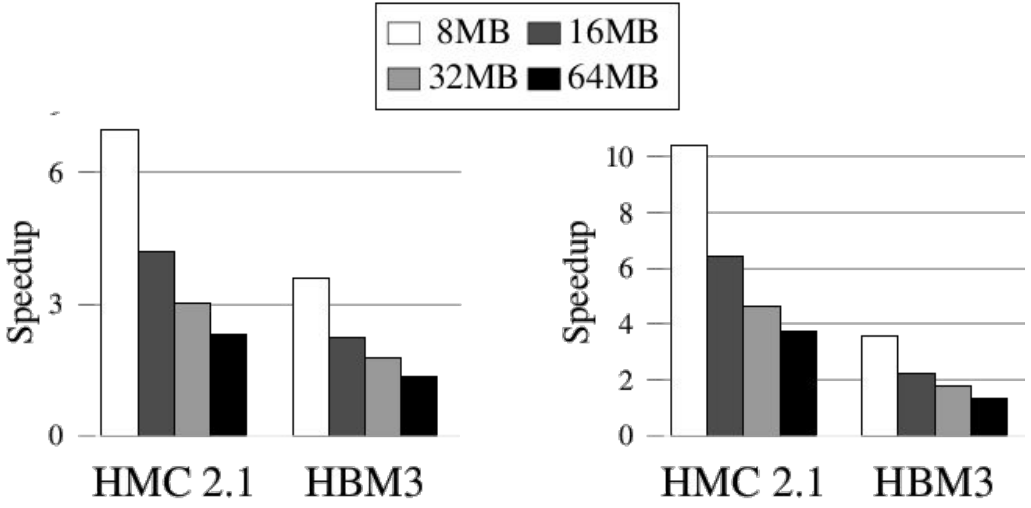
Results: Execution Time

- Data-streaming applications (Consumer workloads);



Results: Execution Time

- Data-streaming applications (Database query operators);

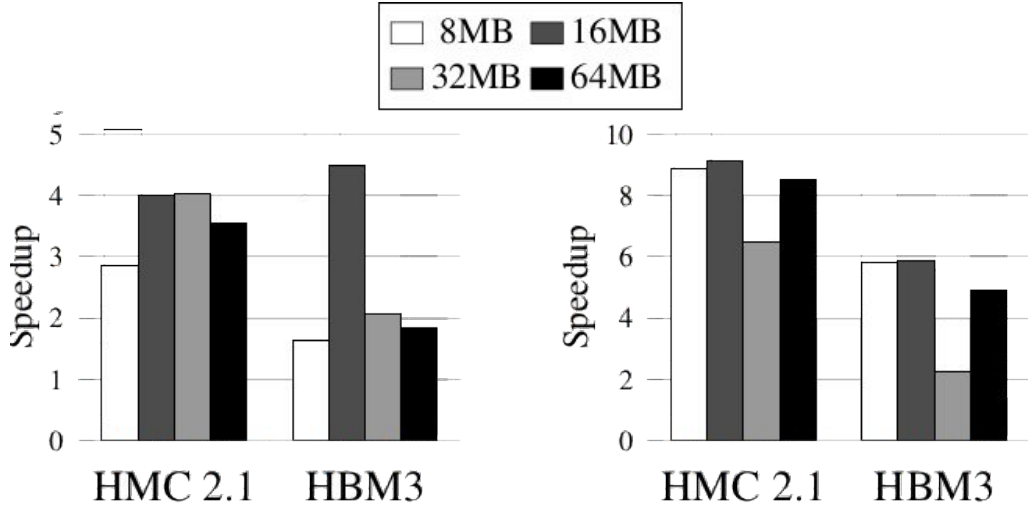


Selection

Projection

Results: Execution Time

- Data-reuse applications (CFD and string matching);



Convolution   Bloom Filter

Understanding Execution Time Results

- HMC 2.1;
  - Average speedup: 8.75x;
  - 32 vaults;
  - 256-byte row buffer;
  - 256-byte maximum request size;
  - 32 memory controllers;

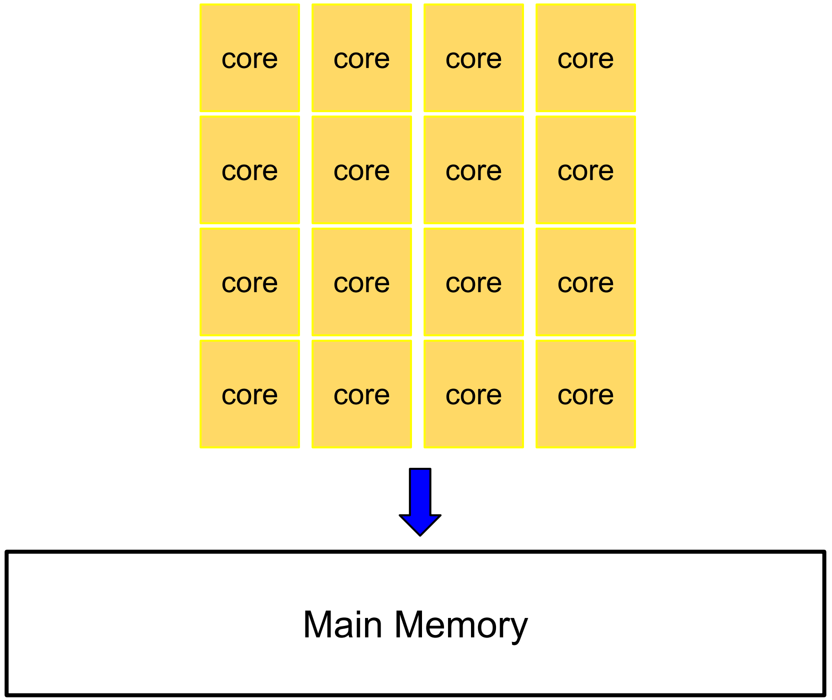
- HBM3;
  - Average speedup: 4.12x;
  - 16 vaults;
  - 1024-byte row buffer;
  - 128-byte maximum request size;
  - Single memory controller;

Results: Execution Time

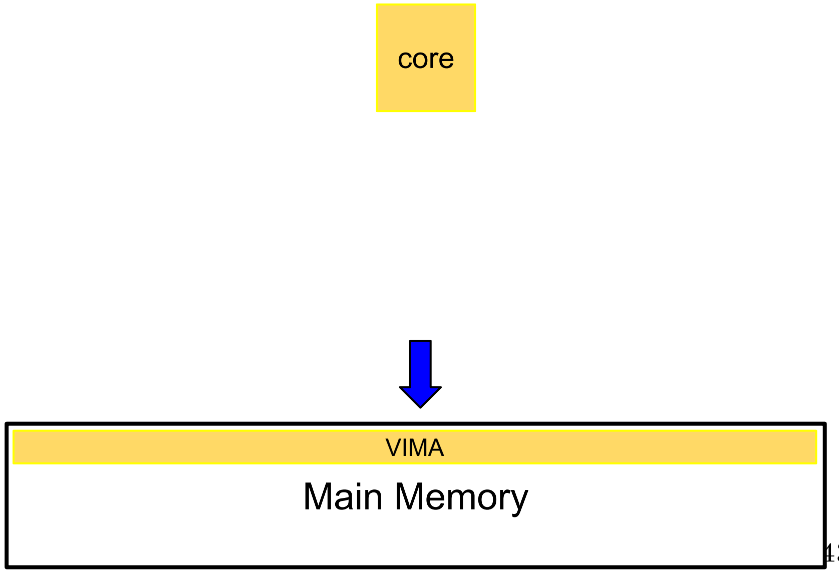
- Significant results;
- Superior results with HMC 2.1 memory;
    - Up to 13x with HMC 2.1 memory;
    - Up to 5x with HBM3 memory;
- Memory device organization impacts performance;

Results: Energy Consumption

- Baseline

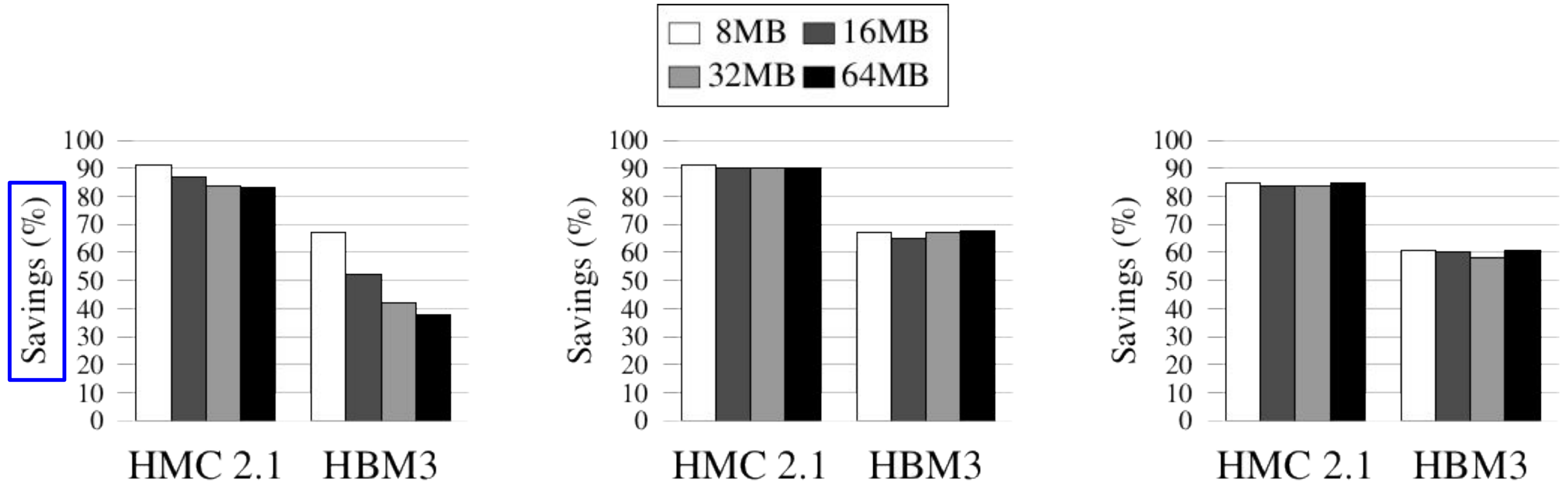


- VIMA



Results: Energy Consumption

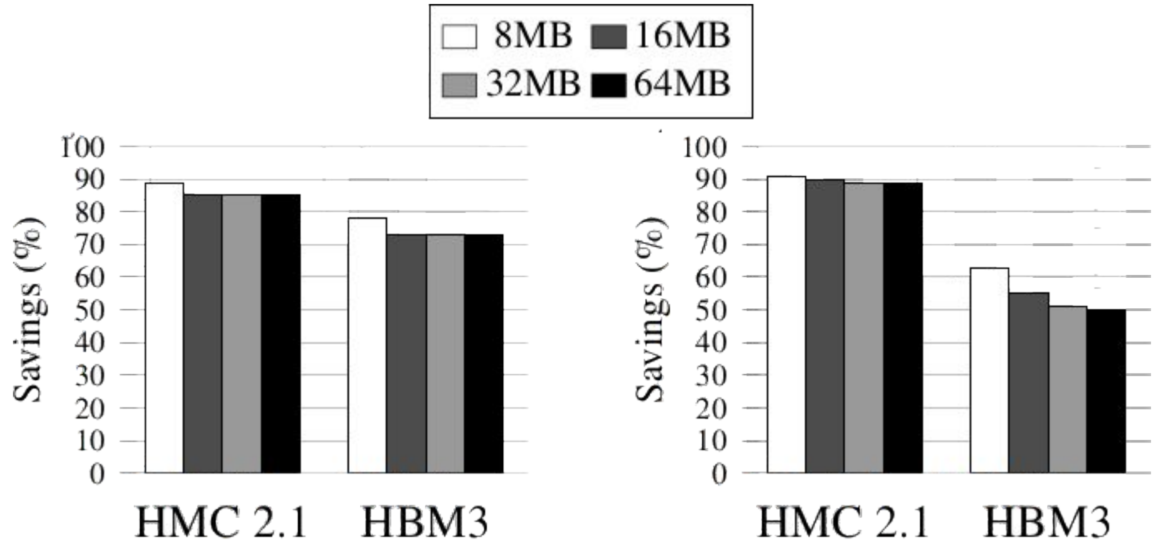
- Estimates based on CACTI and McPAT;
- Data-streaming applications (Consumer workloads);



MemSet MemCopy VecSum

Results: Energy Consumption

- Data-streaming applications (Database query operators);

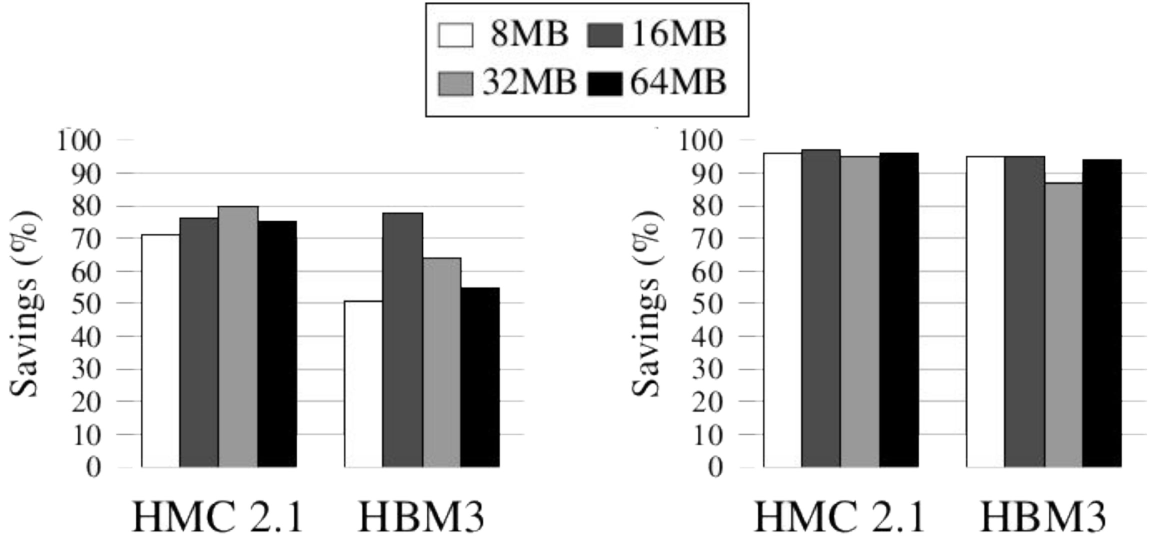


Selection

Projection

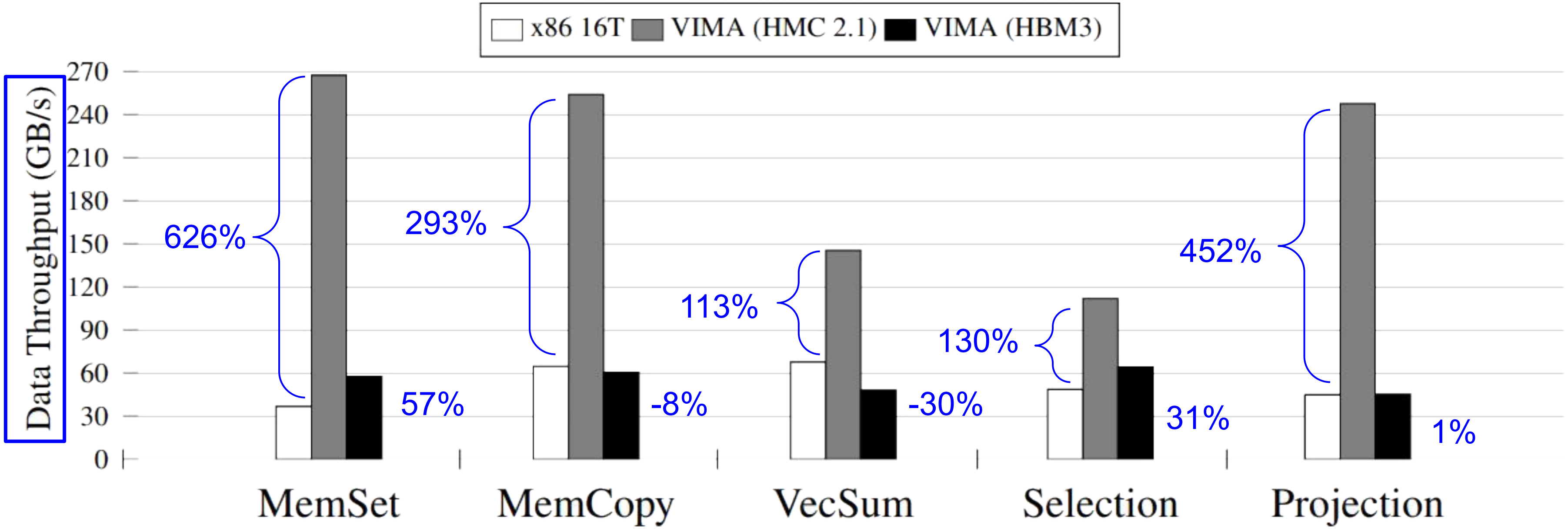
Results: Energy Consumption

- Data-reuse applications (CFD and string matching);
- Unrealistic bloom filter results;



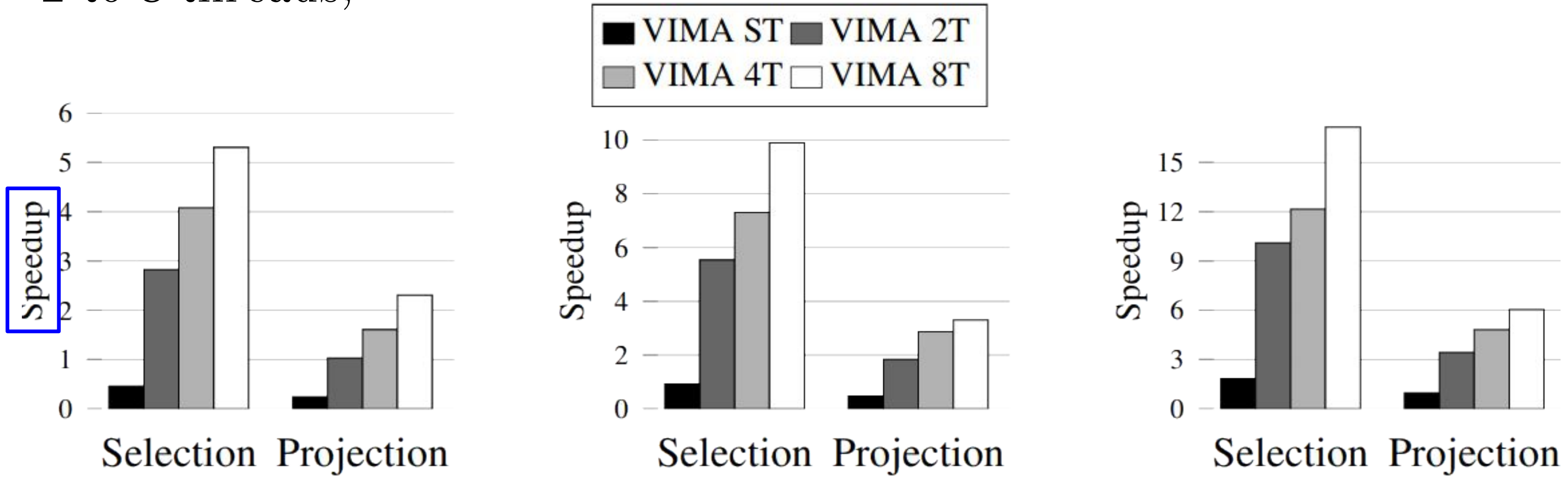
Convolution   Bloom Filter

Results: Data Throughput



Results: Near-Data Multithread

- Database query operators, 64MB input dataset;
- Smaller vector widths;
- 2 to 8 threads;



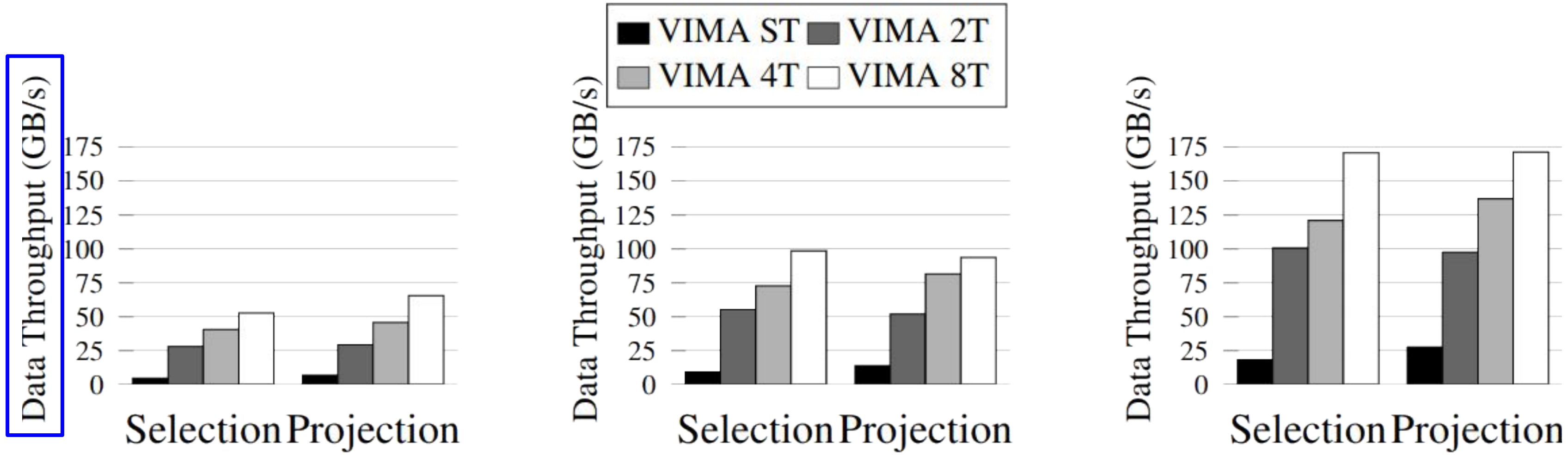
) 256 B Vector

' 512 B Vector

1024 B Vector

Results: Near-Data Multithread

- Improved data throughput with load-ahead mechanism and multiple threads;



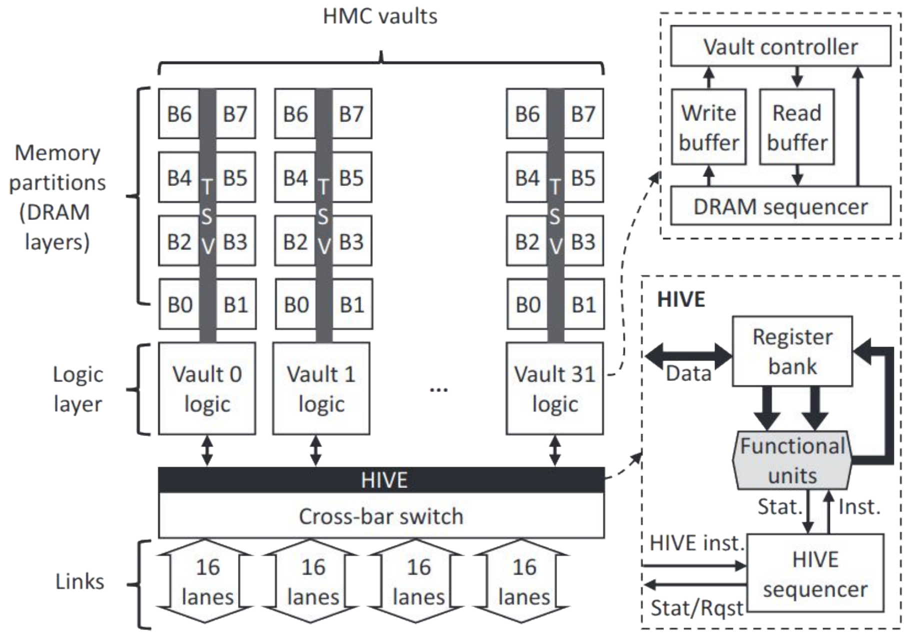
256 B Vector

) 512 B Vector

) 1024 B Vector

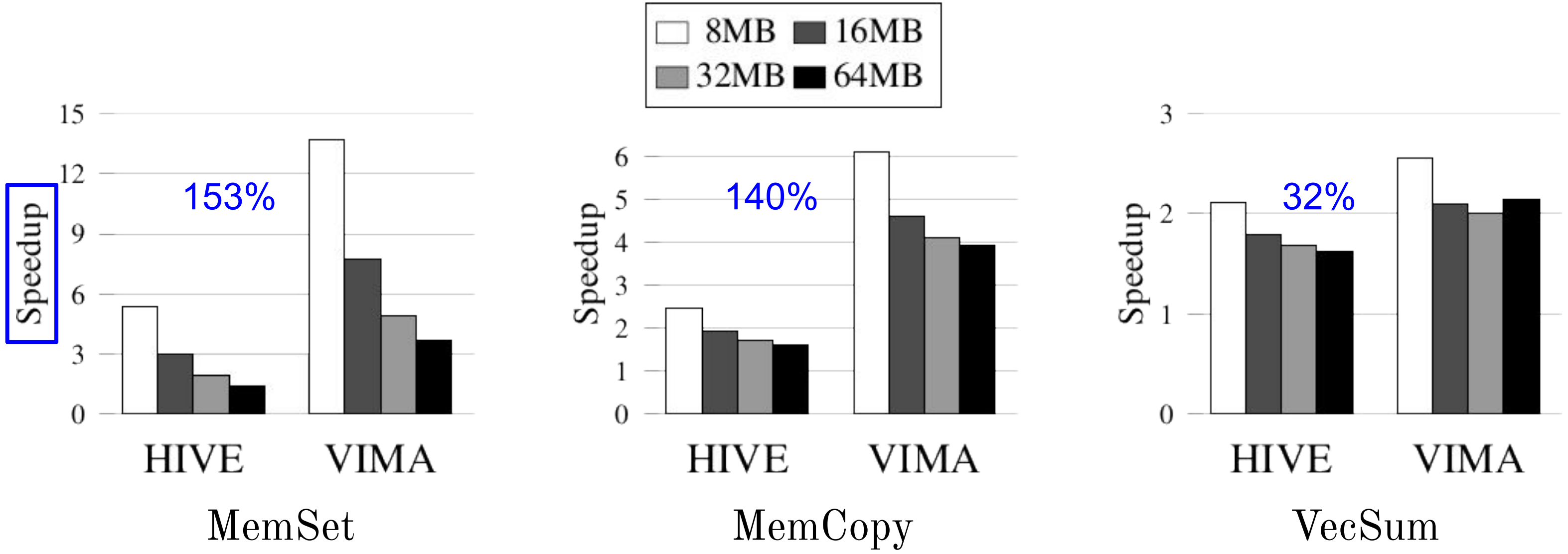
Results: State-of-the-art Comparison

- HIVE (HMC Instruction Vector Extensions)
  - Alves et al., 2016;
  - Similar design;
  - Register bank data storage;
  - Shortcomings:
    - "Imprecise exceptions";
    - More Limited ISA;
    - HMC dependent;
    - No active multithreading;



Results: State-of-the-art Comparison

- Only applications available to the two architectures;



Conclusions

Conclusions

- The memory wall continues to be a significant bottleneck;
  - NDP systems attempt to alleviate the issue;
- We propose Vector-In-Memory Architecture (VIMA);
  - FU-based;
  - Precise exceptions;
  - Improved performance against modern traditional system;
  - Large reduction in energy consumption;
  - Near-data multithreading;
  - Between 32% and 153% faster than the state-of-the-art;

Conclusions

- Hypothesis

“It is possible to provide precise exceptions, improved memory access and multithreading with fine-grain NDP to speedup data streaming applications”

Activities

- VIMA: Idealization, initial development and all extensions;
- VIMA: Extension of Intrinsics-VIMA library;
- Porting all applications to VIMA;
- OrCS: DRAM memory modelling;
- OrCS: HMC Instruction Vector Extensions (HIVE) simulation;
- OrCS: Vector-In-Memory Architecture (VIMA) simulation;
- OrCS: Extension of Pin-based trace generator;
- All experiments featured on presentation and text;
- All data analysis features on presentation and text;
- Participation on several collaboration research papers;
- Lead writing of several research paper manuscripts;

List of Publications

1. CORDEIRO, ALINE S.; **SANTOS, SAIRO R.**; MOREIRA, FRANCIS B.; SANTOS, PAULO C.; CARRO, LUIGI; ALVES, MARCO A. Z. Machine Learning Migration for Efficient Near-Data Processing In: 29th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2021.
2. SANTOS, PAULO C.; MOREIRA, FRANCIS B.; CORDEIRO, ALINE S.; **SANTOS, SAIRO R.**; KEPE, TIAGO R.; CARRO, LUIGI; ALVES, MARCO A. Z. Survey on Near-Data Processing: Applications and Architectures In: Journal of Integrated Circuits and Systems, 2021.
3. CORDEIRO, ALINE S.; **SANTOS, SAIRO R.**; MOREIRA, FRANCIS B.; SANTOS, PAULO C.; ALVES, MARCO A. Z. Efficient Machine Learning execution with Near-Data Processing In: Microprocessors and Microsystems, 2022.
4. **SANTOS, SAIRO R.**; MOREIRA, FRANCIS B.; KEPE, TIAGO R.; SANTOS, PAULO C.; ALVES, MARCO A. Z. Advancing Database System Operators with Near-Data Processing In: 2022 30th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2022.
5. ALVES, MARCO A. Z.; **SANTOS, SAIRO R.**; CORDEIRO, ALINE S.; MOREIRA, FRANCIS B.; SANTOS, PAULO C.; CARRO, LUIGI Vector In Memory Architecture for simple and high efficiency computing In arXiv preprint arXiv:2203.14882.
6. **SANTOS, SAIRO R.**; ALVES, MARCO A. Z. Impacto da Largura do Vetor de Instruções SIMD em Arquiteturas de Processamento Próximo à Memória In XXII Escola Regional de Alto Desempenho da Região Sul (ERAD/RS), 2022.
7. **SANTOS, SAIRO R.**; KEPE, TIAGO R.; MOREIRA, FRANCIS B.; ALVES, MARCO A. Z. Advancing Near-Data Processing with Precise Exceptions and Efficient Data Fetching In: 2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2022.
8. **SANTOS, SAIRO R.**; MOREIRA, FRANCIS B.; KEPE, TIAGO R.; SANTOS, PAULO C.; ALVES, MARCO A. Z. Improved Computation of Database Operators via Vector Processing Near-Data In: The Journal of Supercomputing, 2022. *(pending confirmation)*

Thank you!