Pacific Northwest
NATIONAL LABORATORY

Commercial vs Open ISAs: a High-Performance Data Analytics Perspective

September 7, 2022

Antonino Tumeo

U.S. DEPARTMENT OF ENERGY BATTELLE

PNNL is operated by Battelle for the U.S. Department of Energy

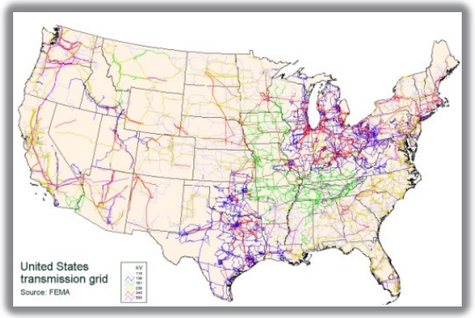
Pacific Northwest
NATIONAL LABORATORY

Graphs are Everywhere

Big Science

Bioinformatics

Community Detection

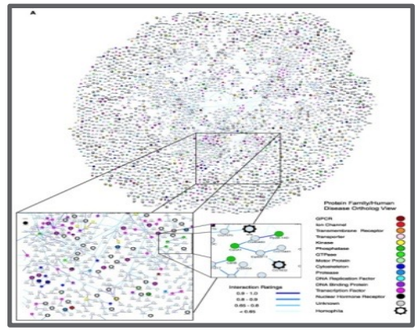


Complex Networks

Graph Databases

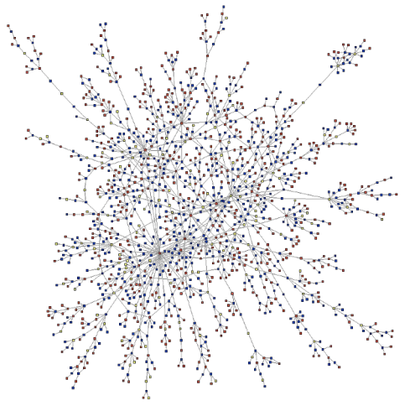
Knowledge Discovery

Language Understanding



Pattern Recognition

Massive Social Networks



facebook surpassed 1 billion active users

- Statistics
  - More than 1 billion active users, even more objects
  - Average user has 130 friends and is connected to 80 community pages, groups, and events
- Graph characteristics
  - **Topology**: Interaction graph is low-diameter and has no good separators
  - **Irregularity**: Communities are not uniform in size
  - **Overlap**: individuals are members of one or more communities
- Sample queries:
  - **Allegiance switching**: identify entities that switch communities
  - **Community structure**: identify the genesis and dissipation of communities
  - **Phase change**: identify significant change in the network structure

Pacific Northwest NATIONAL LABORATORY

Characteristics of Graph Analytics

- Prototypical irregular kernels

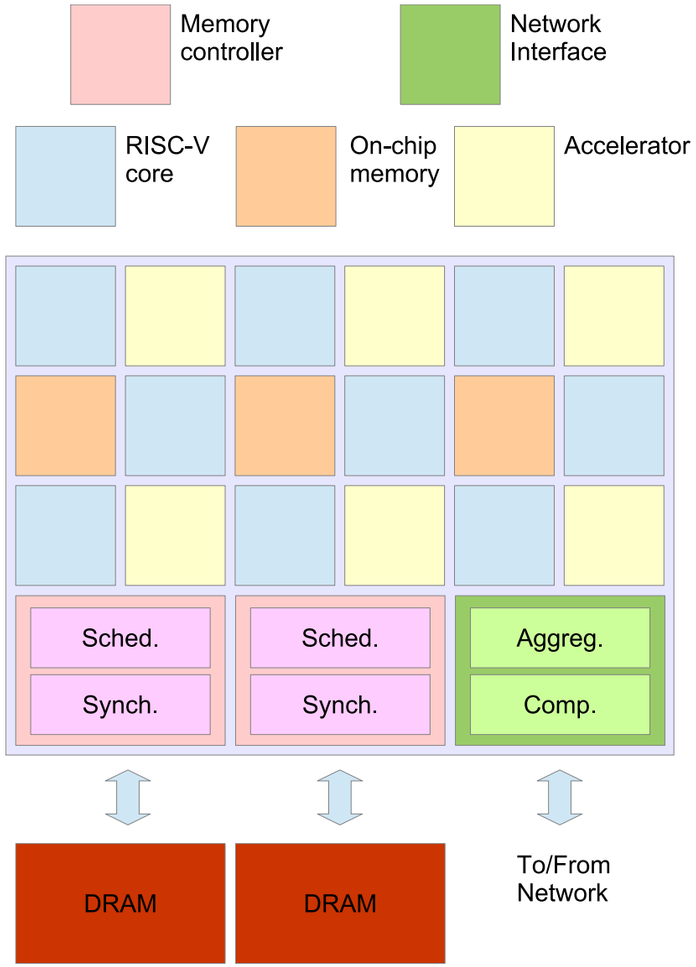
- Irregularity in data structures
  - Very poor spatial and temporal locality
    - ✓ Unpredictable data accesses
    - ✓ Fine grained data accesses
- Irregularity in control
  - Divergent branches
    - ✓ If (vertex==x) z; else k
- Irregularity in communication patterns
  - Unpredictable and fine-grained communication
  - A consequence of irregularity in data structures and in control

Pacific Northwest
NATIONAL LABORATORY

Additional Characteristics

- Very large datasets
  - Way more than the memory available for single cluster nodes
  - Very difficult to partition in a balanced way
- Large amounts of parallelism (e.g., each vertex, each edge in the graph)
- Usually, high synchronization intensity
  - Concurrent activities accessing the same elements of the data structures
- Datasets may be dynamically updated

Strawmen Architecture for P38 Explorations



- "Template" architecture
- Manycore design
- On-chip interconnect
- Application-specific accelerators
- On-chip (configurable) memories
- (Configurable) Memory controllers
- Network-Interface (custom/configurable?)

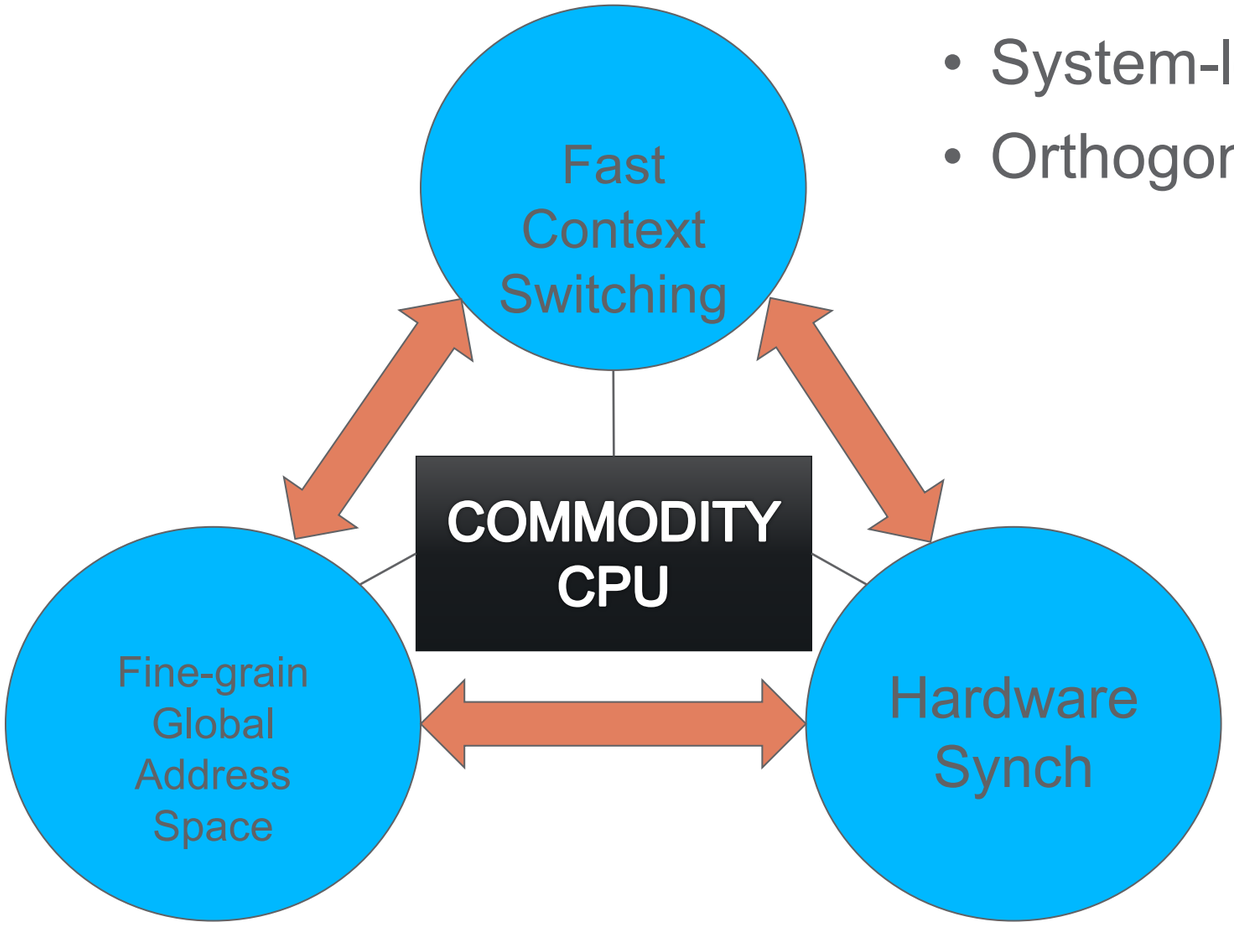
Pacific Northwest NATIONAL LABORATORY

Rationale

Motivation	Hardware characteristic
Workloads are memory bound, many fine-grained memory accesses	Many simple cores to maximize parallelism and memory injection rates
Workloads may exhibit different locality behaviors - (need to understand size of caches, sharing, coherency) We may want to take advantage of used managed on-chip memories	(Configurable) on-chip memories
Opportunities for efficient synchronization, and for on-the-fly optimizations of memory access patterns	(Configurable) memory controllers
Opportunities for optimization of remote operations (e.g., collectives, aggregation, remote atomics)	(Configurable) network interface

Pacific Northwest NATIONAL LABORATORY

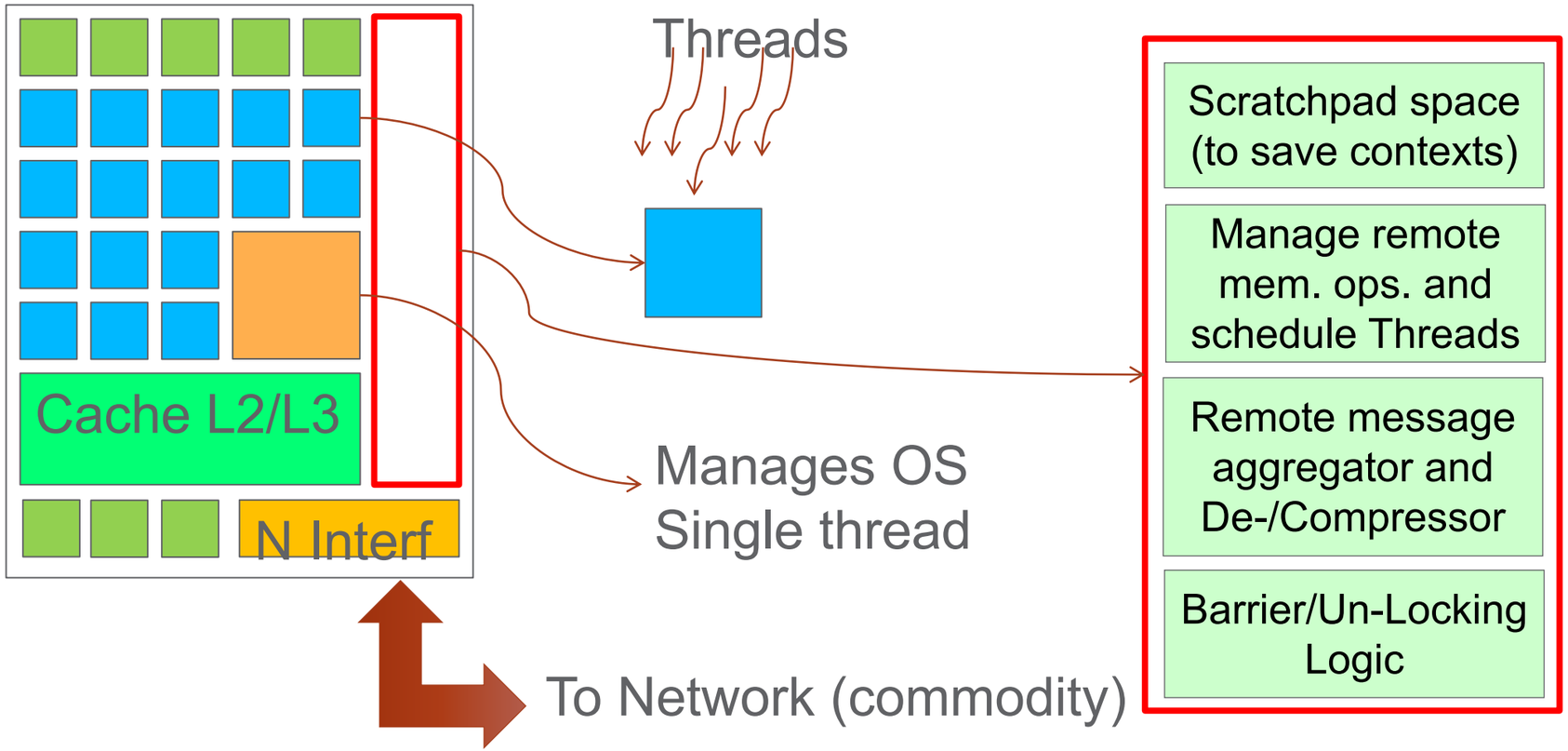
Supporting Irregular Applications



- System-level features
- Orthogonal to GraphBLAS

Possible Node Architecture

- Most scarce resource for this applications is network bandwidth (because of fine grain requests)
- Long-latency accesses are tolerated switching many threads (only if fast context switch)
- Support for context on chip (scratchpad/user controlled local memories)



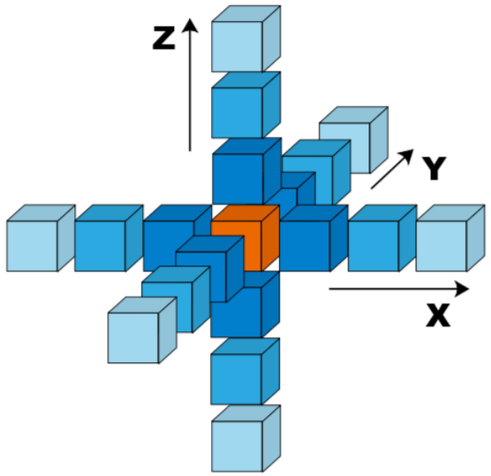
Pacific Northwest
NATIONAL LABORATORY

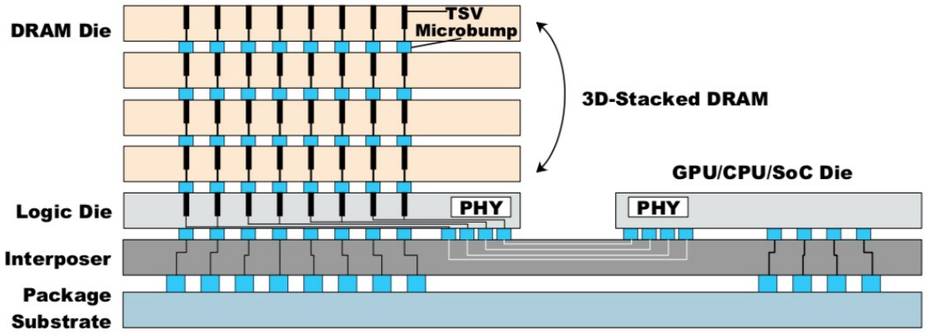
RISC-V Near Memory Explorations for Irregular Applications

- RISC-V platform simulation of new components
  - PIMS: A Lightweight Processing-in-Memory Accelerator for Stencil Computations
  - MAC: Memory Access Coalescer
  - HAM: Hotspot-Aware Manager for 3D Stacked Memory

PIMS: Processing-in-Memory Accelerator for Stencil Computations

- Focuses on stencil computations
  - Typical in Partial Differential Equation solvers
- Exploits in-memory computation
  - Uses the logic die of a 3D-Stacked memory to implement specific custom instructions and related logic
- More “regular” problem, but fundamental kernel in DOE applications and effective proof-of-concept towards more irregular algorithms

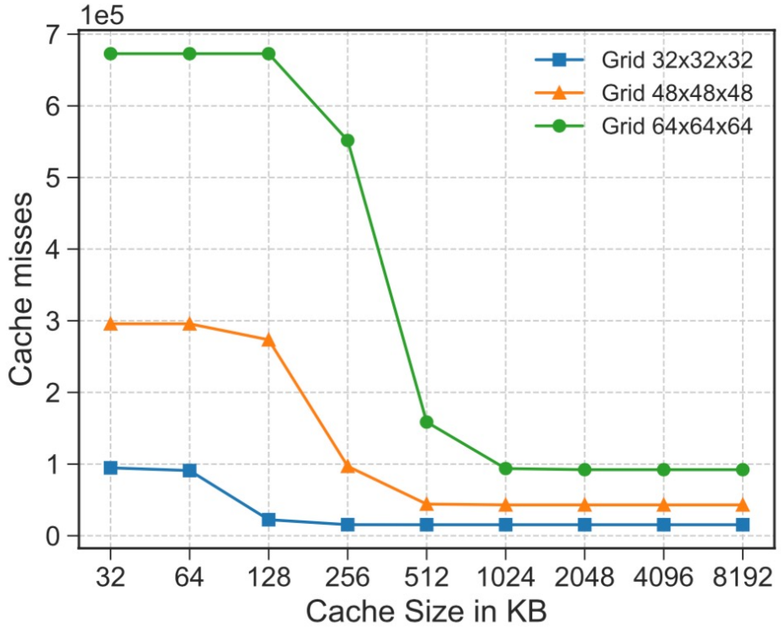




[J. Li, X. Wang, A. Tumeo, B. Williams, J. Leidel, Y. Chen, “PIMS: A Lightweight Processing-in-Memory Accelerator for Stencil Computations”, MEMSYS 2019]

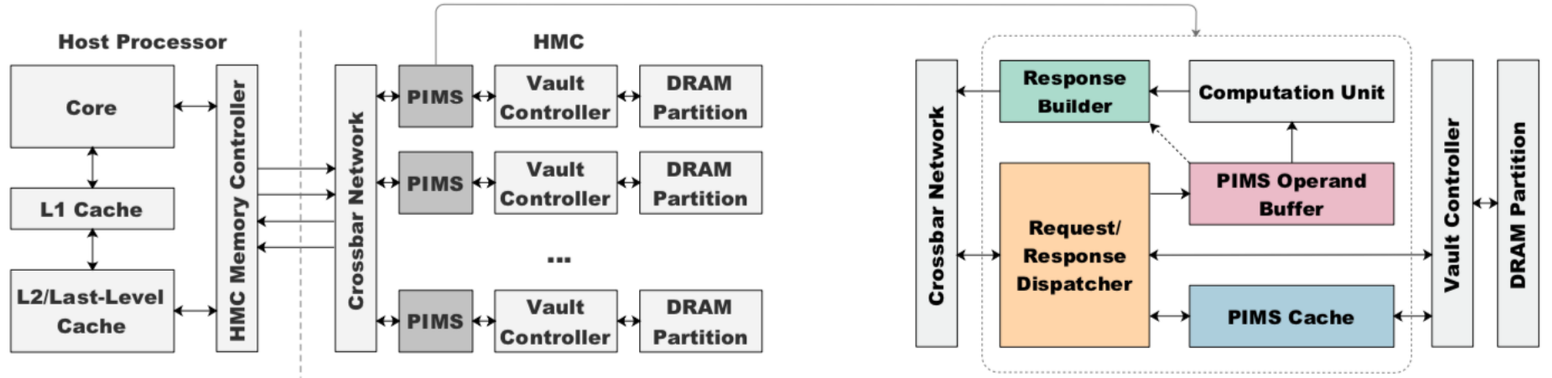
PIMS: Motivation

```
double *data-container, ***grid_a, ***grid_b, c[4];

/* Initialize data containers and coefficients */
data_a = malloc( sizeof(double) * (dim_x+6)*(dim_y+6)*(dim_z+6) );
data_b = malloc( sizeof(double) * (dim_x+6)*(dim_y+6)*(dim_z+6) );
init(data_a, data_b, c);

/* Generate contiguous layout of addresses */
grid_a = malloc( sizeof(double **) * (dim_x + 6) );
grid_b = malloc( sizeof(double **) * (dim_x + 6) );

/* Initialize stencil grids */
for(i = 0; i < (dim_x + 6); i++) {
   grid_a[i] = malloc( sizeof(double *) * (dim_y + 6) );
   grid_b[i] = malloc( sizeof(double *) * (dim_y + 6) );

   for(j = 0; j < (dim_y + 6); j++) {
      grid_a[i][j] = data_a+ (dim_z + 6) * ( j + (dim_y + 6) * i);
      grid_b[i][j] = data_b+ (dim_z + 6) * ( j + (dim_y + 6) * i);
   }
}
/* Iterations */
for(t = 0; t < T; t ++) {
   for( i = 3; i < (dim_x + 3); i++ ) {
      for (j = 3; j < (dim_y + 3); j++) {
         for (k = 3; k < (dim_z + 3); k++) {
            grid_b[i][j][k] = c[0] * grid_a[i][j][k]
                        + c[1] * (grid_a[i-1][j][k] + grid_a[i+1][j][k]
                                + grid_a[i][j-1][k] + grid_a[i][j+1][k]
                                + grid_a[i][j][k-1] + grid_a[i][j][k+1])
                        + c[2] * (grid_a[i-2][j][k] + grid_a[i+2][j][k]
                                + grid_a[i][j-2][k] + grid_a[i][j+2][k]
                                + grid_a[i][j][k-2] + grid_a[i][j][k+2])
                        + c[3] * (grid_a[i-3][j][k] + grid_a[i+3][j][k]
                                + grid_a[i][j-3][k] + grid_a[i][j+3][k]
                                + grid_a[i][j][k-3] + grid_a[i][j][k+3])
         }
      }
   }
 swap(grid_a, grid_b); // swap grid pointers
}
```



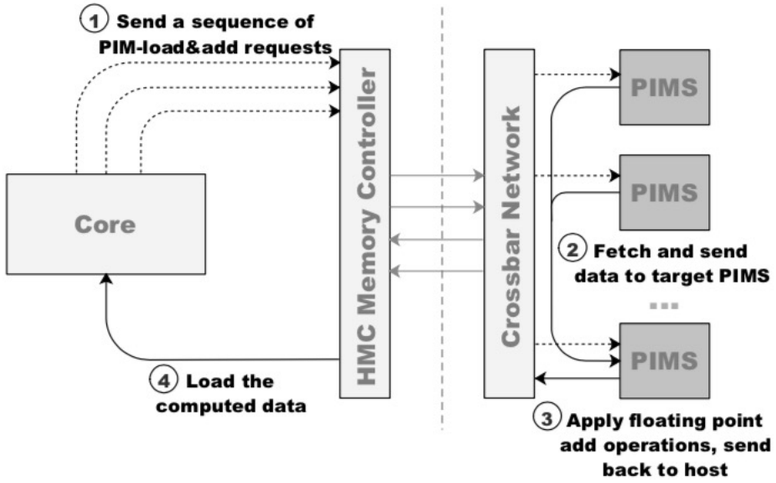
- Cache reuse varying cache size and size of the stencil grid

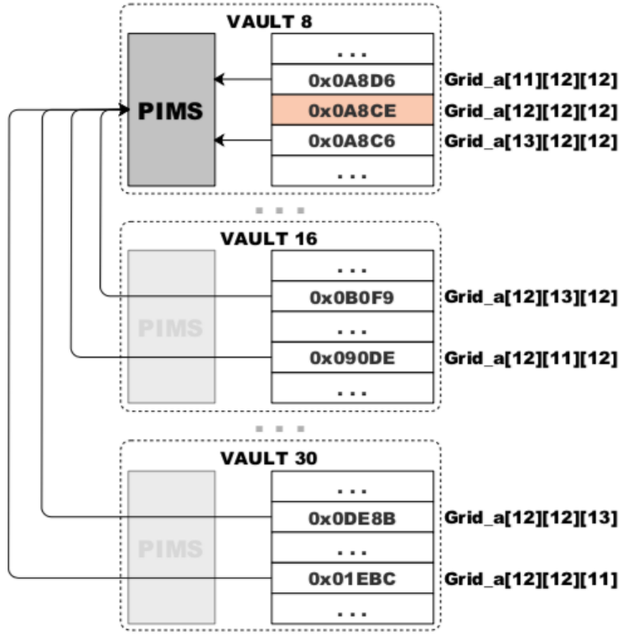
PIMS: Architecture



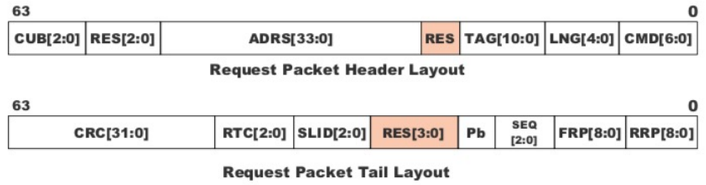
- HMC 2.1 specs
  - One accelerator unit per Vault
  - Implements a specific Custom Instruction with the related (lightweight) logic
  - ADD operations of the innermost stencil loop
- Programming Model
  - Data segmented depending if the processing is offloaded to the in-memory units
  - In-memory operations are expressed as specialized instructions of the host processor
- Communication
  - HMC packetized protocol allows accelerators to communicate each other directly
  - Accelerators do not issue requests, only responses (from other accelerators)
- No in-memory address translation
- No cache coherence between in-memory cache and host processor cache

PIMS: Employing HMC packet-based protocol



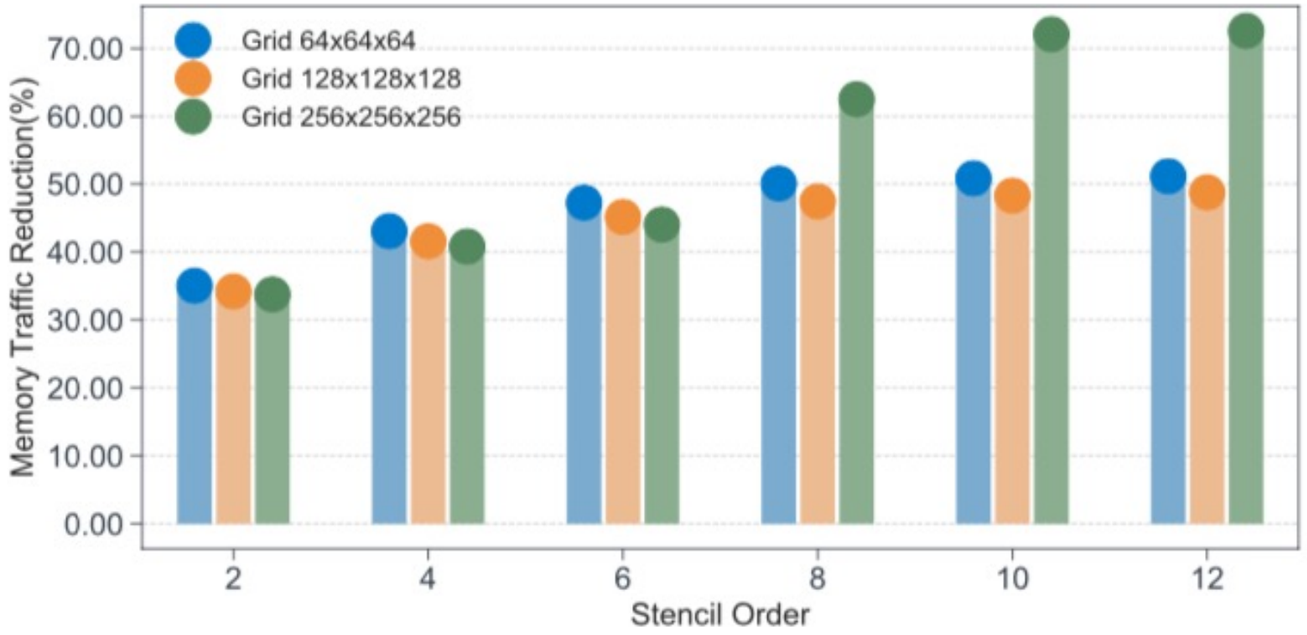


- Example for a 2-order, 7-point stencil
- Data of neighbors needs to be sent to the vault of the central point
- Destination ID needs to be present in the packet before being flushed into HMC
  - 32 vaults = 5 bits
  - 1 bit in header, 4 bits in the tail



PIMS: Main Results

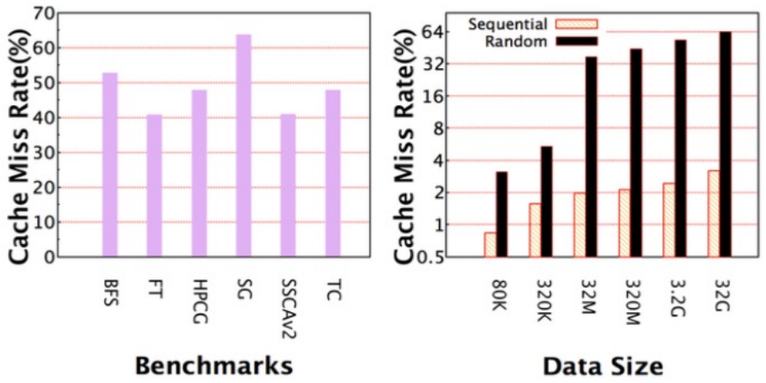
- Memory Traffic Reduction



- Key result: significant reduction of traffic between host processors and memory for stencil operations
  - Now performed in memory
- Increases bandwidth efficiency
- Use of a memory side cache also reduces the number of bank conflicts

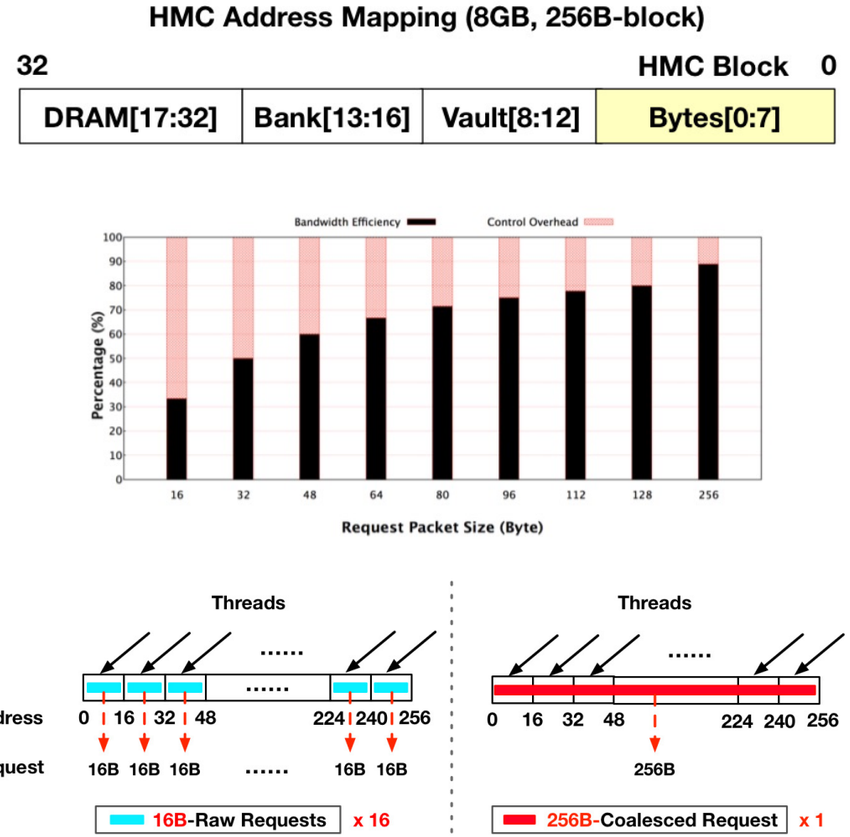
MAC: Memory Access Coalescer for 3D-Stacked Memory

- Miss Rate Analysis

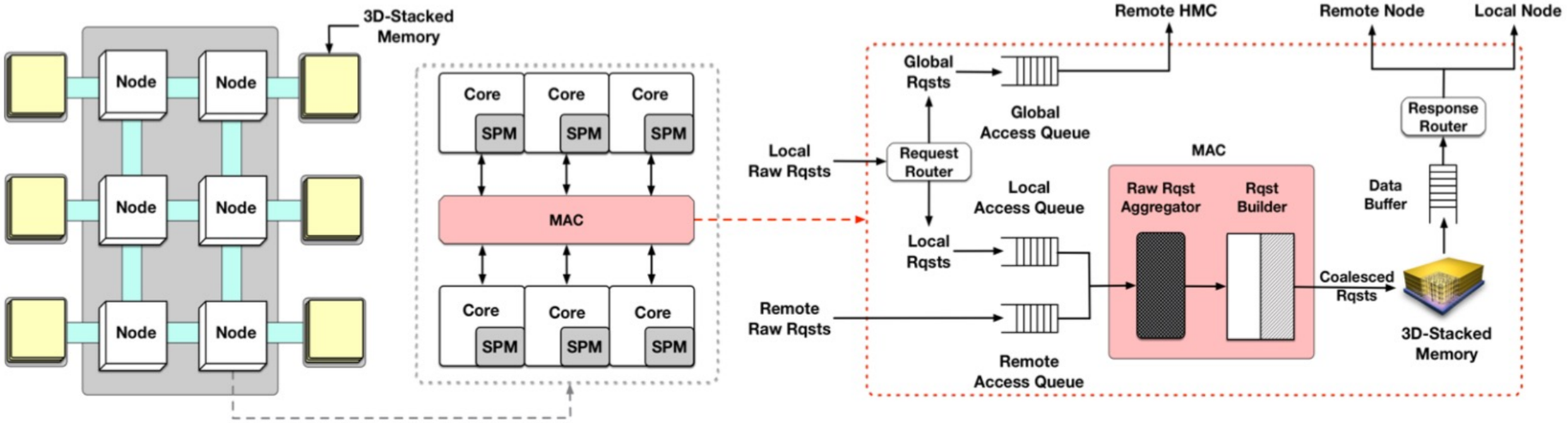


[Xi Wang, Antonino Tumeo, John D. Leidel, Jie Li, Yong Chen: MAC: Memory Access Coalescer for 3D-Stacked Memory. ICPP 2019: 2:1-2:10]

- HMC Address Mapping and Efficiency



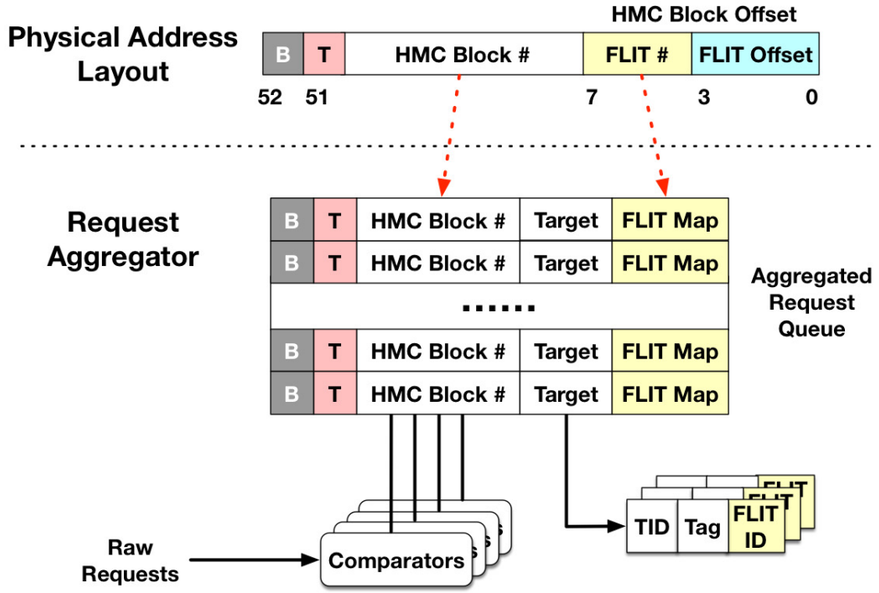
MAC: Architecture

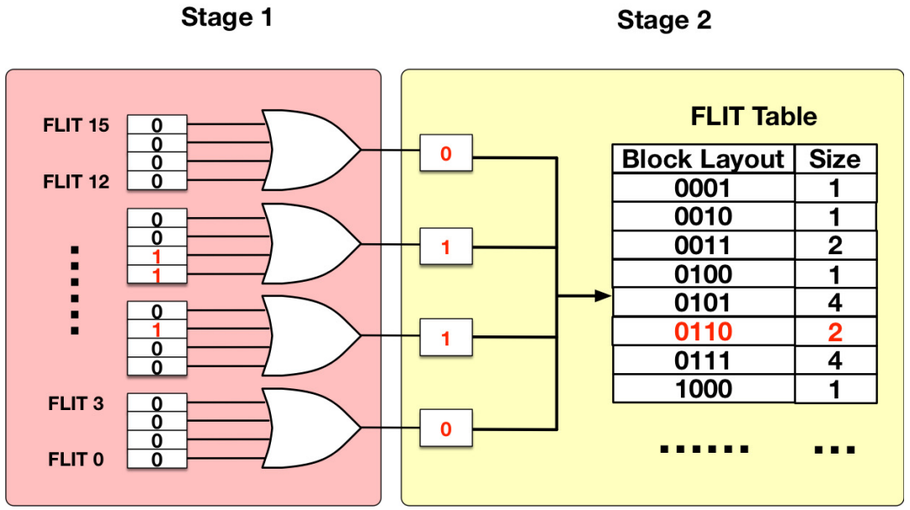


- Multinode manycore design
  - No caches, scratchpad memory - inspired by GoblinCore 64, P38
  - Each node, directly attached to an HMC stack

MAC: Design

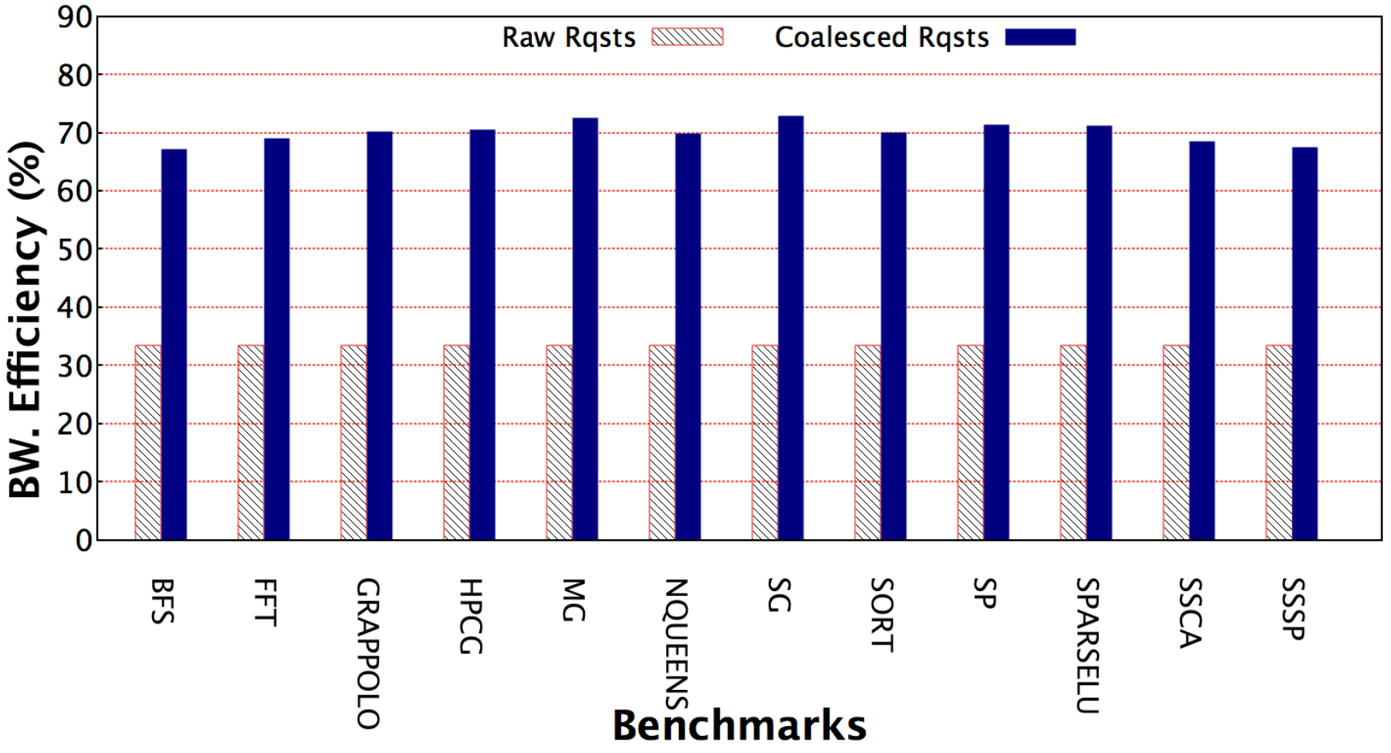
- MAC is composed of two components:
  - *Raw Request Aggregator:* responsible for grouping the small raw requests depending on request types and physical addresses
  - *HMC Request Builder*. The HMC Request Builder prepares the actual HMC transactions, adaptively adjusting their size depending on the amount of mergeable memory operations and latency tolerance afforded.





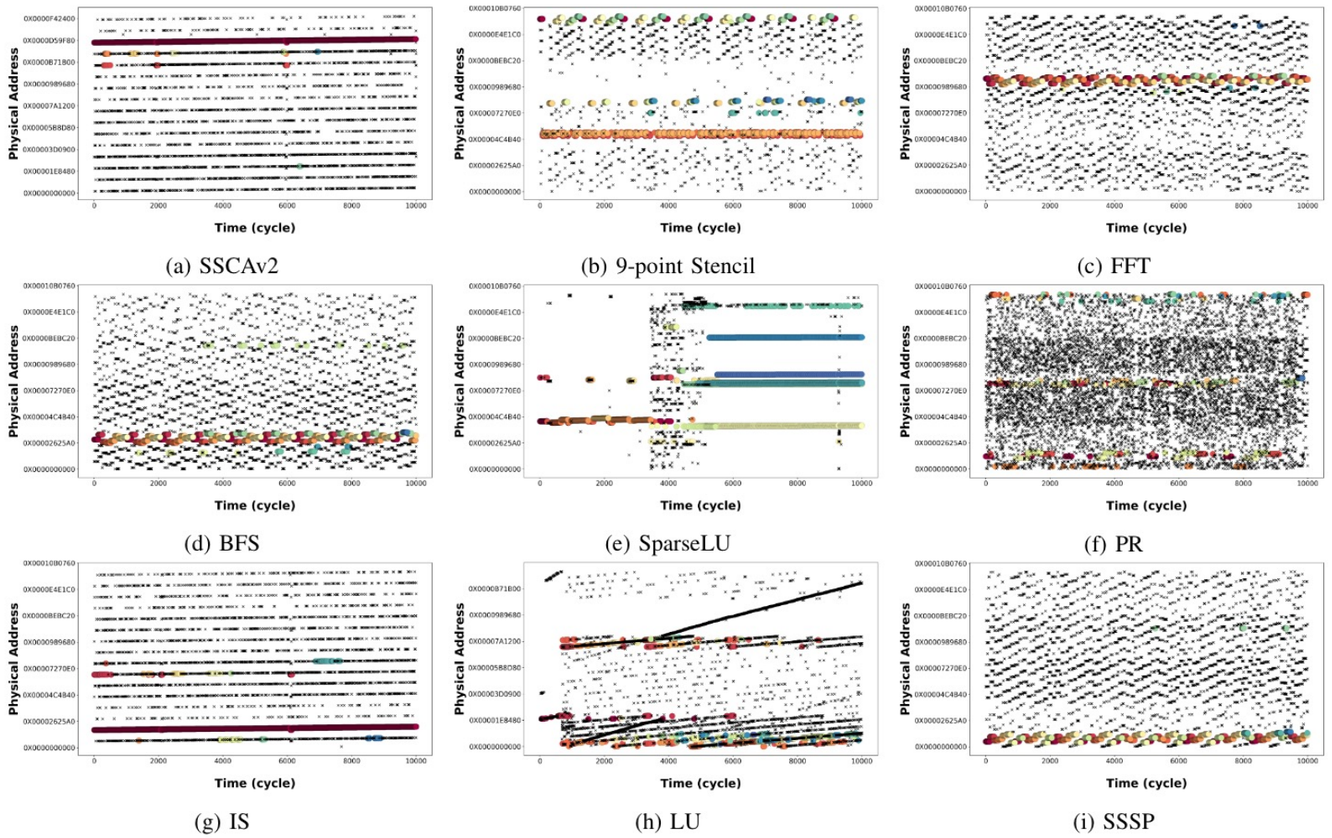
MAC: Main Results

- Bandwidth Efficiency

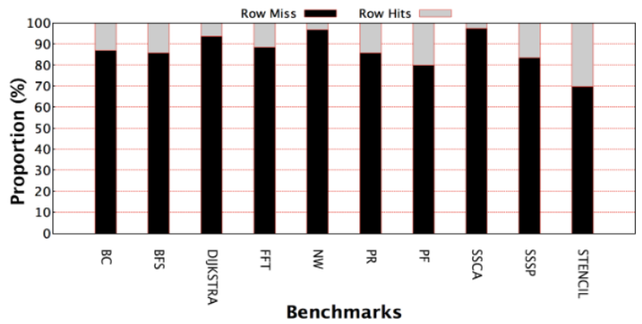


- Key result: increase in memory bandwidth utilization with fine grained data accesses
  - More data, fewer headers/tails
- Reduces bank conflicts
- Provides saving in energy (fewer transactions)
- Overall speed up of memory operations

HAM: Hotspot Aware Manager for 3D-Stacked Memory



Row-buffer hits and misses

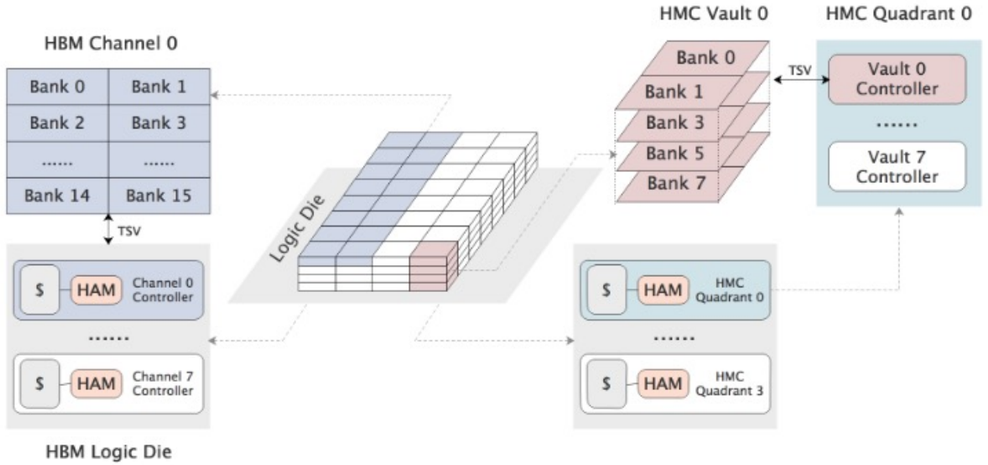


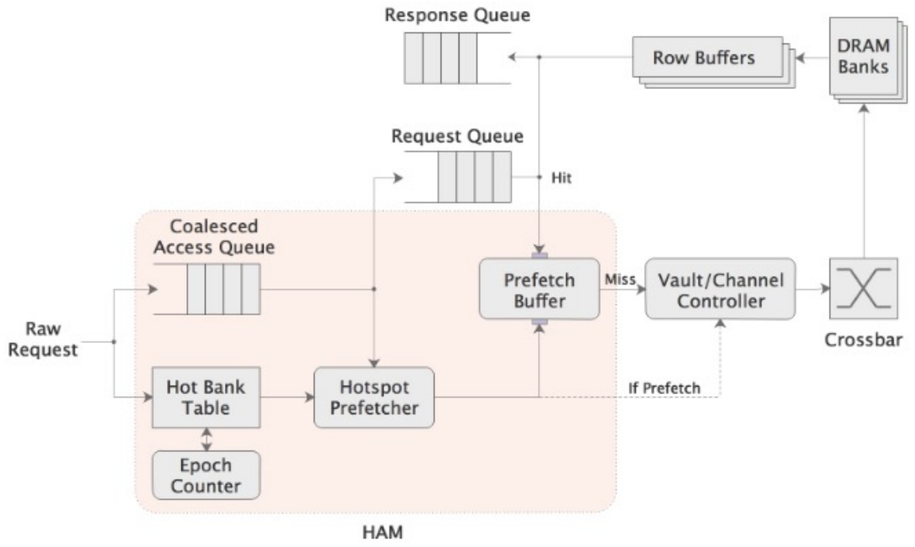
[X. Wang, A. Tumeo, J. D. Leidel, J. Li, Y. Chen: HAM: Hotspot-Aware Manager for Improving Communications With 3D-Stacked Memory. IEEE Trans. Computers 70(6): 833-848 (2021)]

- Motivation: we want to reduce memory hotspots for Irregular Applications on 3D Stacked Memory

HAM: Architecture

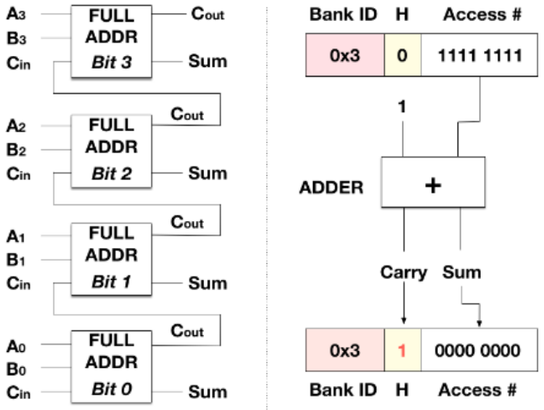
- HAM is a “near memory” component, i.e., implemented in logic layer of a stacked memory, that:
  - implements a coalescing mechanism (extending MAC)
  - Implements a “hot bank” table to identify hotspots
  - Implements a hotspot prefetcher that preloads data of “hot” memory location into memory - side buffers
  - supports a dynamic page policy



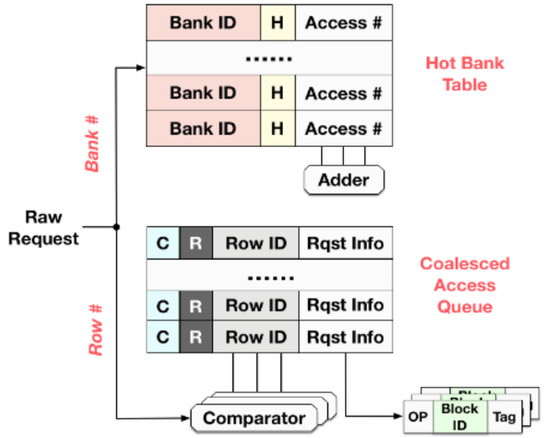


HAM: Design

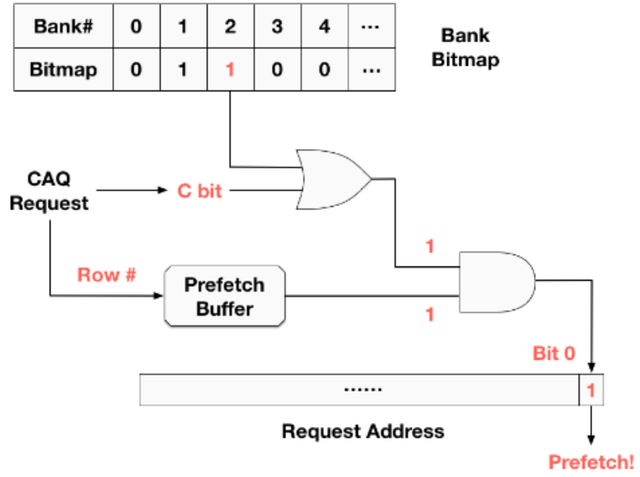
- Coalesced Access Queue (CAQ),
- Hot Bank Table (HBT)
- Hotspot Prefetcher
- Prefetch Buffer



(b) Carry-Driven Update



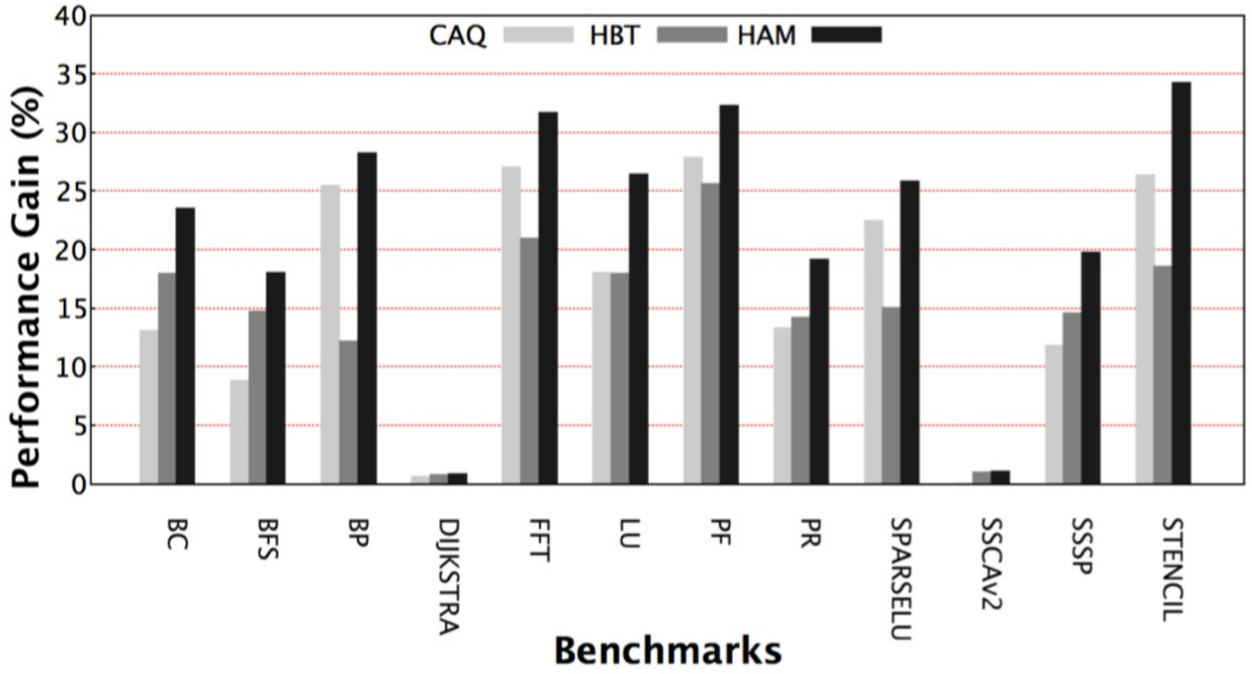
(a) CAQ and HBT Design



(c) Hotspot Prefetcher

HAM: Main Results

- Performance Improvement



- Coalescing and hotspot detection with prefetching provide improvements up to 30% in performance
- Applications with repetitive access patterns gets the best benefits
- Energy reduction and bank conflicts reduction as a consequence

Pacific Northwest
NATIONAL LABORATORY

These modifications are all ISA independent

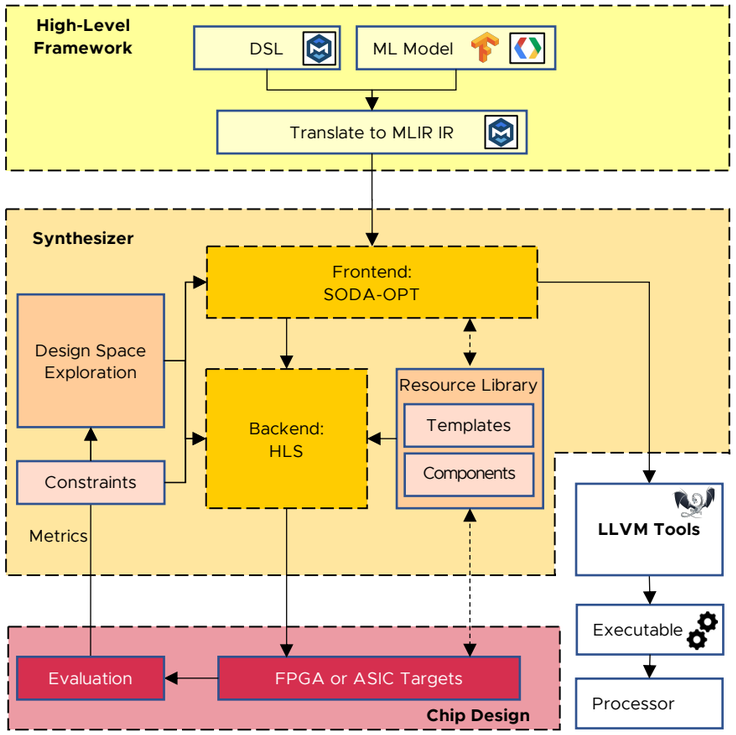
- They manly focus on enabling new research at the memory interface
- They would be possible exploiting other closed ISAs
- Other types of exploration, however, would be more difficult:
  - Extended address space
  - Closely vs loosely coupled accelerators and extensions
  - Multithreaded designs…

Pacific Northwest NATIONAL LABORATORY

ARM HPC Design Characteristics

- Mostly focusing on many efficient cores
- Mostly looking at reducing latencies rather than tolerating them
- Scalable Vector Extensions (SVE) focusing on (dense linear algebra)
  - Support for efficient synchronization and scatter-gather
- Solutions like A64FX have a better flop/memory bandwidth ratio than other designs
  - Provide high-performance with less regular workloads (e.g., HPCG)
- Different process for providing custom instructions
  - Looking more at providing custom Intellectual Property Blocks and accelerators
- But a more mature ecosystem in terms of available libraries, compilers, system software
  - European Project Initiative take: use ARM host core with RISC-V accelerators

SODA Synthesizer: Overview



[M. Minutoli, V. G. Castellana, C. Tan, J. Manzano, V. Amatya, A. Tumeo, D. Brooks, G-Y. Wei: SODA: a New Synthesis Infrastructure for Agile Hardware Design of Machine Learning Accelerators. ICCAD 2020: 98:1-98:7]

[J. Zhang, N. Bohm Agostini, S. Song, C. Tan, A. Limaye, V. Amatya, J. Manzano, M. Minutoli, V. G. Castellana, A. Tumeo, G-Y. Wei, D. Brooks: Towards Automatic and Agile AI/ML Accelerator Design with End-to-End Synthesis. ASAP 2021: 218-225]

- A modular, multi-level, interoperable, extensible, **open-source hardware compiler** from **high-level programming frameworks to silicon**
- Compiler-based frontend, leveraging the MultiLevel Intermediate Representation (MLIR)
- **Compiler-based backend**, leveraging state-of-the-art High-Level Synthesis (HLS) techniques, as well as a Coarse-Grained Reconfigurable Array (CGRA) generator
- Generates **synthesizable Verilog** for a variety of targets, from Field Programmable Gate Arrays (FPGAs) to Application Specific Integrated Circuits (ASICs)
- Optimizations at all levels are performed as **compiler optimization** passes

[N. Bohm Agostini, S. Curzel, J. Zhang, A. Limaye, C. Tan, V. Amatya, M. Minutoli, V.G. Castellana, J. Manzano, A. Tumeo: Bridging Python to Silicon: The SODA Toolchain. IEEE Micro Magazine, to appear.]

Pacific Northwest NATIONAL LABORATORY

Conclusions

- Introduced architectural challenges for irregular workloads
- Described ongoing architectural exploration using RISC-V
  - Focusing on data-intensive applications leveraging 3D-Stacked memory
- Discussed qualitatively the tradeoffs in using a commercial ISA like ARM