Are Task-Based Programming Models Suitable for Processing in Memory (PIM) Clusters?

David Krasowska

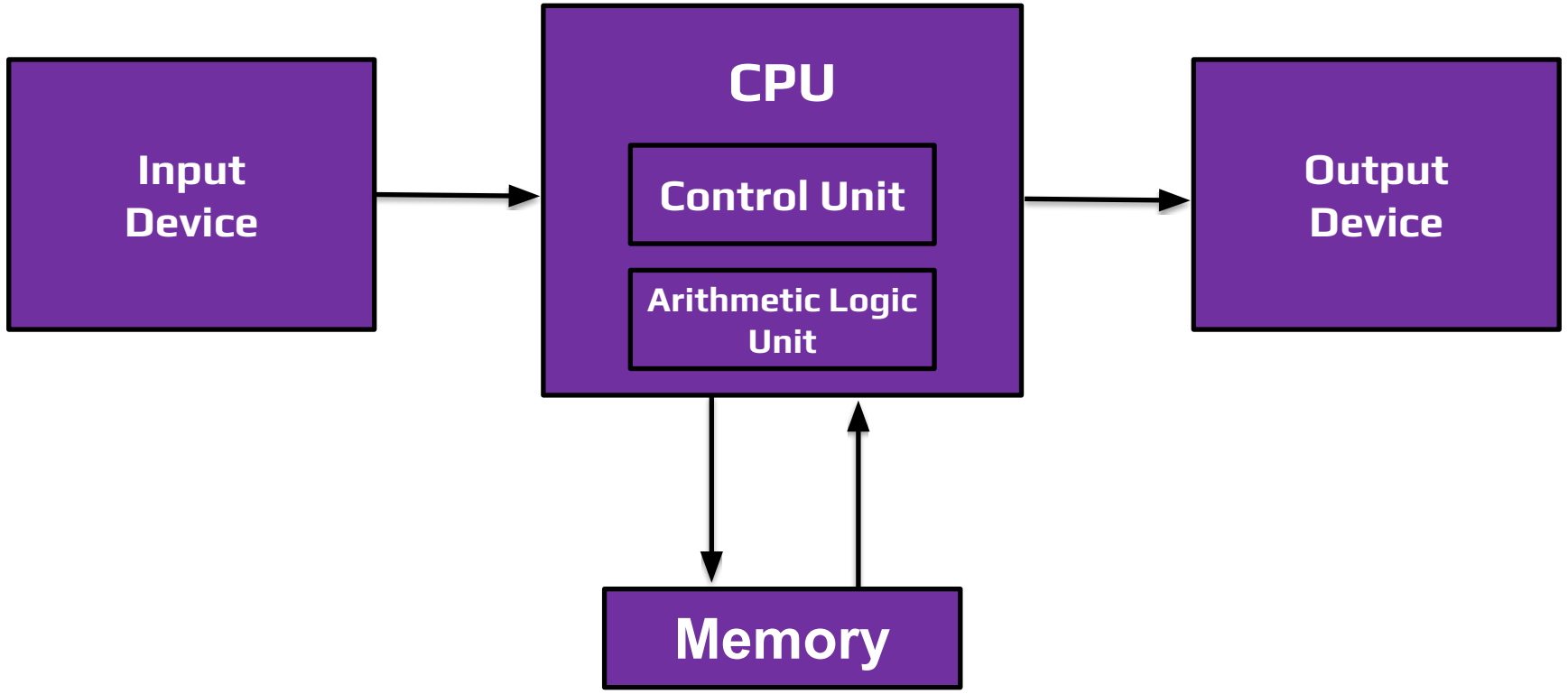
Legion Retreat 12/04/2024

Los Alamos NATIONAL LABORATORY EST. 1943

Plab Northwestern

The data movement bottleneck **… sucks**

Compute Centric Architecture



Processing in Memory (PIM)

Lower Latency: Data is processed in/near memory, reducing retrieval time

Efficient Design: Smaller, simpler processors with local memory reduce complexity

High Parallelism: Large number of processors enables simultaneous data processing

Ideal for Data-Intensive Workloads: AI, genomics, databases, HPC

Different types of PIM solutions

DRAM DIMM (this talk)

- Speed up CPU computation

High-bandwidth memory (HBM)

- Speed up GPU and FPGA computation

Compute Express Link (CXL)

- Large in memory filtering

Different types of PIM solutions

DRAM DIMI

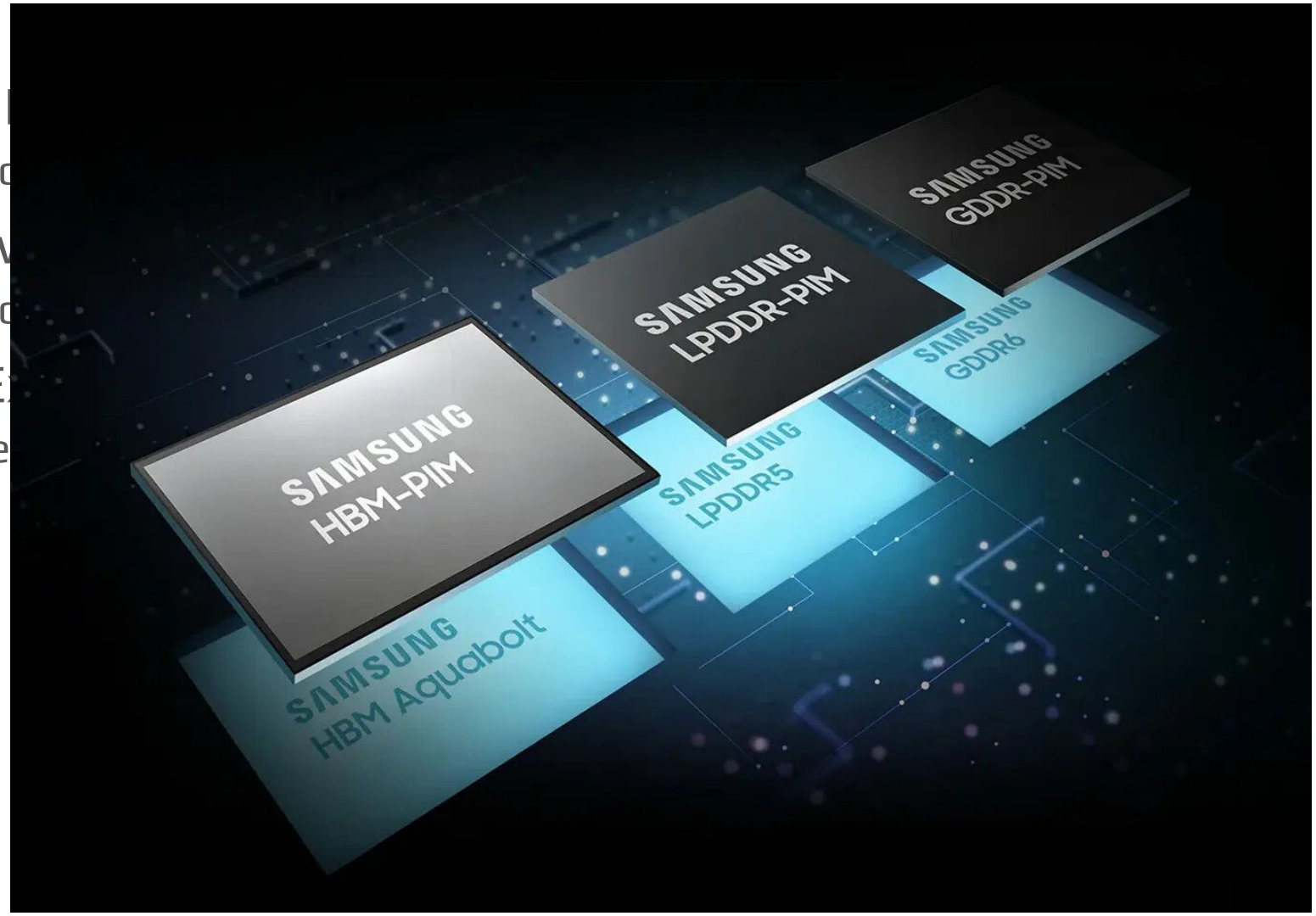
- Speed

High-bandv

- Speed

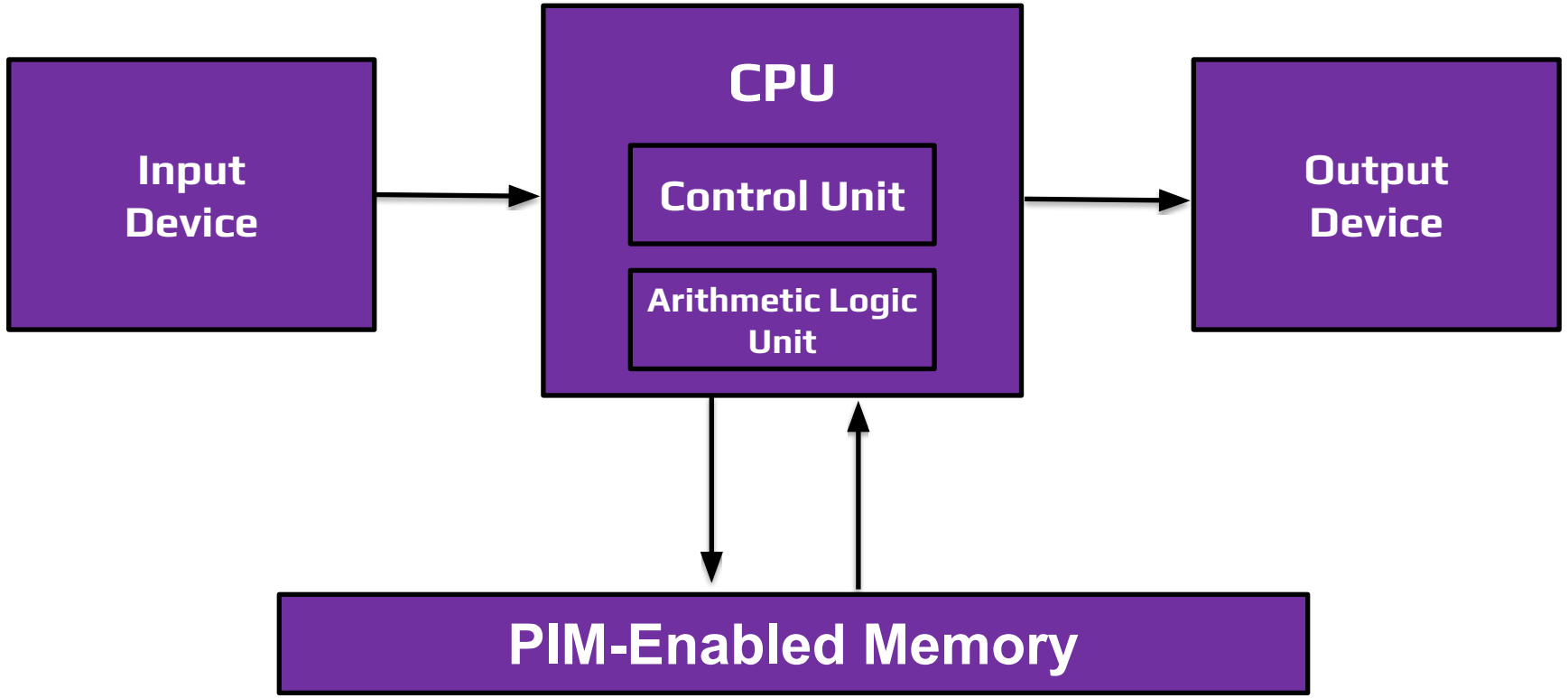
Compute E

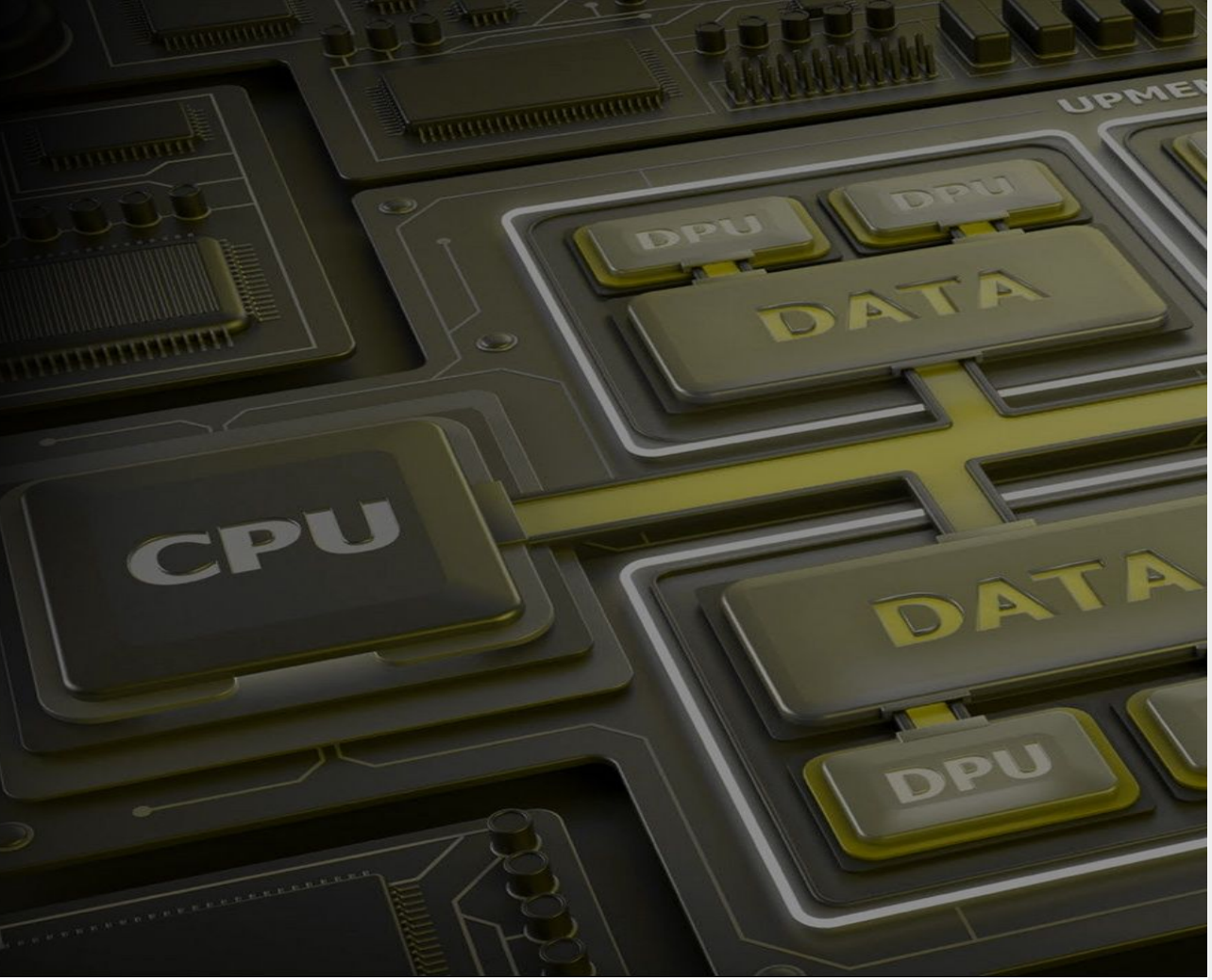
- Large



https://samsung.com

New view of the machine



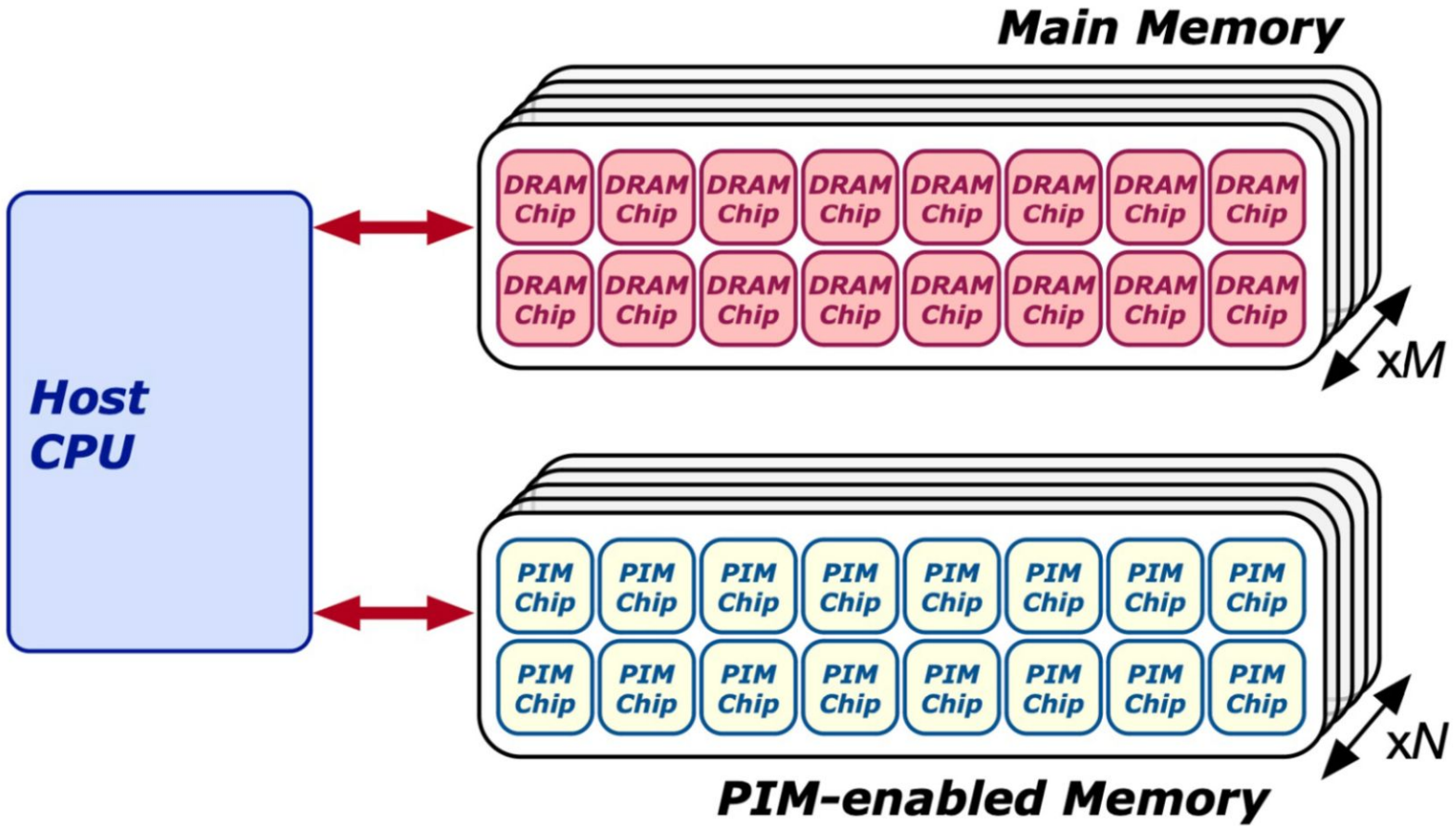


Hardware configuration

- **UPMEM PIM DIMM** with 128 DPUs.
- Each DPU has 24 threads supporting 32-bit RISC instructions (64-bit capable).
- Single-socket server supports 2,560 DPUs (64 per rank) with 256GB PIM-DRAM.

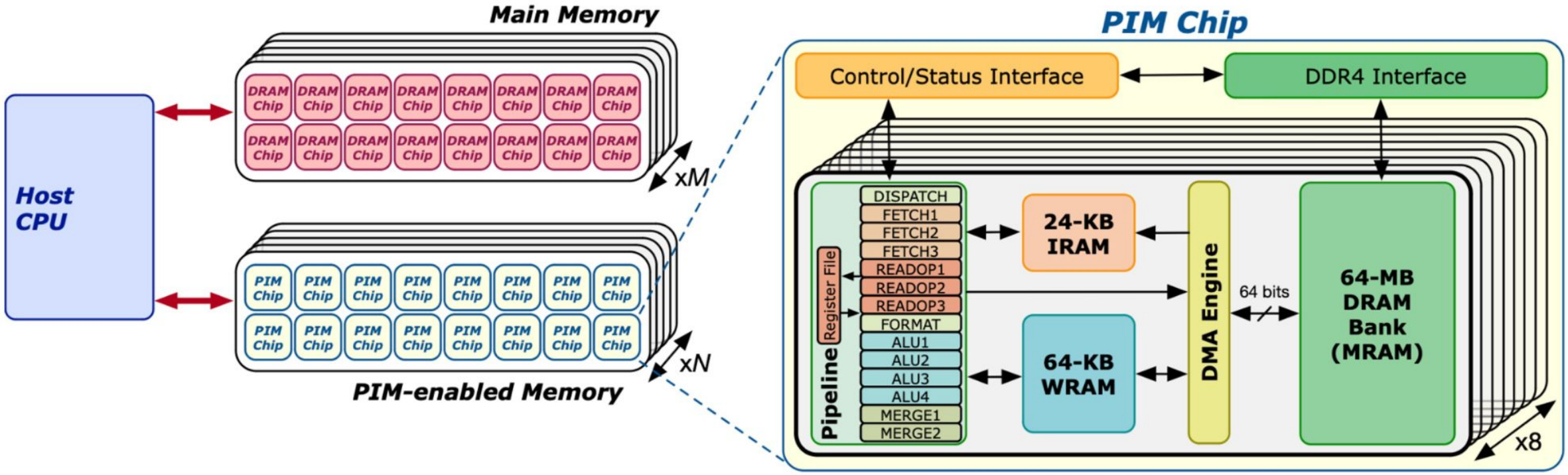
https://www.upmem.com

What is the UPMEM architecture?



https://safari.ethz.ch/

What is the UPMEM architecture?



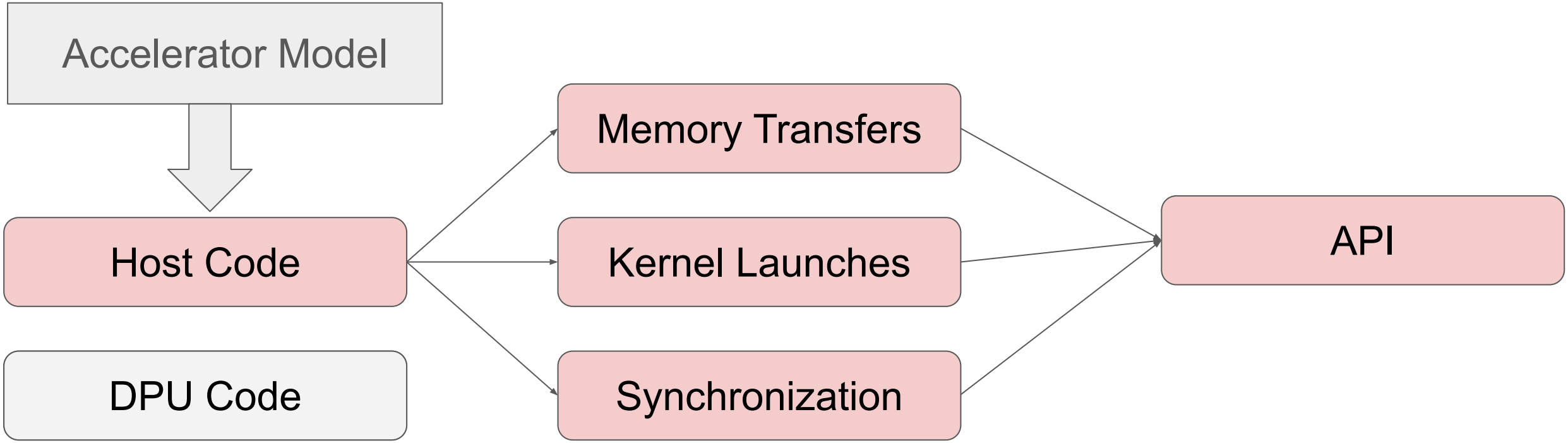
https://safari.ethz.ch/

Limitations of the UPMEM Architecture

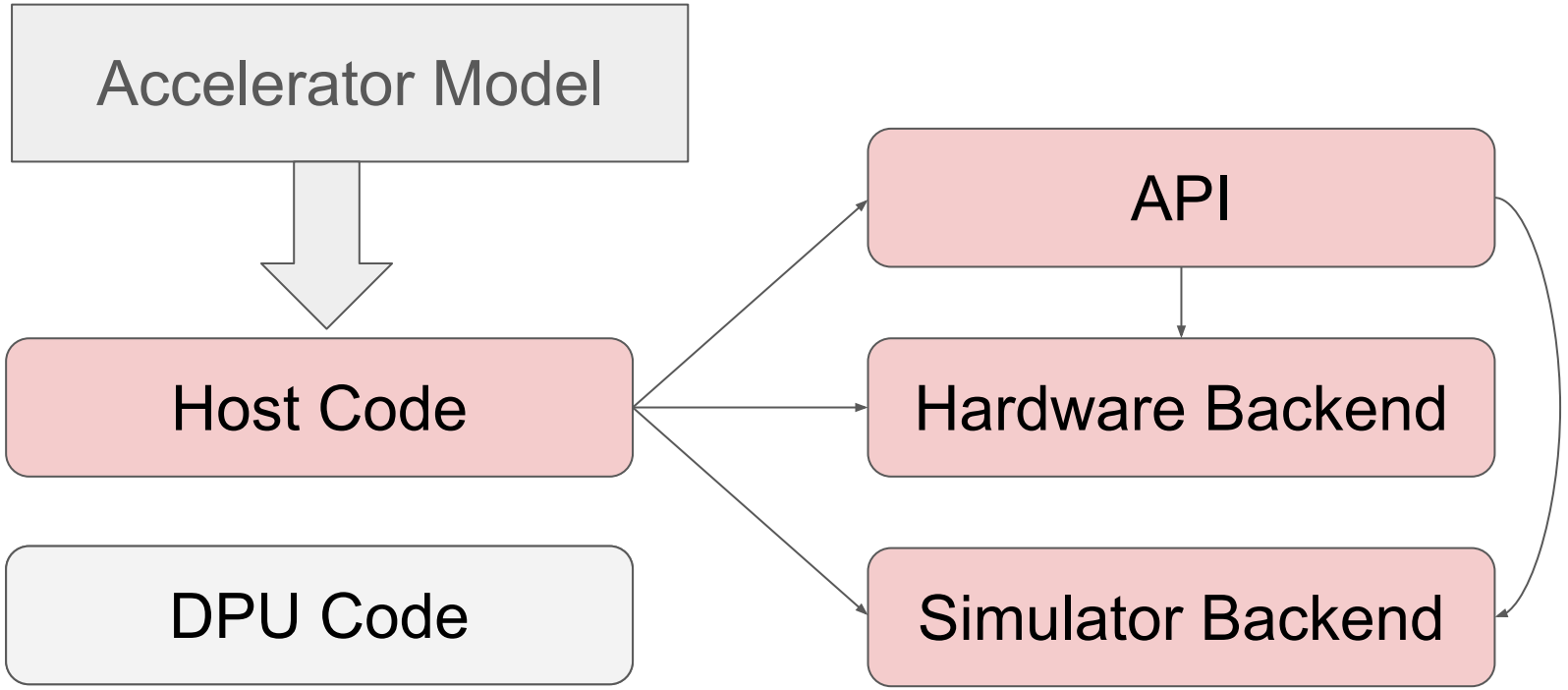
1) Lack of communication network across processors
2) No hardware floating point unit

How do you program with UPMEM?

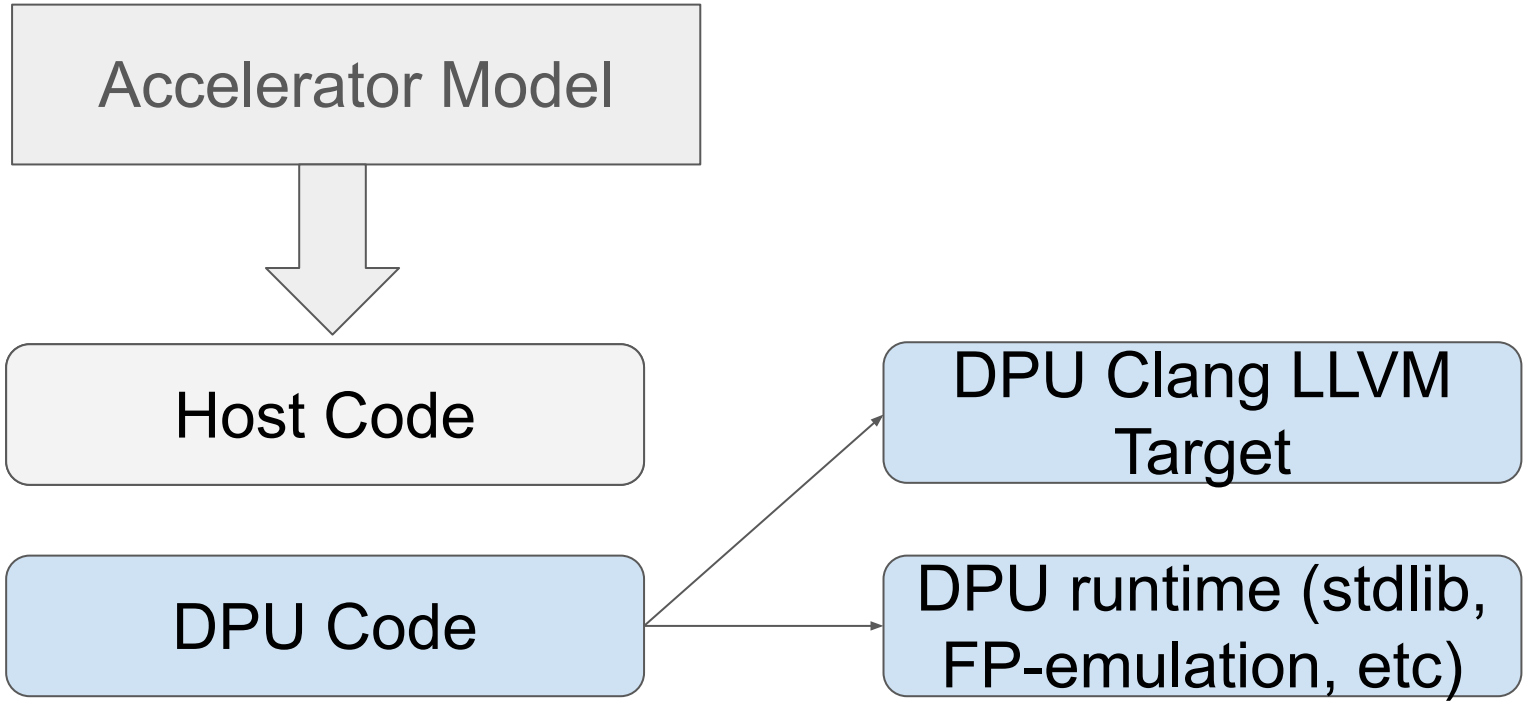
Model of UPMEM



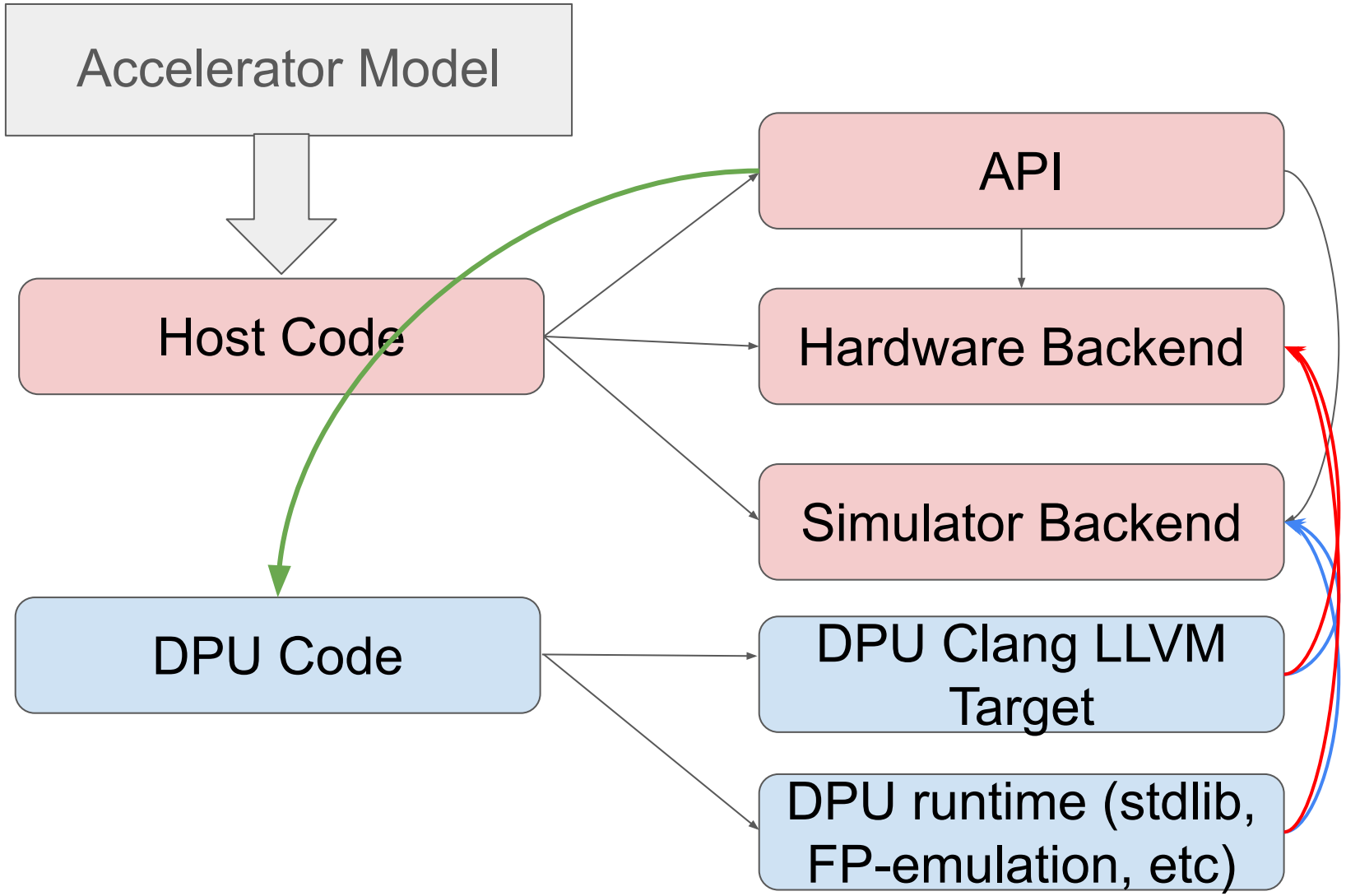
Model of UPMEM



Model of UPMEM



Model of UPMEM



How well does the Legion task model map onto processing in memory architectures?

"Achieving high performance and power efficiency on future architectures will require programming systems capable of reasoning about the structure of program data to facilitate efficient **placement** and **movement** of data"

https://legion.stanford.edu/overview/

Legion utilizes large task graphs to **minimize** kernel invocation and communication latencies

Support UPMEM in the Realm backend

Processor::DPU_PROC

- Each DPU is a registered processor in Realm

Memory::DPUMRAMMemory

- MRAM is registered memory that has affinity to the host RAM

UPMEM provides an asynchronous function callback interface for event synchronization (memory transfers and kernel execution)

Mapper interface will allow us to shard the indexspace amongst the DPUs

Current status (as this a work in progress)

The good:

- UPMEM DPUs are controlled by Legion/Realm host code
  - Event interface handles completion notifications for forward progress within the application (fences etc)
  - Launch kernels + transfer data based on Legion dependency graphs
- Ability to use Legion/Realm notions of data on the UPMEM device

The bad:

- Unable to scale past 64 DPUs due to an address mapping bug
- Complex collective communication patterns haven’t been fully tested
  - (reductions, all-to-all, etc)
- Larger applications need to be ported to evaluate Legion-PIM performance benefits

Related work: Simple-PIM

State-of-the-art programming model for UPMEM PIM

Model abstraction supporting any dimension and size of array

- Host code and device code
- Collective communication across devices

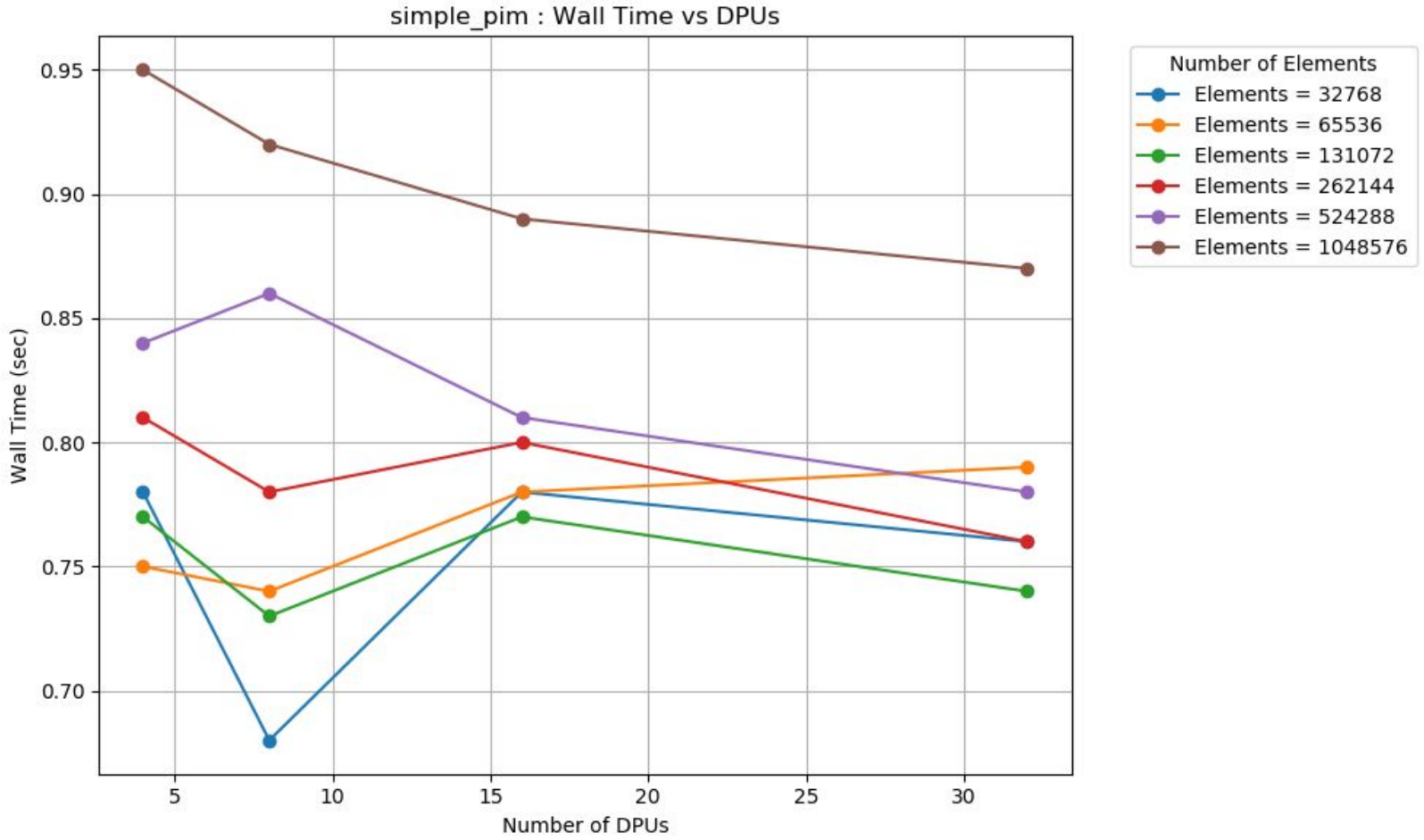
Optimizations to the DPU device code

- Operator strength reductions
- Loop unrolling
- Avoiding boundary checks

Jinfan Chen, Juan Gómez-Luna, Izzat El Hajj, Yuxin Guo and Onur Mutlu, "SimplePIM: A Software Framework for Productive and Efficient In-Memory Processing" , International Conference on Parallel Architectures and Compilation Techniques (PACT), 2023.

Best current state-of-the art (Simple-PIM)

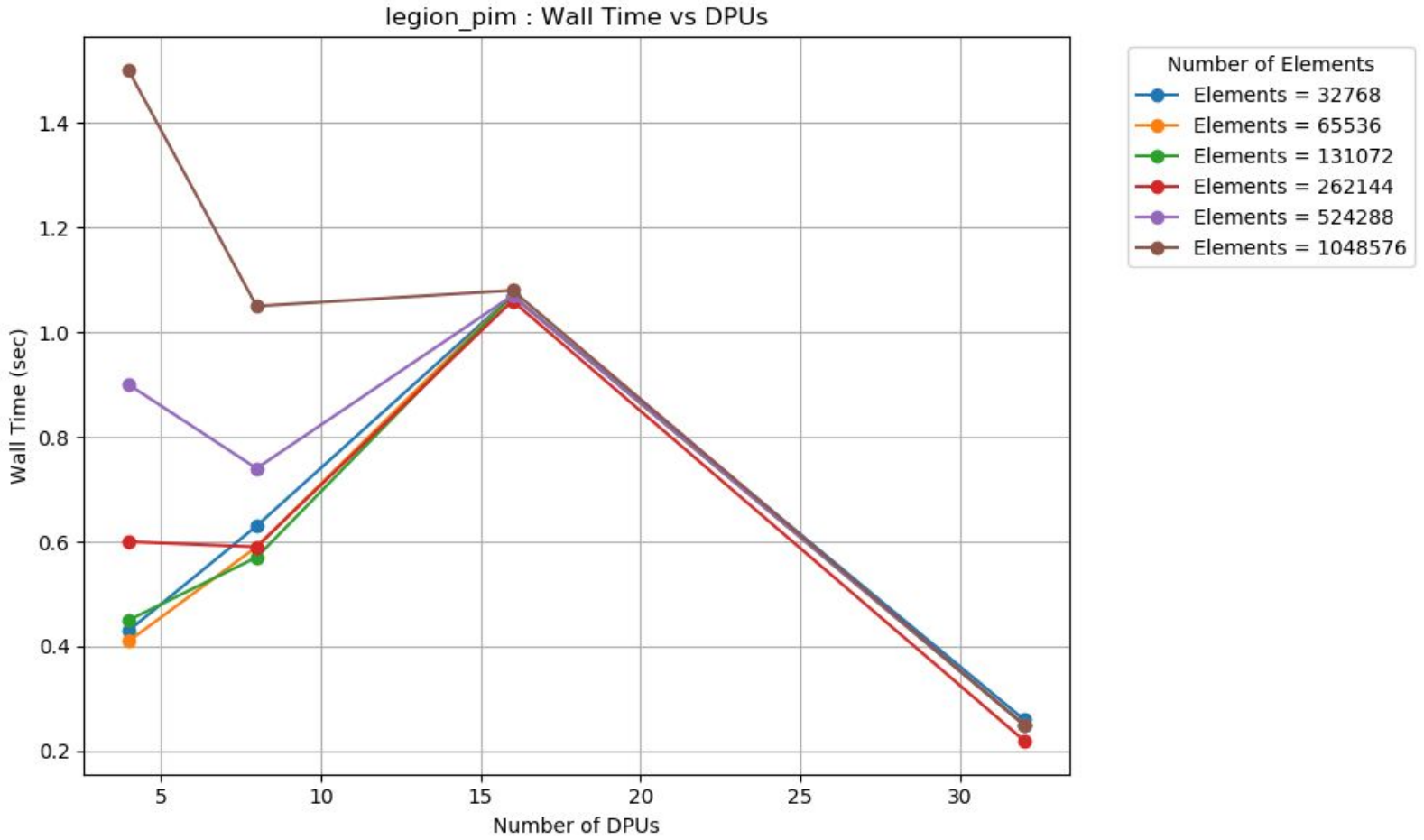
averaged over n=20 trials for DAXBY



https://github.com/CMU-SAFARI/SimplePIM

Legion-PIM (this work)

averaged over n=20 trials for DAXBY



https://github.com/PrescienceLab/upmem-legion

Experience

DPU-Clang is not like NVCC

- No ability to link a static library like librealm or liblegion without major engineering

Cannot adequately build libc++ from forked LLVM provided by UPMEM

Larger runtime overhead due to granularity of the processor

- 128 DPU processors per DIMM managed by Realm

Physical non-global address space for the device

- Each DPU has memory from 0x0MB to 0x64MB

Future Work

On simple benchmarks, achieving scaling across the full 2560 DPUs

Port more complex benchmarks with high degree of concurrency to compare against the prior state-of-the-art

Run multi-node simulator experiments combined with a regression model

- Lack of a multi-node UPMEM PIM cluster

Contact: krasow@u.northwestern.edu

https://krasow.dev