CoolPIM: Thermal-Aware Source Throttling for Efficient PIM Instruction Offloading

Lifeng Nai*† Ramyad Hadidi† He Xiao† Hyojong Kim†
Jaewoong Sim‡ Hyesoon Kim†

†Georgia Institute of Technology
* Google, ‡Intel Labs

IPDPS-32 | May 2018

Disclaimer: This work does not relate to Google/Intel Labs

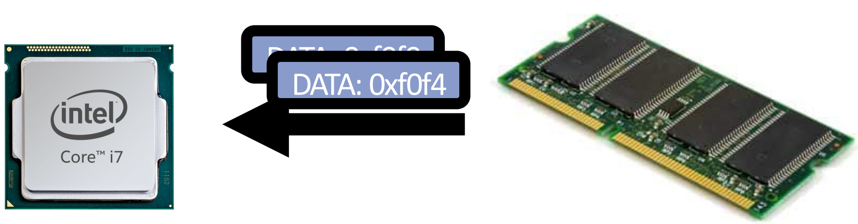
Processing-in-Memory

Processing-in-memory (PIM) is regaining attention for energy efficient computing

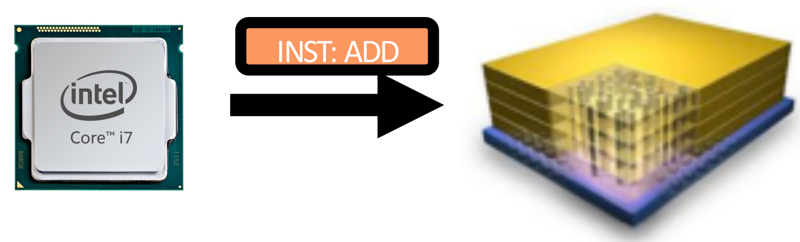
- Graph Workloads: Data-Intensive, Little Data Reuse

Basic Concept: Offload compute to memory

- Reduce *costly* energy consumption of data movement
- Enable using large internal memory bandwidth



Conventional Data Processing



Processing-in-Memory

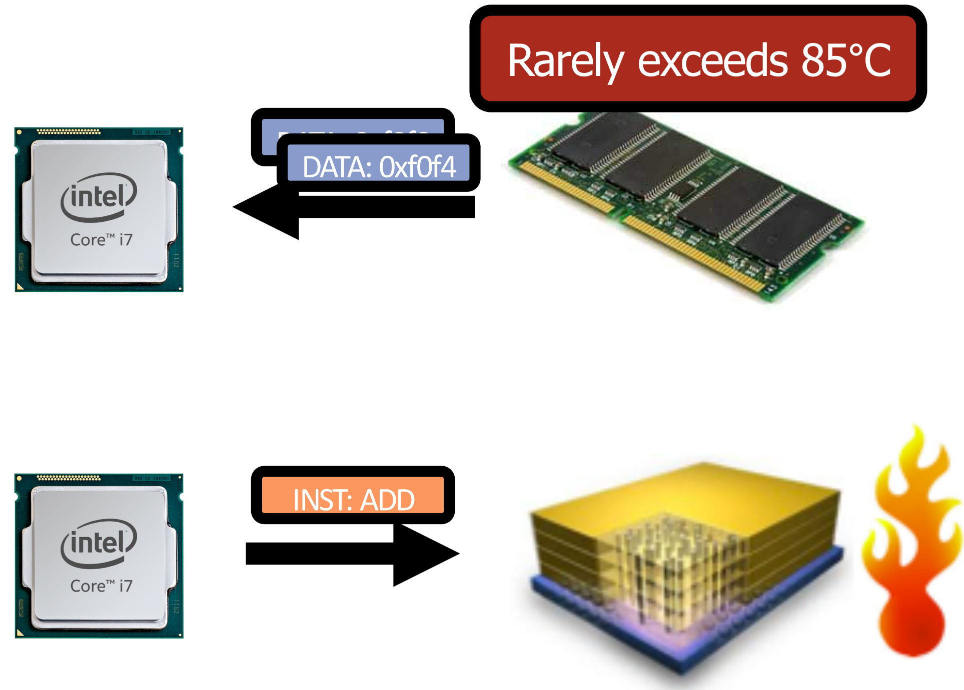
Thermal Challenge in PIM

PIM could increase memory temperature beyond normal operating temperature (85°C)

- High BW (hundreds of GBs ~ TBs) from 3D-stacked memory
- Less effective heat transfer compared to DIMMs
- **PIM would make these thermal problems worse!**

Too Hot Memory Stack?

- Slower processing for memory requests
- Decreasing overall system performance



CoolPIM keeps the memory "***Cool***"
to achieve better PIM performance

Outline

Introduction

Hybrid Memory Cube
- Background
- Thermal Measurements & Thermal Modeling of Future HMC

CoolPIM
- Software-Based Throttling
- Hardware-Based Throttling

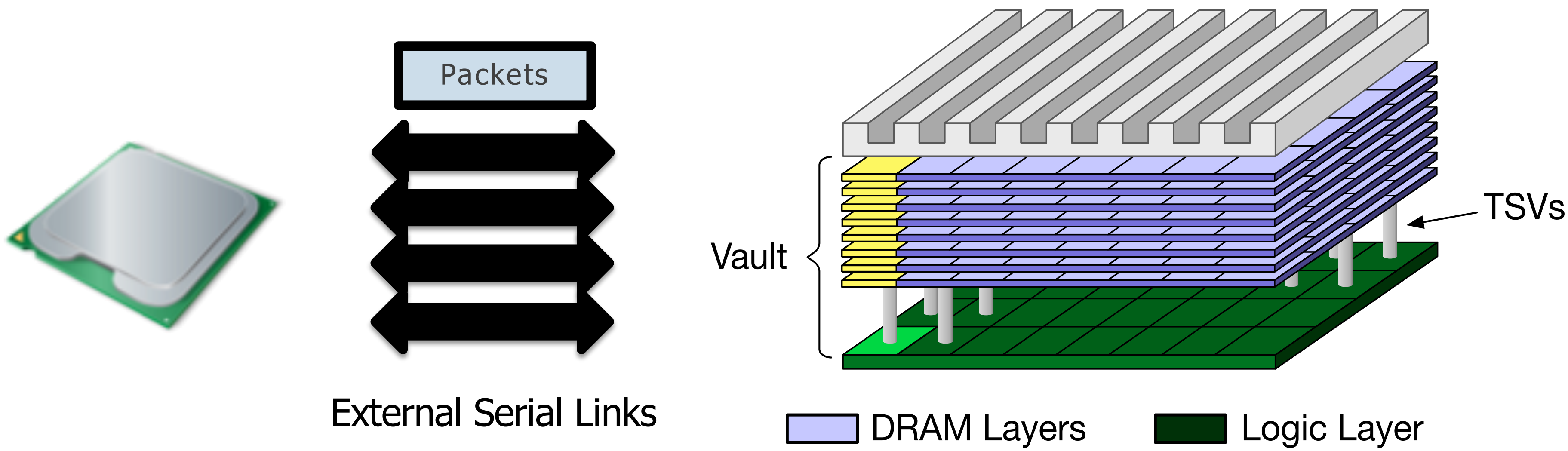
Evaluation

Conclusion

A Hybrid Memory Cube (HMC) from Micron

- Multiple 3D-stacked DRAM layers + one logic layer with TSVs
- Vaults: equivalent to memory channels
- Full-duplex serial links between the host and HMC

No PIM functionality for existing HMC products yet



Background

PIM Offloading in Future HMC

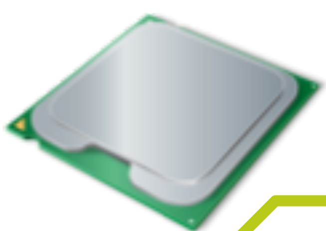
Instruction-level PIM supported in future HMC (HMC 2.0)

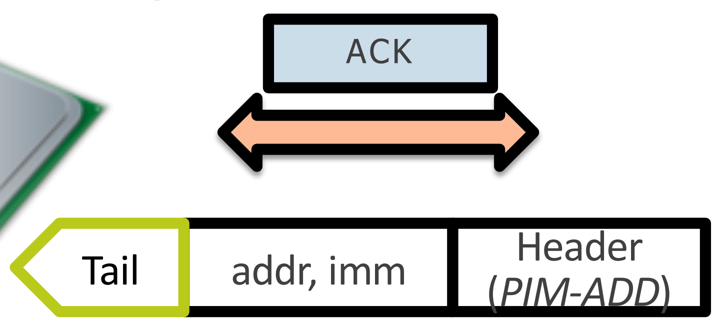
- Perform **Read-Modify-Write (RMW)** operations atomically
- Similar to READ/WRITE packets; just different **CMD** in the **Header**
- **No HMC 2.0 product yet!**

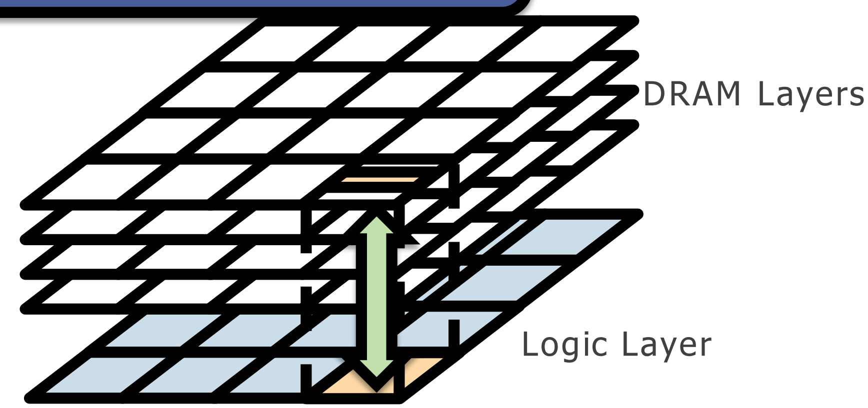
Type	HMC 2.0 PIM Instruction
Arithmetic	Signed Add
Bitwise	Swap, bit write
Boolean	AND/NAND/OR/NOR/XOR
	qual/greater

Q: Can we offload all the PIM operations to HMC?
What is the thermal impact of PIM in future HMC?

PIM-ADD (addr, imm)







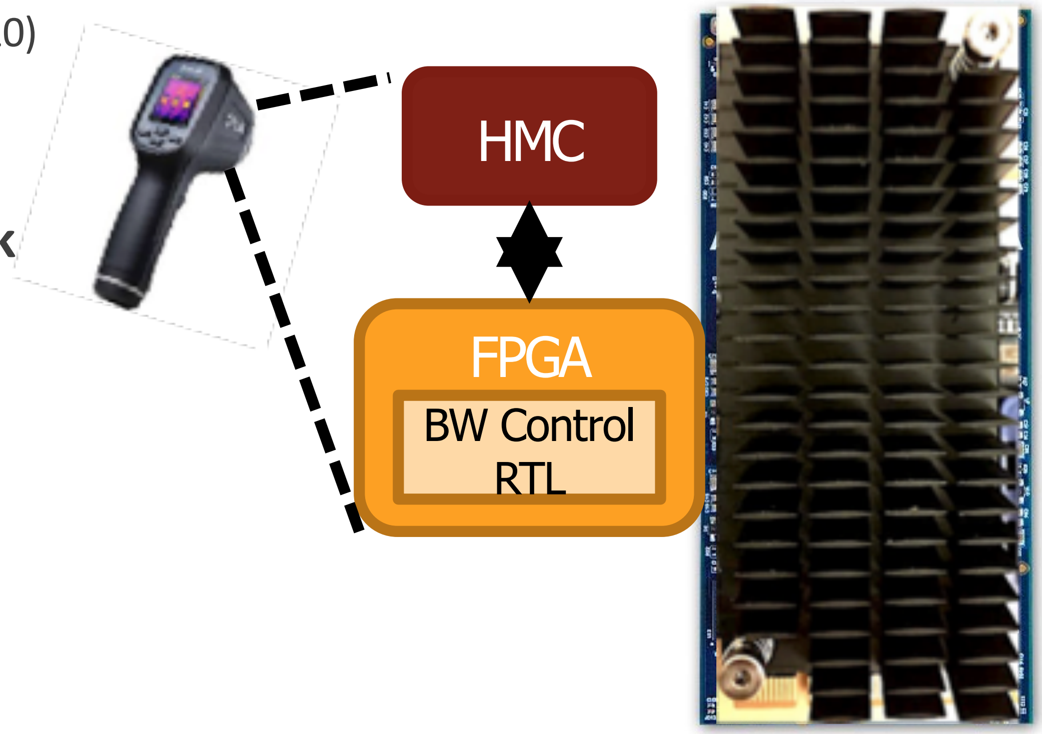
Existing HMC Thermal Measurement (1)

Experiment Platform (Pico SC-6 Mini System)

- Intel Core i7 + FPGA Compute Modules (AC-510)
  - AC-510: 4GB HMC 1.1, Kintex Ultrascale

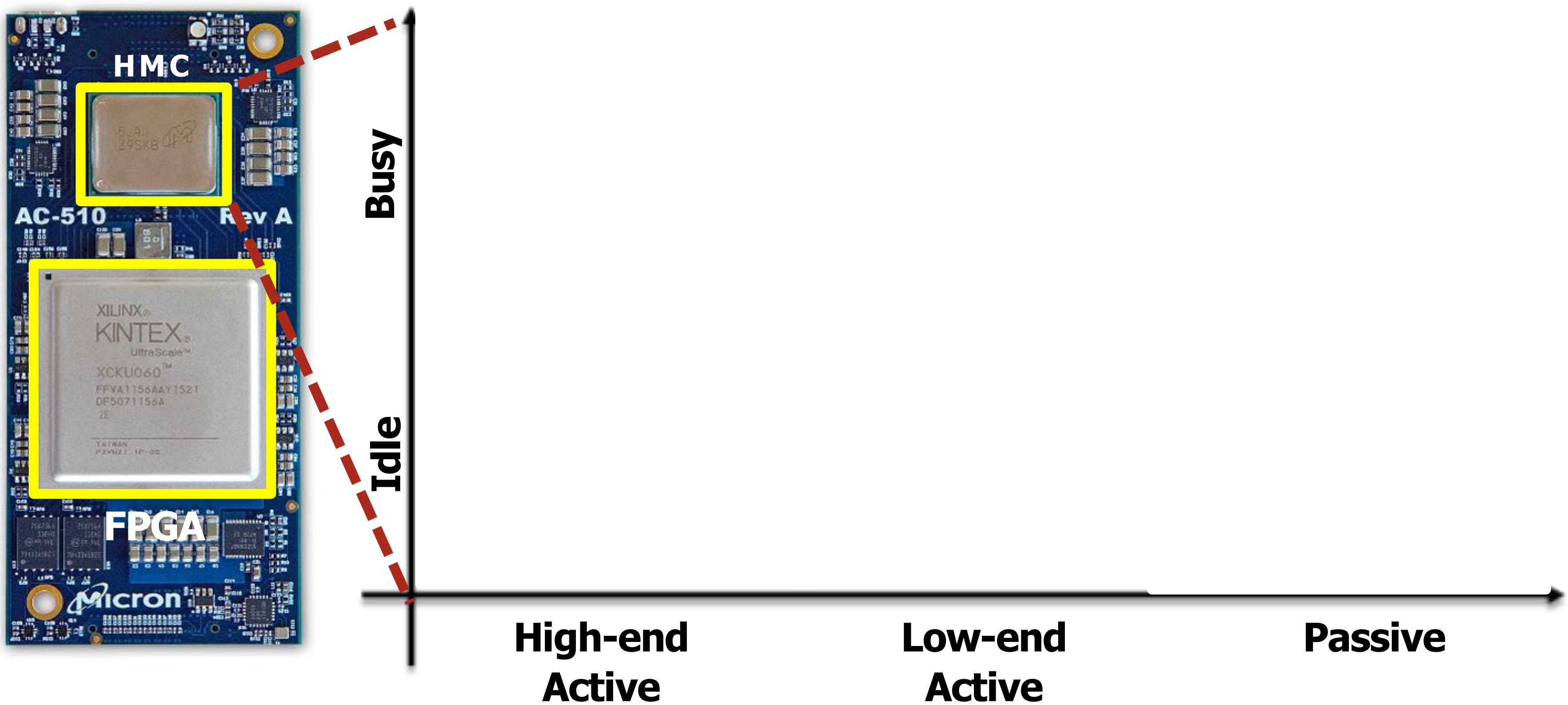
Measure the temperature on the heat sink

- Controlling memory BW via FPGA
- Applying three different cooling methods
  - High-End Active Heat Sink
  - Low-End Active Heat Sink
  - Passive Heat Sink



HMC 1.1 has no PIM functionality!

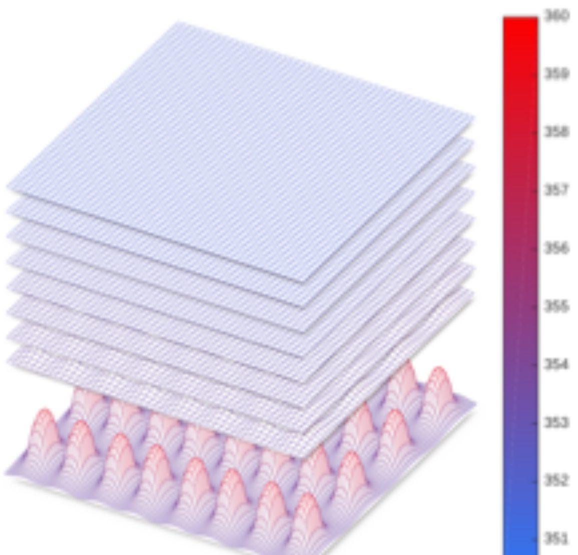
Existing HMC Thermal Measurement (2)

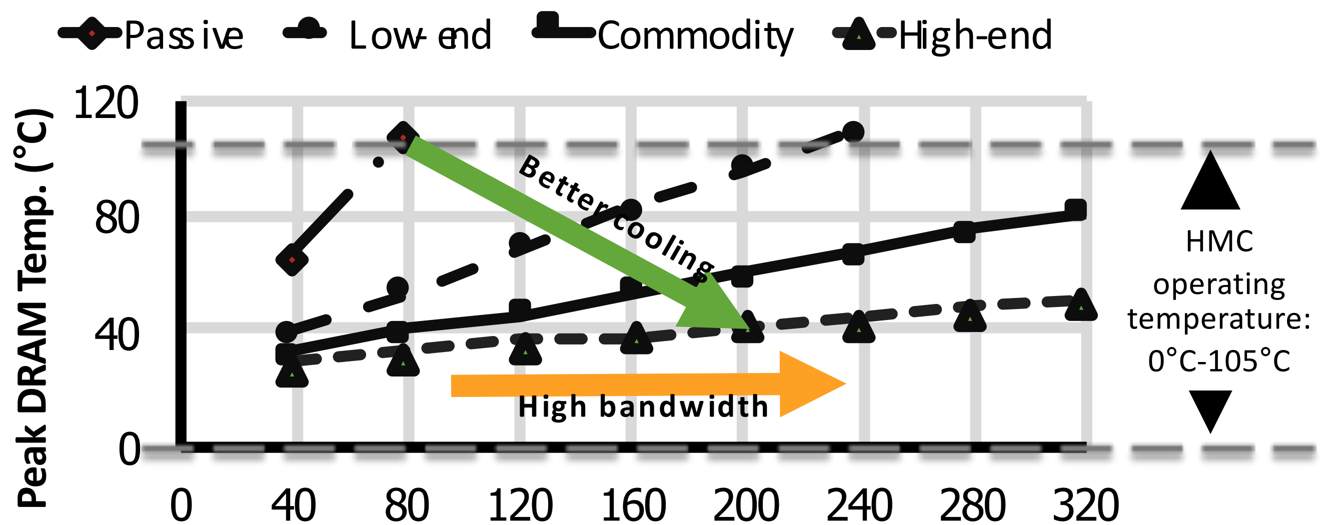


Future HMC Thermal Modeling

Thermal modeling for HMC 2.0 with commodity-server active cooling

- HMC 2.0 (w/o PIM) would reach 81°C at a full external BW (320GB/s)
  - We validated our thermal model against the measurements on HMC 1.1



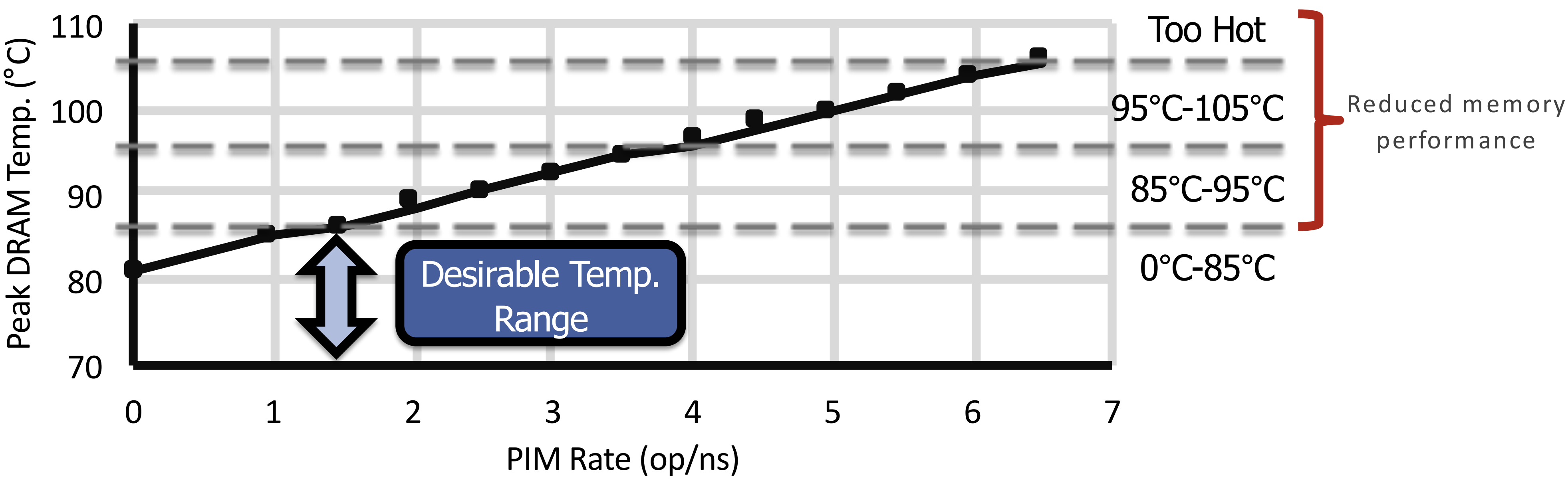


We need at least commodity-server cooling to benefit from PIM!

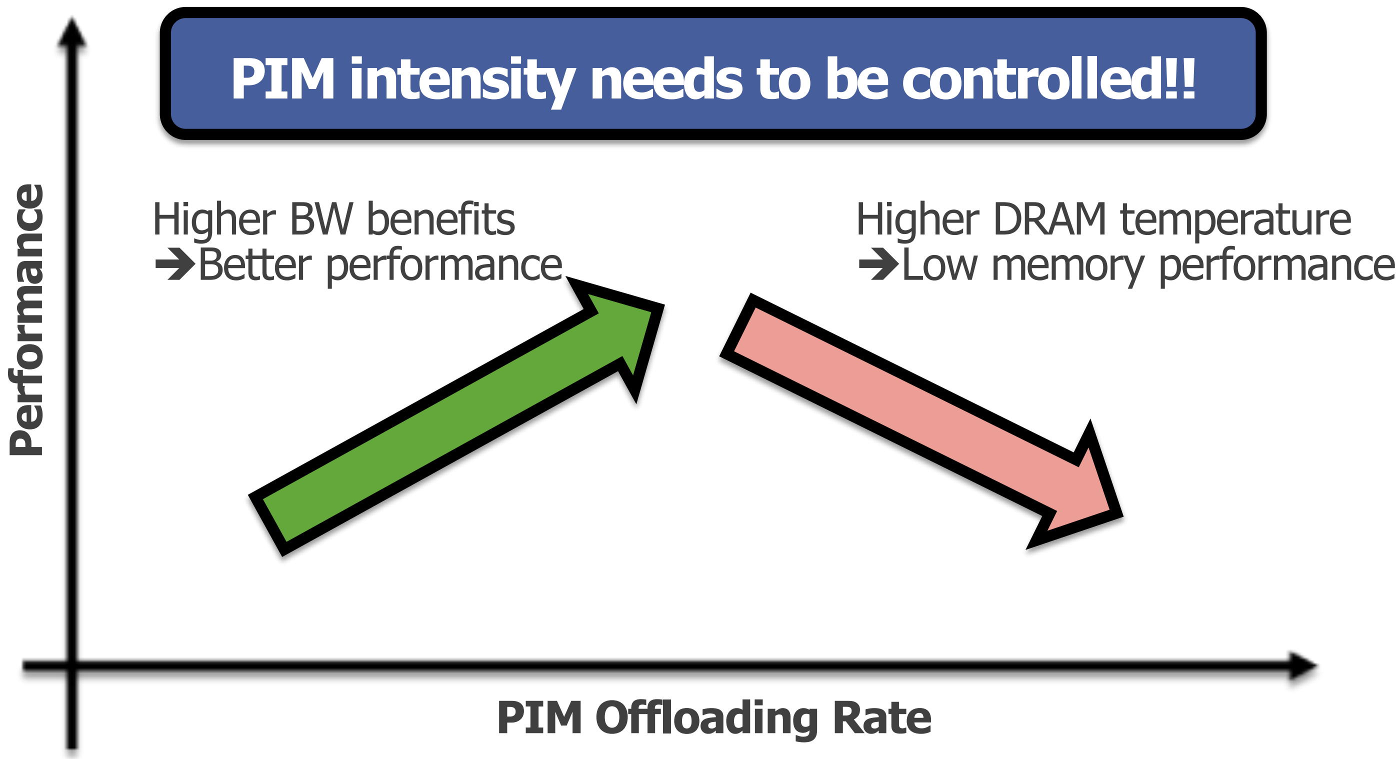
Thermal Impact with PIM in HMC 2.0

PIM increases memory temperature due to power consumption of logic and DRAM layers.

- In our modeling, the maximum PIM offloading rate is 6.5 PIM ops/ns
- A high offloading rate could reduce memory performance for cool down



Performance Trade-off of PIM



CoolPIM

Controls PIM Intensity with Thermal Consideration

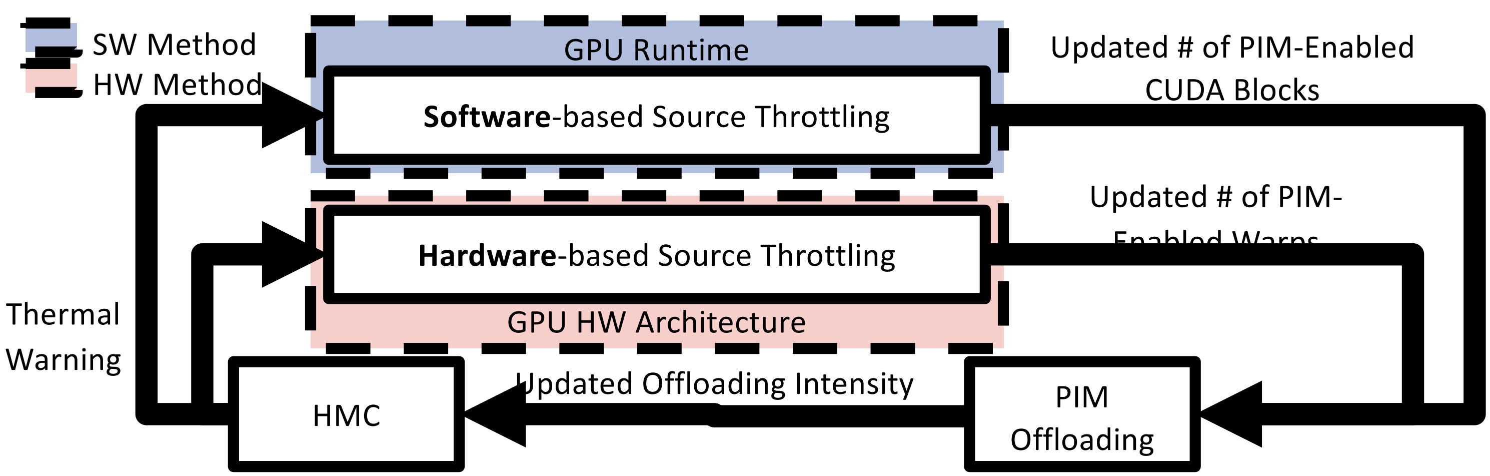
CoolPIM: Overview

We propose two methods for GPU/HMC

1) A SW mechanism with no hardware changes
2) A HW mechanism with changes in GPU architectures

Dynamic source throttling based on thermal warning messages from HMC

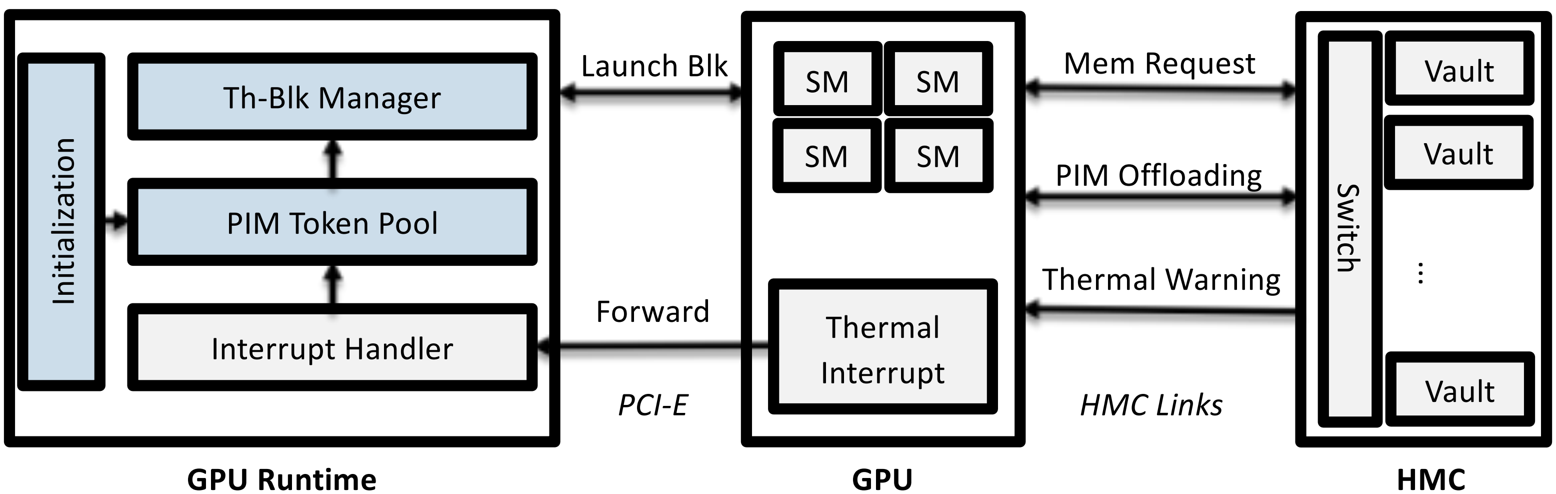
- Thermal warning -> lowers PIM intensity -> reduces internal temperature of HMC



Software-Based Throttling

GPU runtime implements some components to control PIM intensity

- **PIM Token Pool (PTP)**
  - # of maximum thread blocks that are allowed to use PIM functionality
- **Thread Block Manager**
  - Check PTP and launch PIM code if tokens are available
- **Initialization**
  - Estimate the initial PTP size based on static analysis at compile time



Code Generation for non-PIM code

The GPU compiler generates PIM-enabled and non-PIM kernels at compile time

- Source-to-source translation
- IR-to-IR translation

```
Void cuda_kernel(arg_list)
{
  for (int i=0; i<end; i++)
  {
    uint addr = addrArray[i];
    PIM_Add(addr, 1);
  }
}
```

Original PIM Code

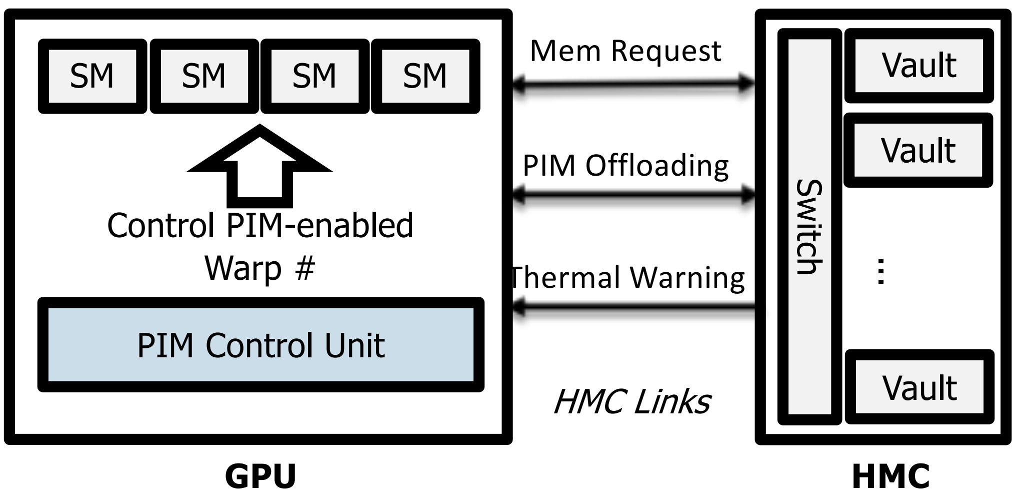
```
void cuda_kernel_np(arg_list)
{
  for (int i=0; i<end; i++)
  {
    uint addr = addrArray[i];
    cuda atomicAdd(addr, 1);
  }
}
```

Shadow Non-PIM Code

Hardware-Based Throttling

PIM Control Unit

- Controls # of PIM-enabled warps
- Performs dynamic binary translation
- **See the paper for detail!**



Type	PIM Instruction	Non-PIM
Arithmetic	Signed Add	atomicAdd
Bitwise	Swap, bit write	atomicExch
Boolean	AND, OR	atomicAND/OR
Comparison	CAS-equal/greater	atomicCAS/Max

Evaluation

Methodology

Thermal Evaluation

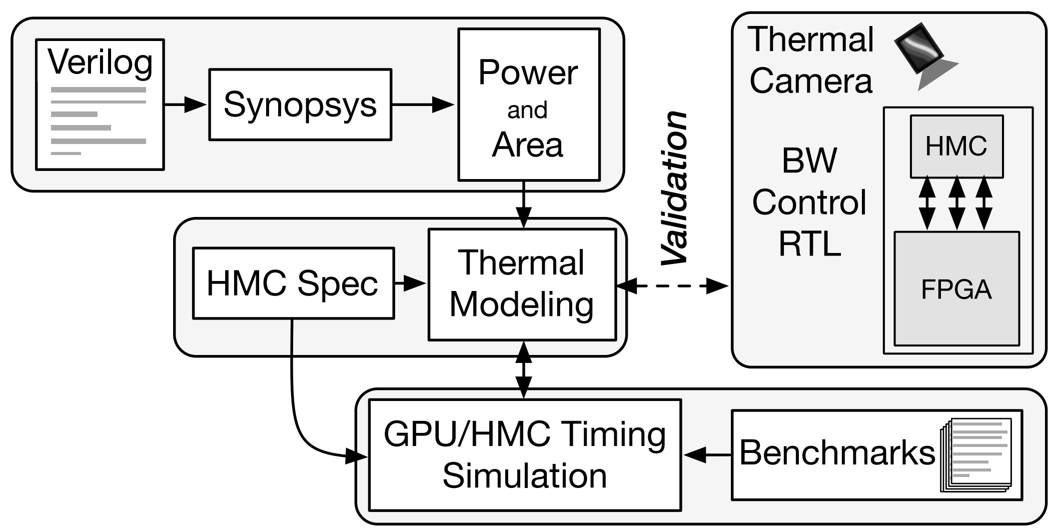
- Temp Measurement: Real HMC 1.1 Platform
- Thermal Modeling: HMC 2.0 using 3D-ICE
- Power & Area: Synopsys (28nm/50nm CMOS)

Performance Evaluation

- MacSim w/ VaultSim

Benchmark

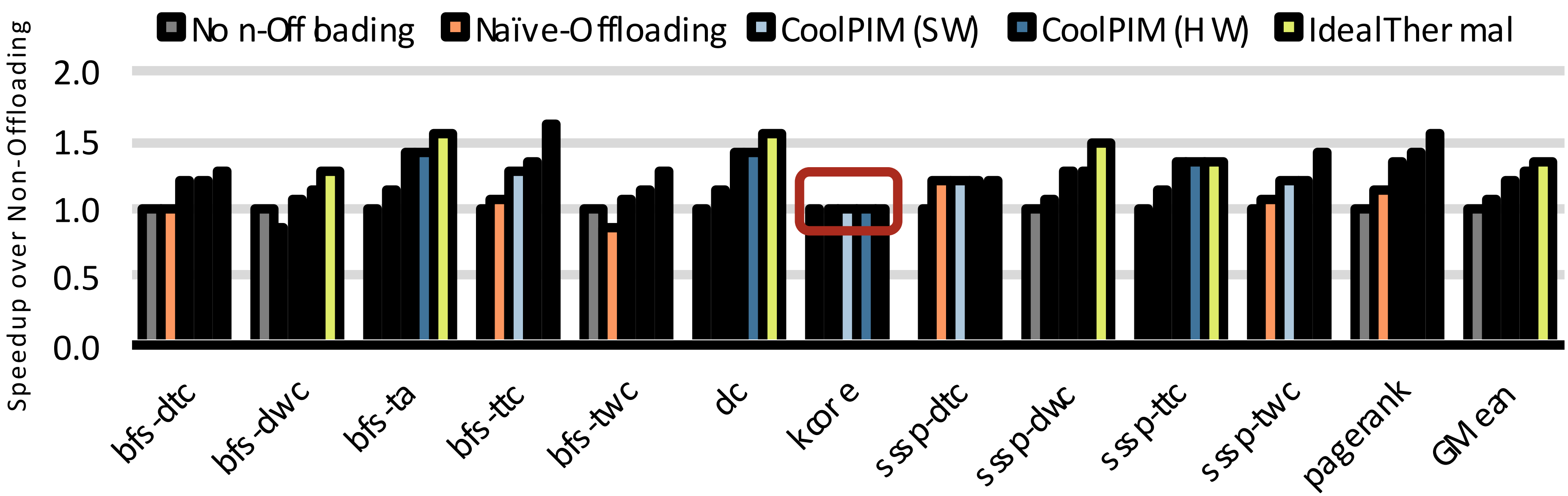
- GraphBIG benchmark with LDBC dataset
  - BFS, SSSP, PageRank, etc…



Performance

Speedup over baseline (Non-Offloading)

- **Naïve/SW/HW**: using a commodity-server active heat sync
- **Ideal Thermal**: with unlimited cooling

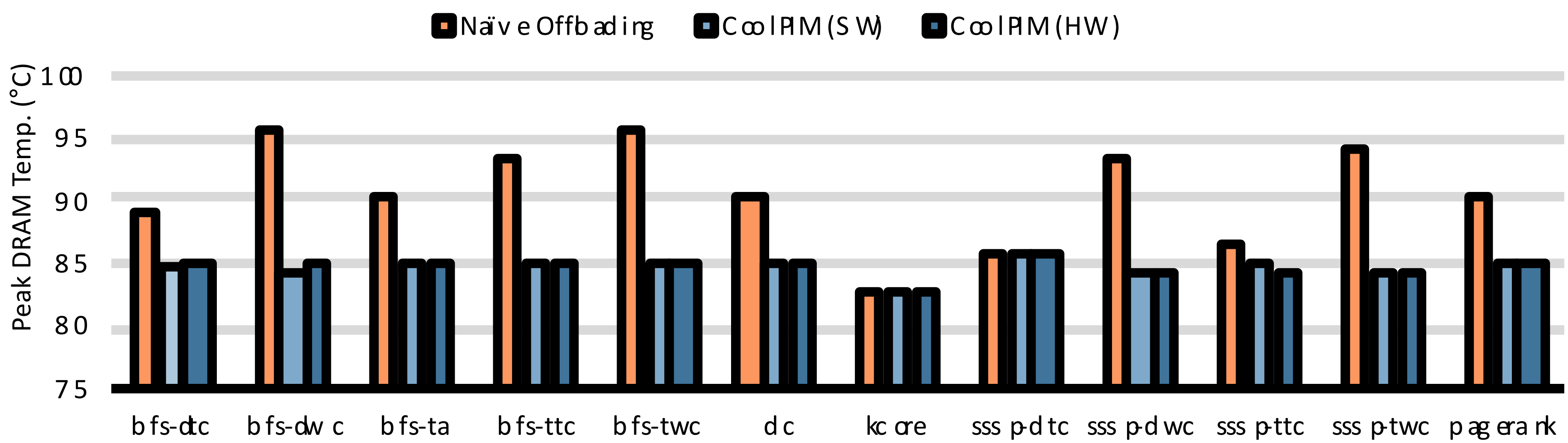


On average, CoolPIM (SW/HW) improves performance by 1.21x/1.25x!

Thermal Analysis

PIM Offloading Rate

- Naïve: 3~4 op/ns → Temperature goes beyond the normal operating region.
- CoolPIM: 1.3 op/ns → No memory performance slowdown



CoolPIM maintains peak DRAM temperature within normal operating temp!

Conclusion

Conclusion

Observation: PIM integration requires careful thermal consideration

- Naive PIM offloading may cause a thermal issue and degrades overall system performance

CoolPIM: Source throttling techniques to control PIM intensity

- Keeps HMC ”Cool” to avoid thermal-triggered memory performance degradation

Results: CoolPIM improves performance by 1.37x over naïve offloading

- 1.2x over non-offloading on average

Thank You

Backup

Typical Cooling Types

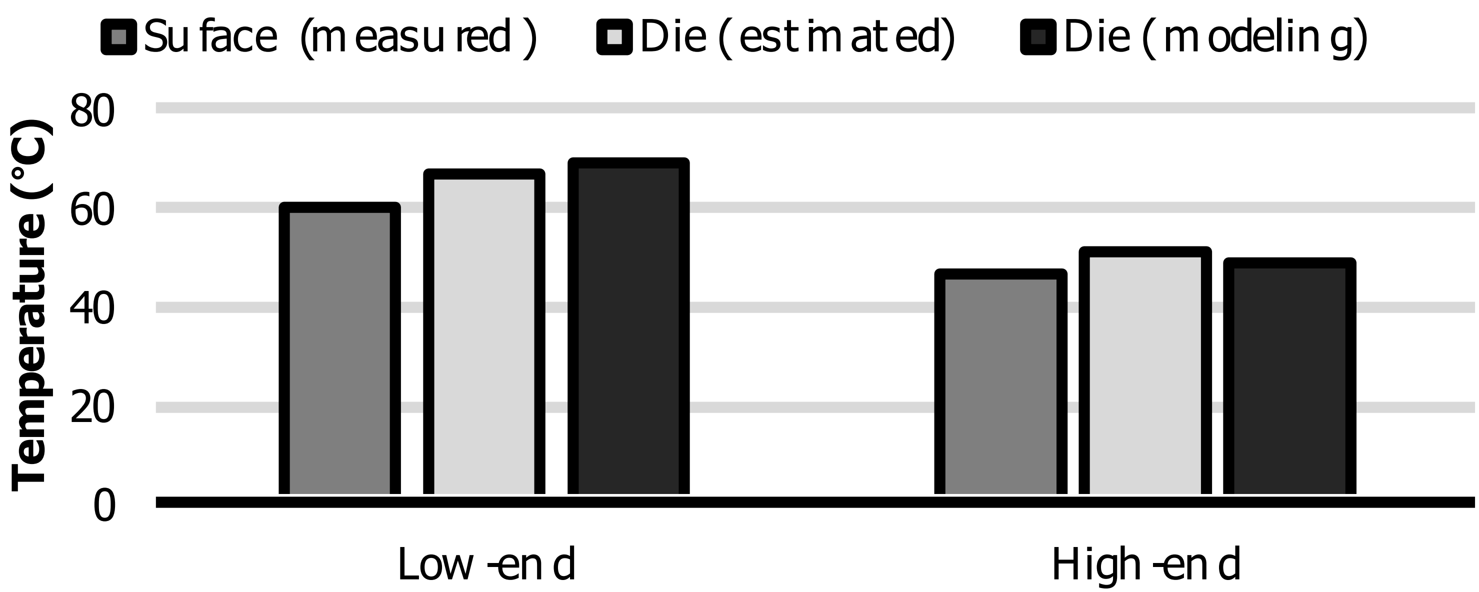
Type	Thermal Resistance	Cooling Power*
Passive heat sink	4.0 °C/W	0
Low-end active heat sink	2.0 °C/W	1x
Commodity-server active heat sink	0.5 °C/W	104x
High-end heat sink	0.2 °C/W	380x

* We assume the same plate-fin heat sink model for all configurations.

Thermal Model Validation

Validate our thermal evaluation environment

- Model HMC 1.1 temperature and compare with measurements

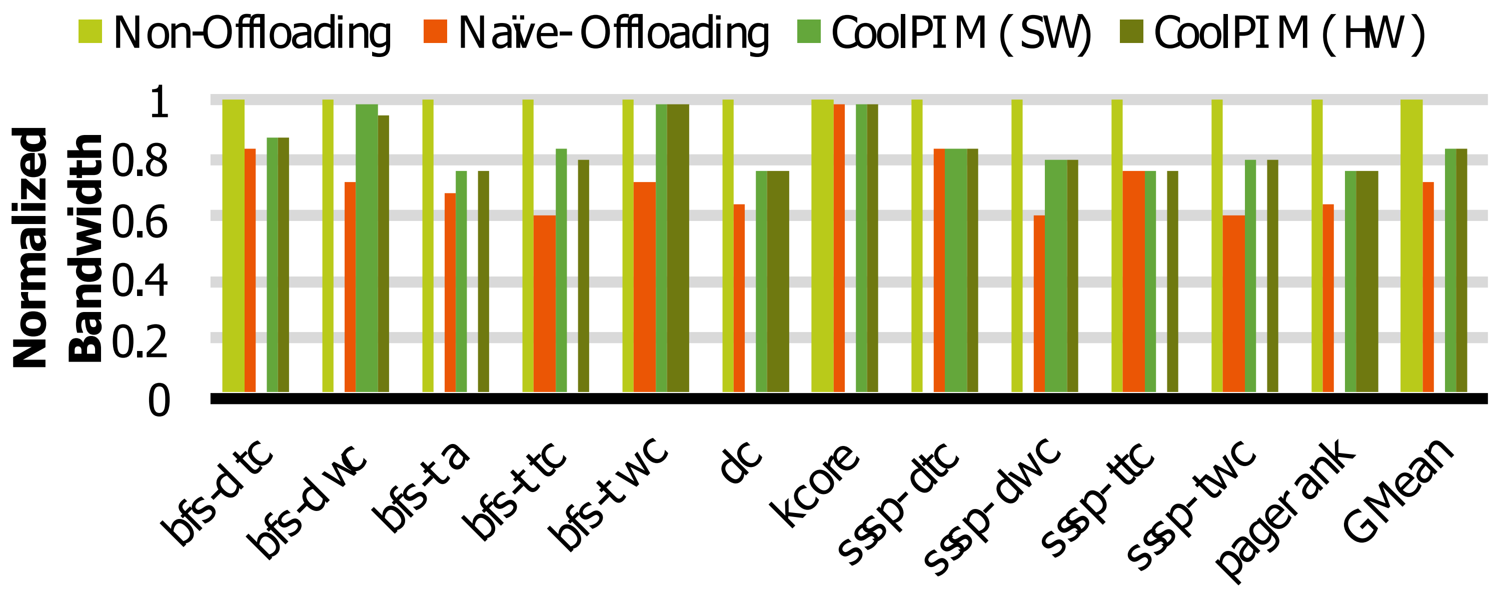


Evaluation Configuration

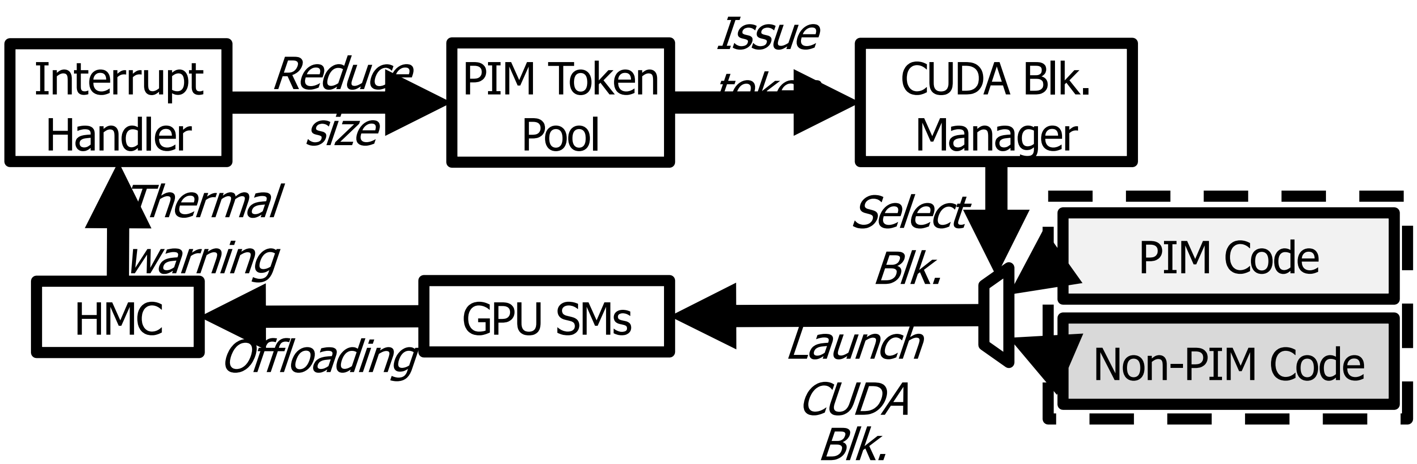
Component	Configuration
Host	GPU, 16 PTX SMs, 32 threads/warp, 1.4GHz 16KB private L1D and 1MB 16-way L2 cache
HMC	8 GB cube, 1 logic die, 8 DRAM dies 32 vaults, 512 DRAM banks tCL=tRCD=tRP=13.75ns,tRAS=27.5ns 4 links per package, 120 GB/s per link 80 GB/s data bandwidth per link DRAM Temp. phase: 0-85 °C, 85-95 °C, 95-105 °C 20% DRAM freq reduction (high temp. phases)

Bandwidth Consumption

Bandwidth consumption normalized to baseline (non-offloading)



Software-Based Throttling



Hardware vs Software

Type	Software-Based	HW-Based
Control Granularity	Thread Blocks	Warps
Control Delay	Long Delay	Short Delay
Design Complexity	Low	High

Hardware vs Software