mNPUsim: Evaluating the Effect of Sharing Resources in Multi-core NPUs

Soojin Hwang*, Sunho Lee*, Jungwoo Kim,

Hongbeen Kim, and Jaehyuk Huh

KAIST School of Computing

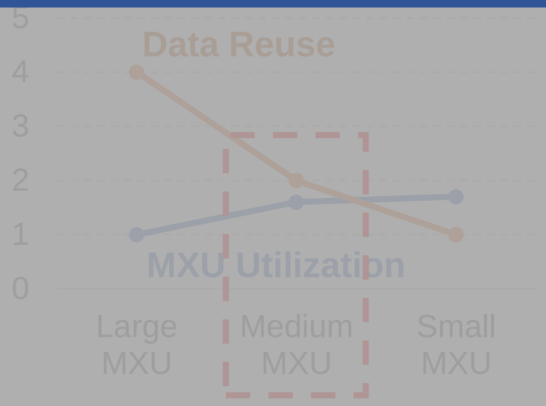
CASYS Computer Architecture and Systems Lab

*: Equal contribution

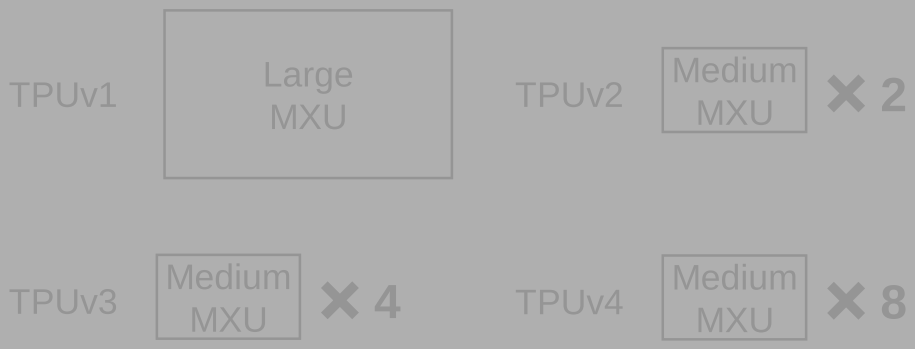
Emergence of Multi-core NPU

		Small NPU	Large NPU
Alexnet (Image Classification)	Less Computation	○	✕
GPT-3 (Large Language Model)	High Computation	✕	○

Multi-core NPU is necessary!



Medium-size Matrix Multiplication Unit (MXU)

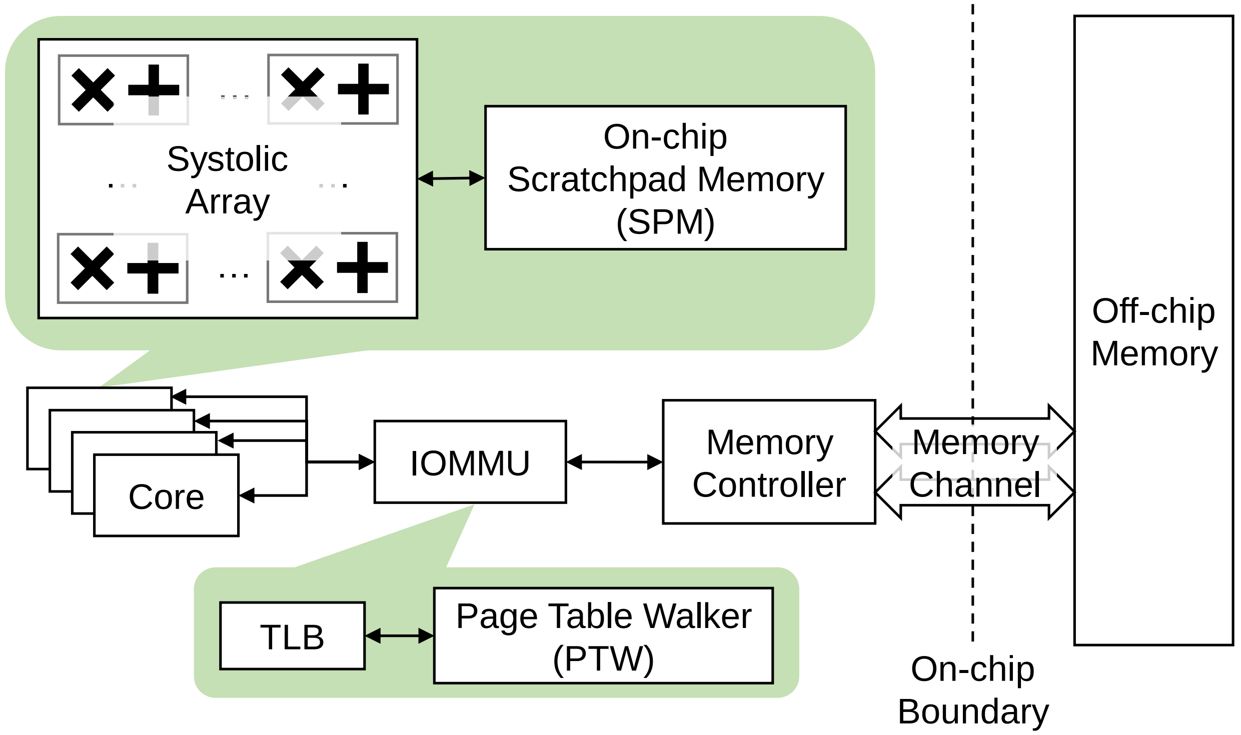


Multi MXU Configuration

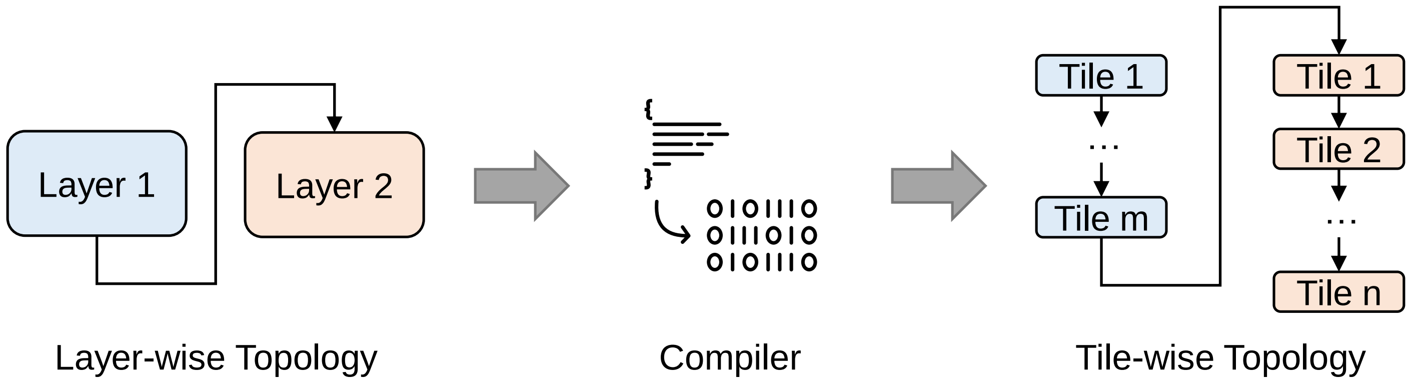
1) Norrie et al., “Google’s Training Chips Revealed: TPUv2 and TPUv3”, Hot Chips Symopsium, 2020

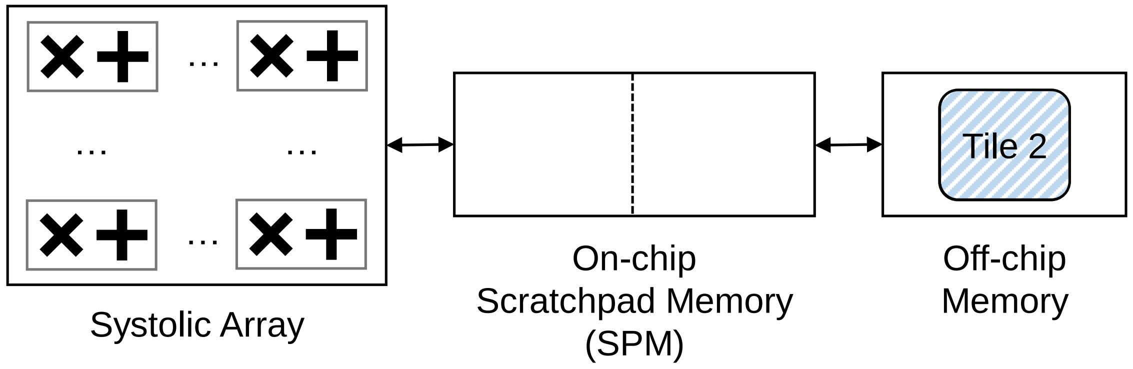
Multi-core NPU Architecture

- Computation
  - Per-core: Systolic array
- Memory
  - Per-core: On-chip scratchpad memory
  - Entire-core: IOMMU, memory controller/channel, off-chip memory



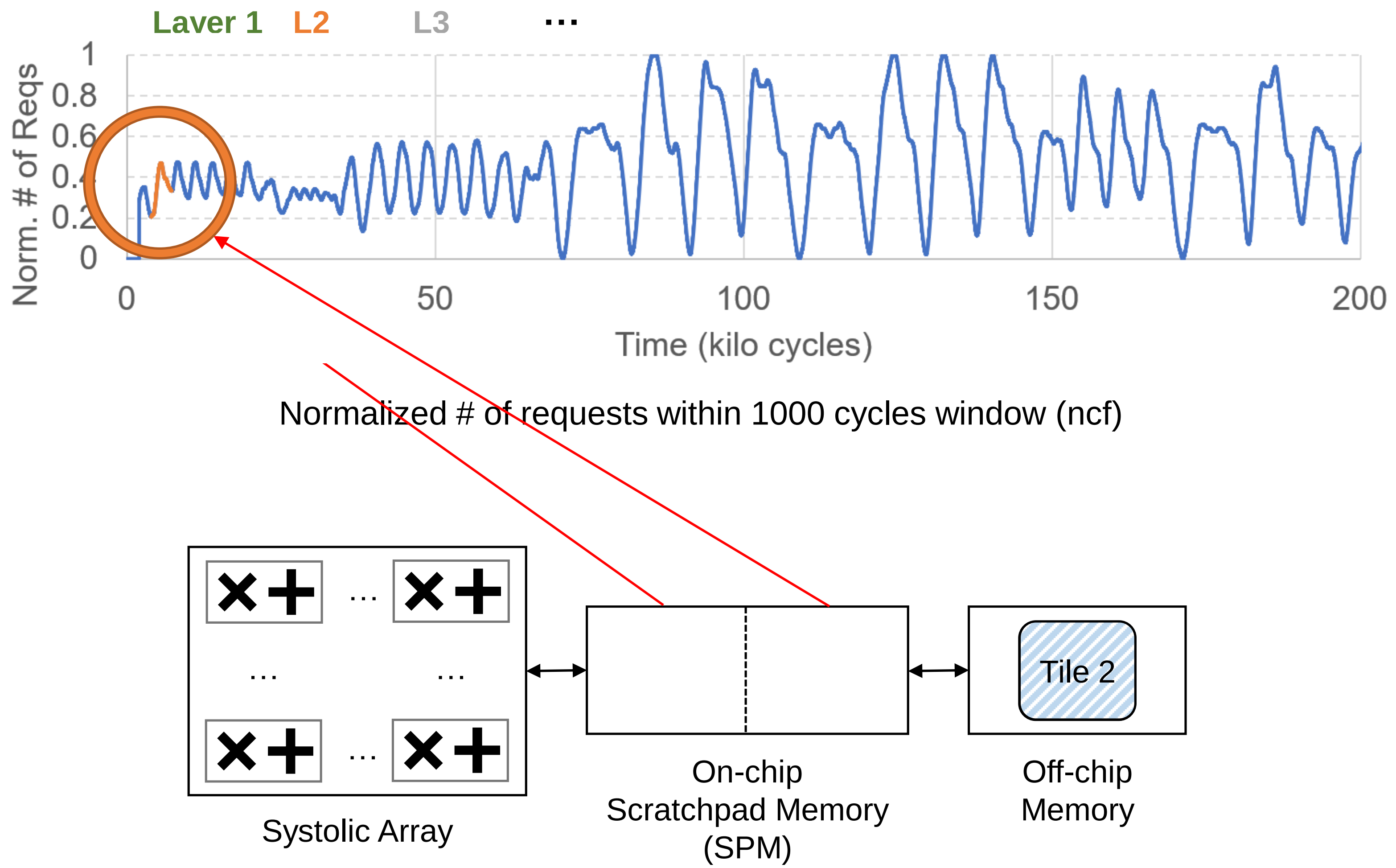
NPU Execution Model





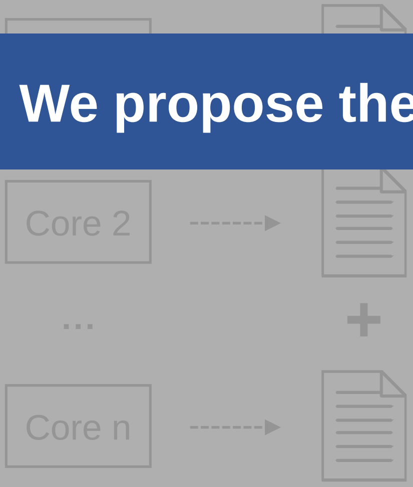
Tile & Double buffering → Memory Burstiness!

NPU Memory Requests Burstiness



Existing NPU Simulators

- Weakness: Ignore interference between cores
- Single-core iteration
  - Multi-core result = Naïve cycle sum of single-core results (w/o interference)
- Fixed-cycle based memory simulation
  - Adopting analytical modeling in off-chip memory access simulation



We propose the dynamic multi-core NPU simulator!

Single-core Iteration

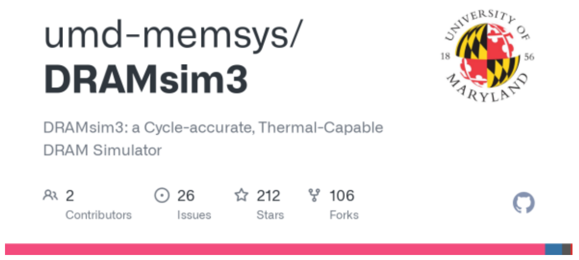
Transit Latency	Traffic / Bandwidth (cycles)
Page Table Walk	100 (cycles/level)
…	…
Off-chip Memory Access	100 (cycles)

Fixed-cycle based Memory Simulation

mNPUsim: A Cycle-accurate Multi-core NPU Simulator

KAIST

mNPUsim



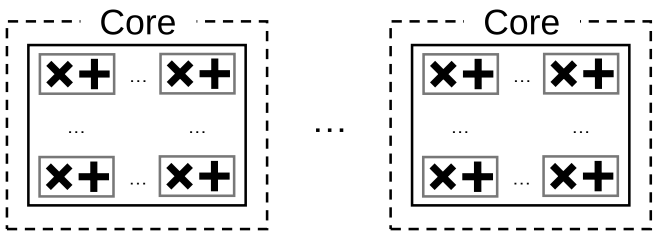
umd-memsys/
DRAMsim3

DRAMsim3: a Cycle-accurate, Thermal-Capable DRAM Simulator

2	26	212	106
Contributors	Issues	Stars	Forks

DRAMsim3[1)]-integrated dynamic off-chip memory simulation

Open-sourced & Artifact-evaluated



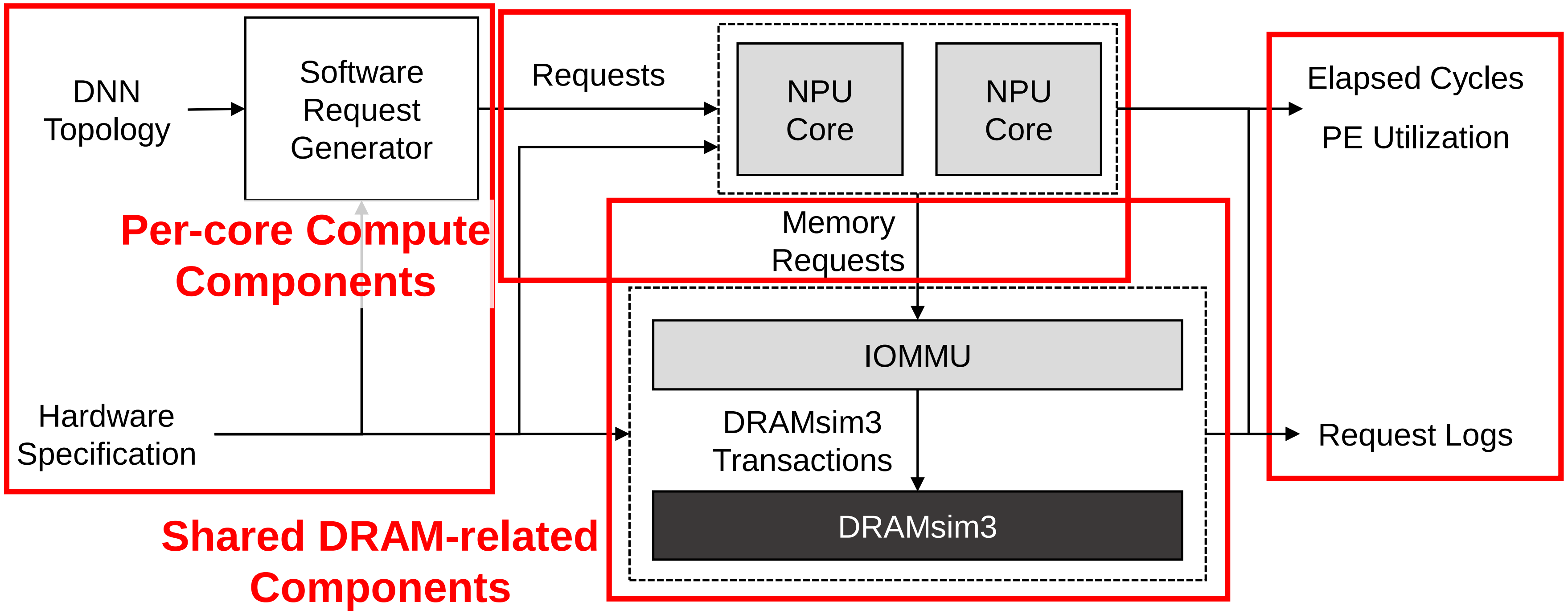
Multi-core simulation

1. Architecture
2. Network
3. Off-chip Memory
4. On-chip Memory
5. Execution Mode

Various configuration inputs

1) Li et al., "DRAMsim3: A cycle-accurate, thermal-capable DRAM simulator", CAL vol. 19, no. 2 (2020)

Execution Flow of mNPUsim



[illegible]

[illegible]

[illegible]

[illegible]

Shared Resource Analysis with mNPUsim

Methodology

- Benchmarks: 8 machine learning workloads
- Simulator configuration
  - NPU: TPUv4 configuration[1)]
  - Off-chip memory: HBM2

Type	Model
CNN	Resnet50 (res) Yolo-tiny (yt) AlexNet (alex)
RNN	Selfish-RNN (sfrnn) DeepSpeech2 (ds2)
Recommendation	DLRM (dlrm) NCF (ncf)
Attention	gpt2 (gpt2)

Benchmarks

Cloud-scale NPU (TPU-modeling)	
Systolic Array	128 x 128
On-chip SPM	36MB
Frequency	1GHz
TLB	8-way 2048 entry per NPU
# of PTW	8 per NPU
Off-chip Memory (HBM2-modeling)	
Bandwidth	128GB/s per NPU
Capacity	4GB per NPU
Frequency	1GHz

Simulator Configuration

1) Google, “CloudTPU”, https://cloud.google.com/tpu/docs/system-architecture-tpu-vm

KAIST

Methodology

- Two metrics
  - Performance: relative speedup
  - Fairness[1]: balance of speedup between cores

$$speedup_k = \frac{exec.\,time\ of\ baseline}{exec.\,time\ of\ k^{th}\ core}$$

$$\boxed{performance_i = geometric\ mean\ of\ speedup\ for\ i^{th}\ mix\ workload}$$

$$slowdown_k = \frac{1}{speedup_k}$$

$$\mu_i = average\ of\ slowdown\ for\ i^{th}\ mix\ workload$$

$$\sigma_i = standard\ deviation\ of\ slowdown\ for\ i^{th}\ mix\ workload$$

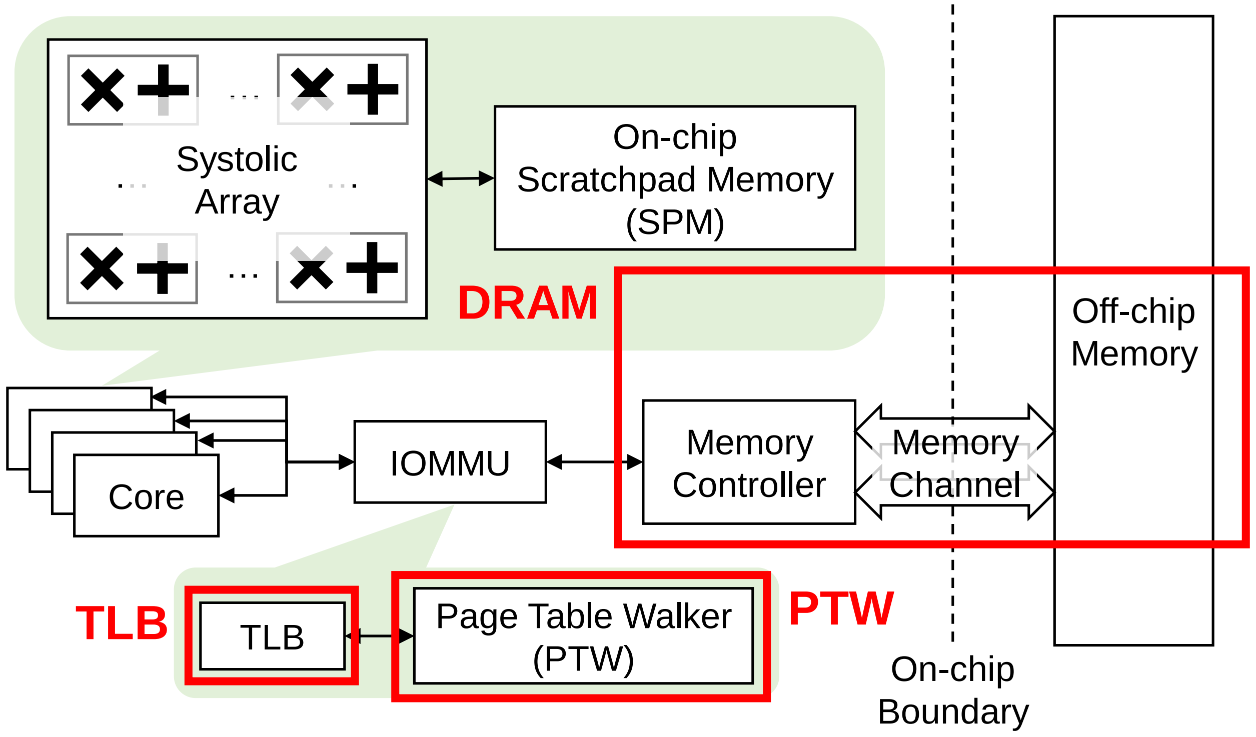
$$\boxed{fairness_i = 1 - \frac{\sigma_i}{\mu_i}}$$

1) Van Careynest et al., “Fairness-aware scheduling on single-ISA heterogeneous multi-cores”, PACT’13

KAIST

Design Space

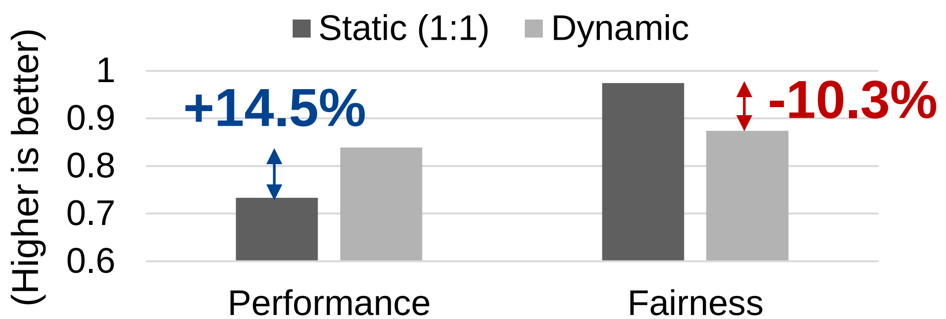
- Three levels of resource sharing
  - DRAM-only sharing: DRAM components only sharing
  - DRAM & PTW sharing: Plus PTW sharing
  - DRAM & PTW & TLB sharing: Plus TLB sharing



KAIST

Shared DRAM Bandwidth: Experiment

- Setup: DRAM-only sharing
  - No address translation, dual-core NPU
- Compares three sharing schemes
  - Baseline: Ideal (per-core monopolization)
  - Static: ½ (for NPU 0), ½ (for NPU 1)
  - Dynamic: Dynamically sharing whole resources
- Dynamic sharing improves **+14.5%** performance over static
  - By sacrificing fairness **-10.3%**

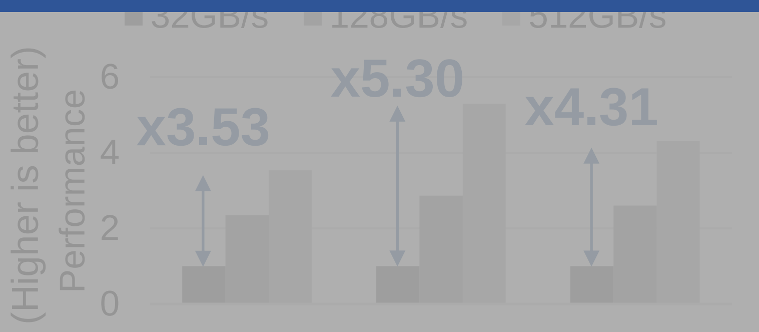


Geomean of performance and fairness with regard to sharing scheme

Shared DRAM Bandwidth: Analysis

- Setup: DRAM-only sharing
  - No address translation, single-core NPU
- Result:
  - Higher bandwidth, better performance (geomean: **x4.31**)
  - Different workload, different sensitivity (sfrnn: **x3.53**, ncf: **x5.30**)
- Analysis:

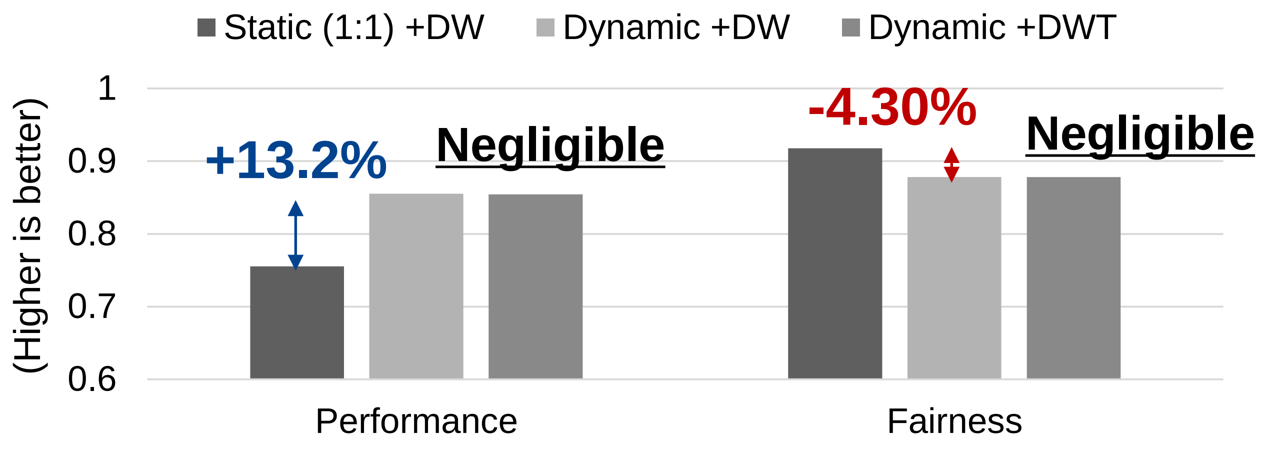
1. DRAM bandwidth is a crucial resource
2. Dynamic DRAM bandwidth sharing is better



Performance with respect to DRAM bandwidth

Shared IOMMU

- Setup: Dual-core NPU
- Experiment 1. DRAM & PTW sharing
  - Performance improvement: **+13.2%**
  - Fairness drop: **-4.30%**
  - Reason: **Burstiness of requests**
- Experiment 2. DRAM & PTW & TLB sharing
  - **Negligible** difference due to negligible TLB capacity contention

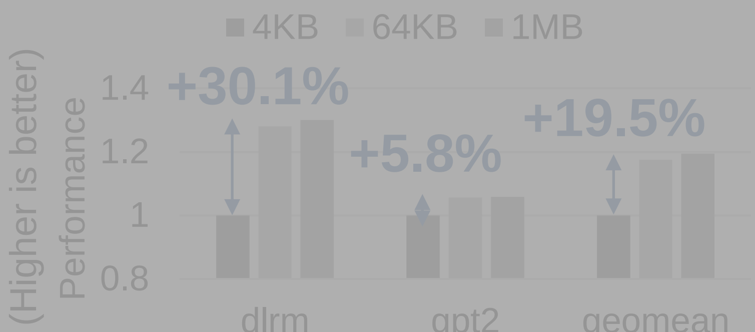


Geomean of performance and fairness with regard to IOMMU sharing

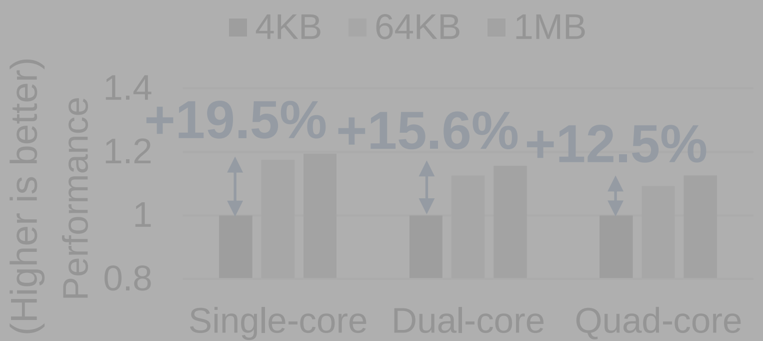
Scalable Page Size

- Page size candidates from ARM64[1]
  - 4KB (Baseline), 64KB, 1MB
- Single-core: **+19.5%** performance improvement (dlrm: **+30.1%**, gpt2: **+5.8%**)
- Multi-core: Only **+12.5%** improvement in quad-core
  - Other contention between cores reduces PTW effect

Huge page is better choice (especially for less number of cores)



Single-core performance



Multi-core performance

1) ARM developer, https://developer.arm.com

More Information on Paper

- Quad-core NPU experiment
- Off-chip memory utilization in single/dual-core evaluation
- Contention sensitivity
- Performance distribution affected by co-runners
- Workload mapping in multi-NPU
- Guideline of using mNPUsim

Conclusion

- Propose a cycle-accurate multi-core NPU simulator: **mNPUsim**

- Evaluate and compare shared resource management techniques

Management Techniques	**Performance**	**Fairness**
Dynamic sharing of DRAM bandwidth	**+14.5%**	**-10.3%**
Dynamic sharing of page table walker	**+13.2%**	**-4.3%**
Dynamic sharing of TLB	Negligible	Negligible
Scalable page size	**+19.5%** (single-core) **+12.5%** (quad-core)	Negligible

- Visit and enjoy **https://github.com/casys-kaist/mNPUsim**

KAIST

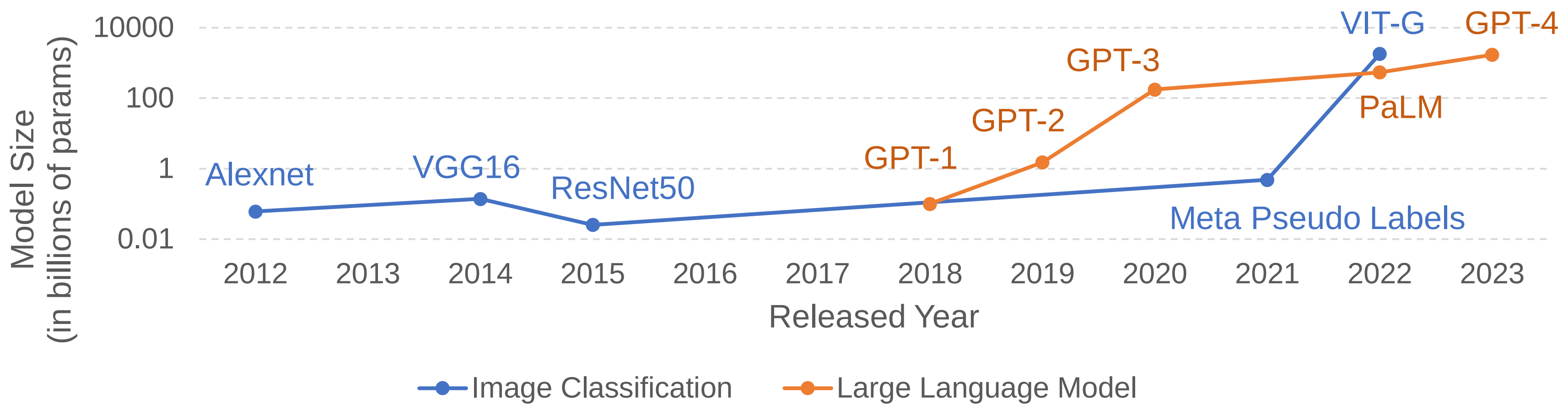
Backup

Discussion (# of PTW req → Scalable Page)

- Increasing total amount of shared resource
  - Not easy due to overhead
  - Can have similar effect by decreasing the demand
  - **Larger page size requires lower number of PTW requests**

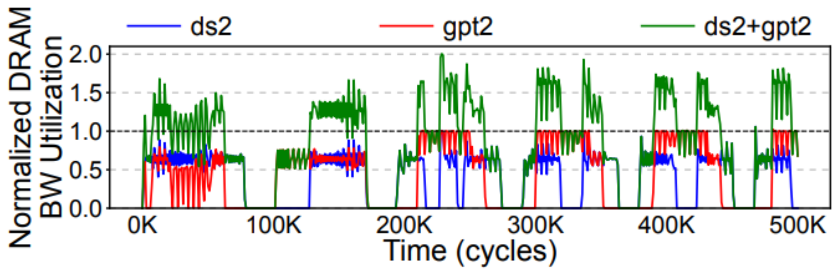
Suggestion. Scalable page size (huge page)

AI Model Size Variation



DRAM Memory Bandwidth Utilization

- Reason of advantages from dynamic sharing:
  - Current bandwidth cannot efficiently deal with NPU burstiness
  - Average bandwidth utilization is not high
  - Burstiness of memory request

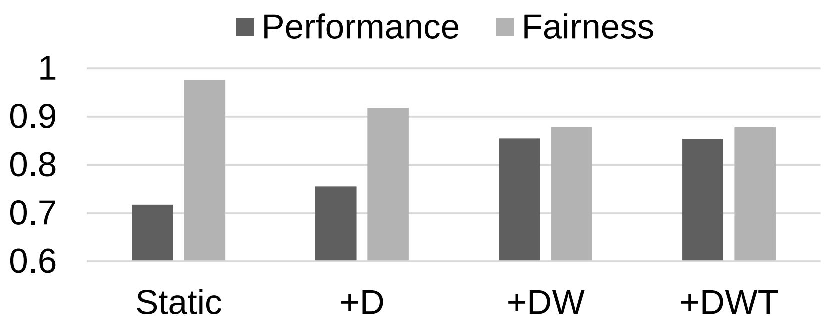


DRAM bandwidth utilization of single/dual-core

Final +DWT

- Effect of dynamic resource sharing (+DWT)
  - Offers **+19.1%** speedup above static sharing
  - Reaches **85.5%** of ideal performance
  - Shows fairness of **0.88**

Total amount of bandwidth and PTW is important

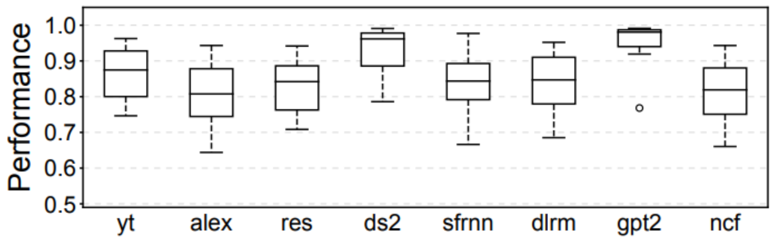


Geomean of performance and fairness

Performance Distribution by Co-runners

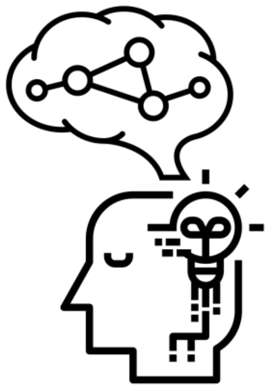
- Effect of co-runner contention
  - Standard deviation distribution: **0.072 ~ 0.106**
- Considering inter-core contention
  - Co-runner problem
  - **Contention-aware scheduling can be helpful**

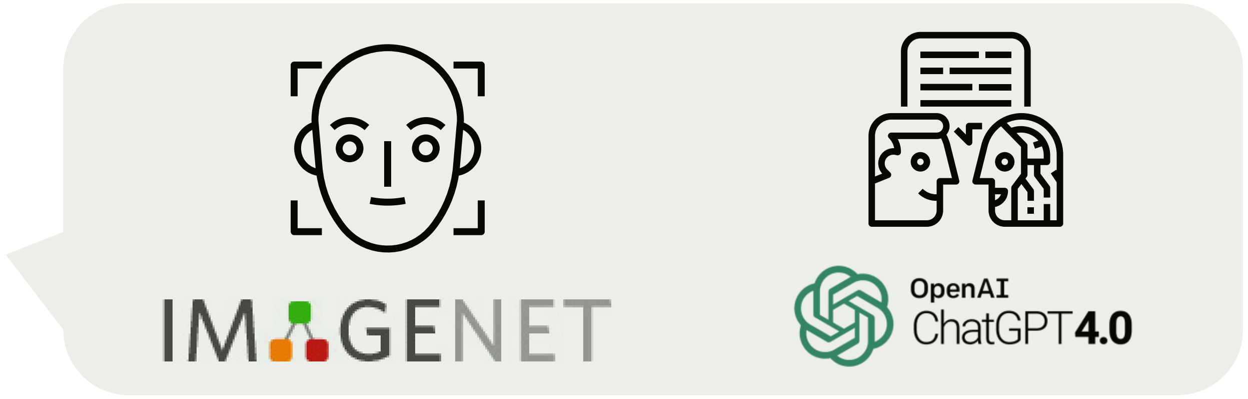
Different workload, different amount of contention
Suggestion. Workload mapping

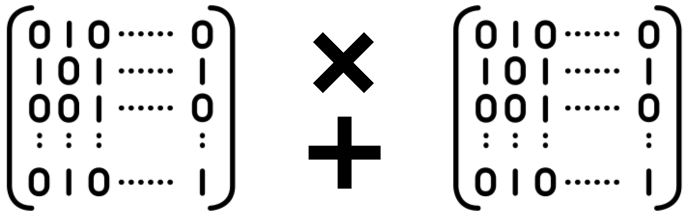


Performance distribution
affected by co-runners

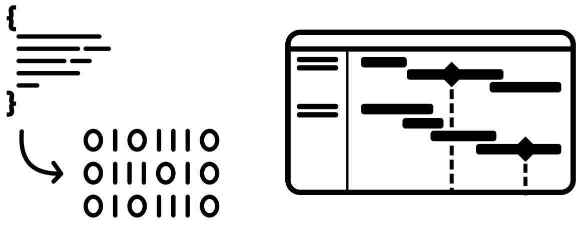
Necessity of NPU Accelerator







Numerous Basic Operations



Compiler-level
Predictable Operations