GCoM:
A Detailed GPU Core Model for Accurate Analytical Modeling of Modern GPUs

Jounghoo Lee[1], Yeonan Ha[1], Suhyun Lee[1], Jinyoung Woo[2],
Jinho Lee[1], Hanhwi Jang[2], and Youngsok Kim[1]

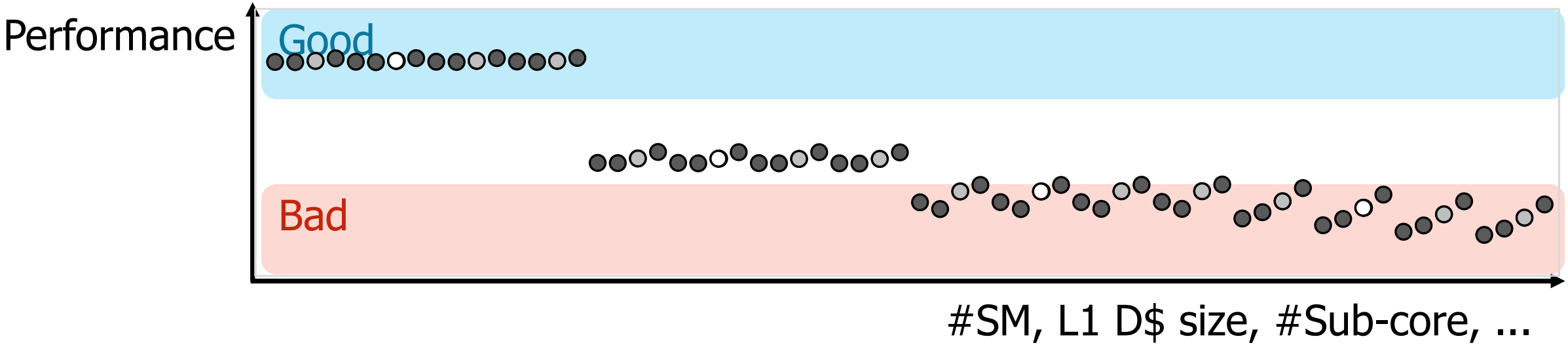
[1]Yonsei University

[2]Ajou University

49th International Symposium on Computer Architecture (ISCA-49)

Importance of Fast Perf. Modeling

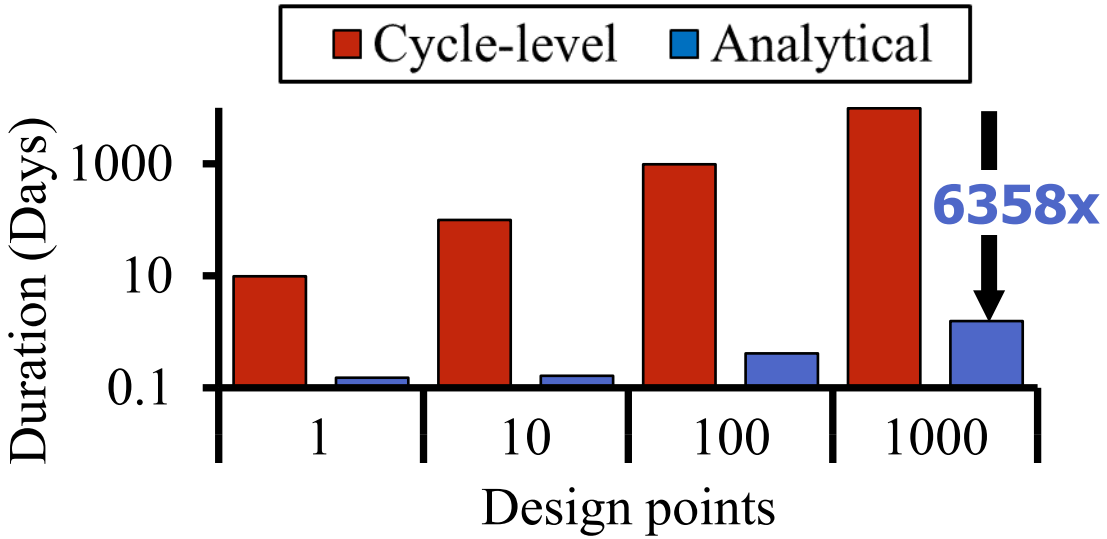
- GPU design space is **huge.**

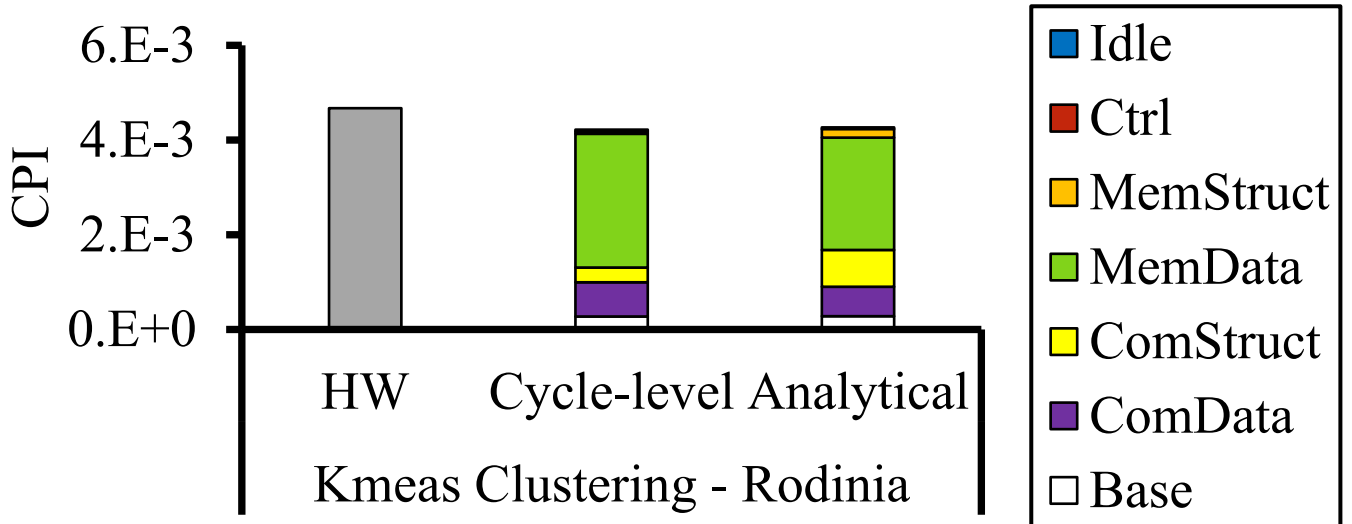


- Cycle-level simulation is the de facto standard method.
- Cycle-level simulation is **accurate** but **too slow**.

Alternative: Analytical Modeling

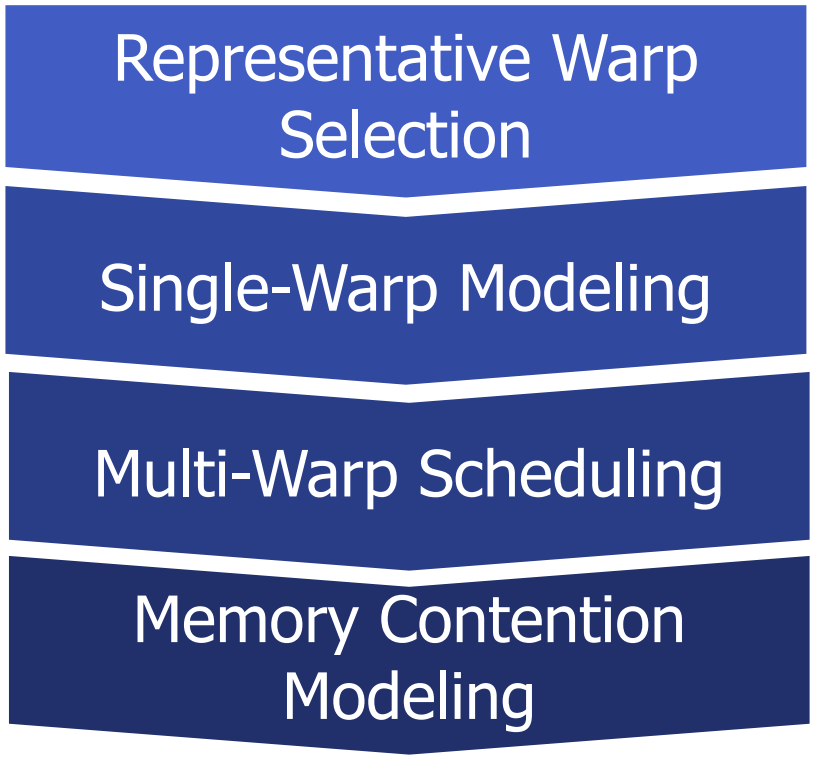
- Analytical models are **fast.**
  - They do not repeat cycle-by-cycle hardware operations.
- Analytical models also **construct CPI stacks.**
  - Prior works use **interval analysis** for accurate insight about bottlenecks.
- e.g., MDM [MICRO '20], GPUMech [MICRO '14]





GPU Interval Analysis

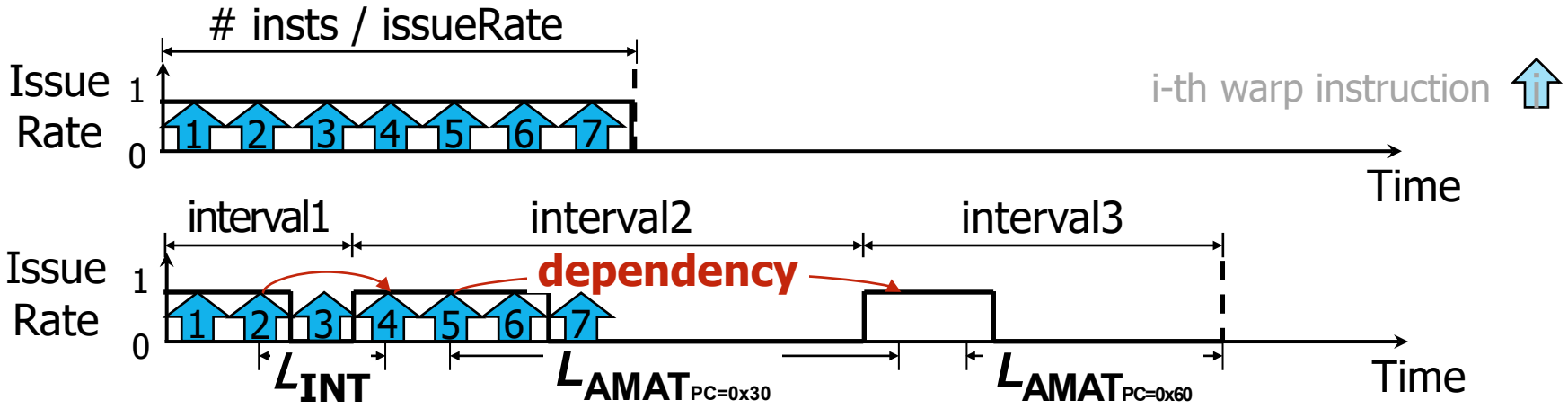
- **Performance = core issue rate** if no stall occurs
  - **Interval**: a sequence of warp instructions followed by stall cycles



- Used to approximate the warps of a kernel
- Split the representative warp into intervals
- Model latency hiding by WLP (warp-level parallelism)
- Model NoC and DRAM contention

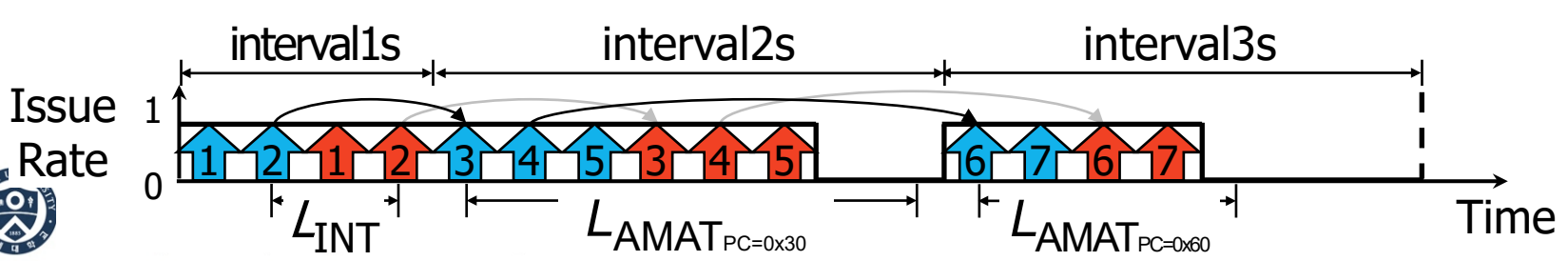
Interval Analysis: Single/Multi-Warp

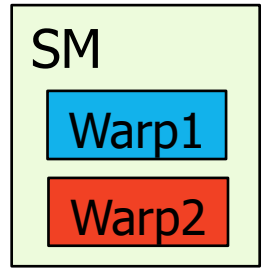
- Data dep. stalls break a single warp into intervals
  - Use average memory access time (AMAT) from cache simulator



	Latency
INT/FP32	2
LD/ST	AMAT

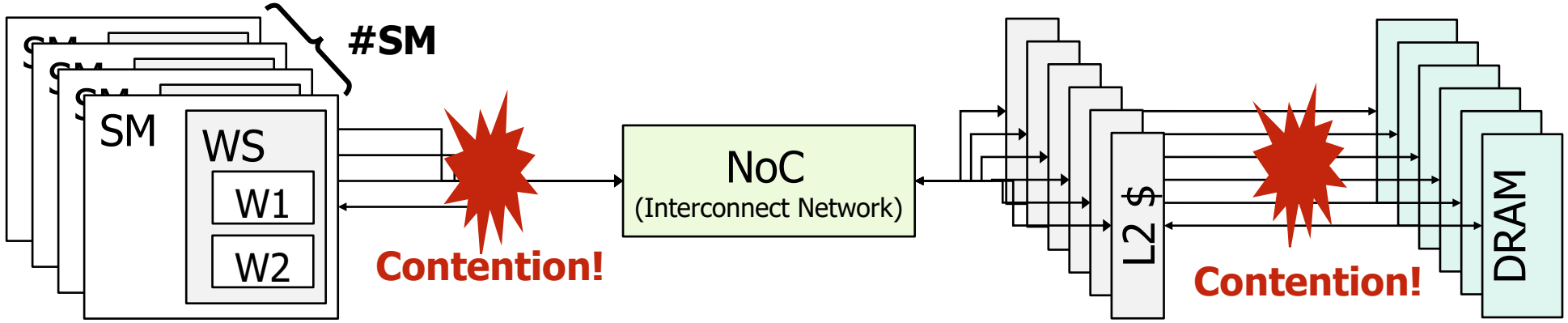
- Model **all warps in an SM as the representative warp**
  - Hide data stalls with respect to warp scheduling policy (e.g., LRR, GTO)



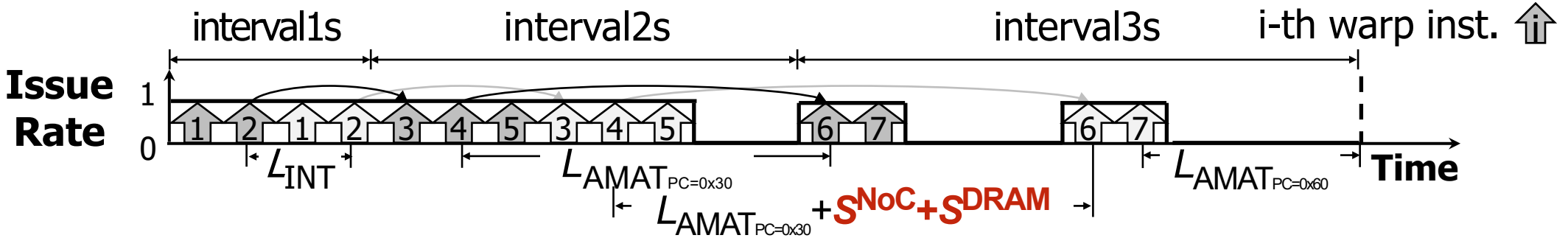


Interval Analysis: Memory Contention

- Limited NoC/DRAM BW incur memory contention.



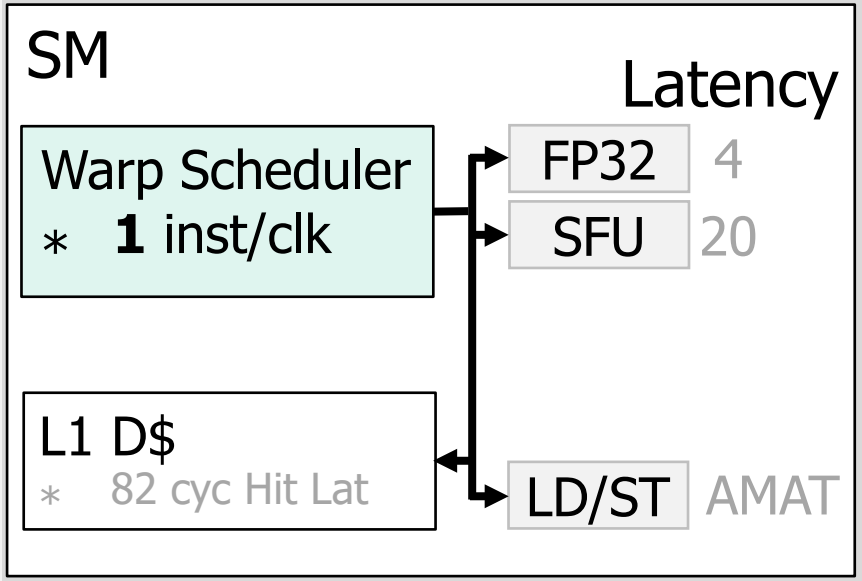
- Assume **the same memory traffic from every SM**



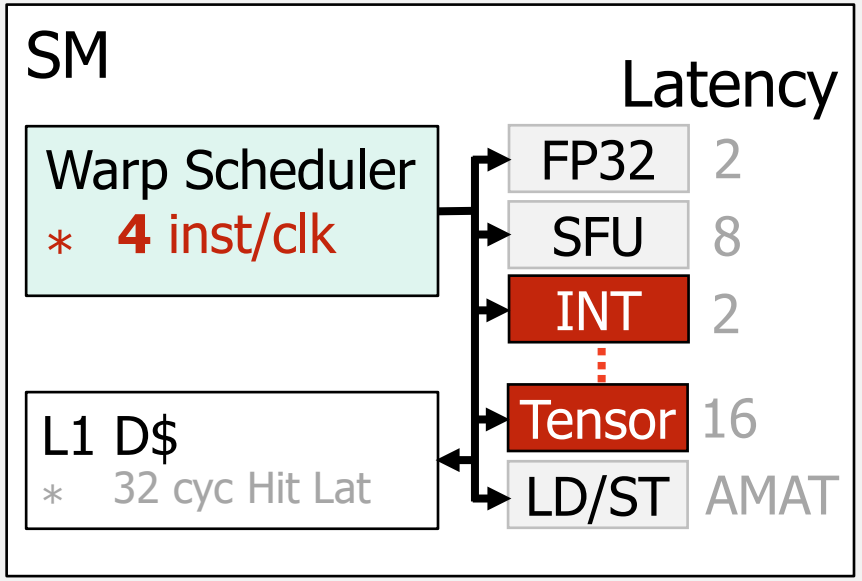
Naïvely Modeling Modern GPUs

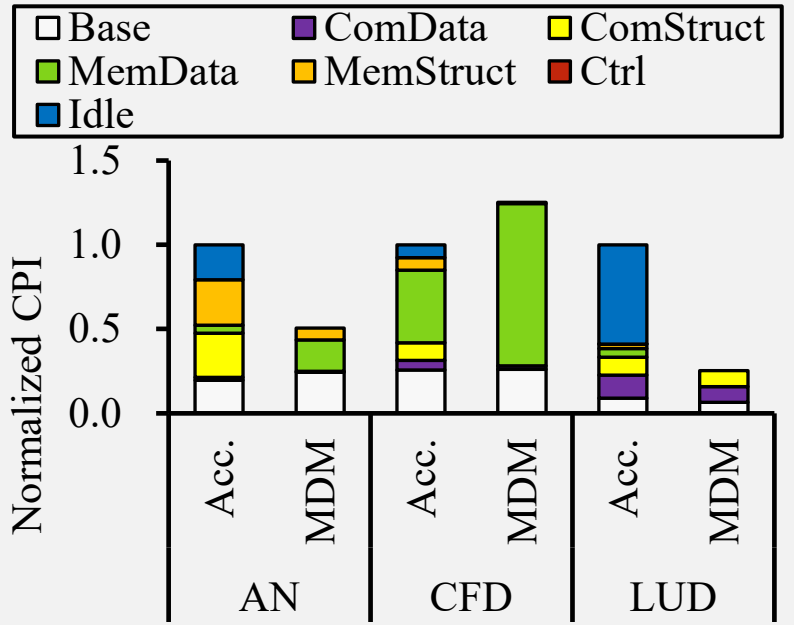
- i.e., tune the parameters of the existing interval models
- **Outdated core assumptions** incur **high errors!**
  - e.g., underestimate structural stalls, memory stalls, idle stalls, ...

MDM's SM Model



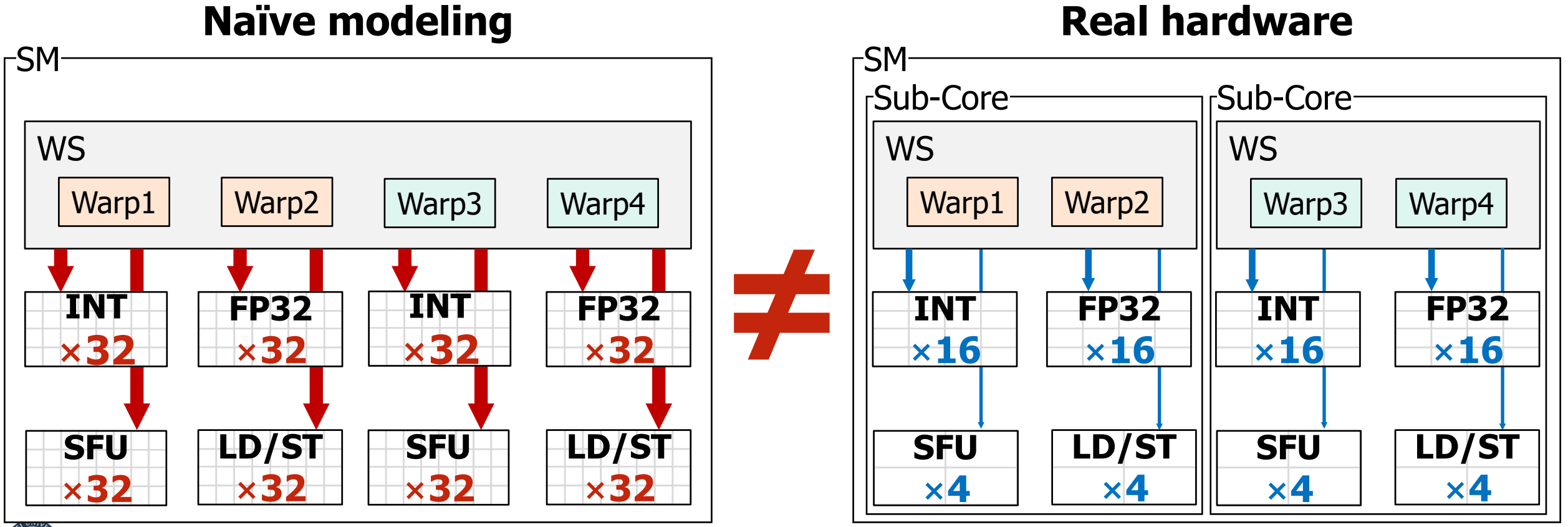
Naïve modeling





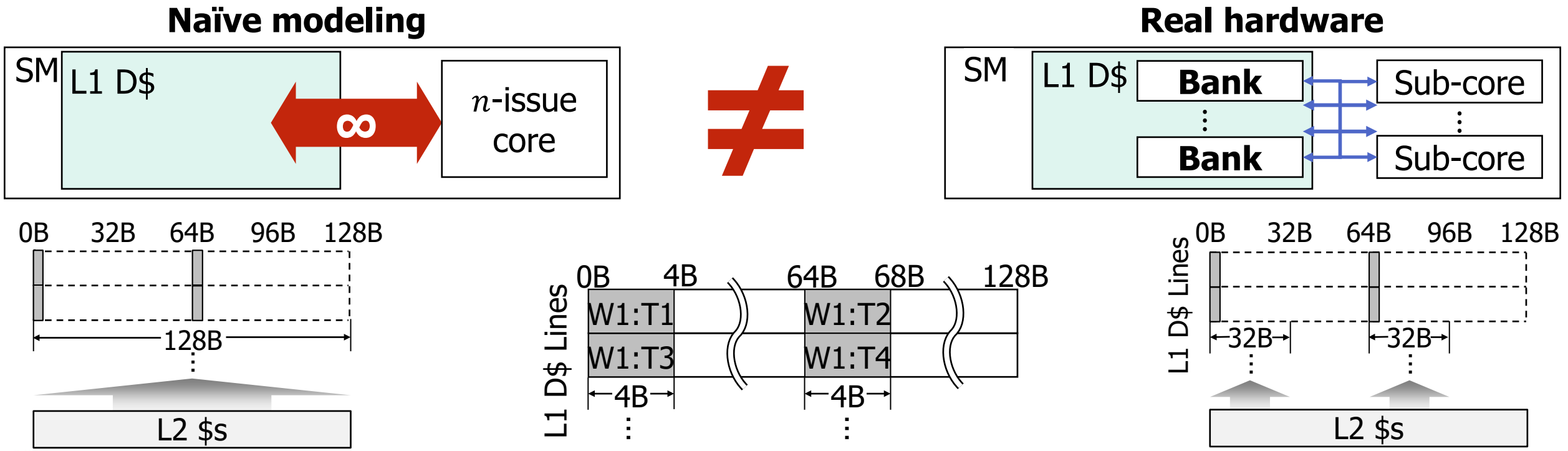
Limitation: Sub-Core-Level

- Naïve modeling cannot capture **ComStruct** stalls.
- … by **assuming sufficient functional unit lanes**



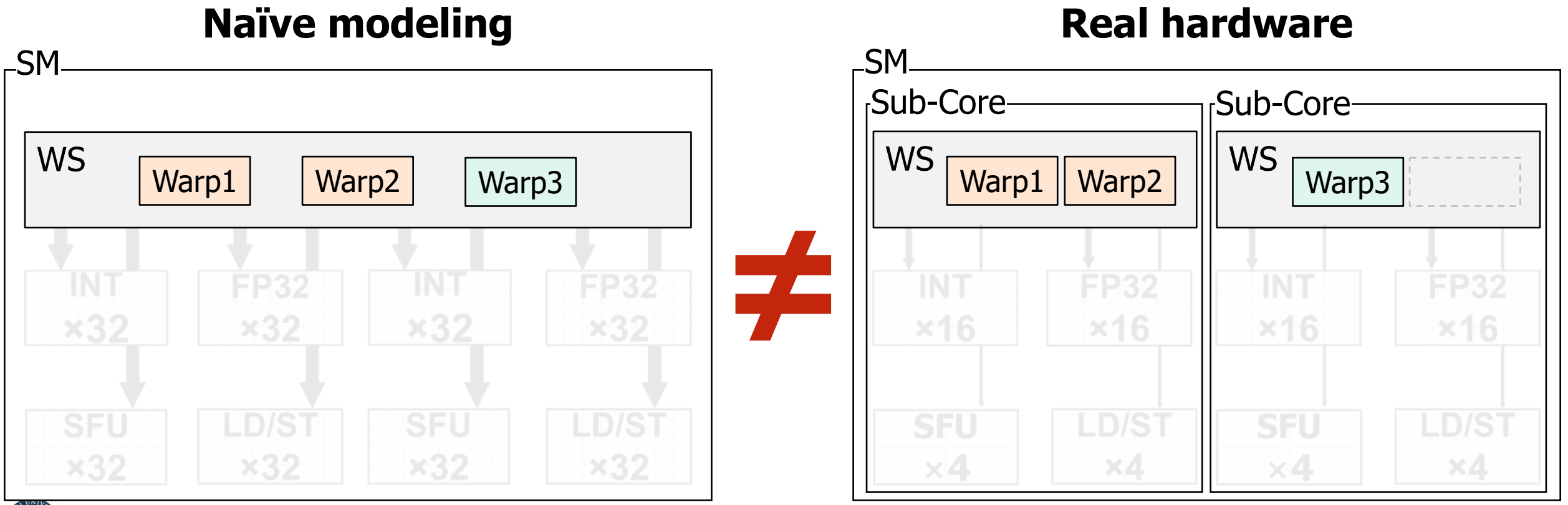
Limitation: SM-Level L1 D$

- Cannot capture **L1-D$ MemStruct** & **MemData** stalls
- … by **assuming unlimited bandwidth** and by **overlooking L1 D$'s sectors**



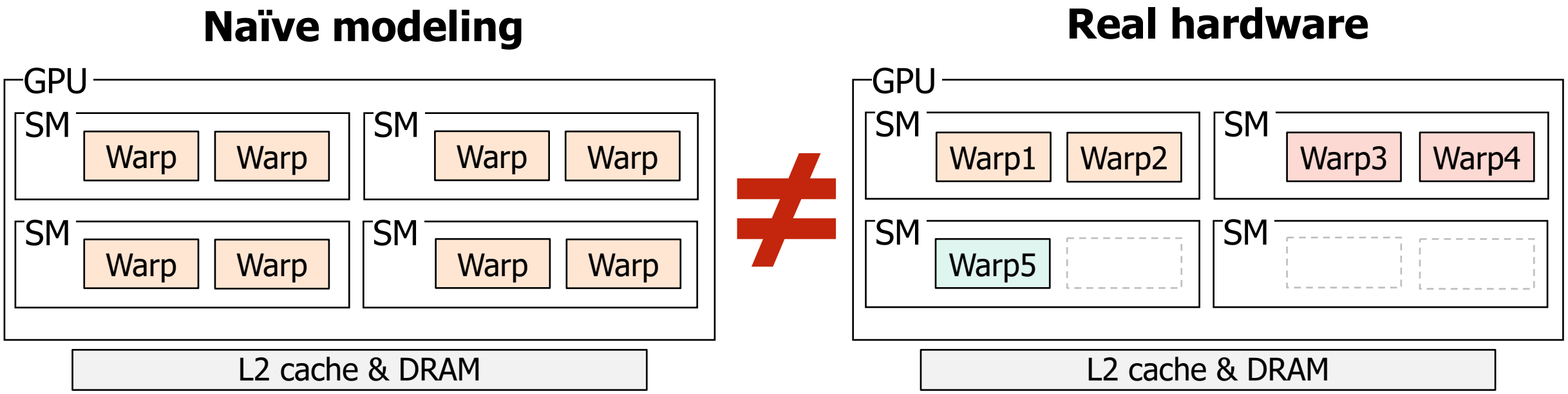
Limitation: SM-Level Load Imbalance

- Cannot capture **intra-SM Idle** stalls
- … by **disregarding multiple warp schedulers** per SM



Limitation: GPU-Level Load Imbalance

- Cannot capture **inter-SM Idle** stalls
- … by **assuming the same # of warps per SM**



Summary of the Limitations

- **Outdated & highly abstract** GPU core model

- Sub-core-level
  - **Limited** functional unit lanes
- SM-level
  - **Banked & sectored** L1 data cache
  - **Intra-SM load** imbalance
- GPU-level
  - **Inter-SM load** imbalance

Contents

- ~~Motivation and Background~~
- GCoM: A Detailed **G**PU **Co**re **M**odel
- Evaluation
- Conclusion

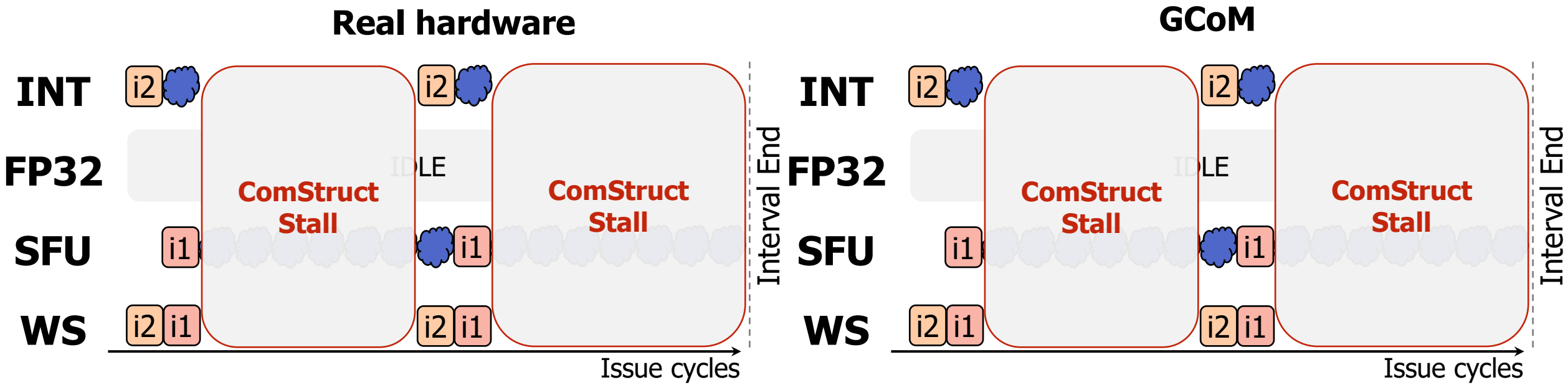
GCoM: Key Ideas

- Hierarchical **microarchitecture-driven** core modeling

- Sub-core level
  - Limited functional unit lanes ➔ Diverse per-FU **initiation intervals**
- SM-level
  - Banked L1 data cache ➔ Estimate the **effective BW** per interval
  - Sectored L1 data cache ➔ Extend **cache simulator**
  - Intra-SM load imbalance ➔ Estimate **WLP per sub-core**
- GPU-level
  - Inter-SM load imbalance ➔ **#warps per SM** considering TB scheduling

GCoM: Sub-Core-Level

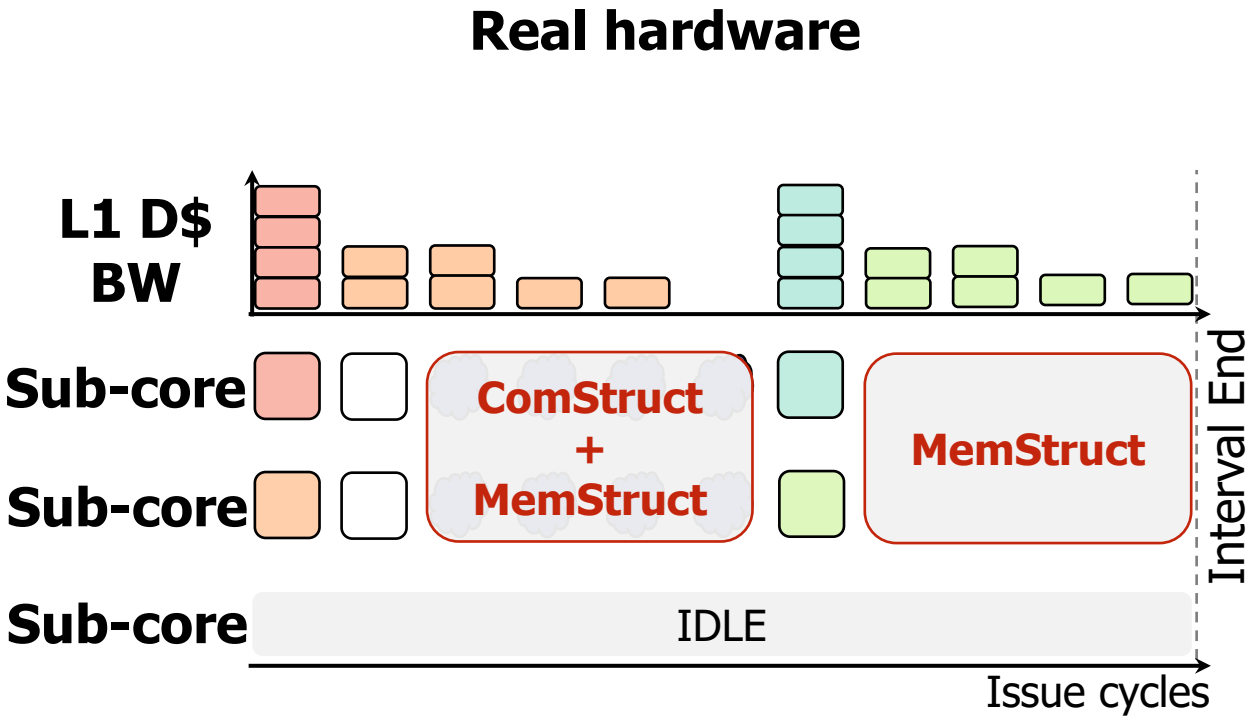
- Capture ComStruct stalls by considering FU lane counts
  - Initiation interval (II) = threadsPerWarp / #FULanes

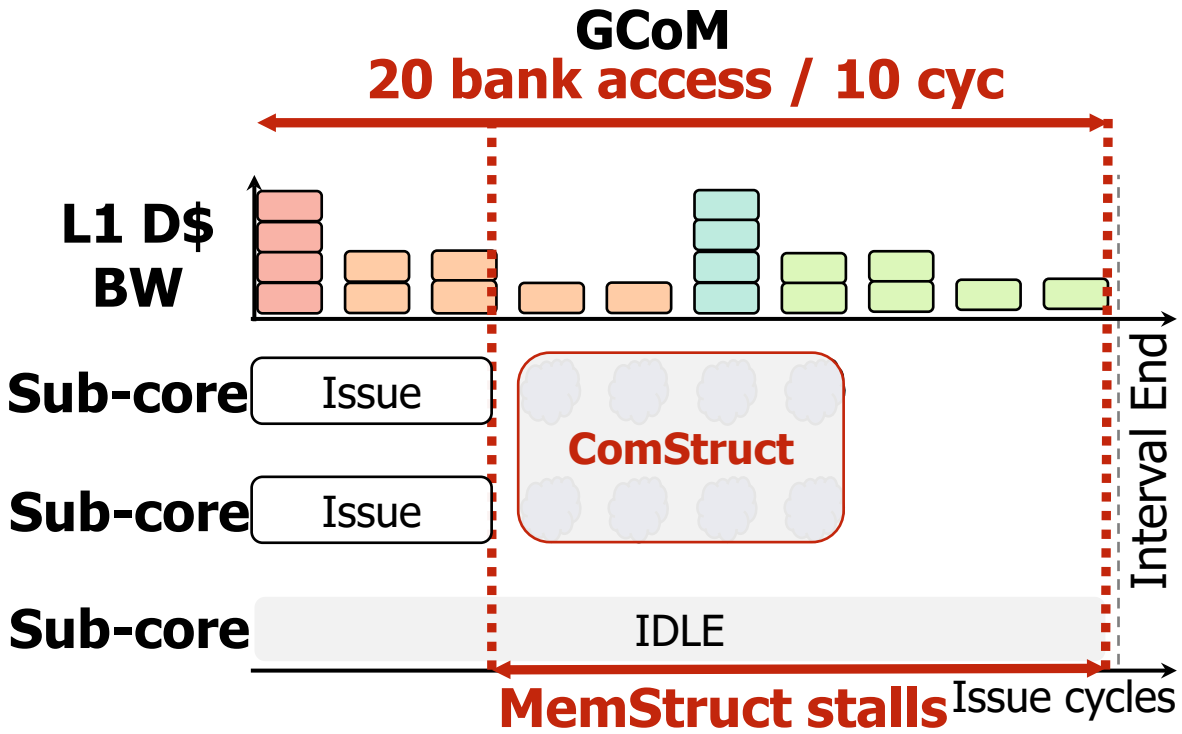


ComStruct = the longest FU's cycles - #insts / issueRate

GCoM: SM-Level Banked L1 D$

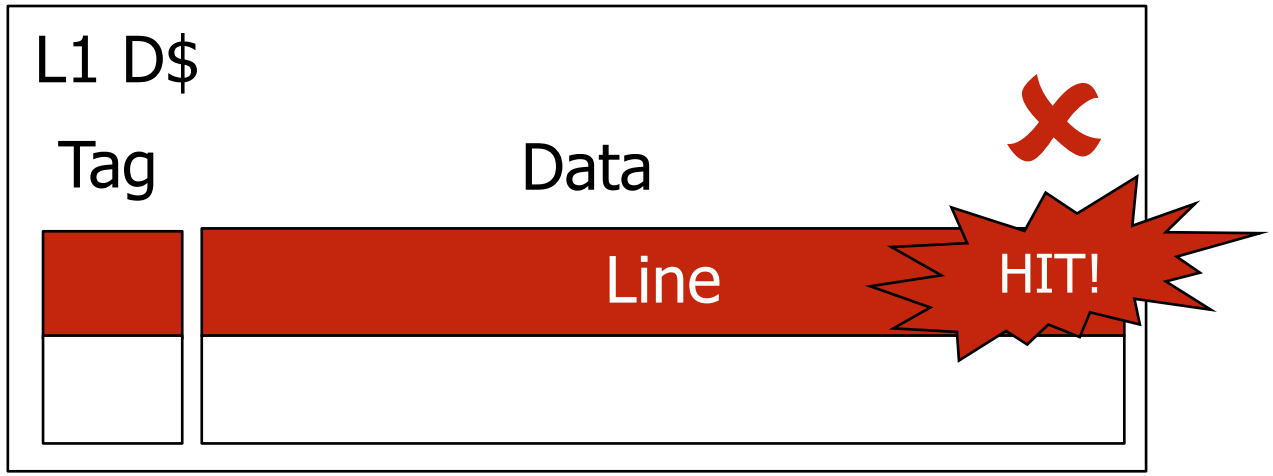
- Capture MemStruct stalls with effective BW per interval
  - Assume sub-cores access L1 D$ banks like the representative warp
- MemStruct stalls can hide ComStruct stalls of a sub-core.



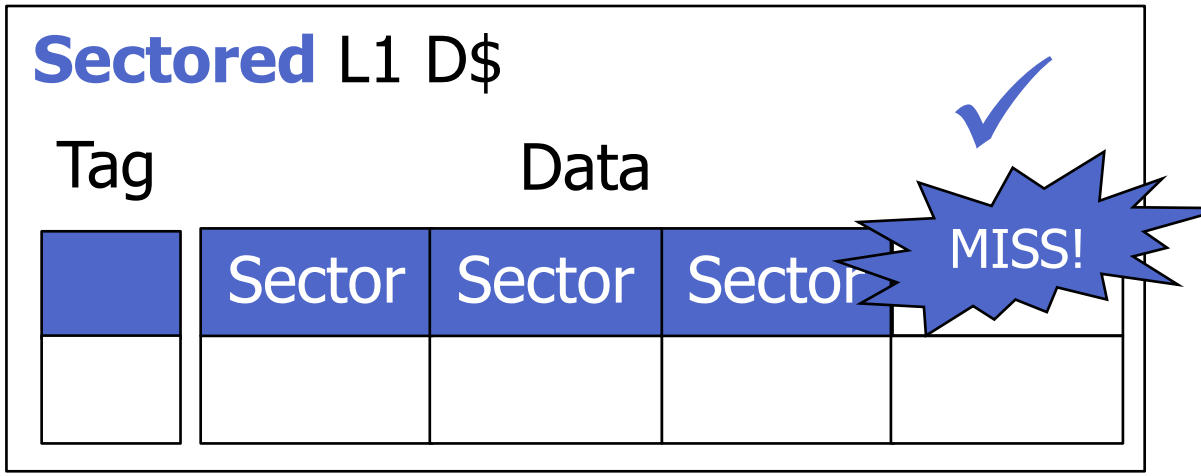


GCoM: SM-Level Sectored L1 D$

- Capture MemData stalls with the extended $ simulator
  - Model the sectored nature of L1 D$s
  - Provide per-sector hit/miss counts & memory coalescing



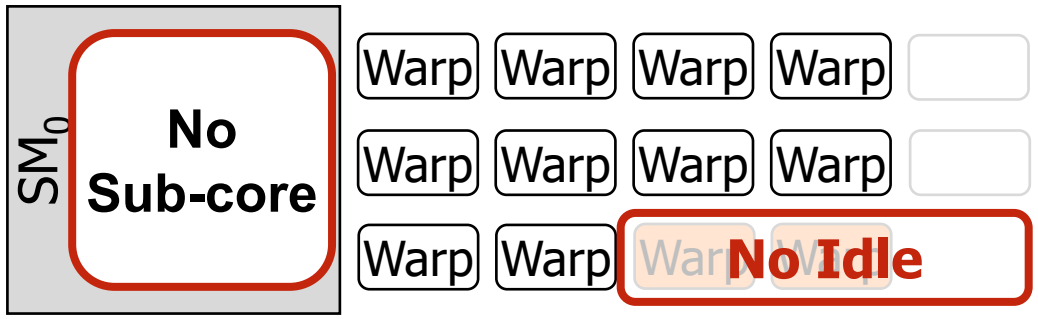
Naïve modeling



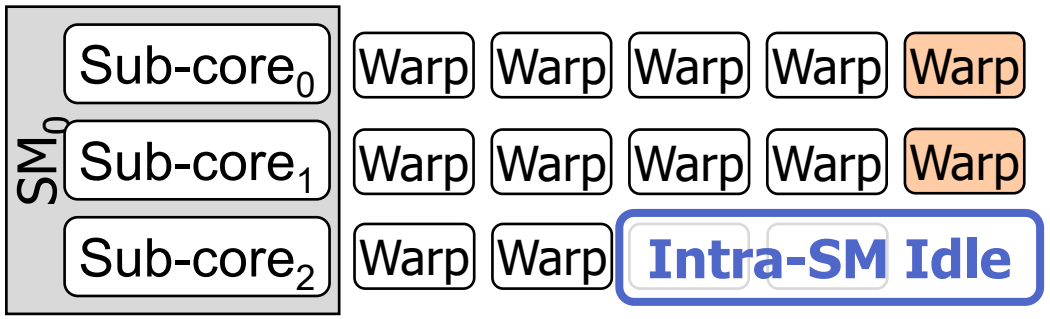
Real hardware & GCoM

GCoM: SM-Level Load Imbalance

- Capture intra-SM Idle stalls by estimating per-sub-core WLP
  - #warps per sub-core may differ



Naïve modeling



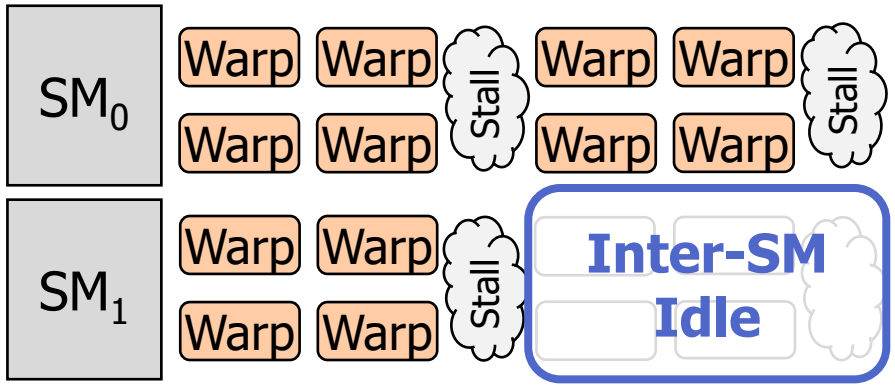
Real hardware & GCoM

Intra-SM Idle = sub-core with max #warps – others in the SM

GCoM: GPU-Level Load Imbalance

- Capture inter-SM Idle stalls using per-SM warp counts
  - Estimate per-SM warp counts with respect to TB scheduling
  - Perform & average interval analysis for each SM

Naïve modeling



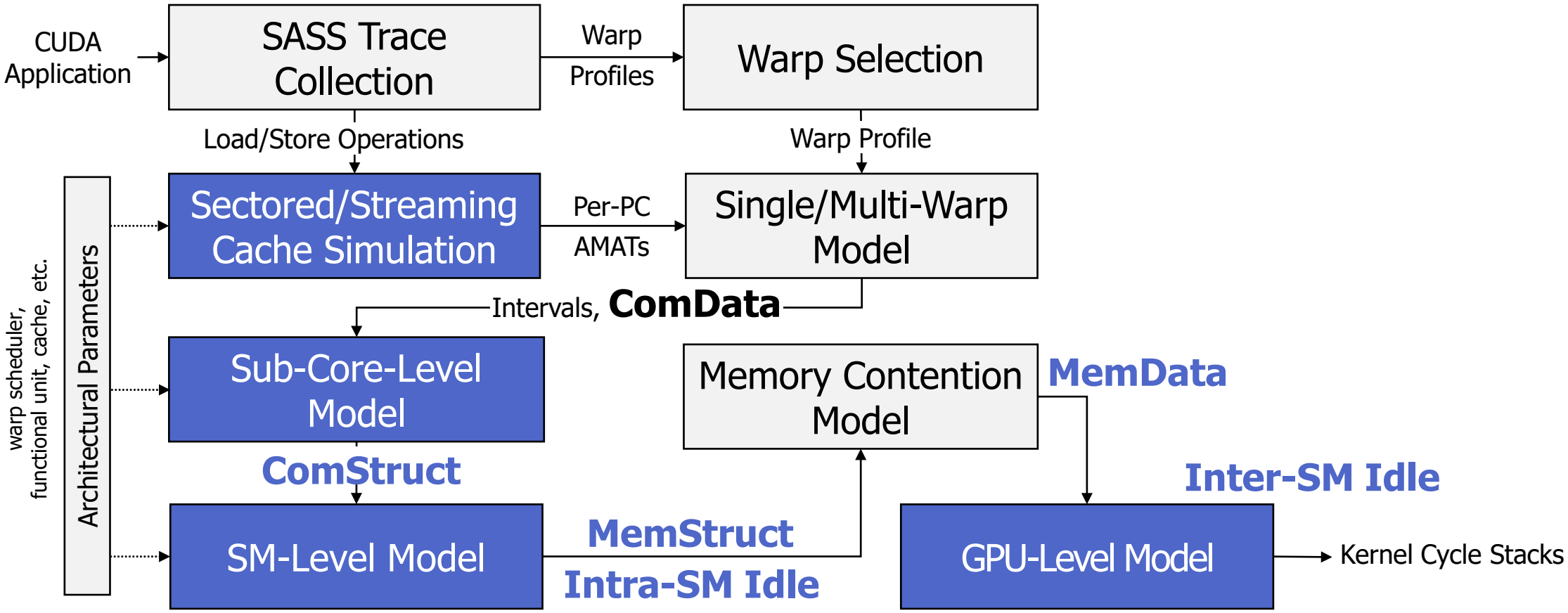
Real hardware & GCoM

Inter-SM Idle = the difference with the slowest SM

GCoM: Summary

- Sub-core-level
  - Limited FU lanes ➔ $C^{Issue}_{k,m}(x) = \frac{I_m \cdot II_m \cdot x}{\#ActSubCores(x) \cdot IssueRate} \quad (I_m \leq I_k)$
- SM-level
  - Banked L1 D$ ➔ $C^{Issue}_{k,L1}(x) = \lceil \frac{b_k}{B^{L1}_k} \rceil \times x$
  - Sectored L1 D$ ➔ $S^{NoC}_i + S^{DRAM}_i$
  - Sub-core load imbalance ➔ $\frac{\sum_{j=0}^{\#SubCoresSM-1} C_{i,j}}{\#SubCoresSM}$
- GPU-level
  - Inter-SM load imbalance ➔ $C^{Kernel} = \frac{\sum_{i=0}^{\#SMs-1} C^{Active}_i + C^{Idle}_i}{\#SMs}$

GCoM Framework



Contents

- ~~Motivation and Background~~
- ~~GCoM: A Detailed **G**PU **Co**re **M**odel~~
- Evaluation
- Conclusion

Experimental Setup

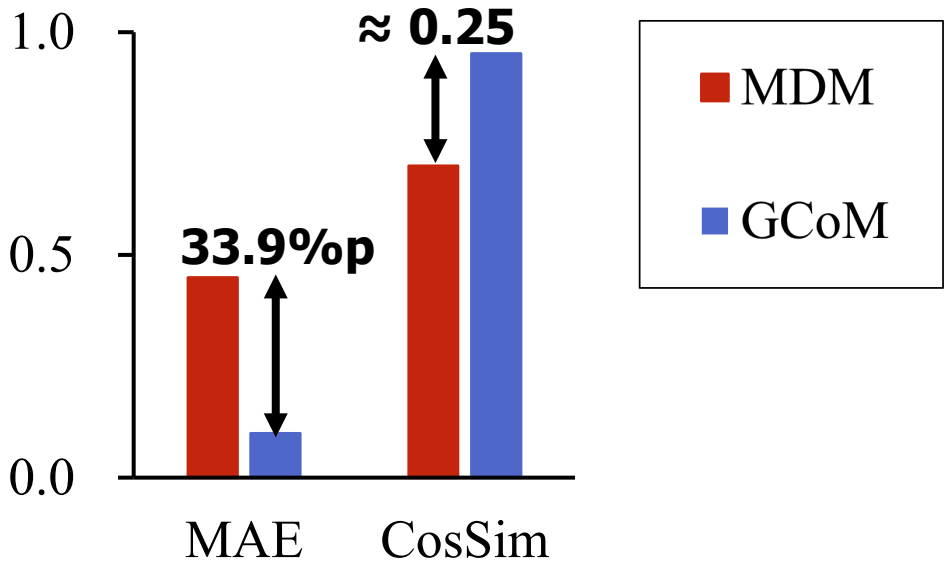
- **GCoM vs. MDM** [MICRO '20] against **Accel-sim** [ISCA '20]
- Build CPI stacks with **GPU Stall Inspector** [ISPASS '16]

- **NVIDIA RTX 2060** configuration
  - Validation against real NVIDIA RTX 2060, RTX 2060 Super, and RTX 3070

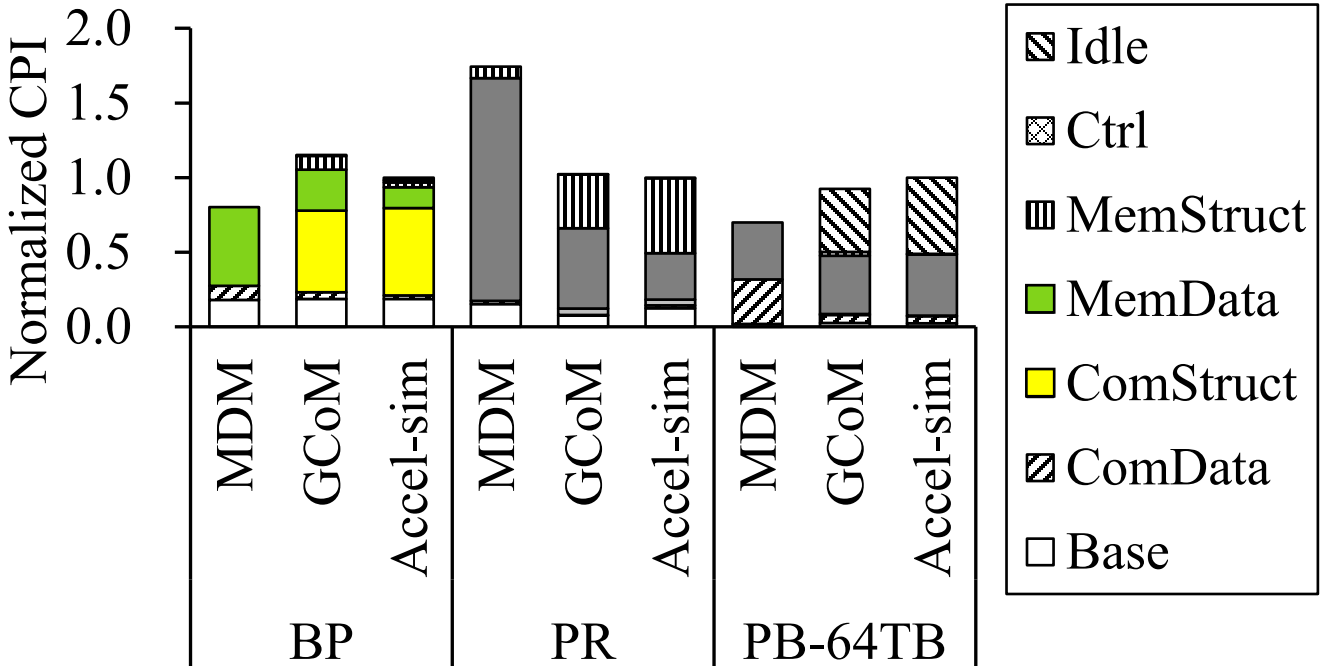
- **31 applications** from various benchmark suites
  - Rodinia, Pannotia, DeepBench, MLPerf, Polybench, Tango
  - Including irregular graph processing, BERT, Tensor Cores

Accurate GPU Modeling

- GCoM significantly improves modeling accuracy.
  - MAE of **10.0 %** vs. MDM's **44.9%**
- GCoM also builds more accurate CPI stacks.
  - Cosine similarity of **0.95** vs. MDM's **0.70**



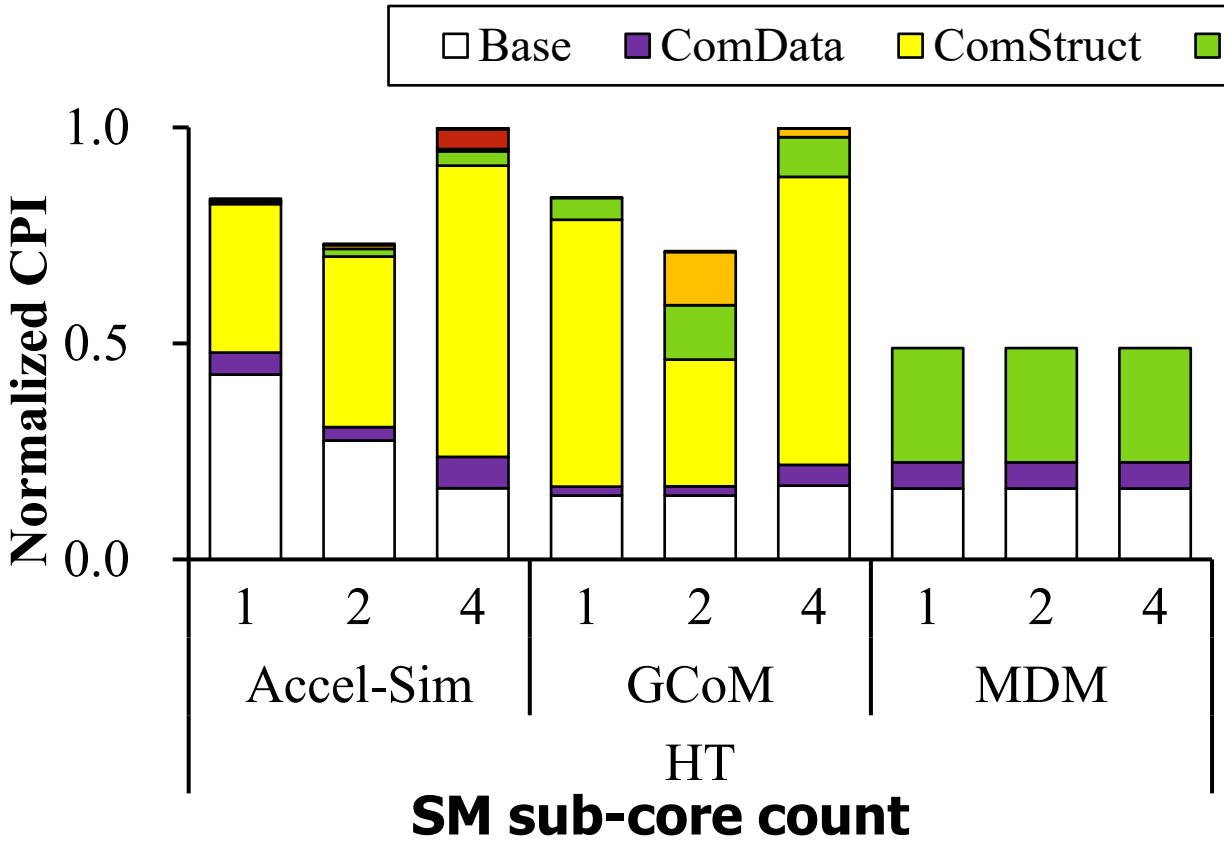
Average accuracy of 31 applications

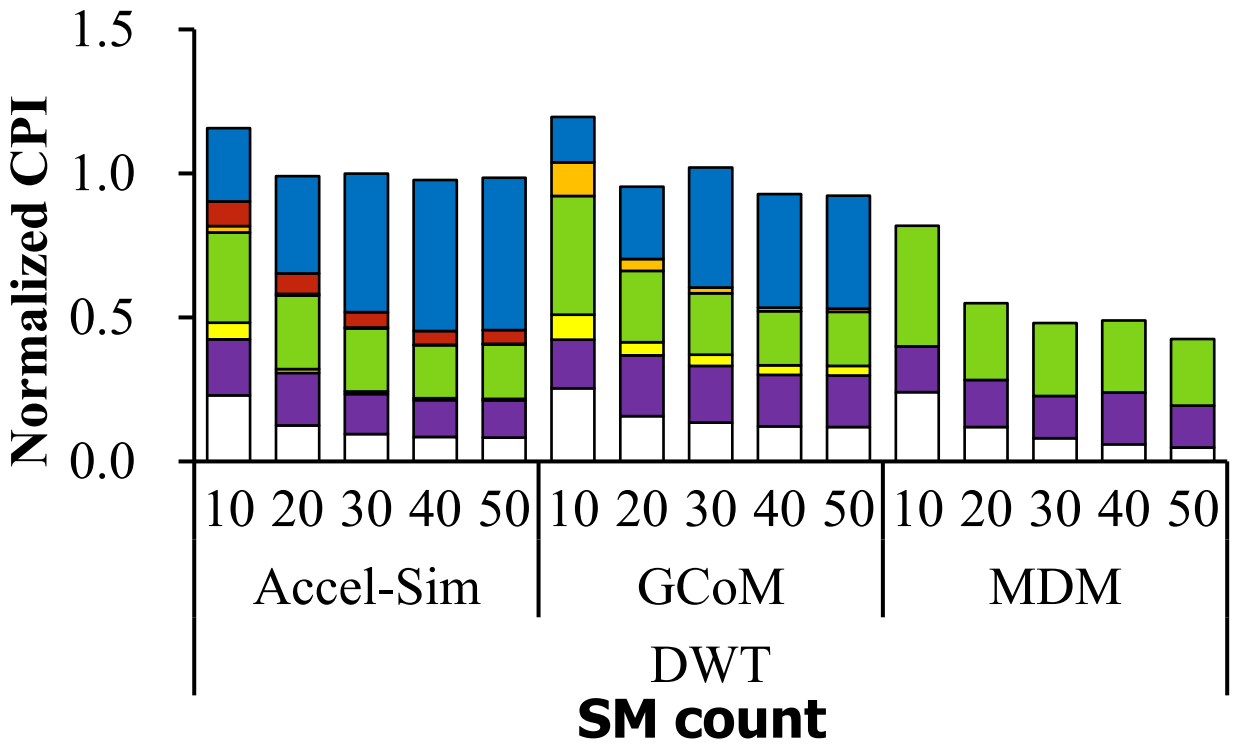


CPI stacks of the representative applications

Accurate Design Space Explorations

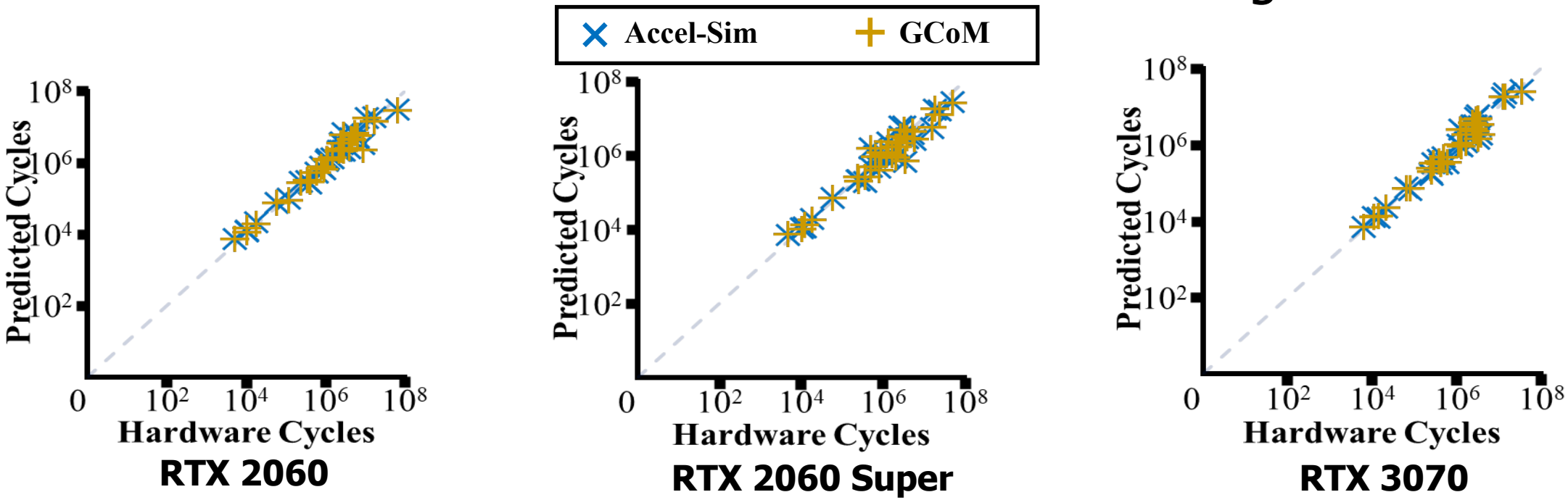
- GCoM accurately captures performance trends.
  - **Per-SM sub-core count**, **SM count**, sectored/non-sectored L1 D$, L1 D$ bank counts, L1 D$ sizes, ...





Accurate Real GPU Modeling

- MAE of **24.4%** against RTX 2060
  - Comparable to Accel-sim
- Applicable to other GPUs: RTX 2060 Super and RTX 3070
  - Can use traces of RTX 2060 to avoid time-consuming trace collection



Conclusion

- Hierarchical microarchitecture-driven GPU core modeling
  - **Sub-core-level**: Limited FU lanes
  - **SM-level**: Banked and sectored L1 D$, intra-SM load imbalance
  - **GPU-level**: Inter-SM load imbalance

- **GCoM**: a **fast** and **accurate** GPU performance model
  - MAE of **10.0%** against Accel-sim, **24.4%** against RTX 2060
  - **345.1x** faster than Accel-sim

Thank You!

- Any questions?
- We are planning to release GCoM soon!

Jounghoo Lee @ Yonsei University
jounghoolee@yonsei.ac.kr
https://hpcp.yonsei.ac.kr/