Mocktails: Capturing the Memory Behaviour of Proprietary Mobile Architectures

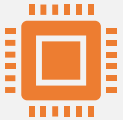
Mario Badr[1], Carlo Delconte[2], Isak Edo[1], Radhika Jagtap[2], Matteo Andreozzi[2], and Natalie Enright Jerger[1]

[1] University of Toronto
[2] Arm

What is Mocktails?

- An open-source tool for synthesizing memory requests
  - Focuses on proprietary mobile workloads running on specialized architectures
- Industry: Generate and distribute models
  - Models hide industry secrets and proprietary details of the workload and architecture
  - Models are **84%** smaller than trace files
- Academia: Generate memory requests from the models to evaluate memory
  - Maximum **7.8%** error on read row hits and **2.8%** error on write row hits

Presentation Outline



Background

- Heterogeneity
- Memory hierarchy
- Statistical simulation

Mocktails

- Modeling Requests
- Uncovering patterns
- Synthesizing

Evaluation

- Memory controller

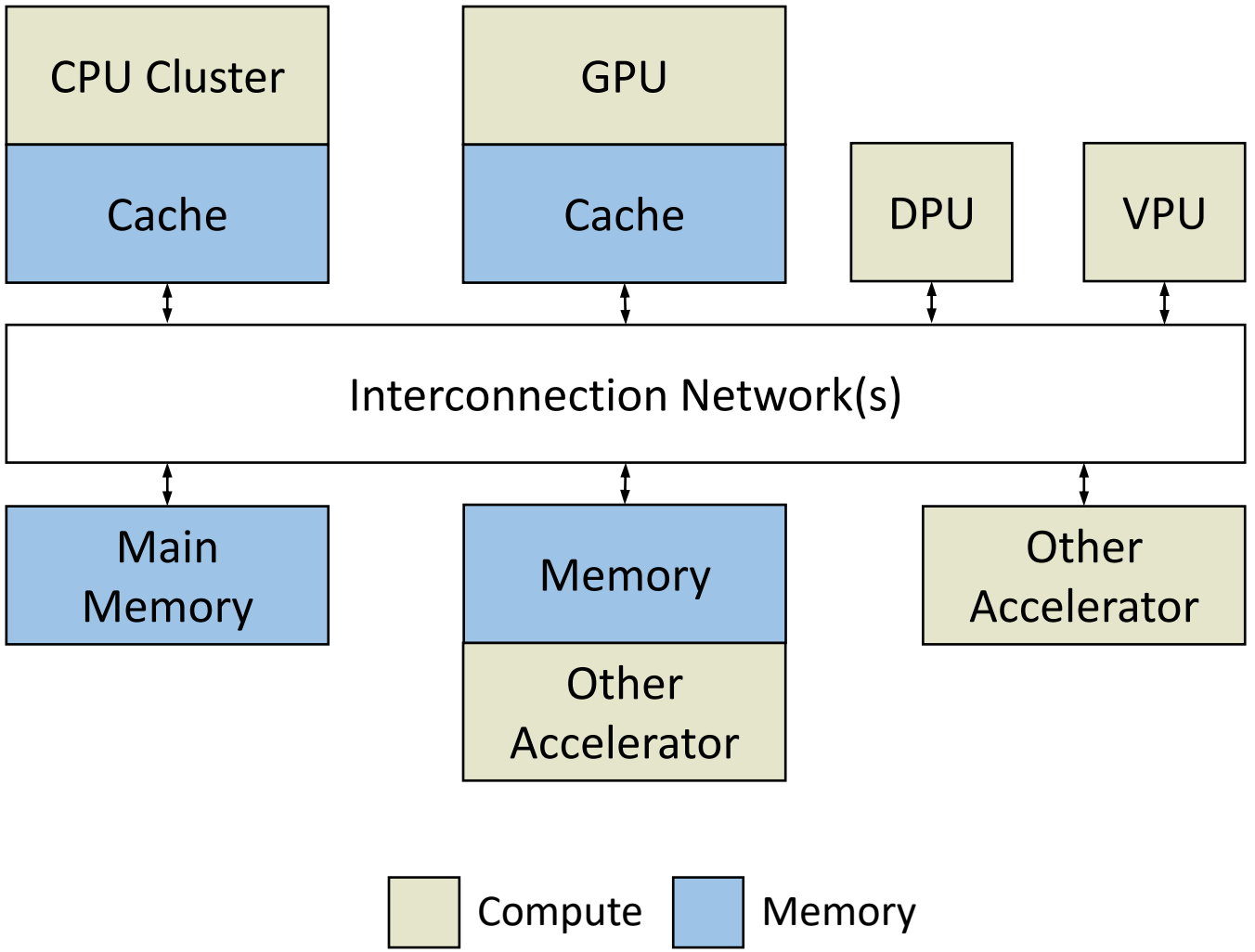
Conclusion

- Summary of results

A System-on-Chip

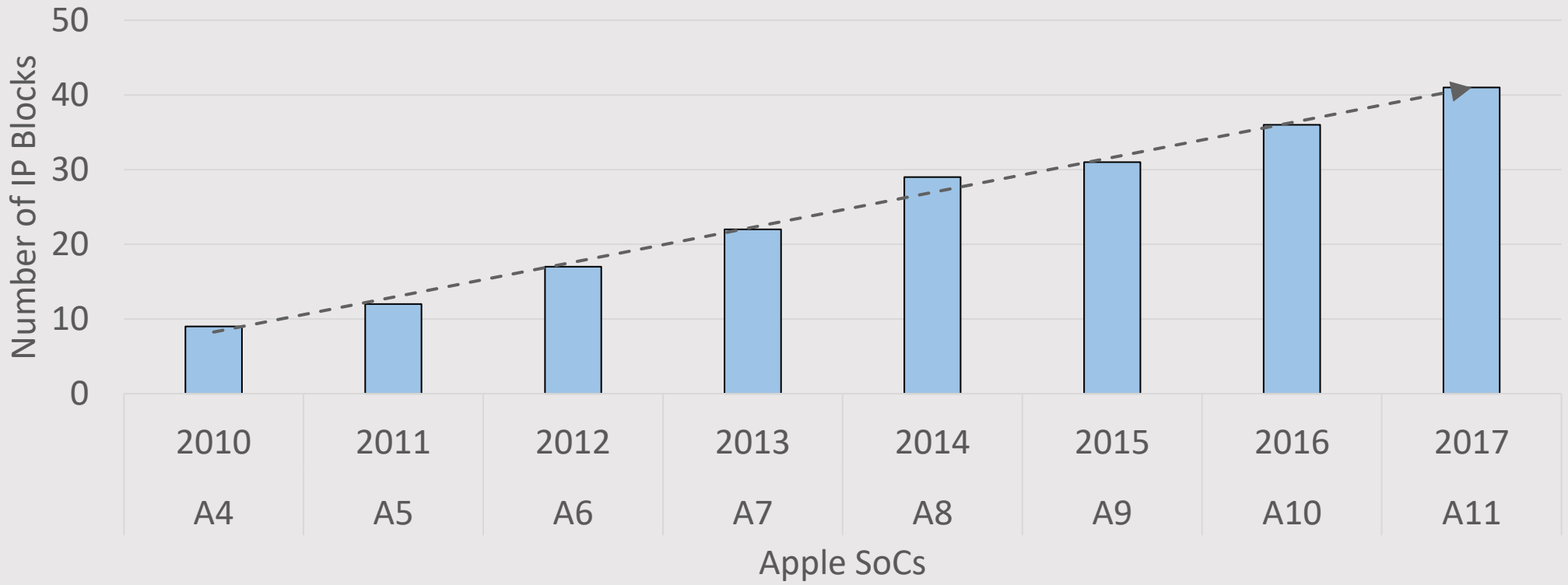
- General-purpose cores (CPU)
- Graphics-Processing Unit (GPU)
- Display-Processing Unit (DPU)
- Video Processing Unit (VPU)

Significant real estate allocated to accelerators.



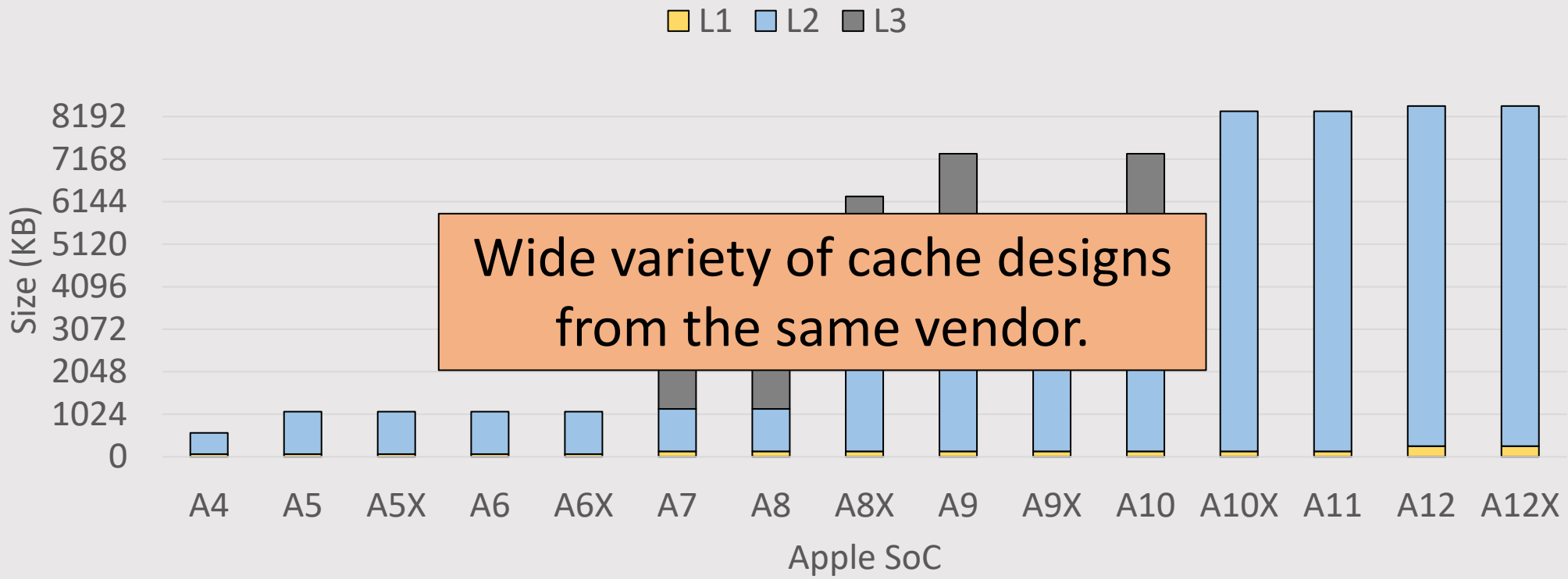
Based on data from: Y. S. Shao, B. Reagen, G. Wei, and D. Brooks, "The Aladdin approach to accelerator design and modeling," IEEE Micro, 2015.

Heterogeneity is Increasing



Based on data from: https://en.wikipedia.org/wiki/Apple-designed_processors

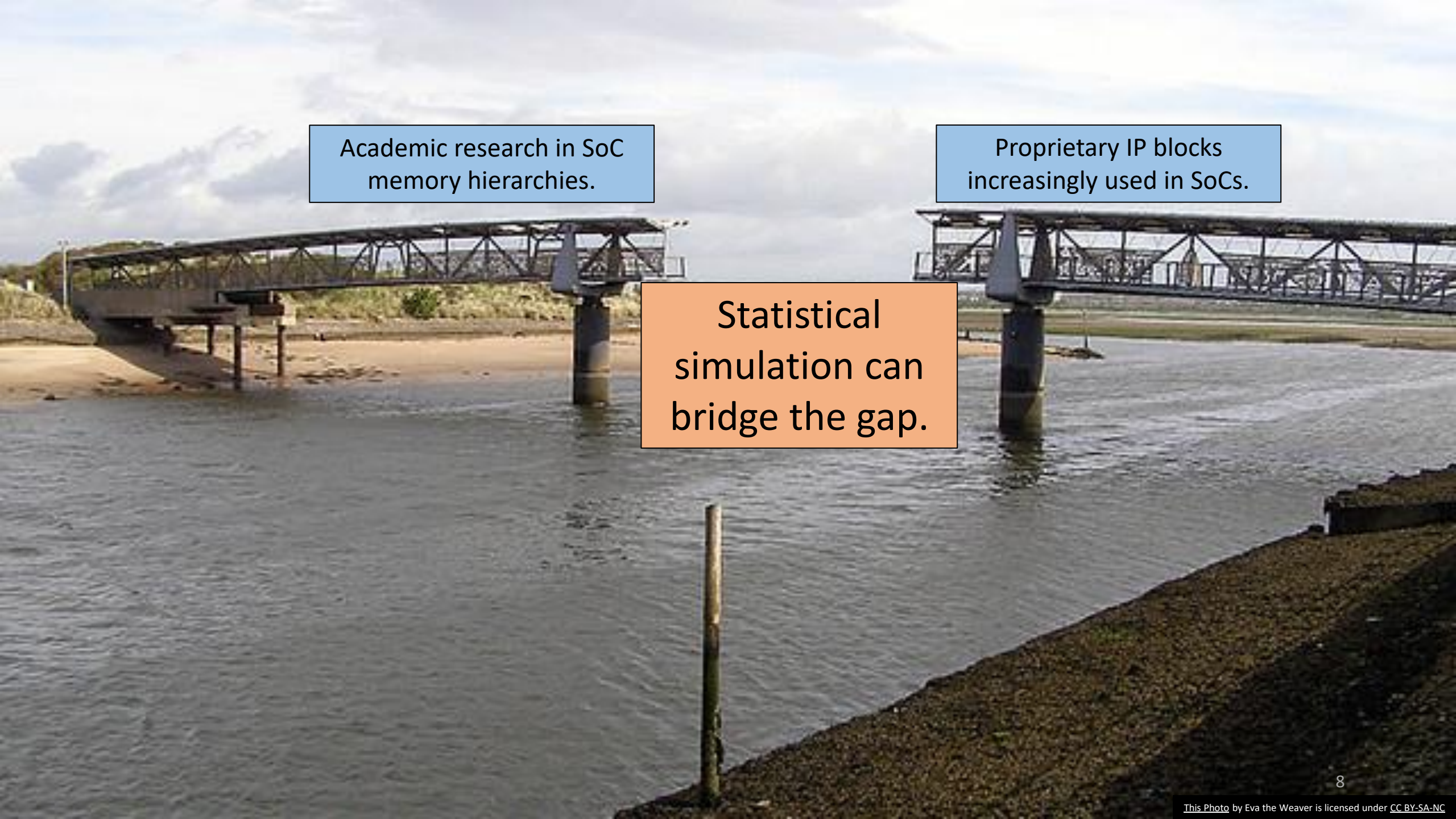
Apple SoCs – The Cache Hierarchy



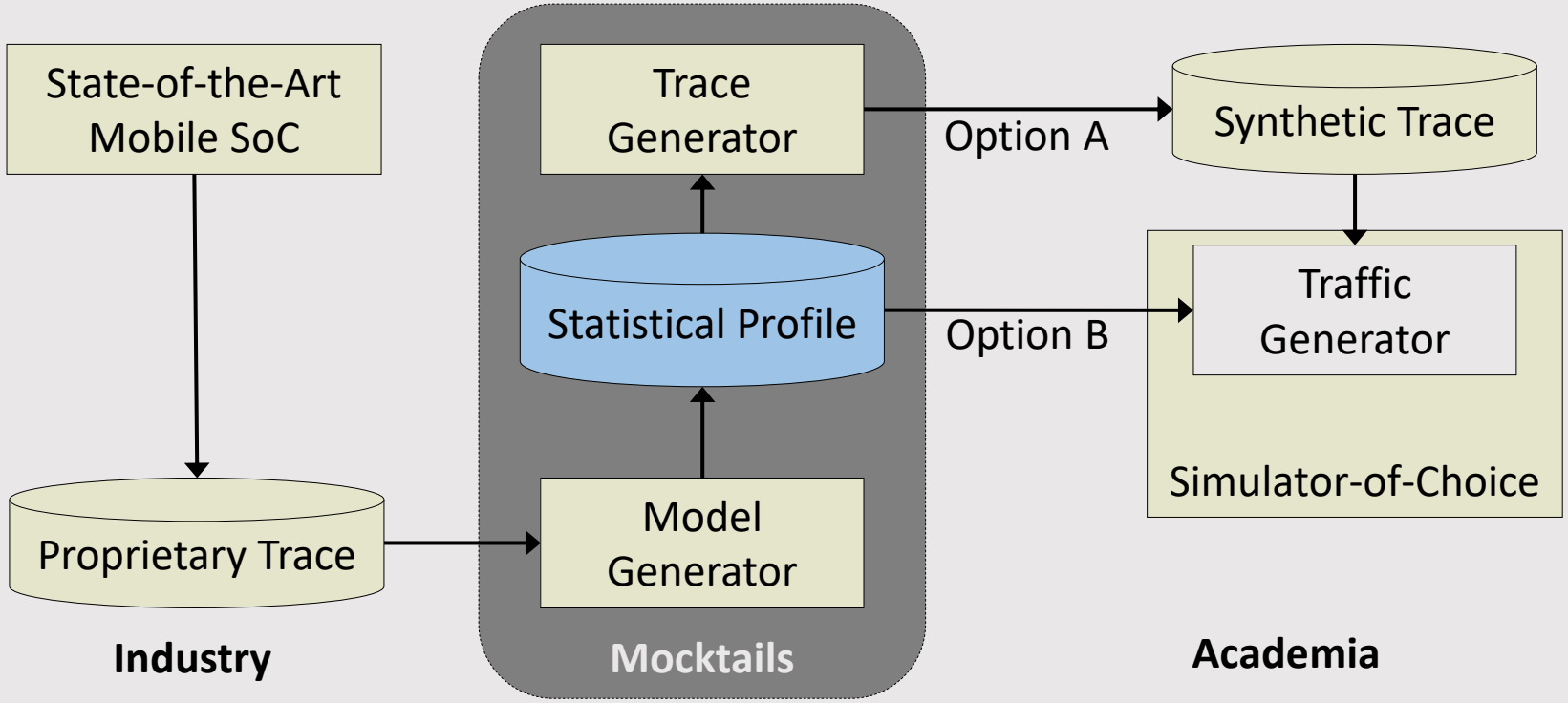
Heterogeneous Systems-on-Chip

- Specialized hardware for commonly used workloads
- Spend area to b iciency
- More IP blocks = varying demands on memory

How do we evaluate the memory hierarchy?

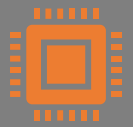


Statistical Simulation



Statistical profiles hide proprietary details.

Up Next: Mocktails



Background

- Heterogeneity
- Memory hierarchy
- Statistical simulation

Mocktails

- Modeling Requests
- Uncovering patterns
- Synthesizing

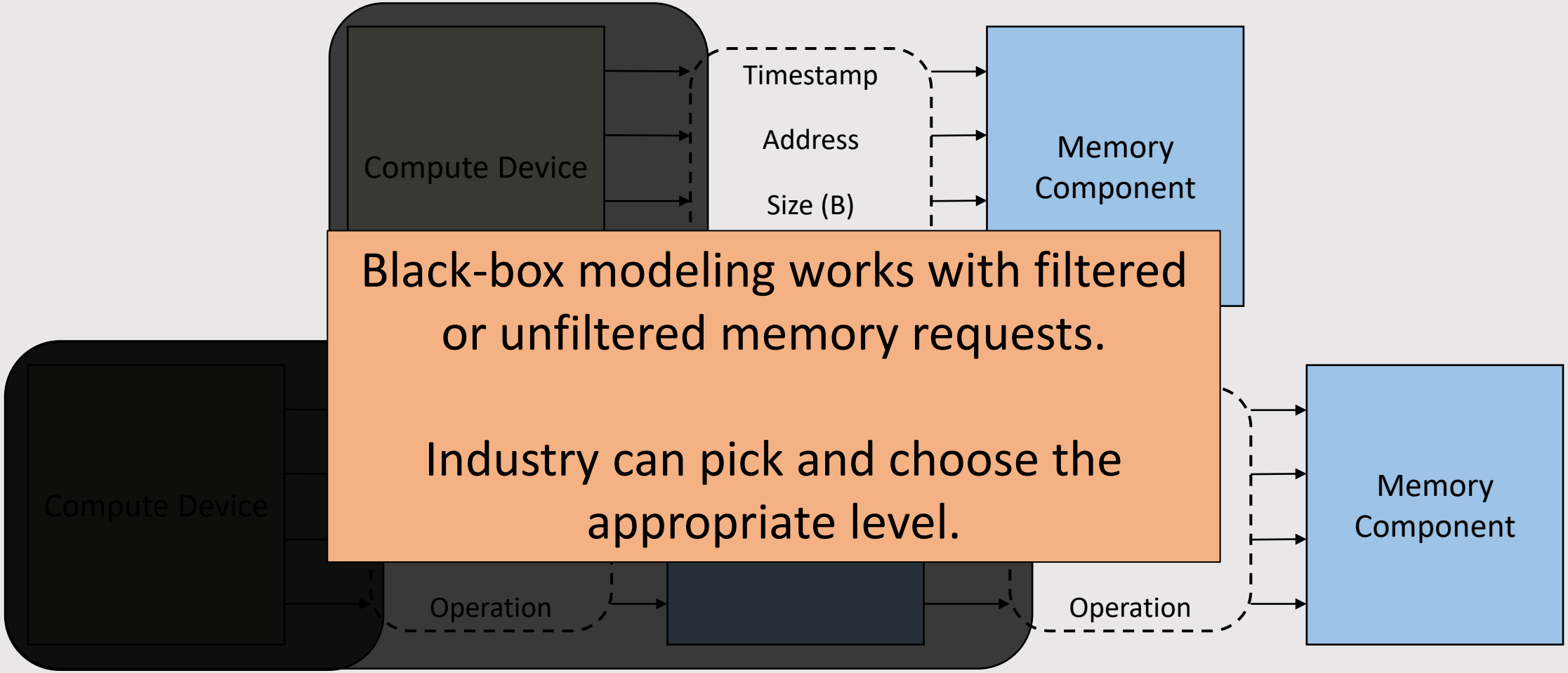
Evaluation

- Memory controller

Conclusion

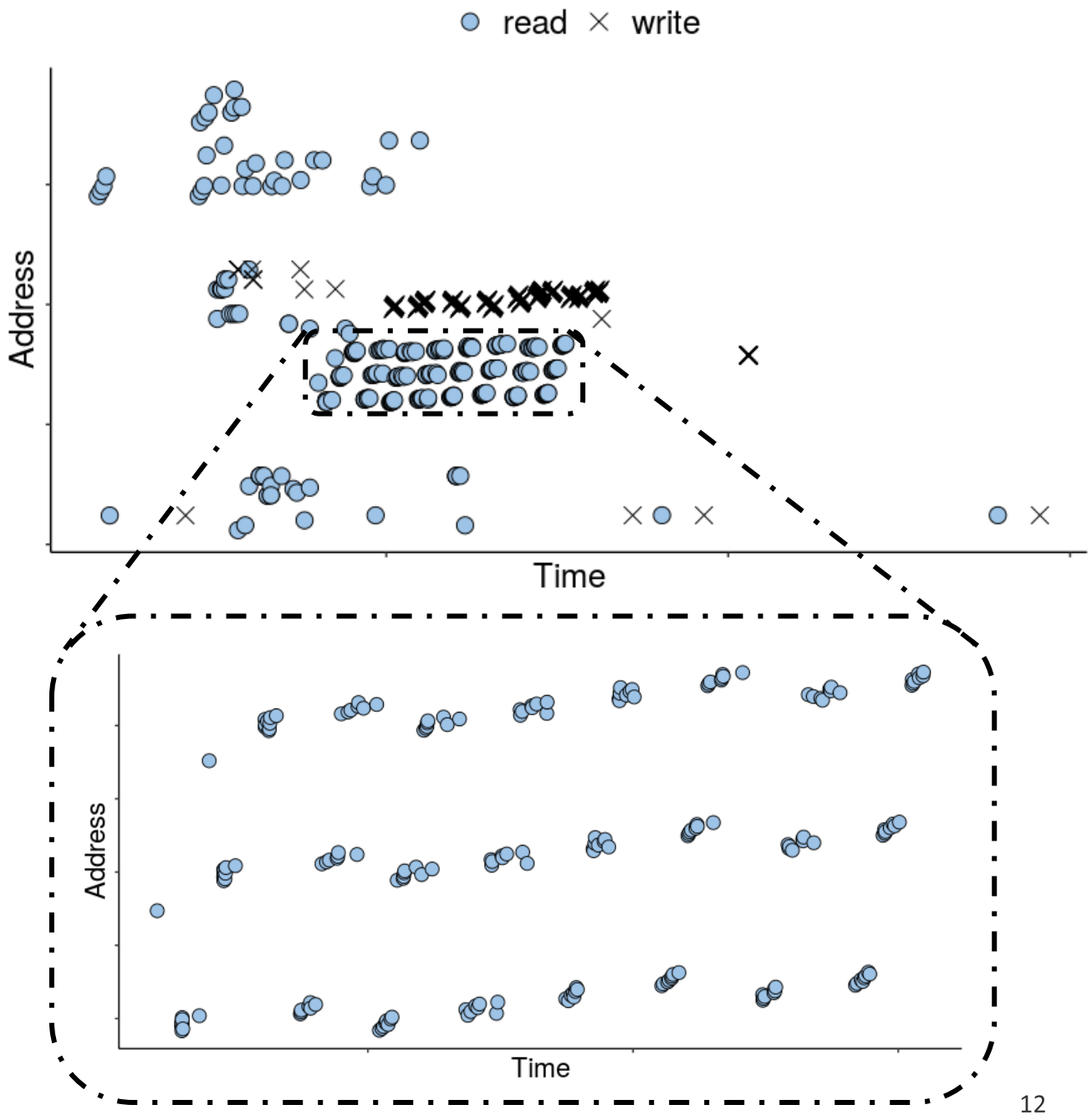
- Summary of results

Modeling a Memory Request



An Example Workload

- Lots of variability
- Hard to find a pattern in the memory accesses
- We can zoom in



Modeling Addresses

- Given a starting address, model th strides
- Stride models used in prior art

Time	Address	Stride
1	D	
2	A	-12
3	C	8
4	B	-4
		4
		0
		16896
8	X	-8
9	Y	4
10	A	-16884
11	Y	16884
12	B	-16880

High variability in strides is difficult to model accurately.

Temporal Partitioning

- Divide the sequence of requests in two
  - Two starting add
  - Two stride mode
- Each partition has different behaviour
- Temporal partitioning used in prior art

	Time	Address	Stride
Time Interval 1	1	D	
	2	A	-12
	3	C	8
	4	B	-4
			4
			0
	8	X	-8
Time Interval 2	9	Y	4
	10	A	-16884
	11	Y	16884
	12	B	-16880

Time interval 2 has high variability in stride values.

Spatial Partitioning

- Divide requests into separate address ranges
- Each partition has different behaviour
- Spatial partitioning used in prior art
  - But tuned for CPUs

Spatial Partition 1

Time	Address	Stride
1	D	
2	A	-12
3	C	8
4	B	-4
5	C	4
		0
		-8
		4

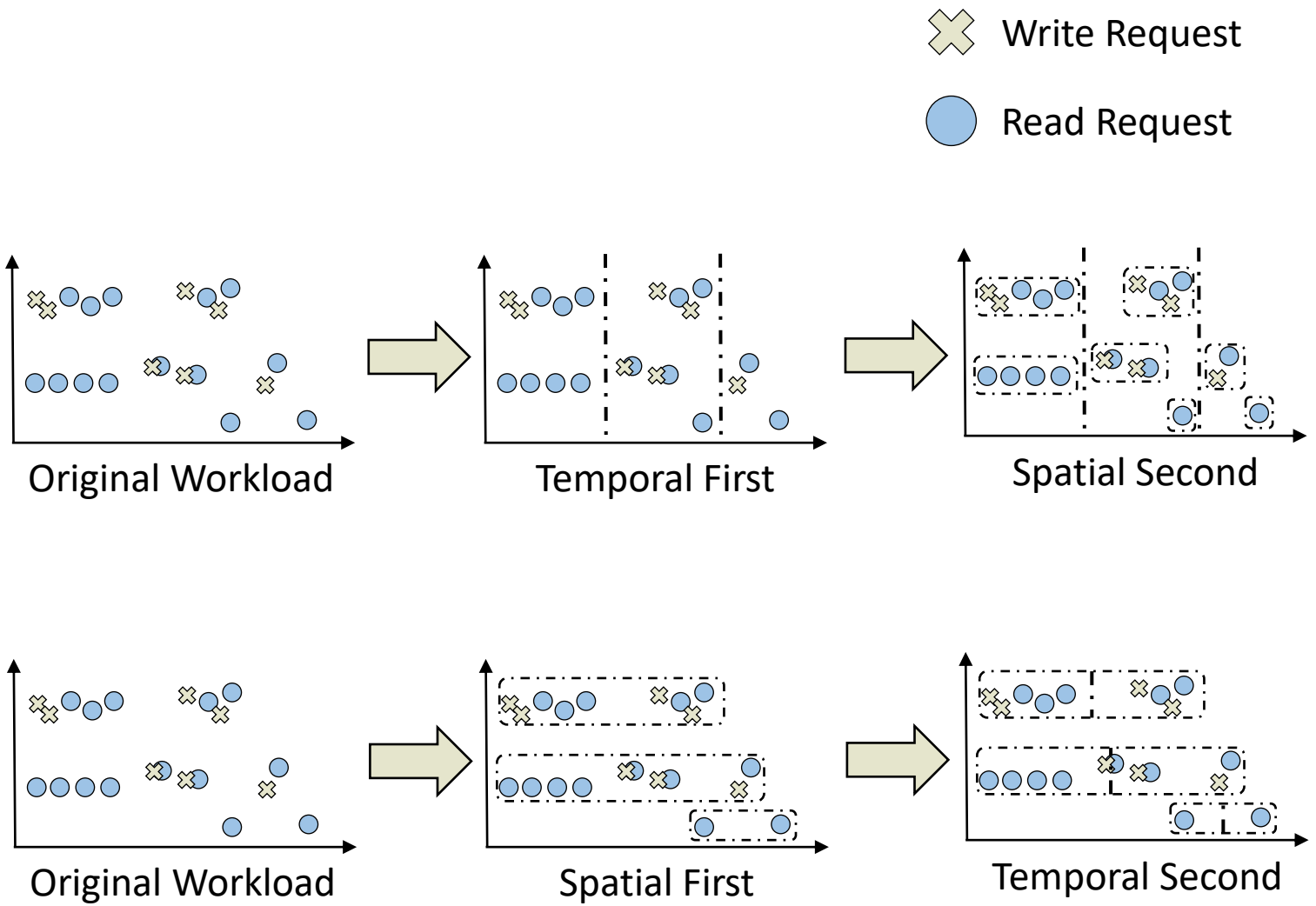
Spatial Partition 2

Time	Address	Stride
7	W	
8	X	-8
9	Y	4
11	Y	0

Spatial partitioning reduces the variance in the stride feature for both partitions.

Partitioning in Two Dimensions

- Temporal: reduces variability in delta times
- Spatial: reduces variability in strides



Carefully Dividing Requests

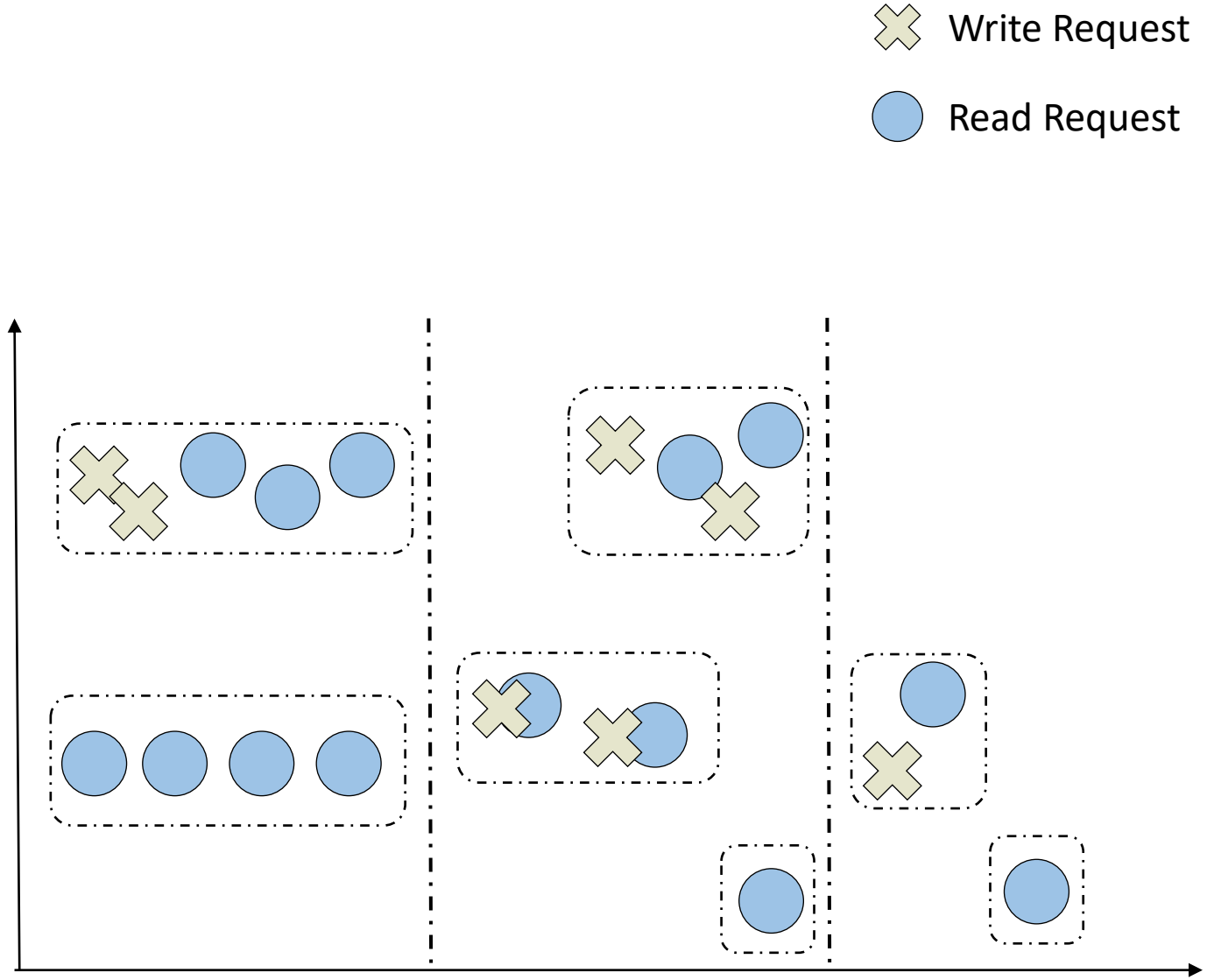
- Dynamic Spatial Partitioning
  - Find reque

Dynamic spatial partitioning adapts to the memory access behaviour of the workload and device.

- Spatial partitioning uncovers variable-sized time intervals
  - Phases with different start times and durations

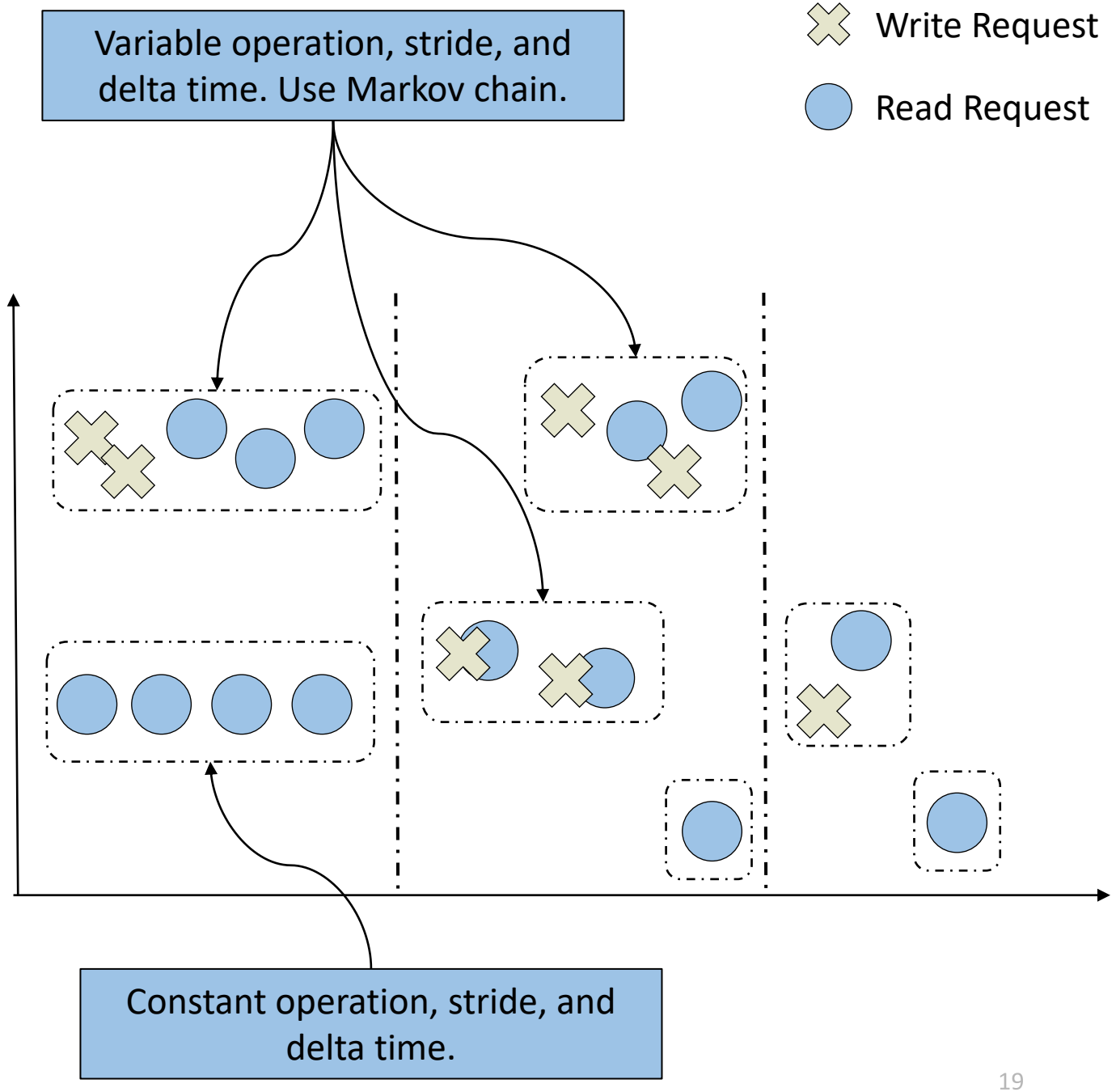
Modeling Each Partition

- Each partition consists of a sequence of memory requests
- Model each partition independently
- Save each partition's:
  - Start time
  - Start address



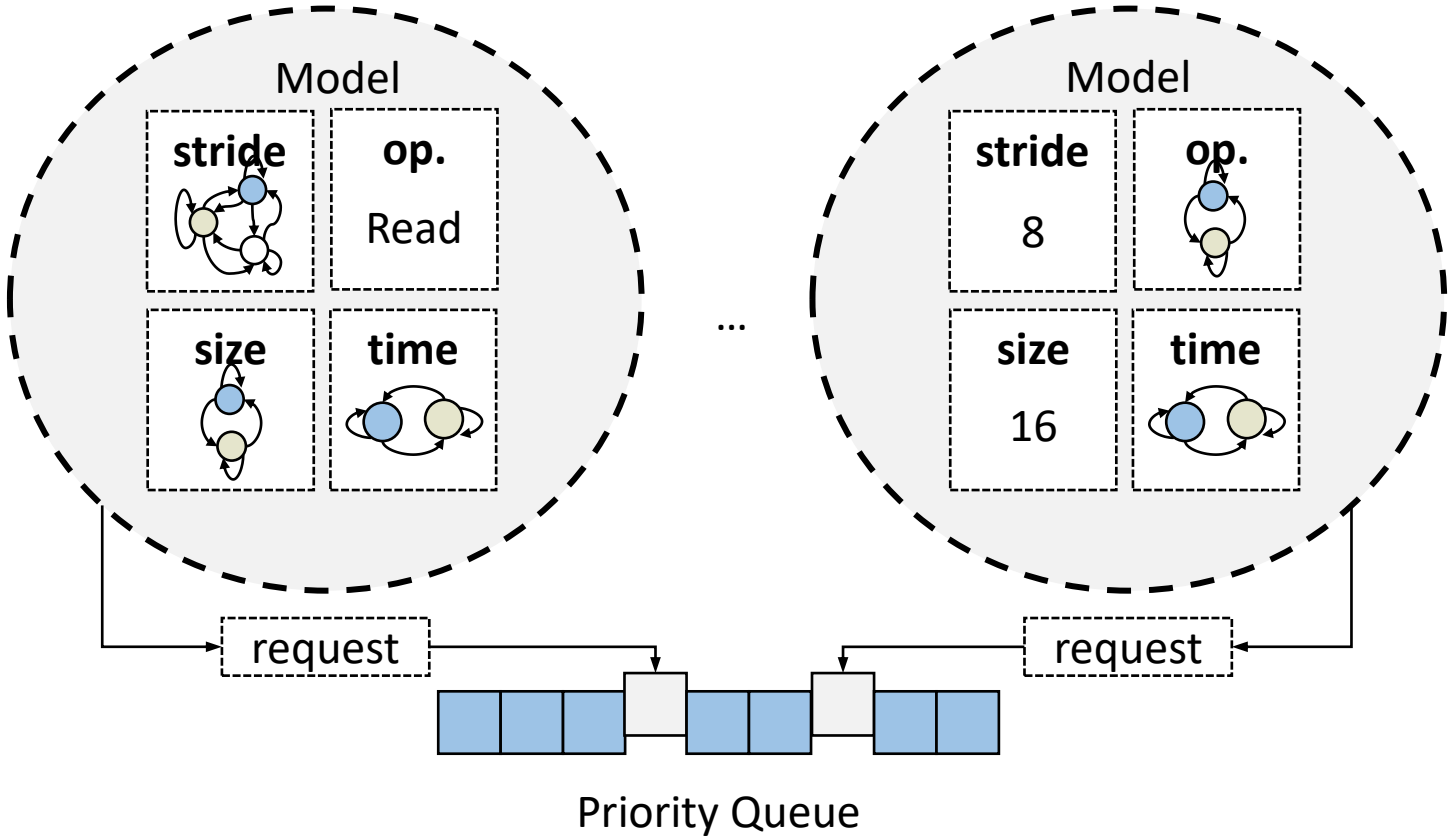
Modeling Each Feature

- Model each feature independently
- Features that do not change:
  - Constant value
- Features that do change:
  - Markov chain

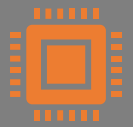


Synthesizing Requests

- Each model is used to generate requests
  - Need initial time and address
- Requests pushed into a priority queue
  - Ordered by timestamp



Up Next: Evaluation



Background

Heterogeneity
Memory hierarchy
Statistical simulation

Mocktails

Modeling Requests
Uncovering patterns
Synthesizing

Evaluation

Memory controller

Conclusion

Summary of results

Methodology

- Proprietary memory access traces from Arm
  - CPU, DPU, GPU, VPU devices

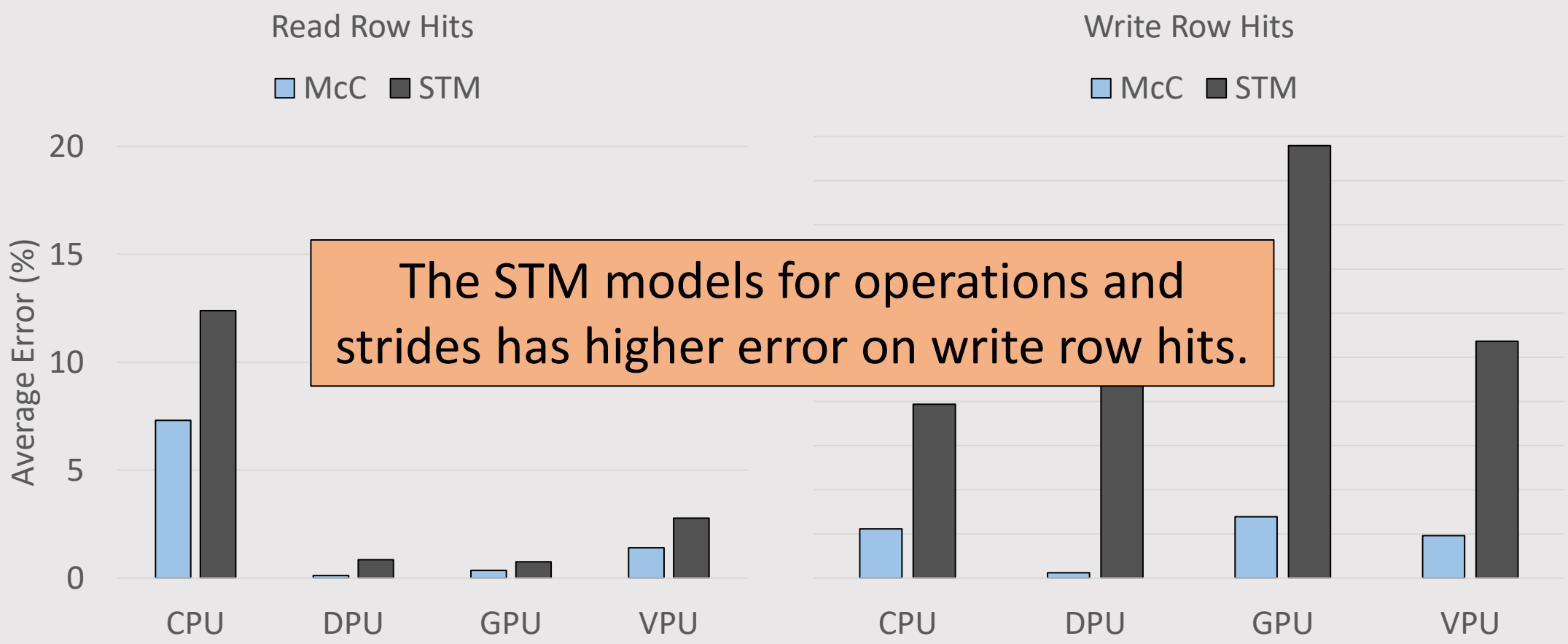
- Trace-based simulation with gem5
  - Baseline: Arm traces
  - Requests sent to main memory over a crossbar

- Memory Controller Configuration
  - 4 channels: Each channel has a read and write queue
  - Dynamic scheduling: First-ready, first-come first-serve

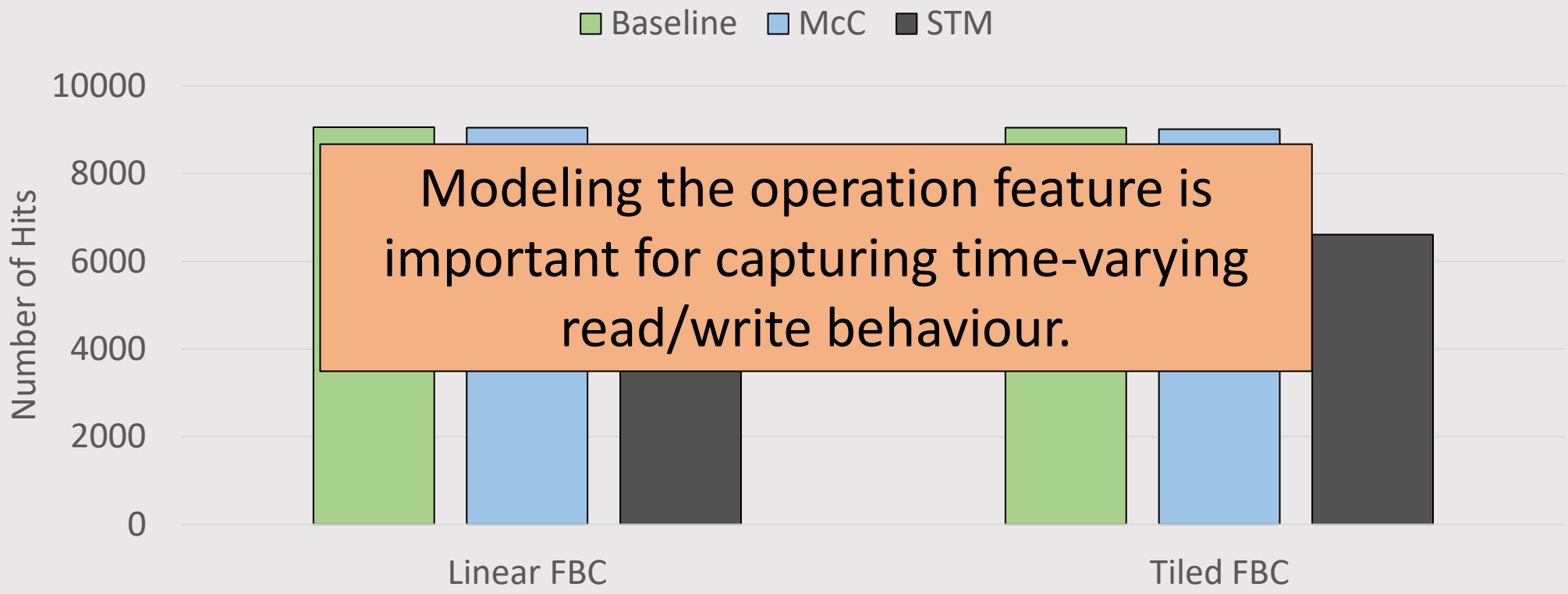
Model Comparison

- Perform hierarchical partitioning
  - Model each partition with two different approaches
  - Configuration: Temporal then Spatial (i.e., 2L-TS)
  - 500,000 cycle time intervals
- Mocktails Approach
  - Markov chain or Constant value for each feature (i.e., the McC model)
- STM Approach
  - A statistical simulation technique for the CPU
  - Weighted coin flip for operation feature
  - Markov chain with history for stride feature
  - Other features use Mocktails approach

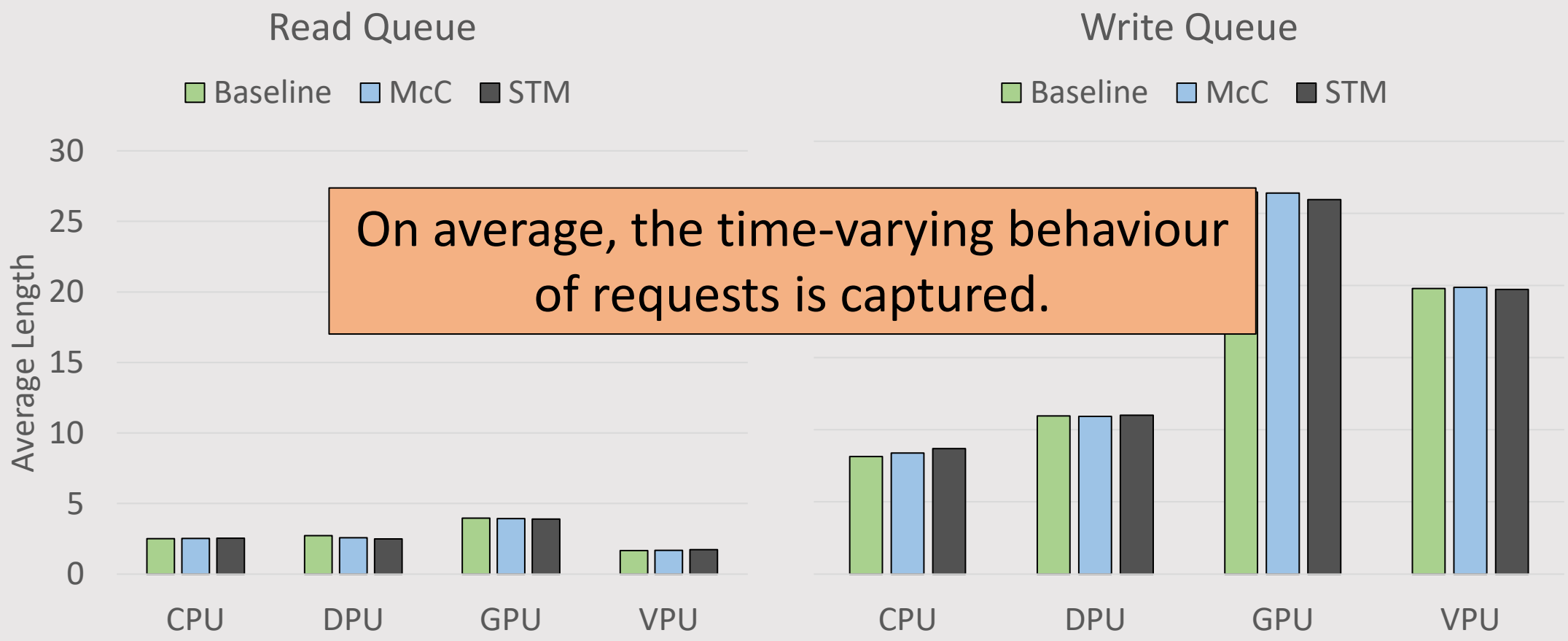
Absolute Accuracy of Row Hits



Write Row Hits (DPU)

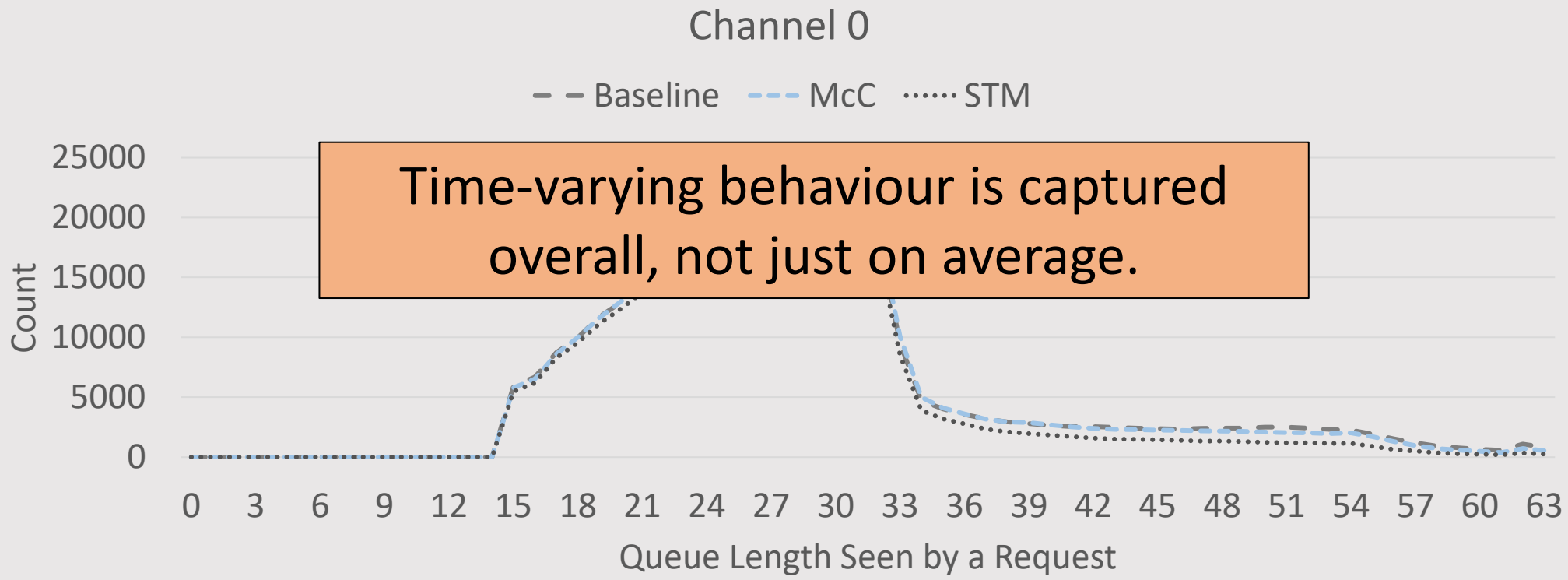


Average Queue Length

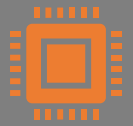


On average, the time-varying behaviour of requests is captured.

Write Queue Length (Manhattan GPU)



Up Next: Conclusion



Background

- Heterogeneity
- Memory hierarchy
- Statistical simulation

Mocktails

- Modeling Requests
- Uncovering patterns
- Synthesizing

Evaluation

- Memory controller

Conclusion

- Summary of results

Summary of Results

- Mocktails is accurate for proprietary mobile applications that use specialized hardware
  - Maximum **7.8%** error on read row hits and **2.8%** error on write row hits
- Mocktails is accurate for traditional CPU benchmarks
  - SPEC CPU2006: **5.6%** average error on L1 cache miss rates
  - More details and exploration in the paper
- Mocktails profiles are distributable
  - **84%** smaller than trace files
  - Do not include proprietary details

Mocktails is Open Source

- https://github.com/mariobadr/statistical-simulation

- Mocktails Models of proprietary IP blocks

- Source code for…
  - Mocktails model/trace generation
  - Prior work comparisons (i.e., STM, HRD)
  - gem5 integration
  - Miscellaneous utilities and libraries