Mosaic: Harnessing the Micro-architectural Resources of Servers in Serverless Environments

MICRO 2024

Jovan Stojkovic, Esha Choukse*, Enrique Saurez*, Íñigo Goiri*, Josep Torrellas

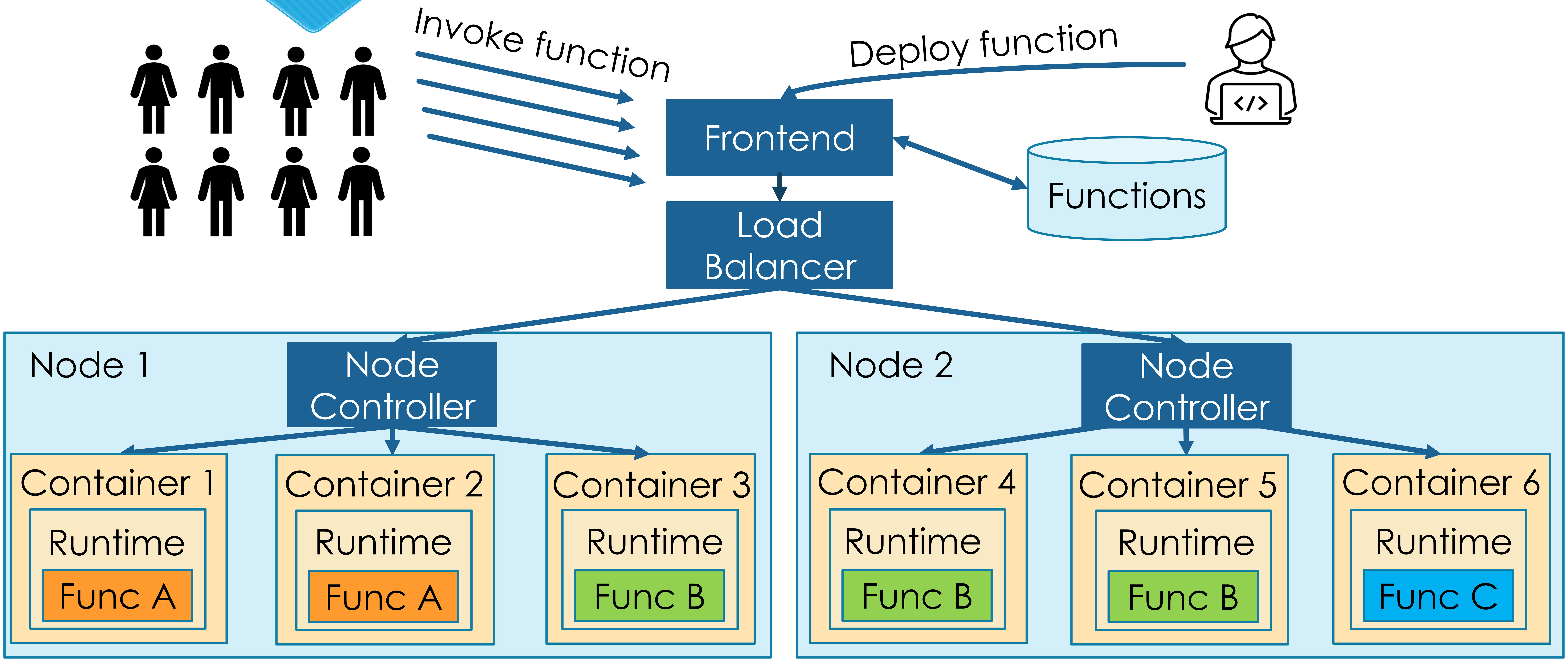
University of Illinois at Urbana-Champaign, *Microsoft Azure Research Systems

Emerging Software in the Cloud: Serverless Computing

Serverless computing

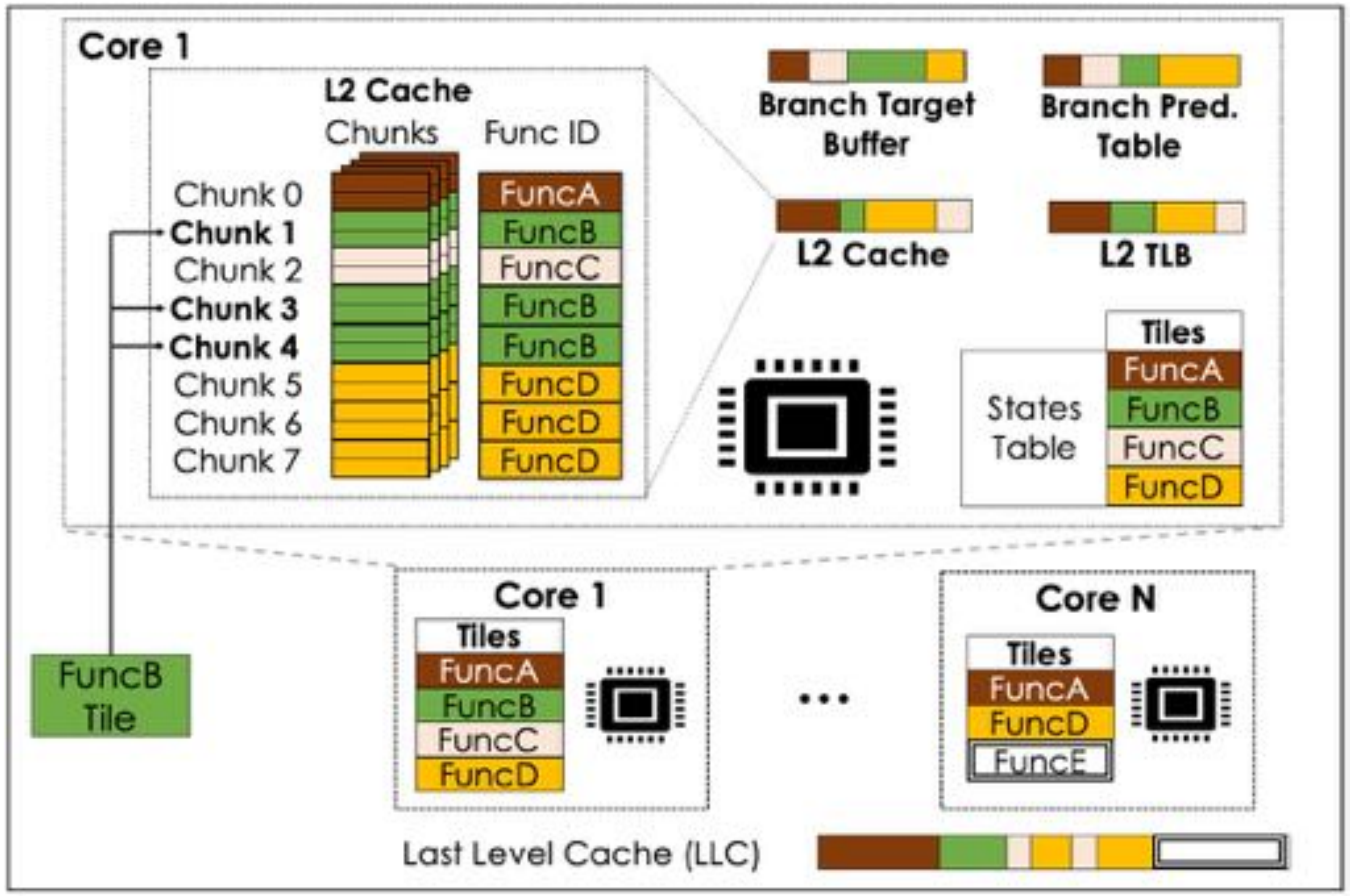
- Users deploy applications, providers provision resources
  - Simple and modular programming
  - Automatic resource scaling
  - Pay-as-you-go model
- Microsoft Azure, AWS Lambda, Google Cloud

How Serverless Computing Works?



Contributions

- Micro-architectural characterization of serverless systems
- **Mosaic:** an architecture for micro-architectural resource efficiency
  - Extends current processors optimized for monolithic applications
- Throughput boost 3.3X, power reduction by 22%



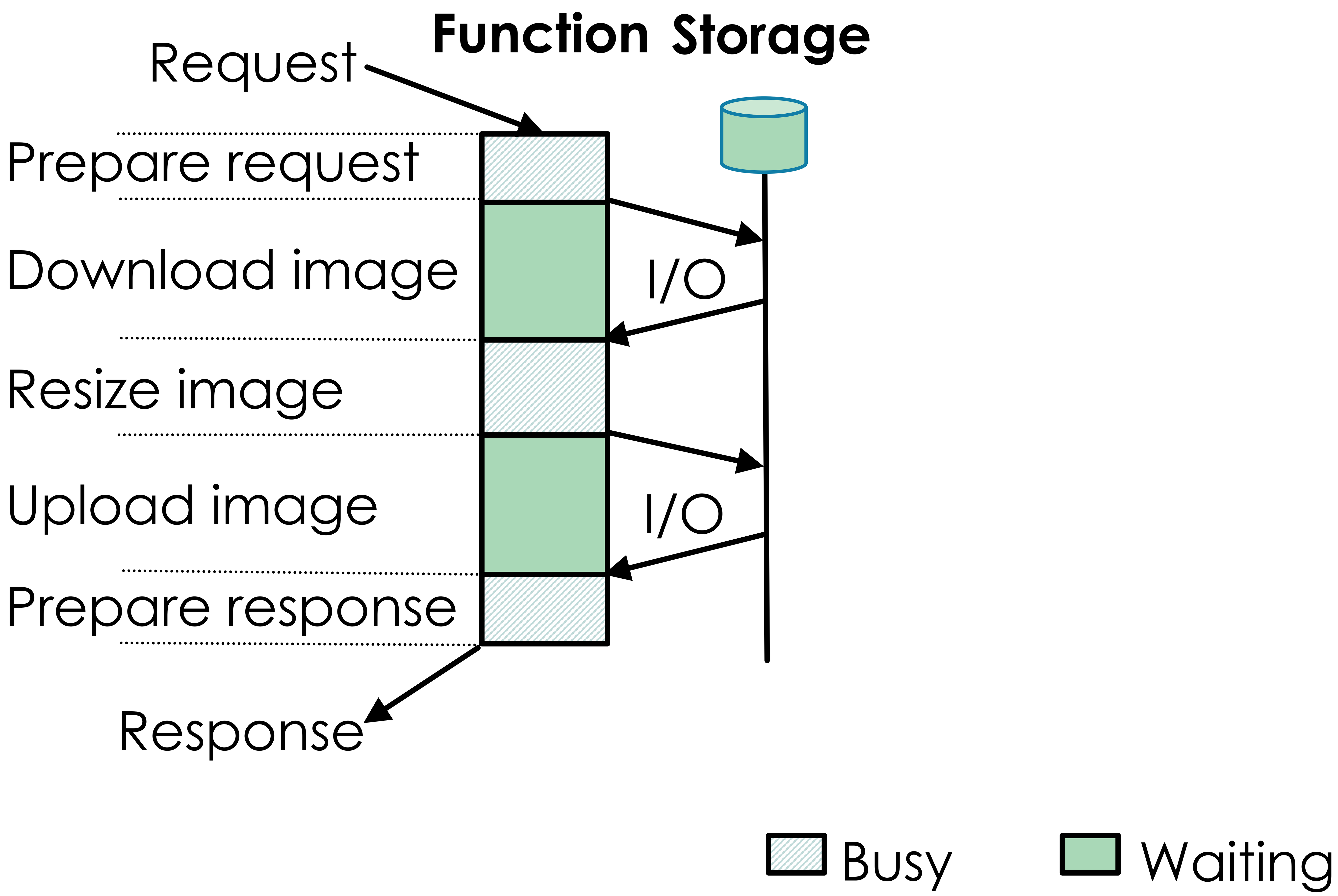
Mismatch Between Current Processors and Serverless Environments

Current Processors	Serverless Environments

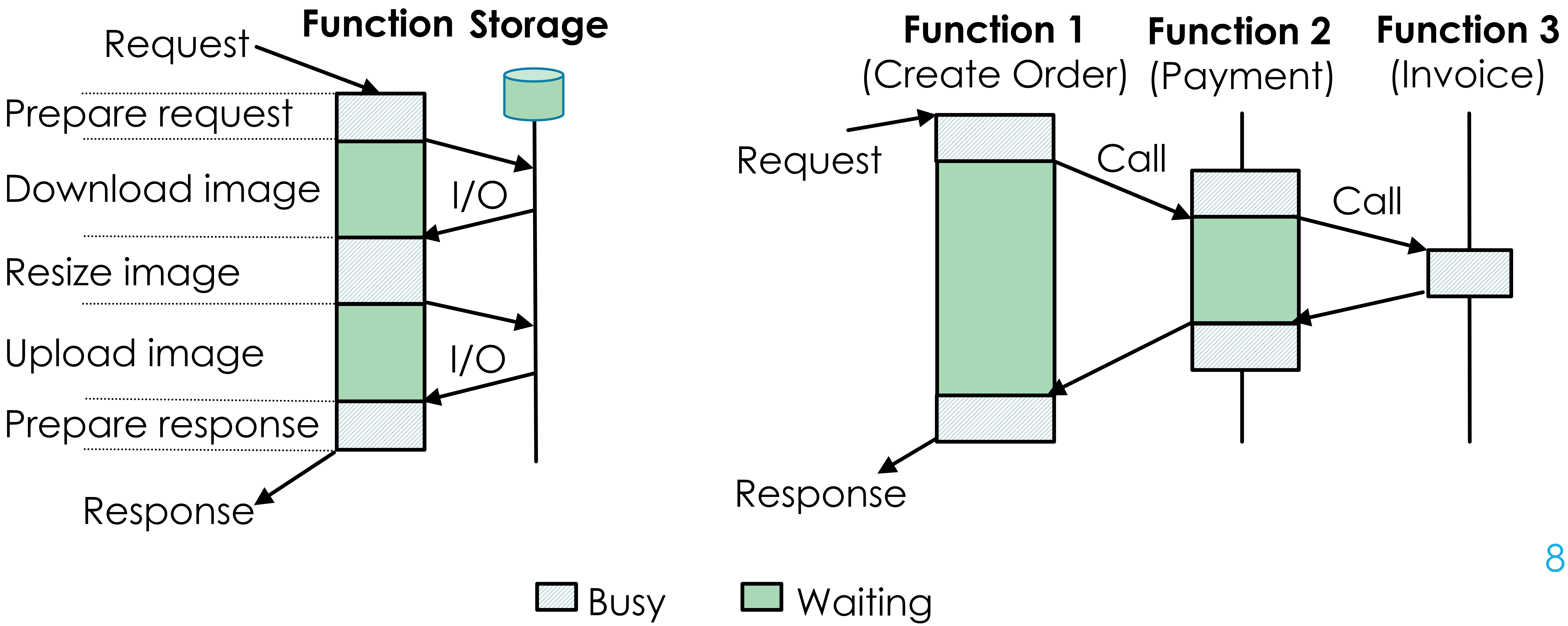
Mismatch Between Current Processors and Serverless Environments

Current Processors	Serverless Environments
Long-running monolithic apps	Short-running functions; dynamic

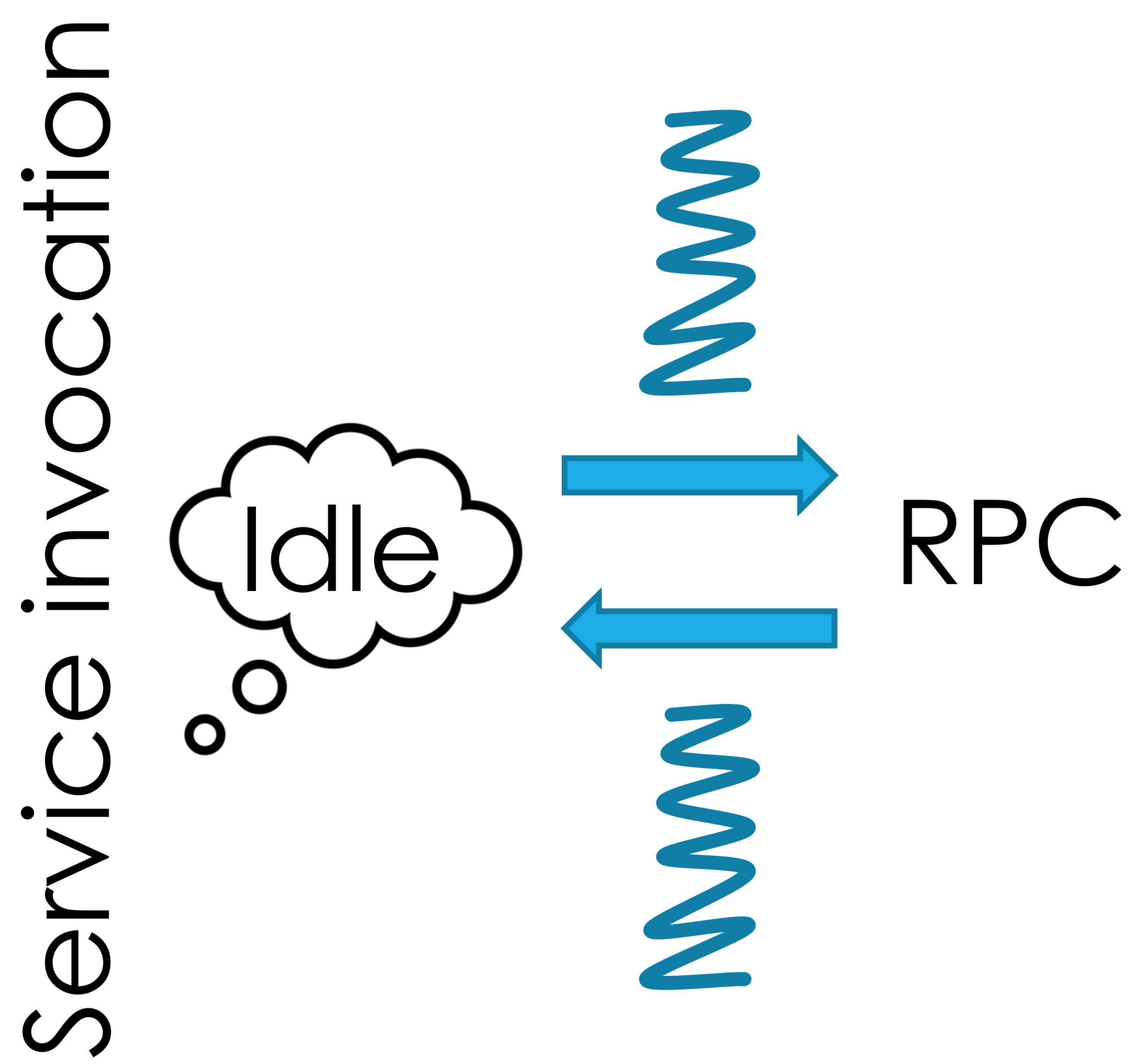
Idle Time Dominates Function Execution



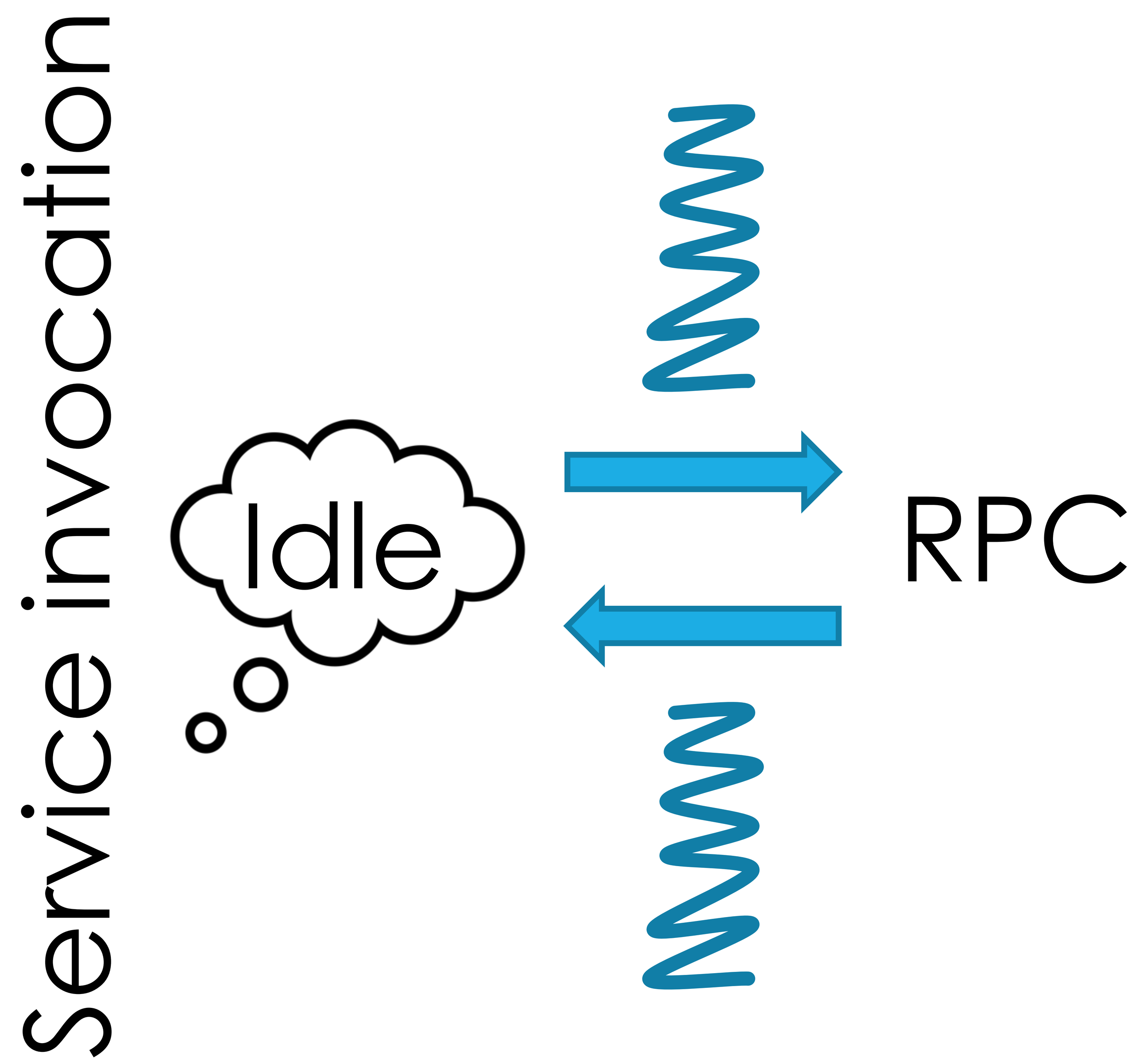
Idle Time Dominates Function Execution



Idle Time Dominates Function Execution



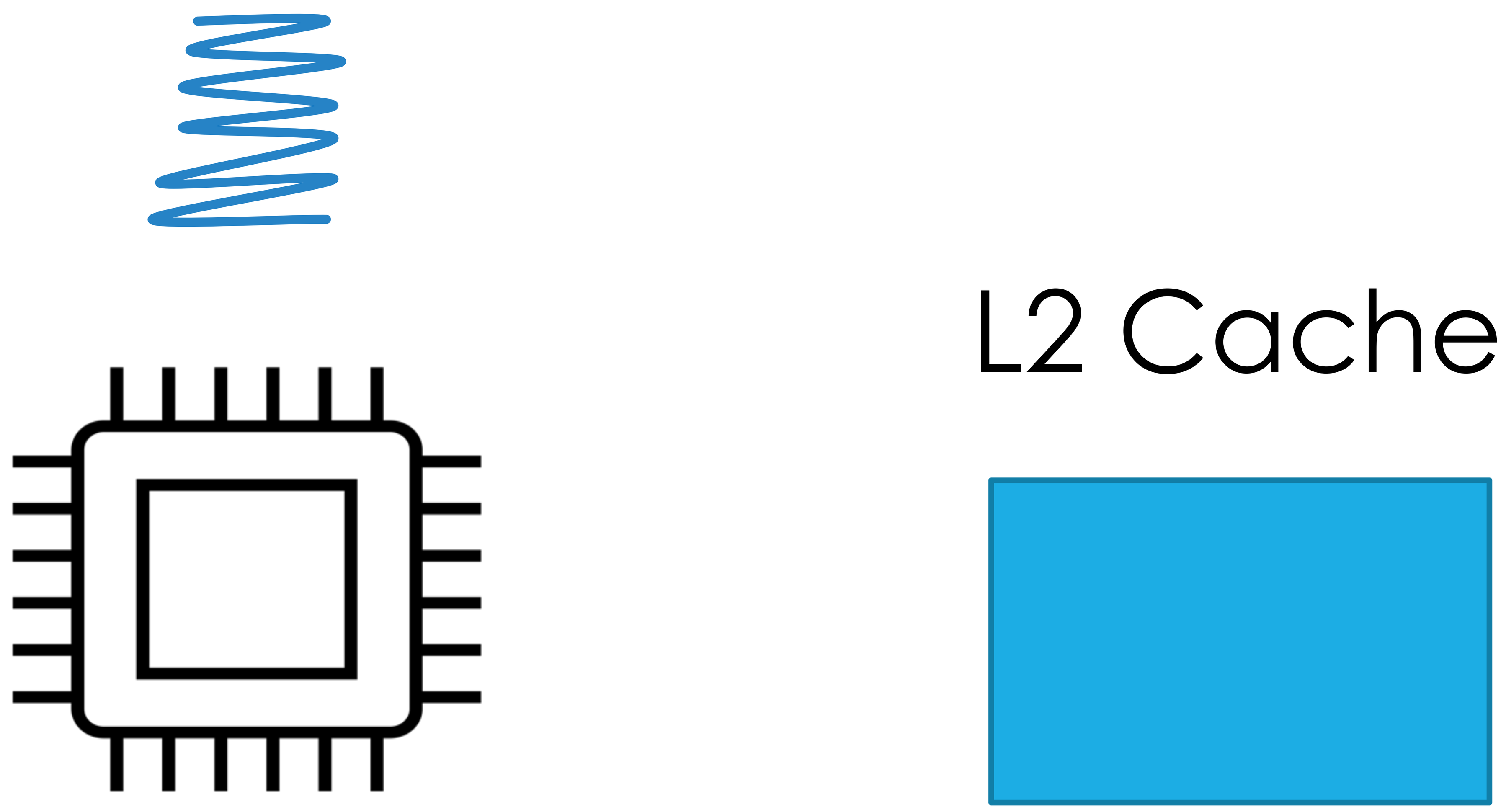
Idle Time Dominates Function Execution



Need to frequently context switch!

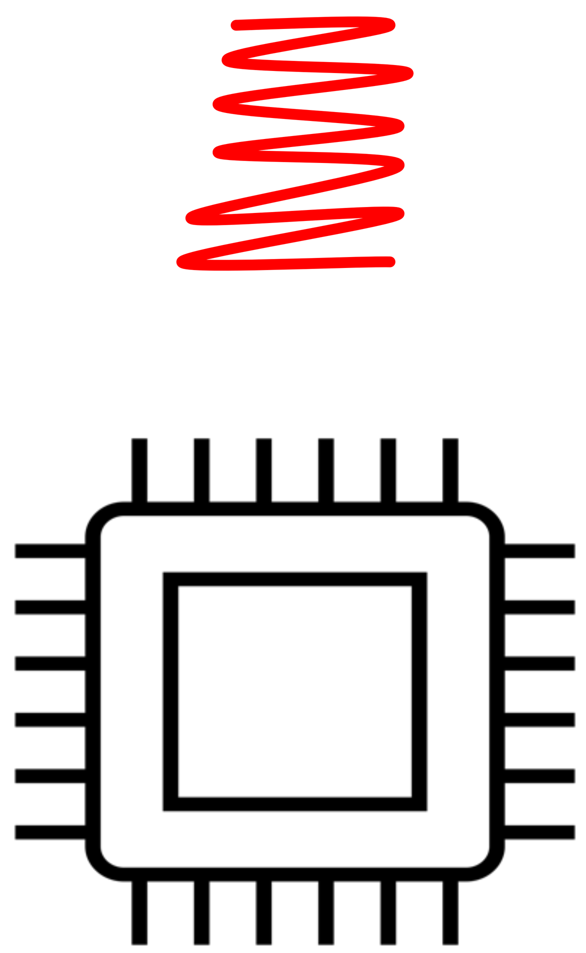
Pollution of Stateful Hardware Structures

- Frequent context-switches interleave executions of different functions on the same core
- Pollution of stateful structures: caches, TLB, branch predictors

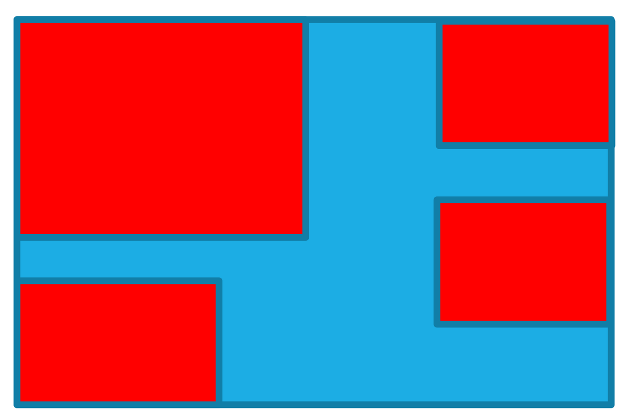


Pollution of Stateful Hardware Structures

- Frequent context-switches interleave executions of different functions on the same core
- Pollution of stateful structures: caches, TLB, branch predictors

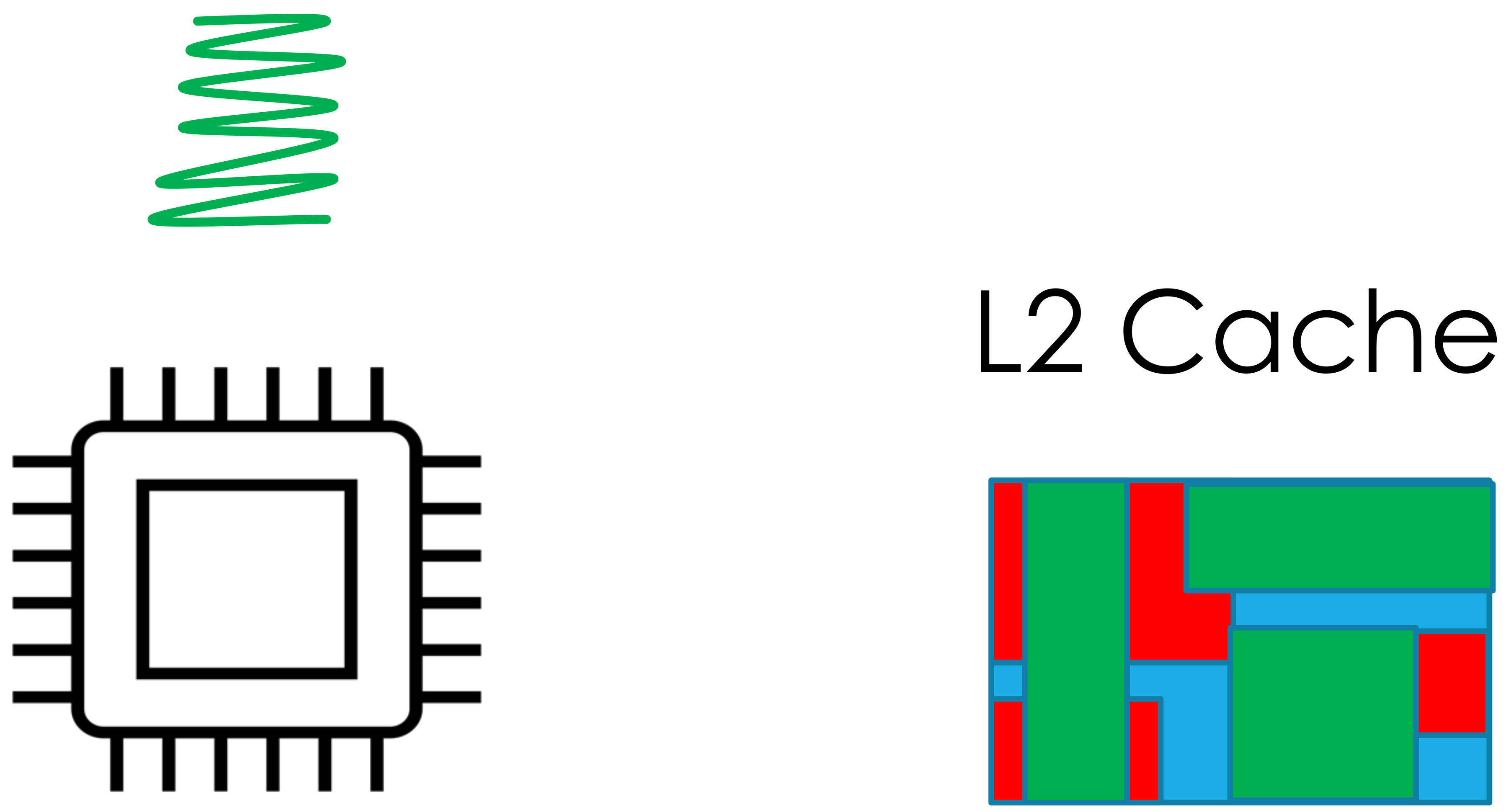


L2 Cache



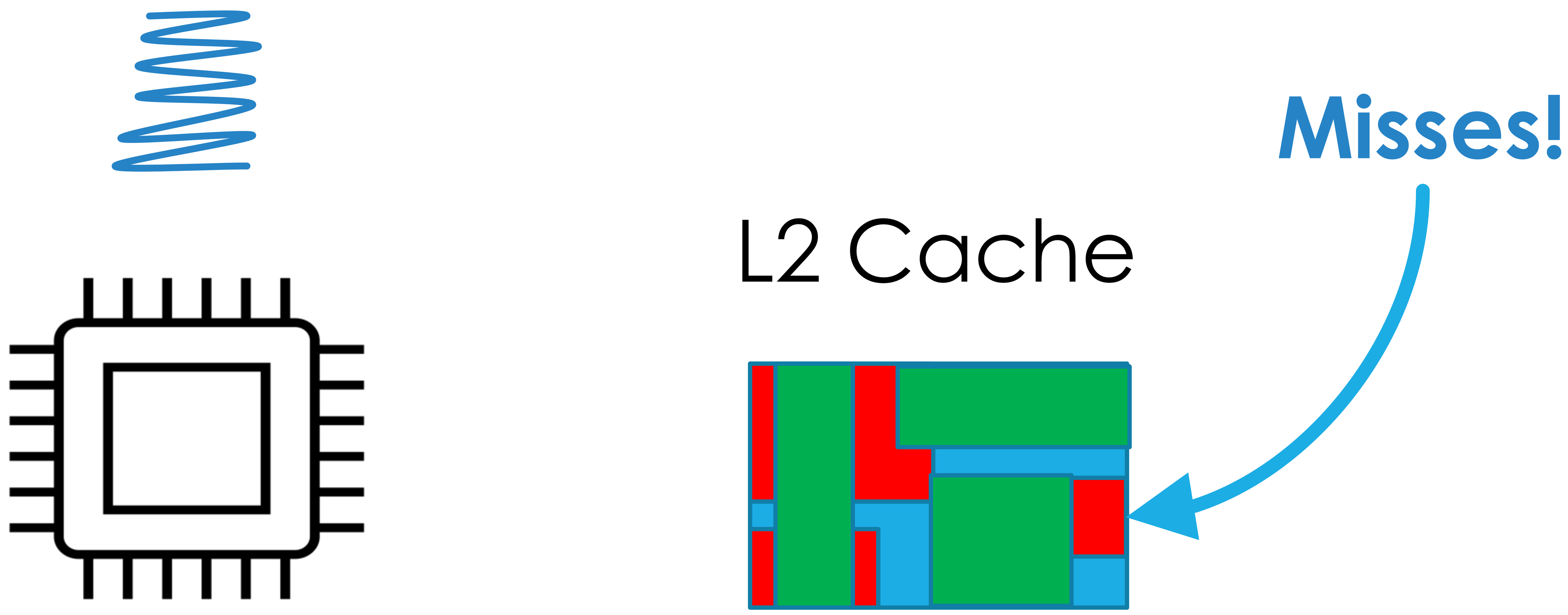
Pollution of Stateful Hardware Structures

- Frequent context-switches interleave executions of different functions on the same core
- Pollution of stateful structures: caches, TLB, branch predictors



Pollution of Stateful Hardware Structures

- Frequent context-switches interleave executions of different functions on the same core
- Pollution of stateful structures: caches, TLB, branch predictors



Pollution of Stateful Hardware Structures

Pollution of Stateful Hardware Structures

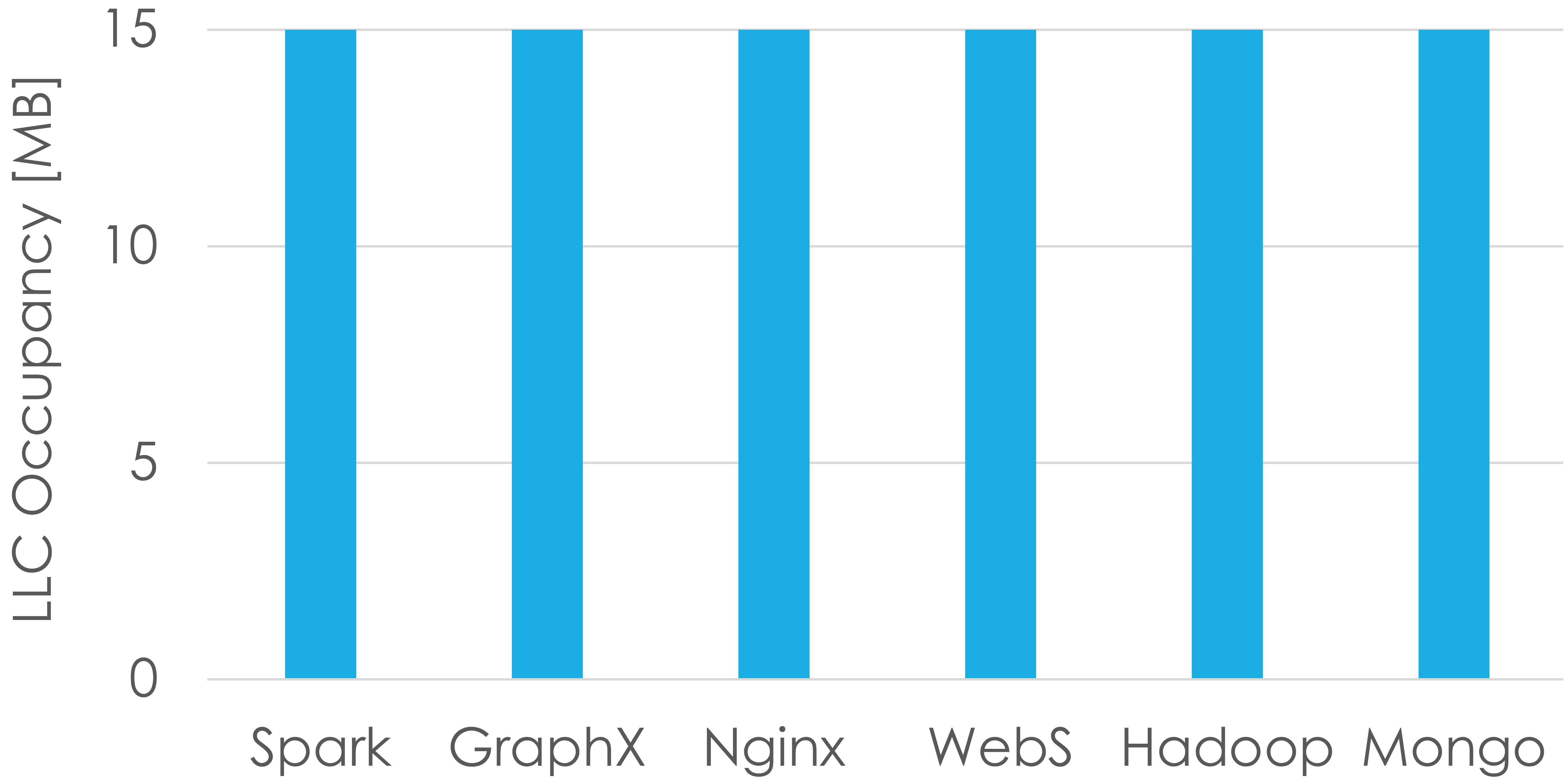
To maintain good function performance, we should save their micro-arch state

Mismatch Between Current Processors and Serverless Environments

Current Processors	Serverless Environments
Long-running, predictable apps	Short-running services; dynamic
Large monolithic applications	Small data, instr, branch footprints

Opportunity: Small-sized Serverless Functions

- Serverless functions with small data, instruction and branch footprints

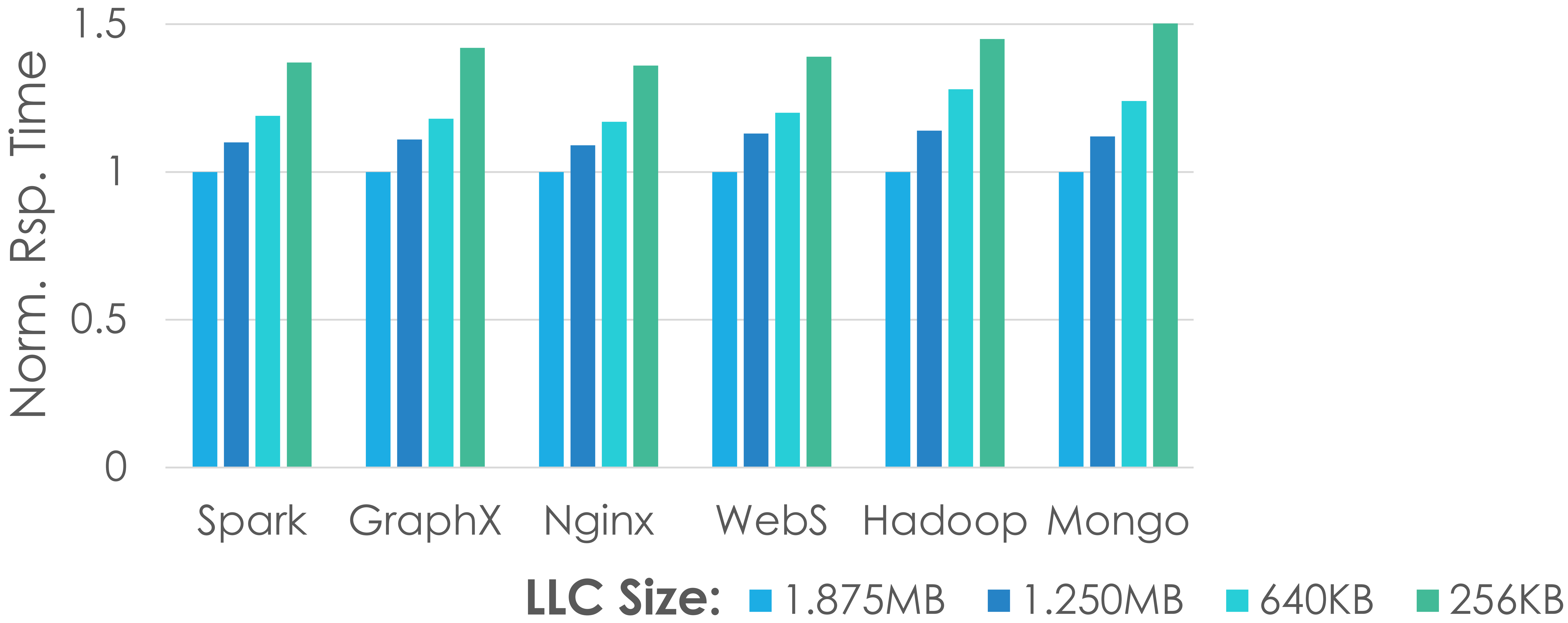


Opportunity: Small-sized Serverless Functions

- Serverless functions with small data, instruction and branch footprints

Opportunity: Small-sized Serverless Functions

- Small size → no need for full-sized large hardware structures

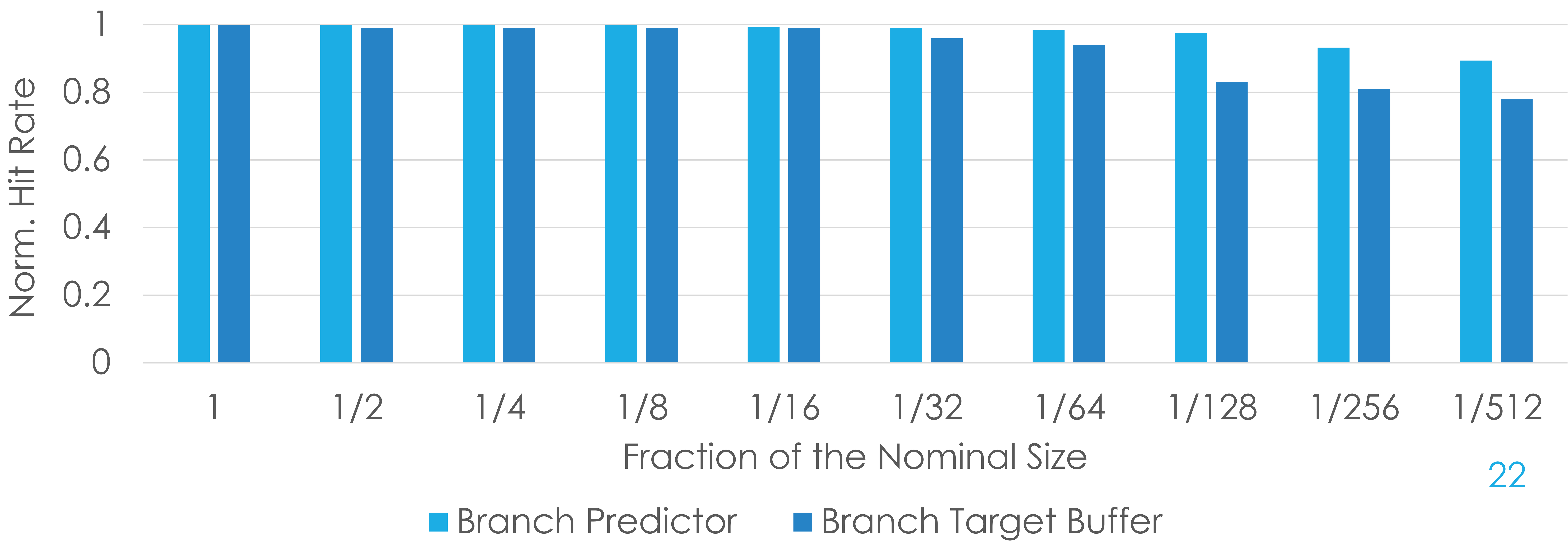


Opportunity: Small-sized Serverless Functions

- Small size → no need for full-sized large hardware structures

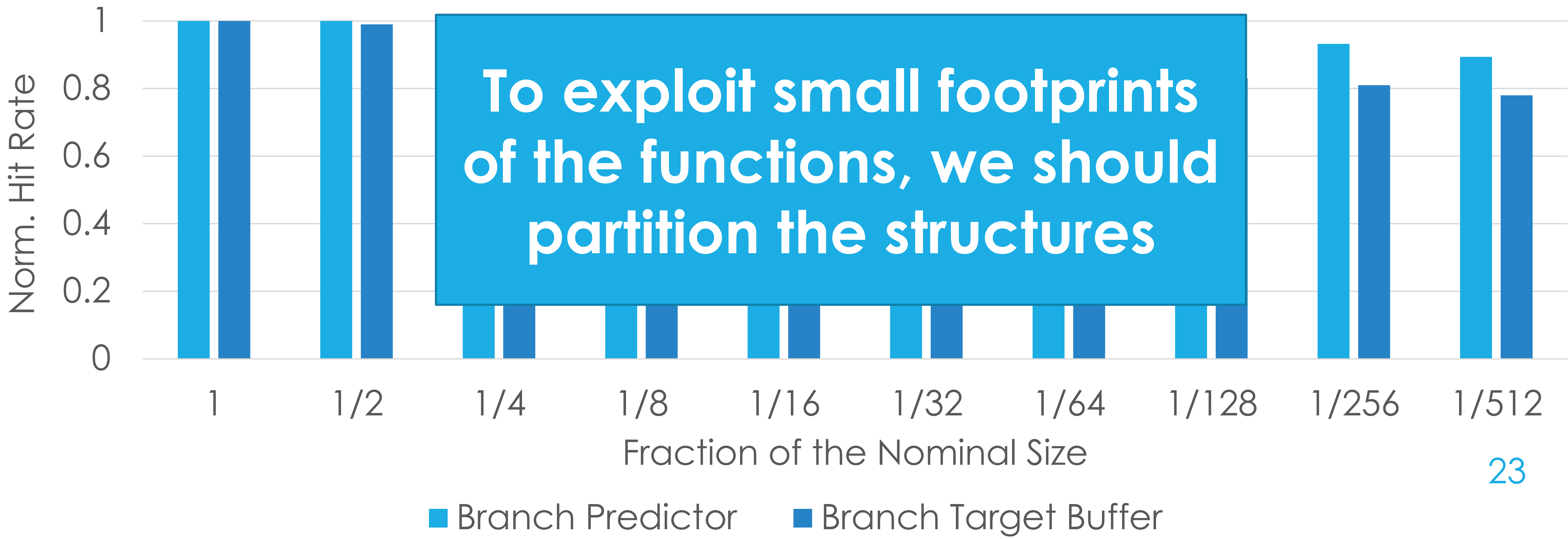
Opportunity: Small-sized Serverless Functions

- Small size → no need for full-sized large hardware structures



Opportunity: Small-sized Serverless Functions

- Small size → no need for full-sized large hardware structures

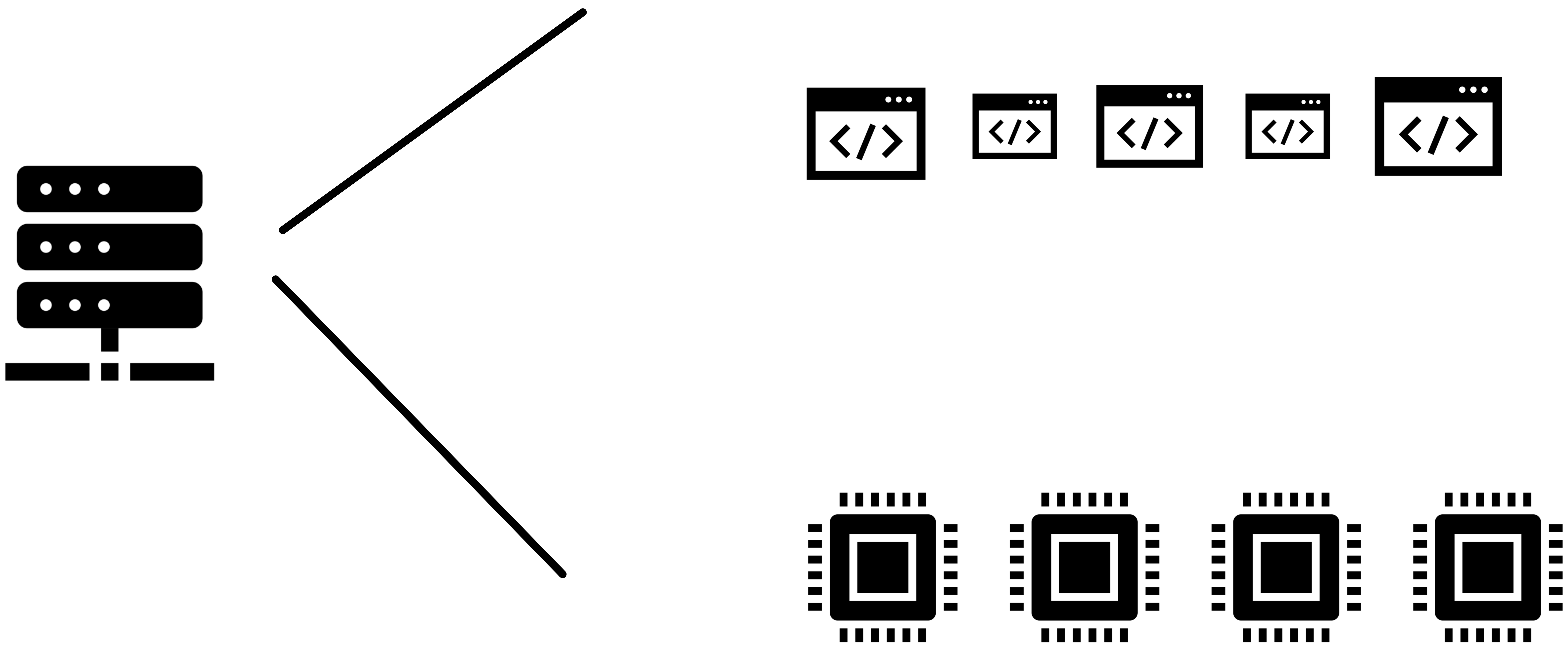


Mismatch Between Current Processors and Serverless Environments

Current Processors	Serverless Environments
Long-running, predictable apps	Short-running services; dynamic
Large monolithic applications	Small data, instr, branch footprints
Optimized for average latency	Focus on tail; diverse functions

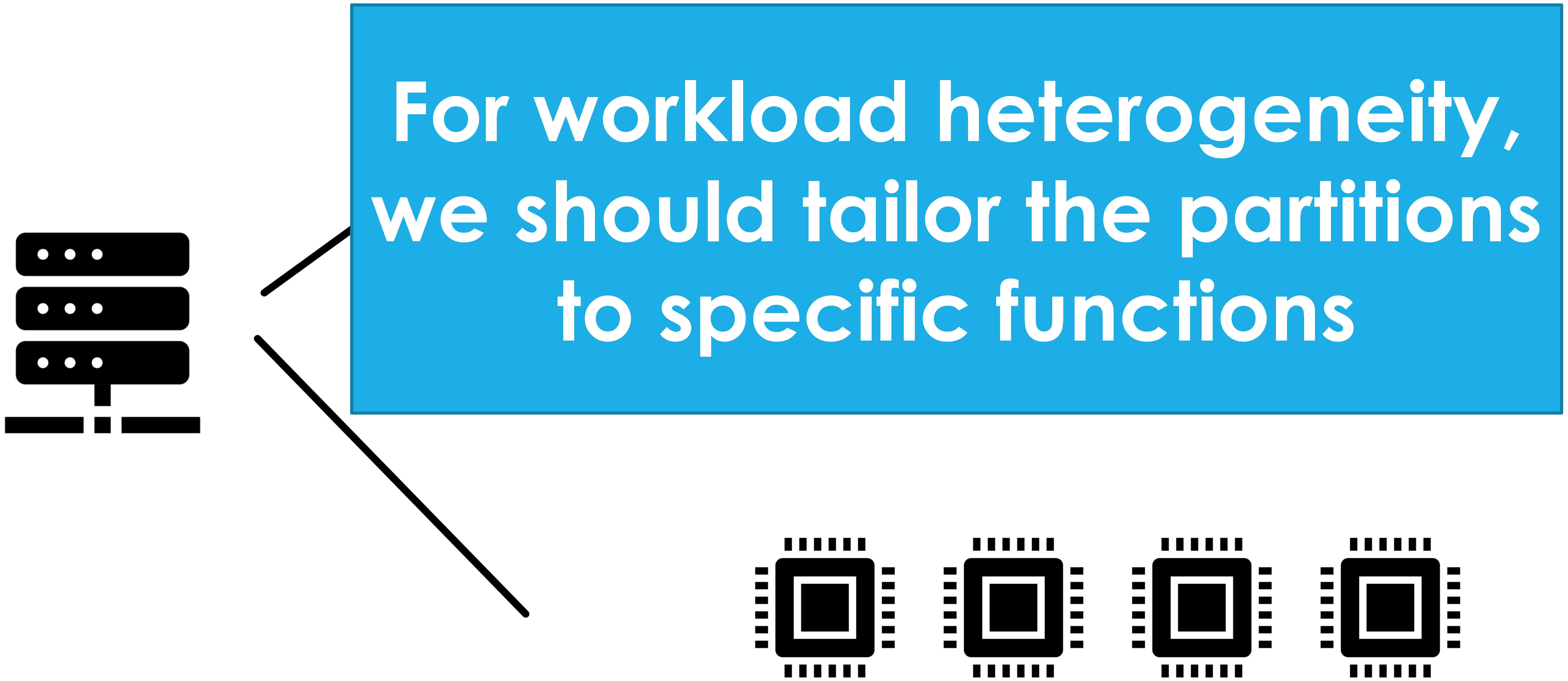
Workload Heterogeneity

- Serverless functions highly diverse



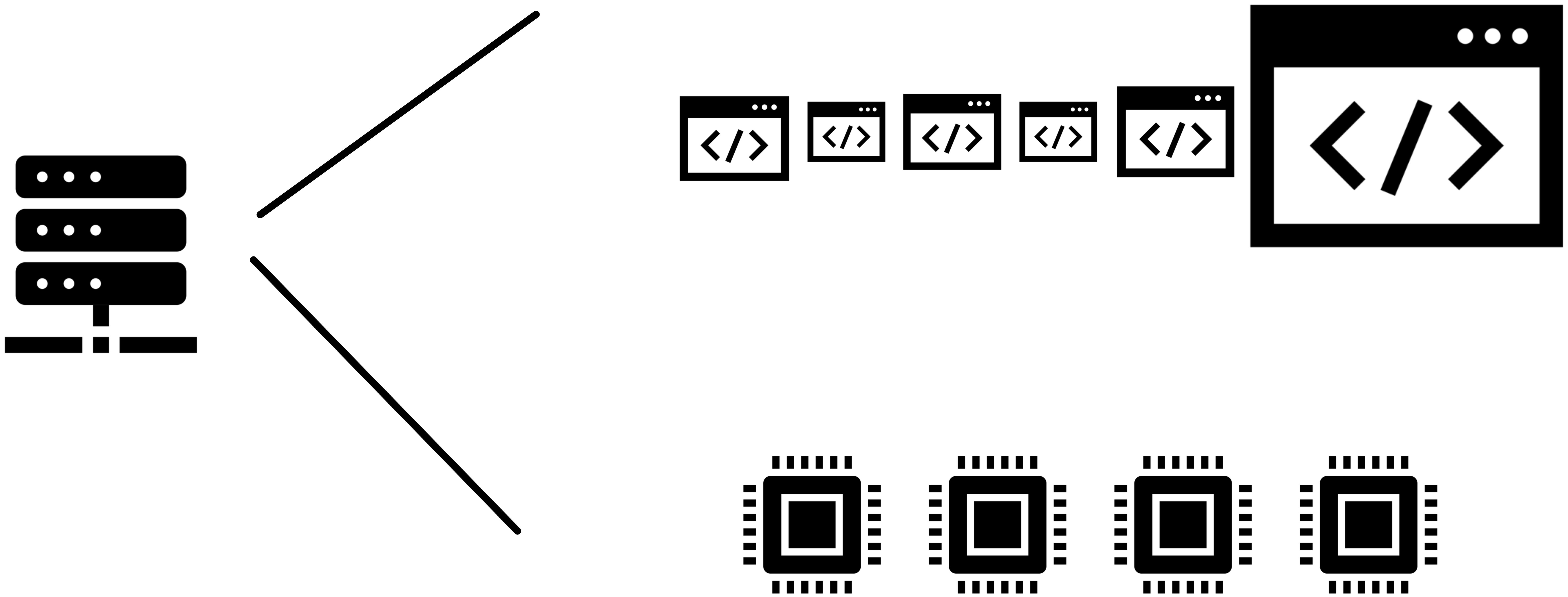
Workload Heterogeneity

- Serverless functions highly diverse



Need for Processor Generality

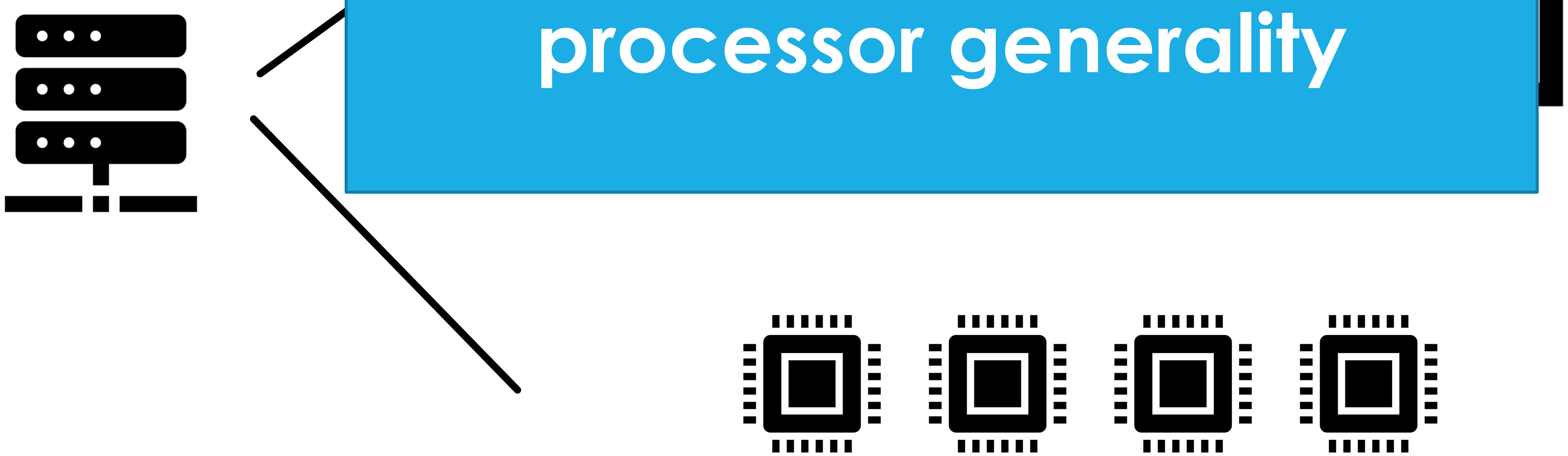
- Serverless functions colocated with monolithic workloads



Need for Processor Generality

- Serverless functions colocated with monolithic workloads

We need to maintain processor generality



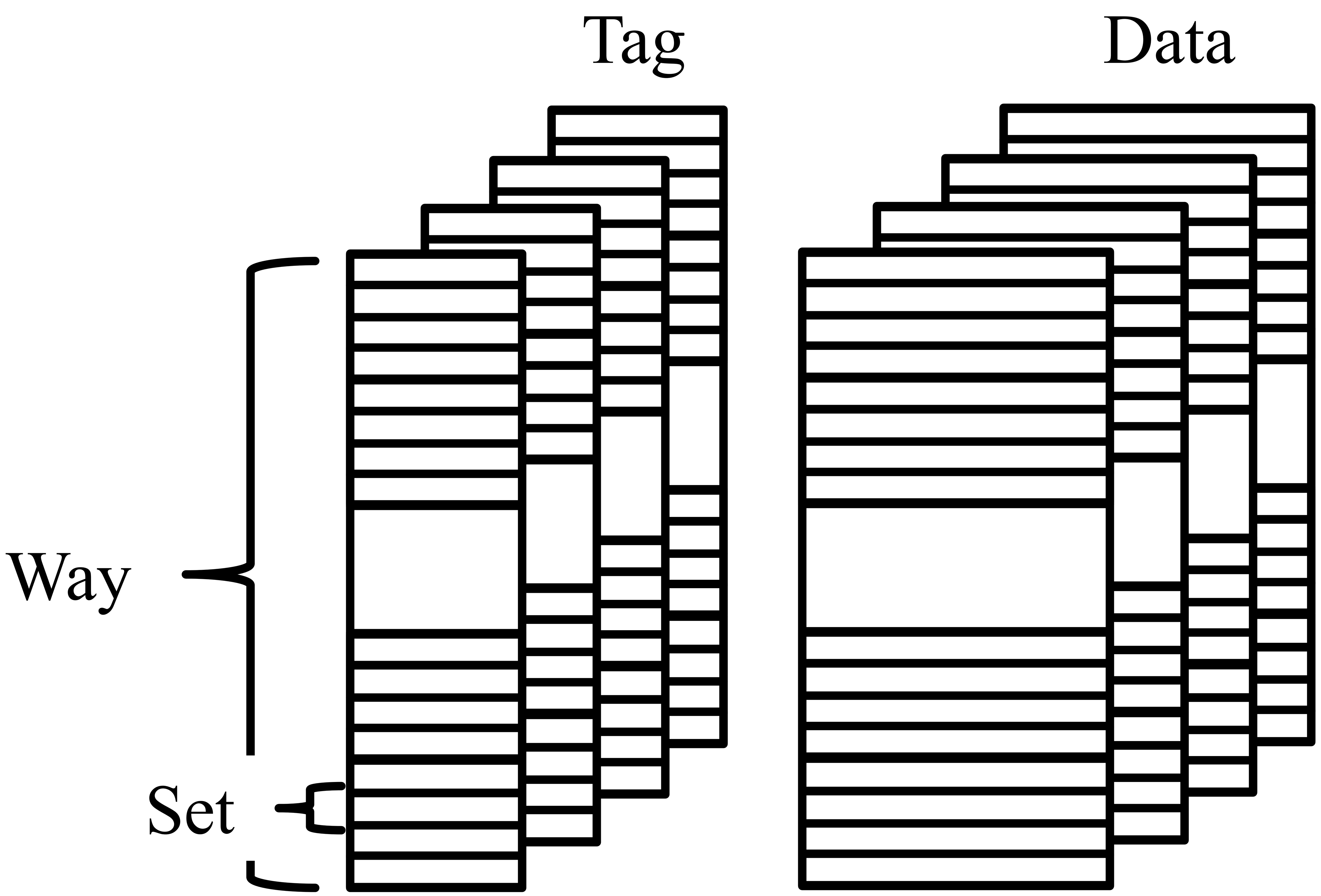
Mosaic: An Architecture for FaaS Environments

- **MosaicCPU** – a processor architecture that efficiently runs both monolithic applications and serverless functions
- **MosaicScheduler** – a software stack for serverless systems that maximizes the benefits of MosaicCPU

Mosaic: An Architecture for FaaS Environments

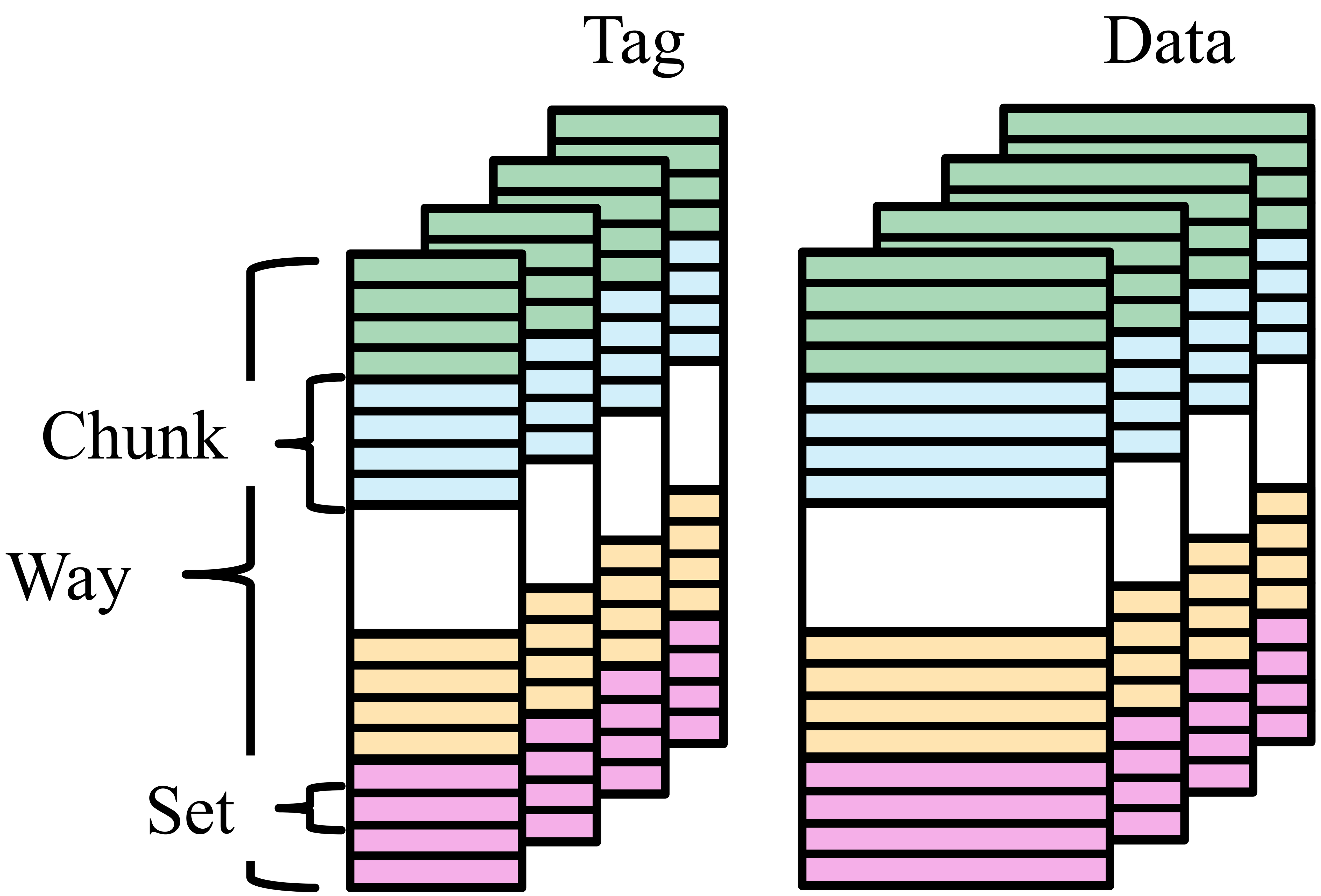
- 4 main principles:
  - **Partition stateful structures into per-function tiles**

Fine-Grained Hardware Partitioning



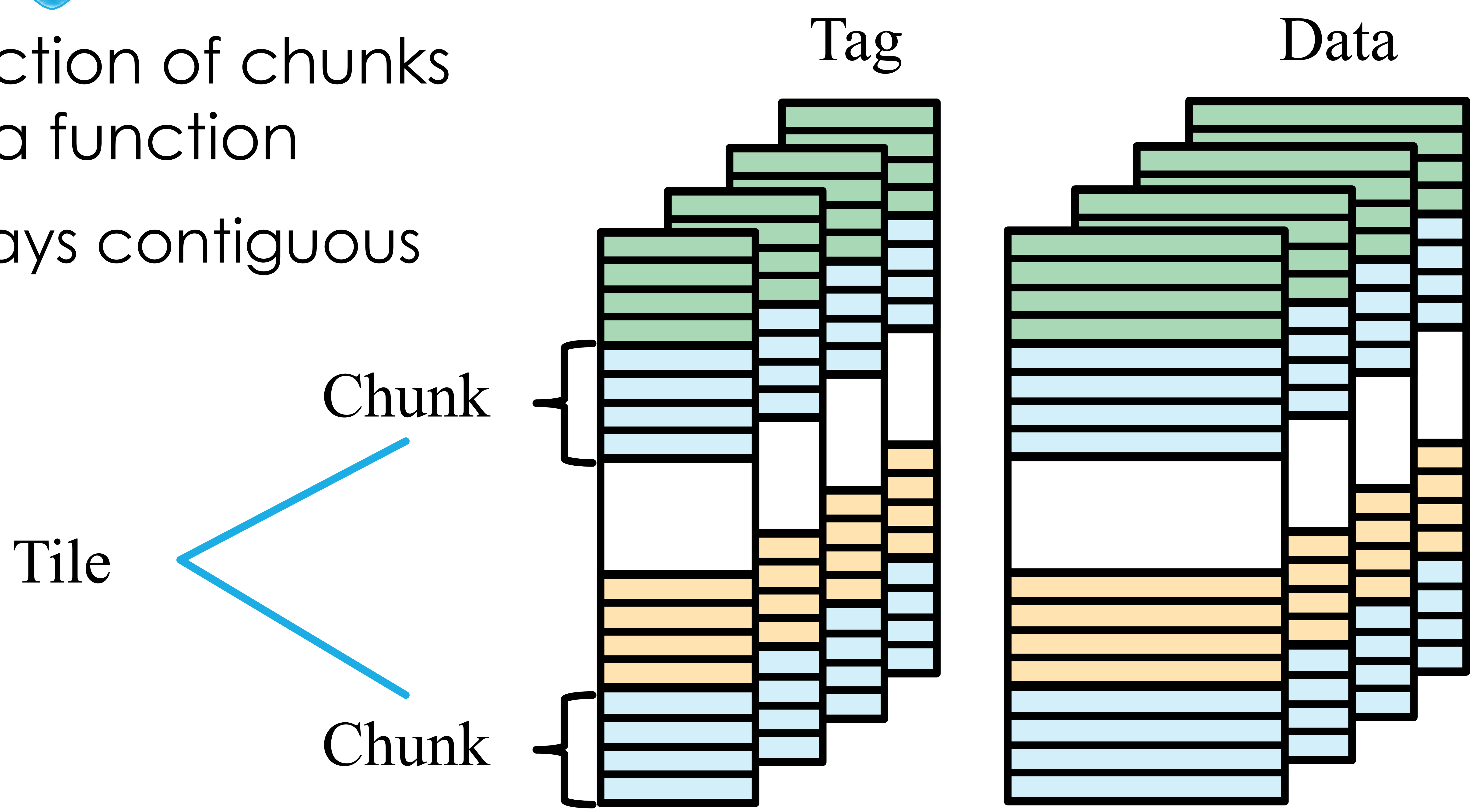
Fine-Grained Hardware Partitioning

- For example:
  - Caches
  - TLBs
  - Branch units

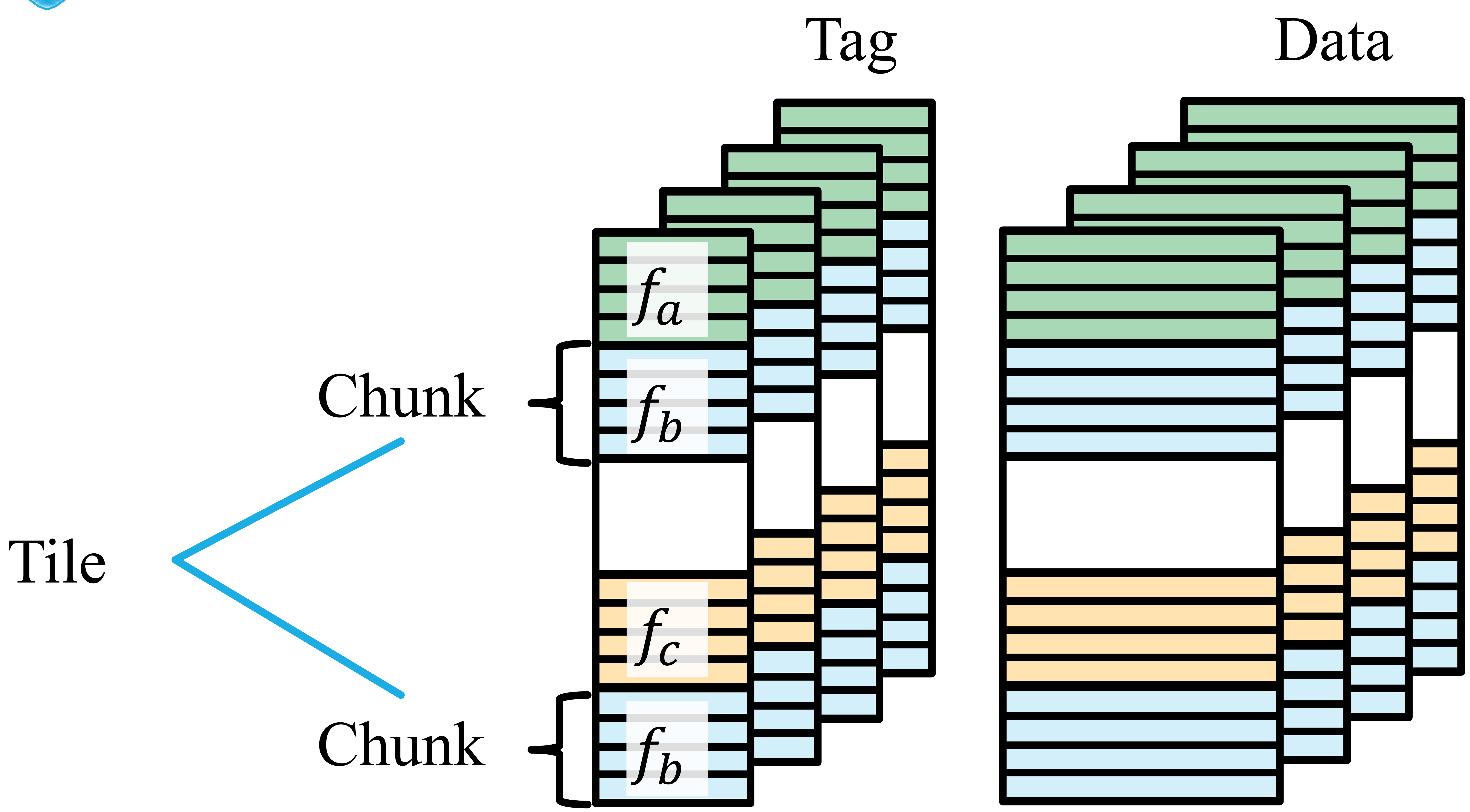


Fine-Grained Hardware Partitioning

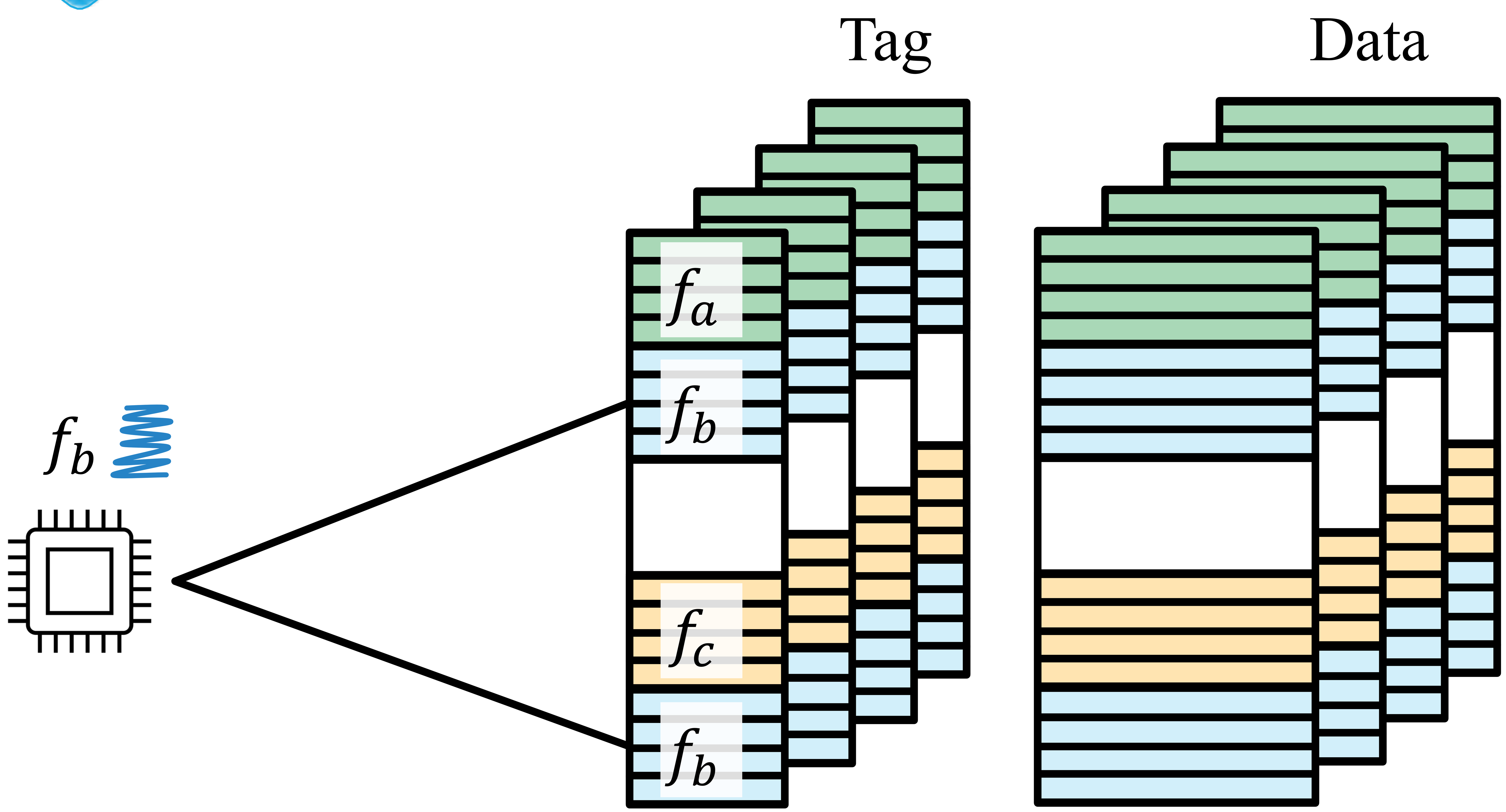
- **Tile** = Collection of chunks owned by a function
  - Not always contiguous



Fine-Grained Hardware Partitioning

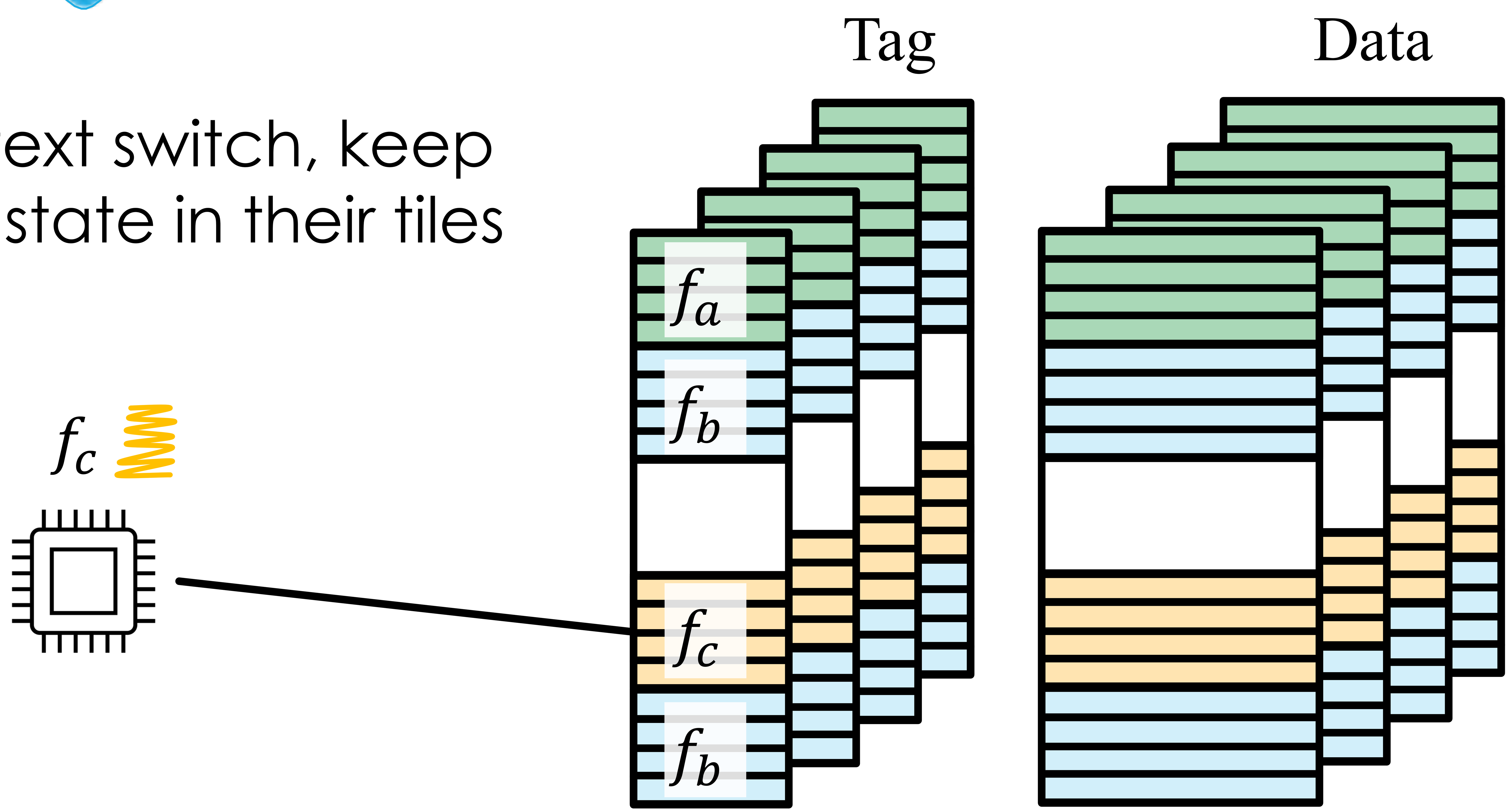


Fine-Grained Hardware Partitioning



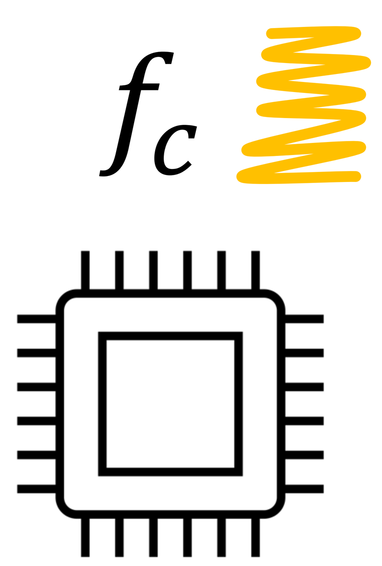
Fine-Grained Hardware Partitioning: Improve Performance

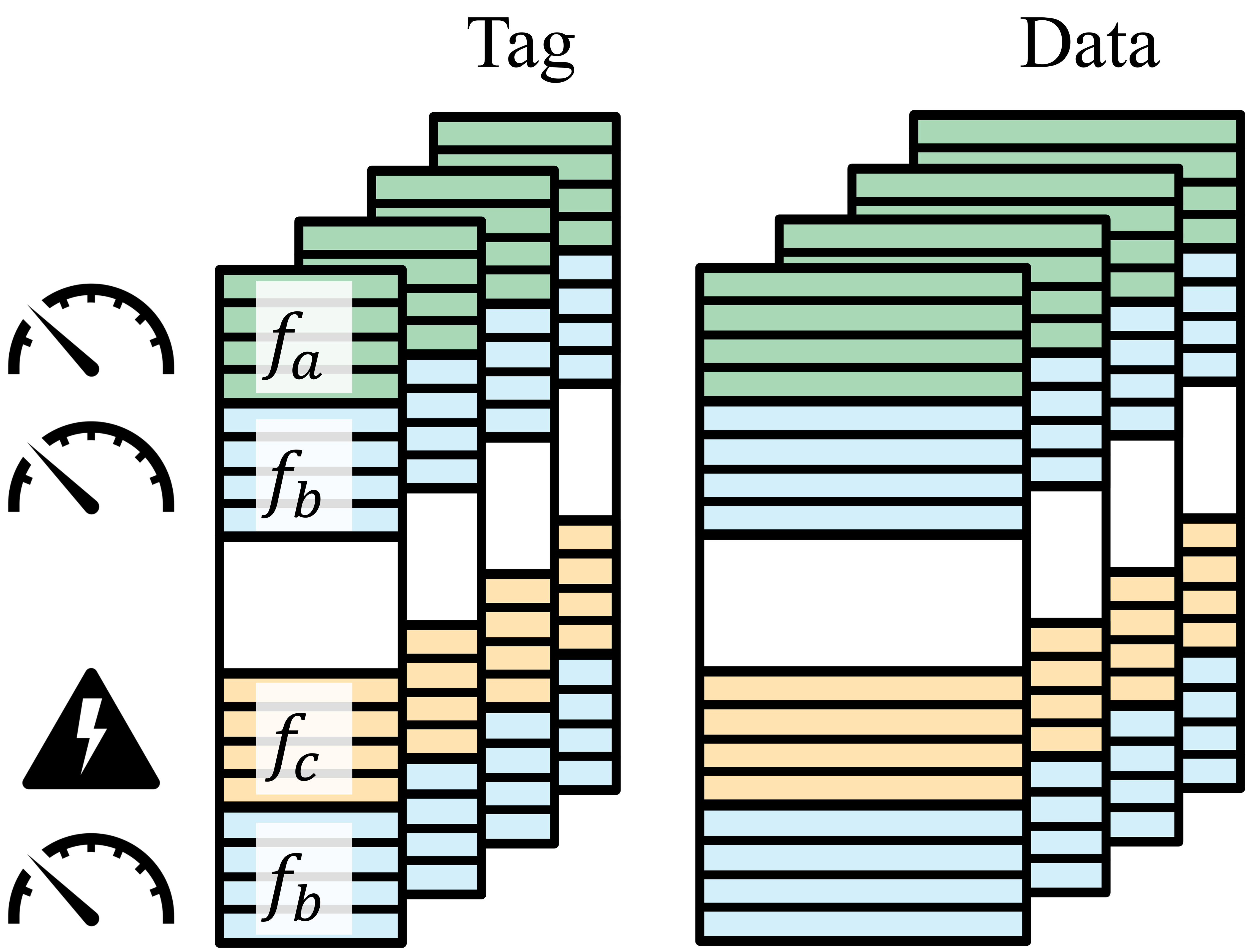
- On a context switch, keep functions' state in their tiles



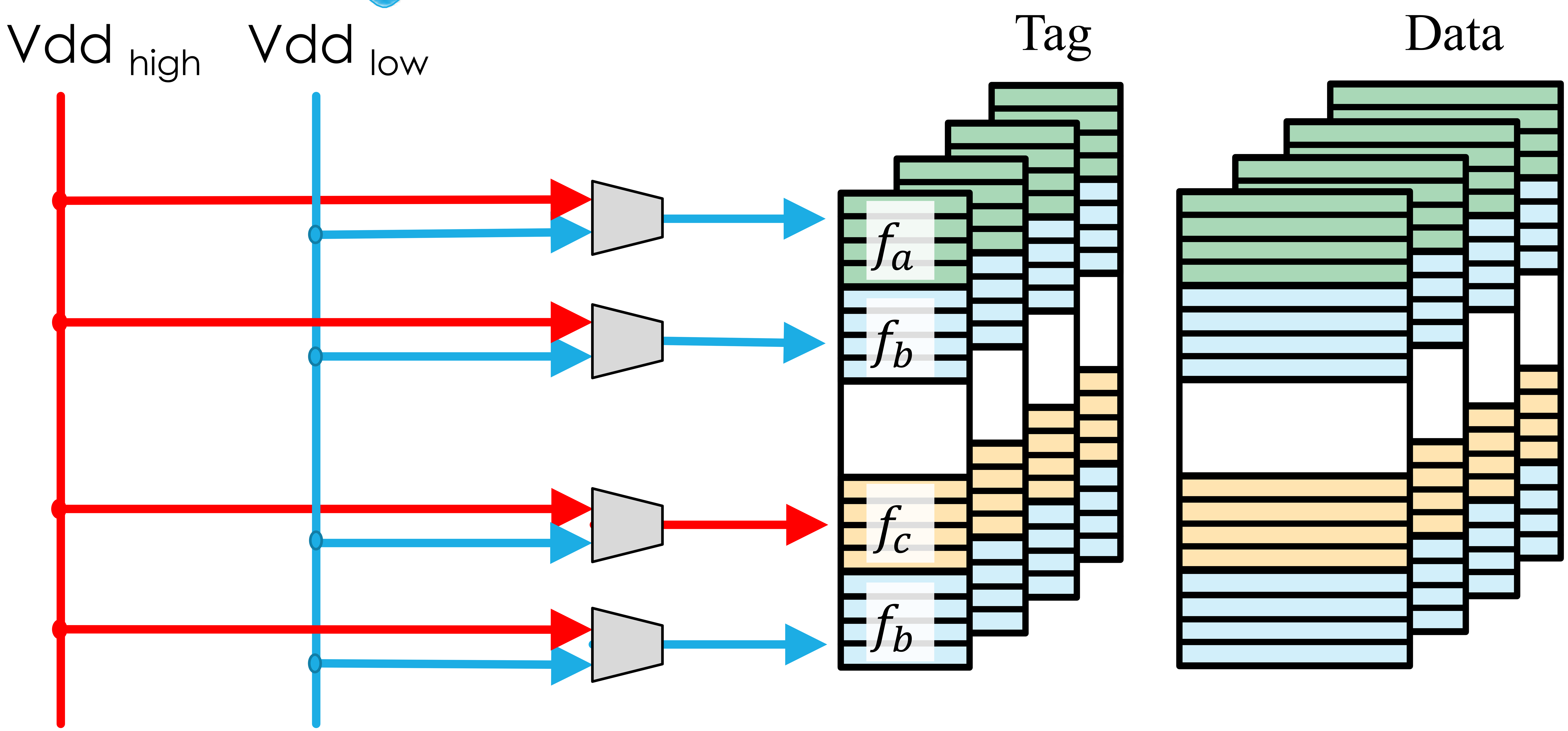
Fine-Grained Hardware Partitioning: Improve Power Efficiency

- Inactive chunks at low voltage, saving power





Fine-Grained Hardware Partitioning: Improve Power Efficiency

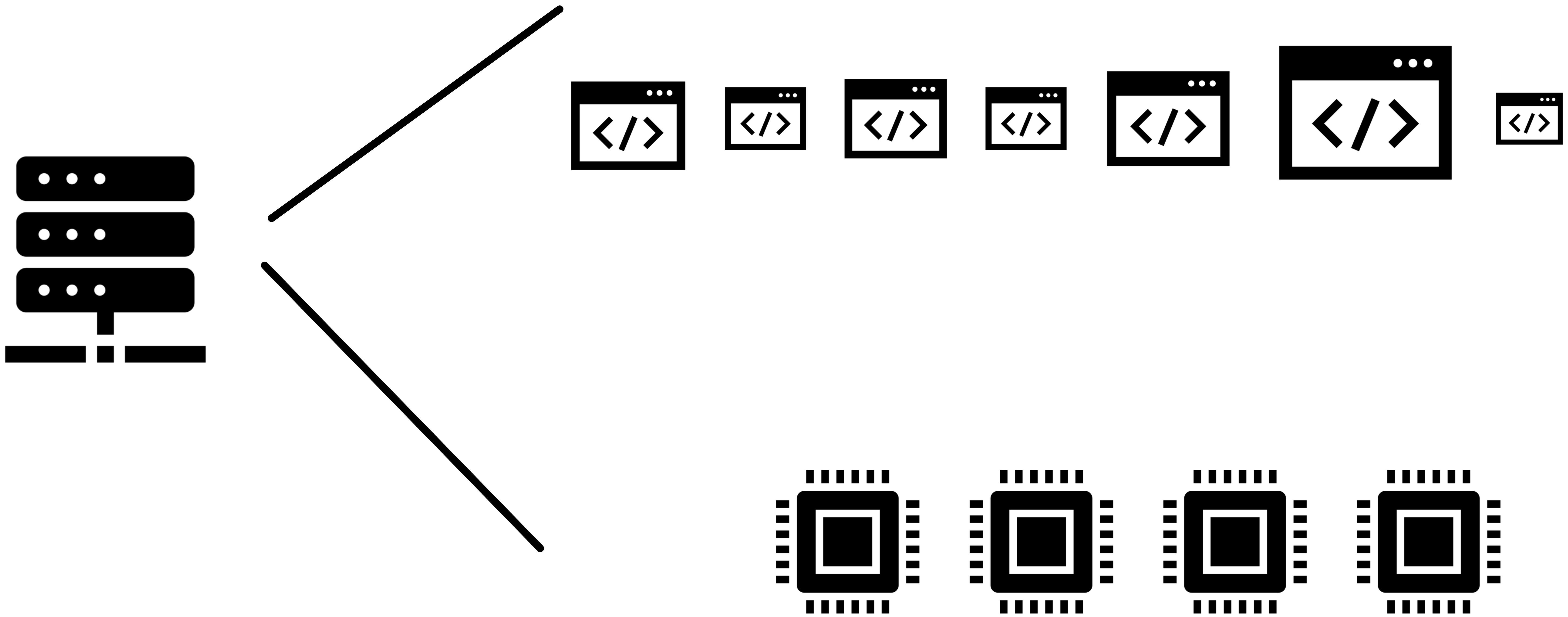


Mosaic: An Architecture for FaaS Environments

- 4 main principles:
  - Partition stateful structures into per-function tiles
  - **Size per-function tiles based on individual function needs**

Per-Function Structure Sizing

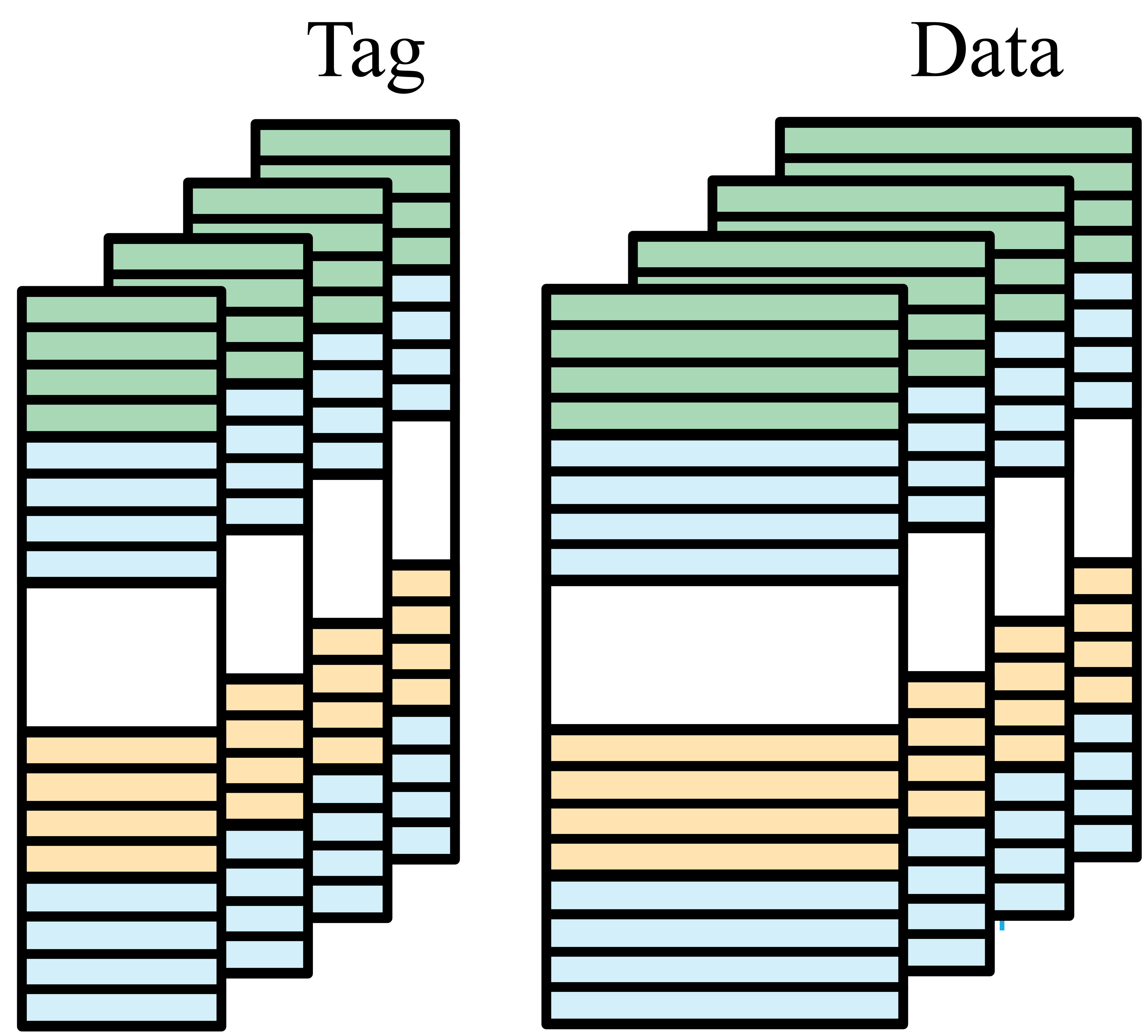
- Serverless functions highly diverse → need non-uniform tile sizes
- Non-uniform tiles can cause fragmentation → need non-contiguity



Accessing a Function's Tile

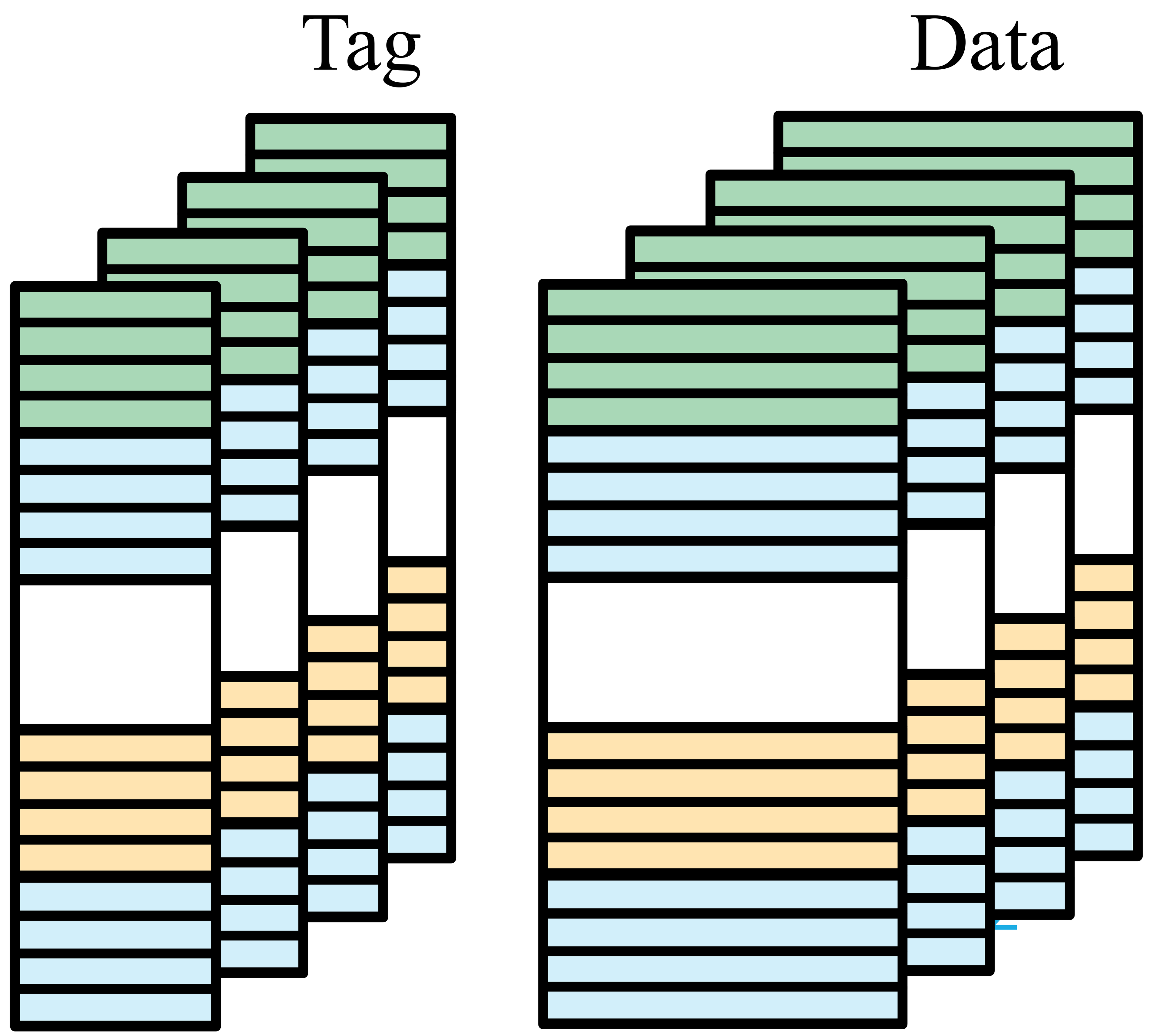
PA:

Tag	Set	Offset
45..13	12..6	5..0

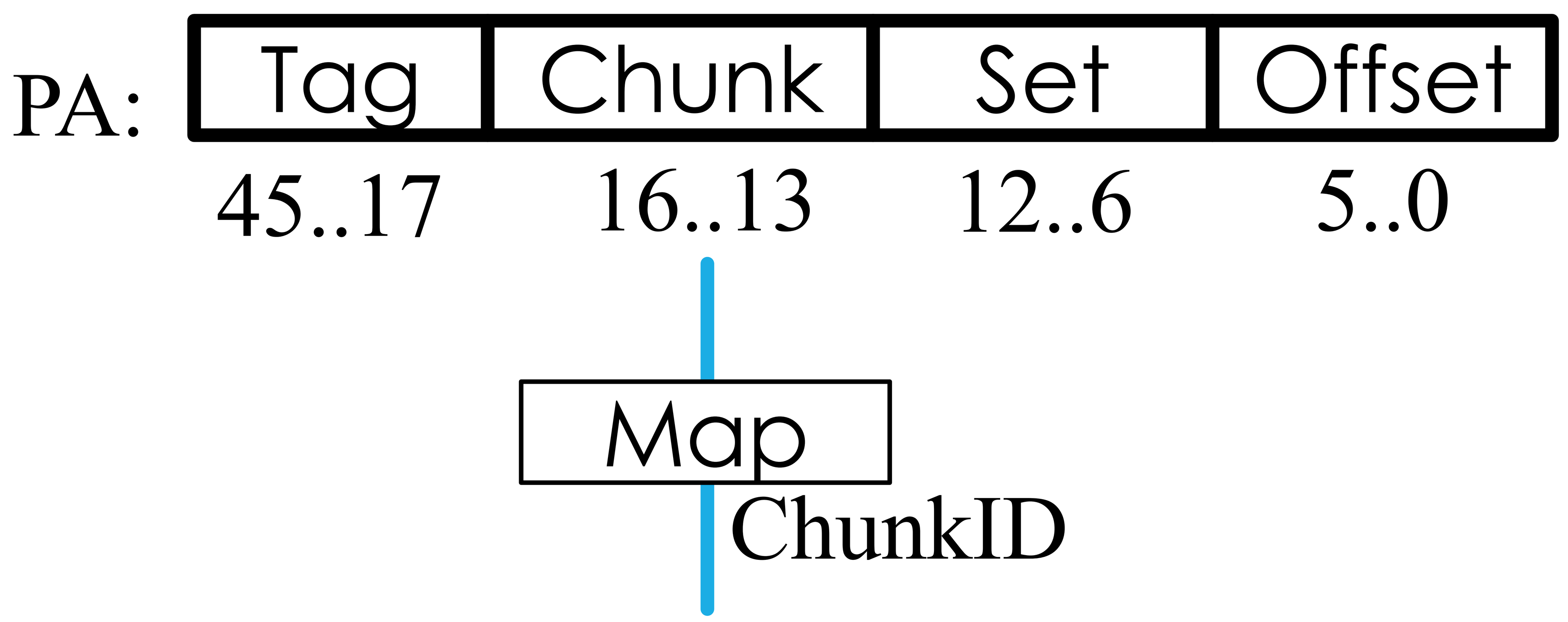


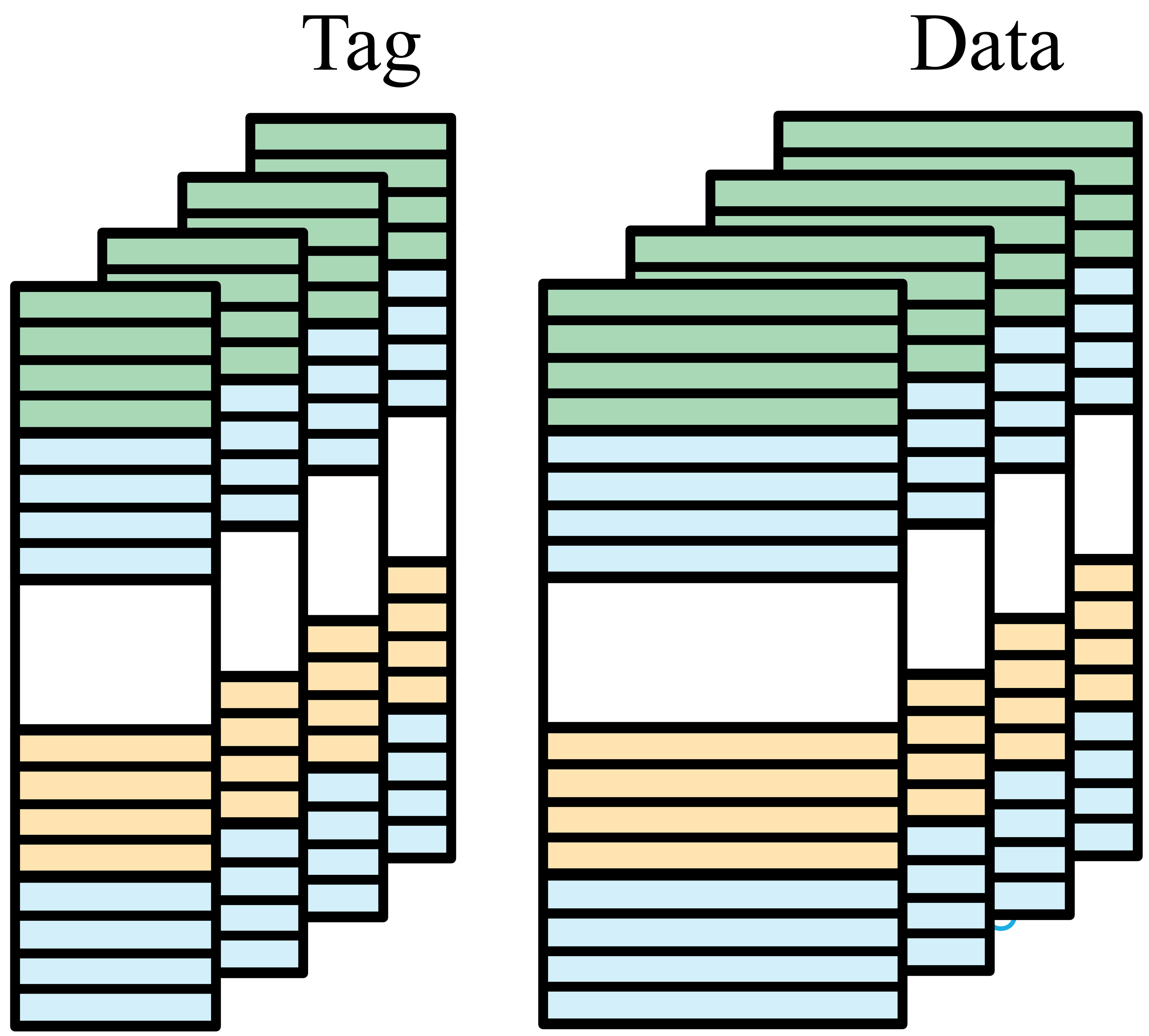
Accessing a Function's Tile

PA:	Tag	Chunk	Set	Offset
	45..17	16..13	12..6	5..0

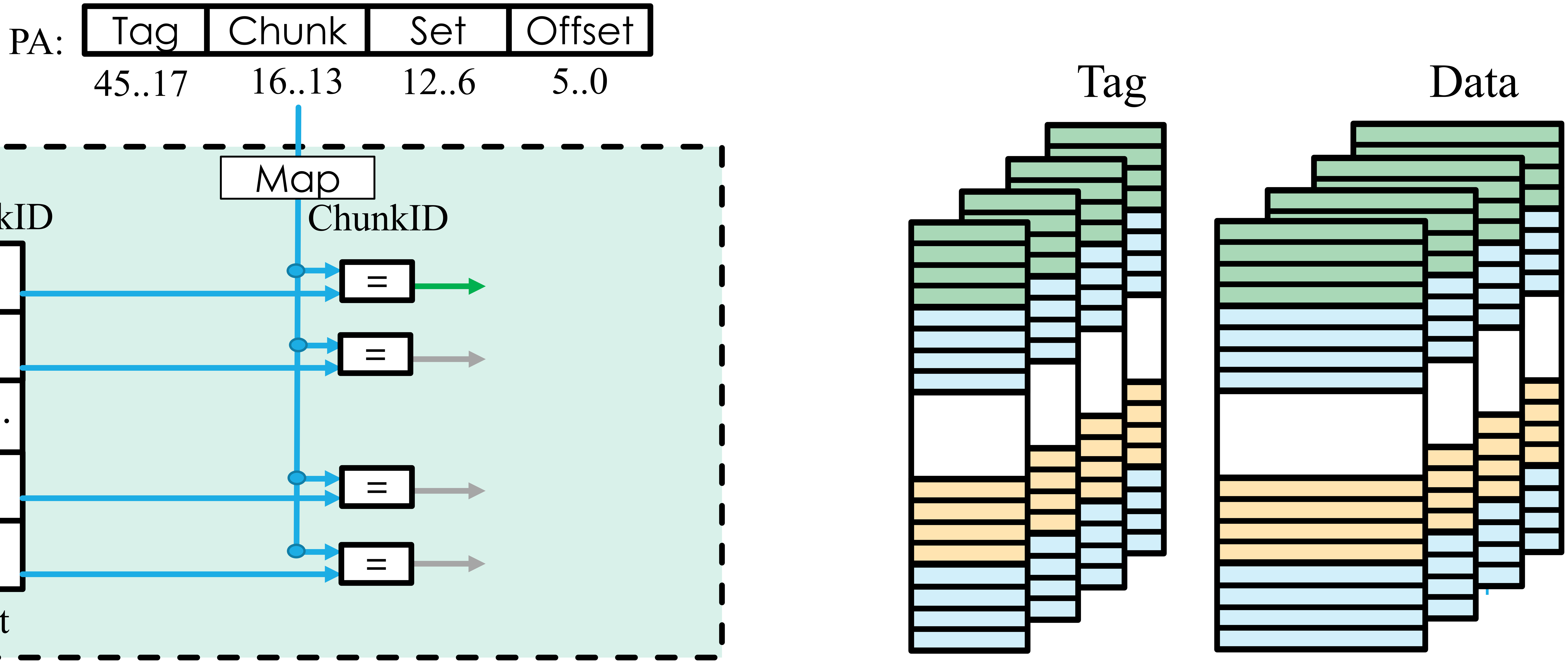


Accessing a Function's Tile

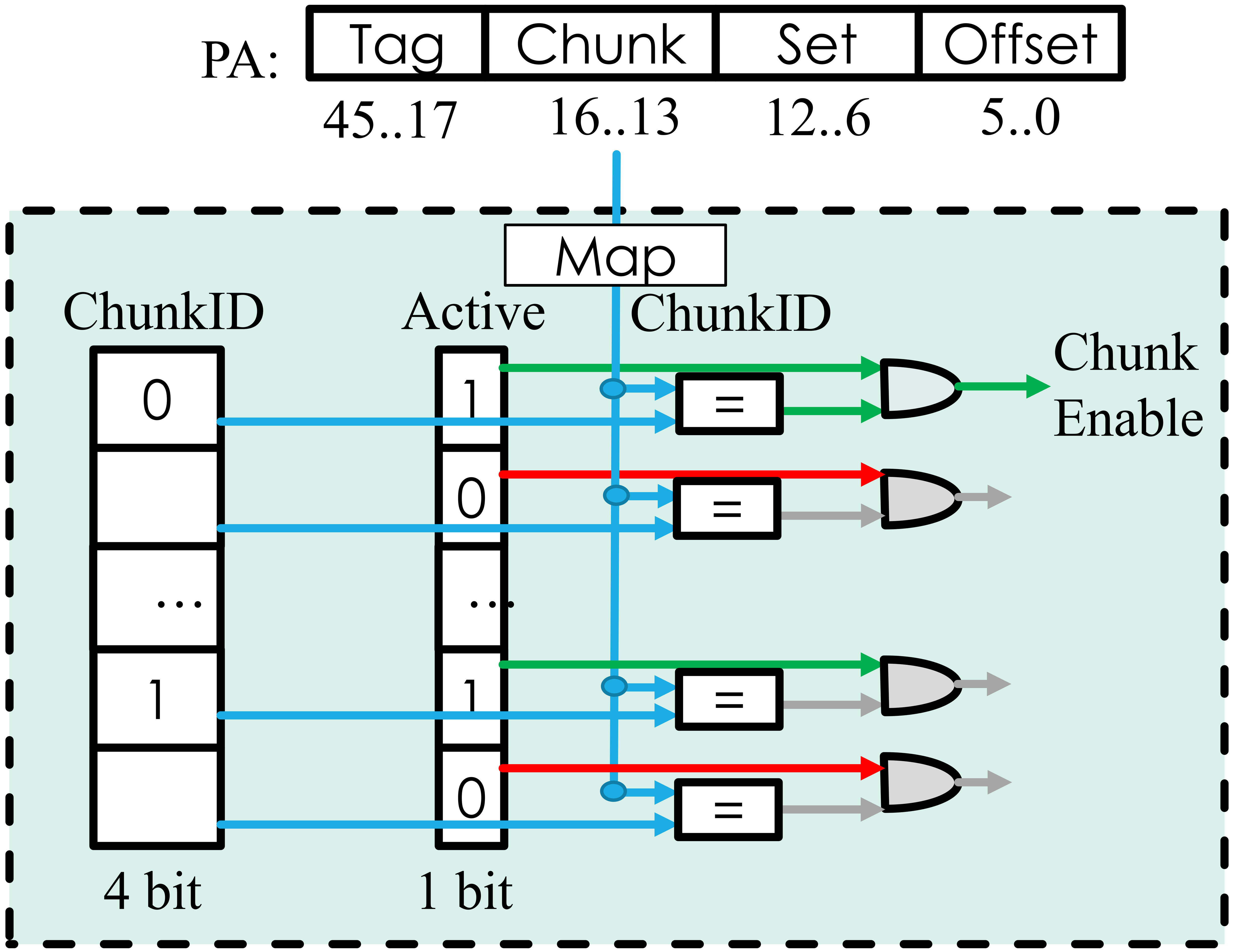


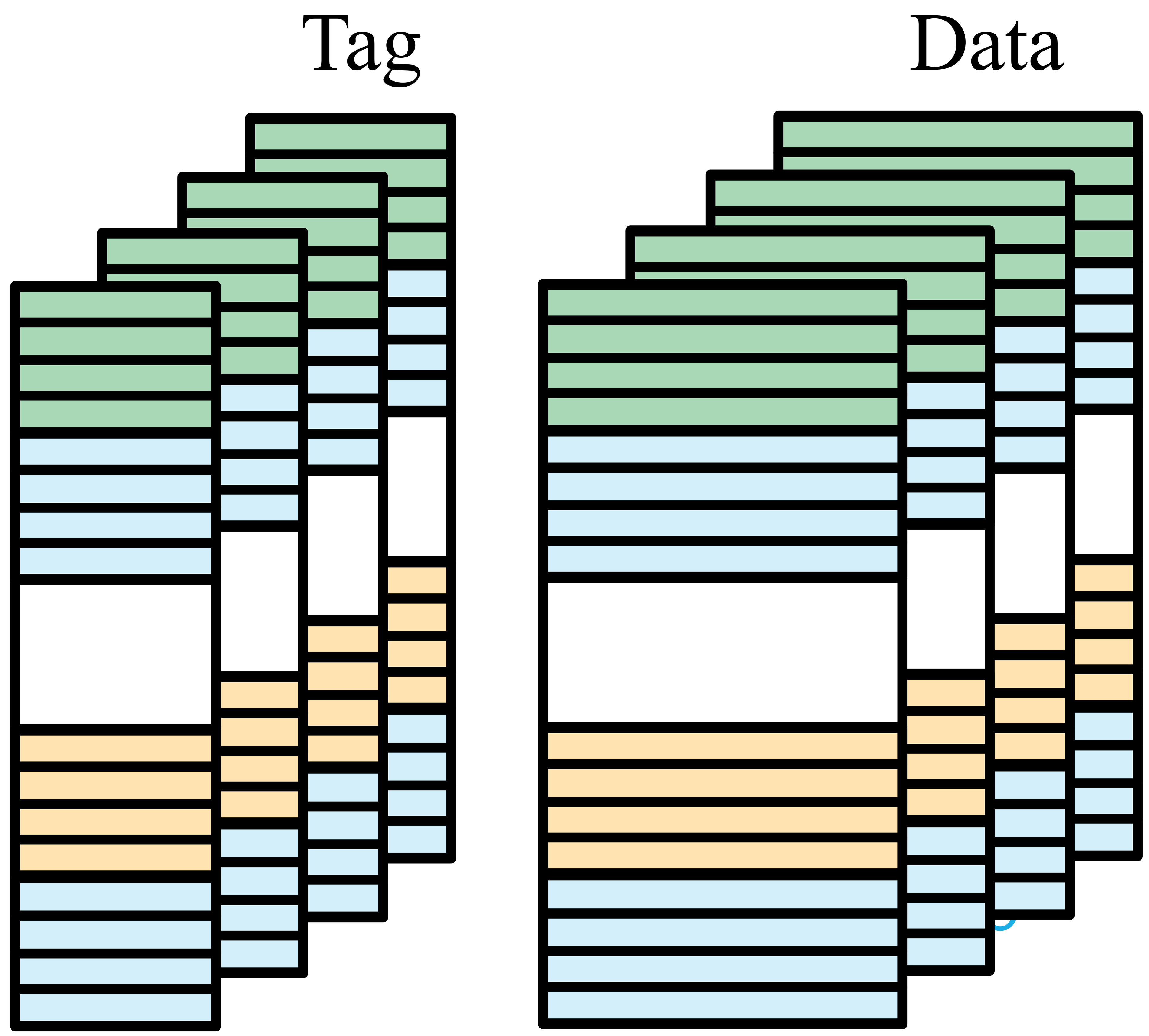


Accessing a Function's Tile

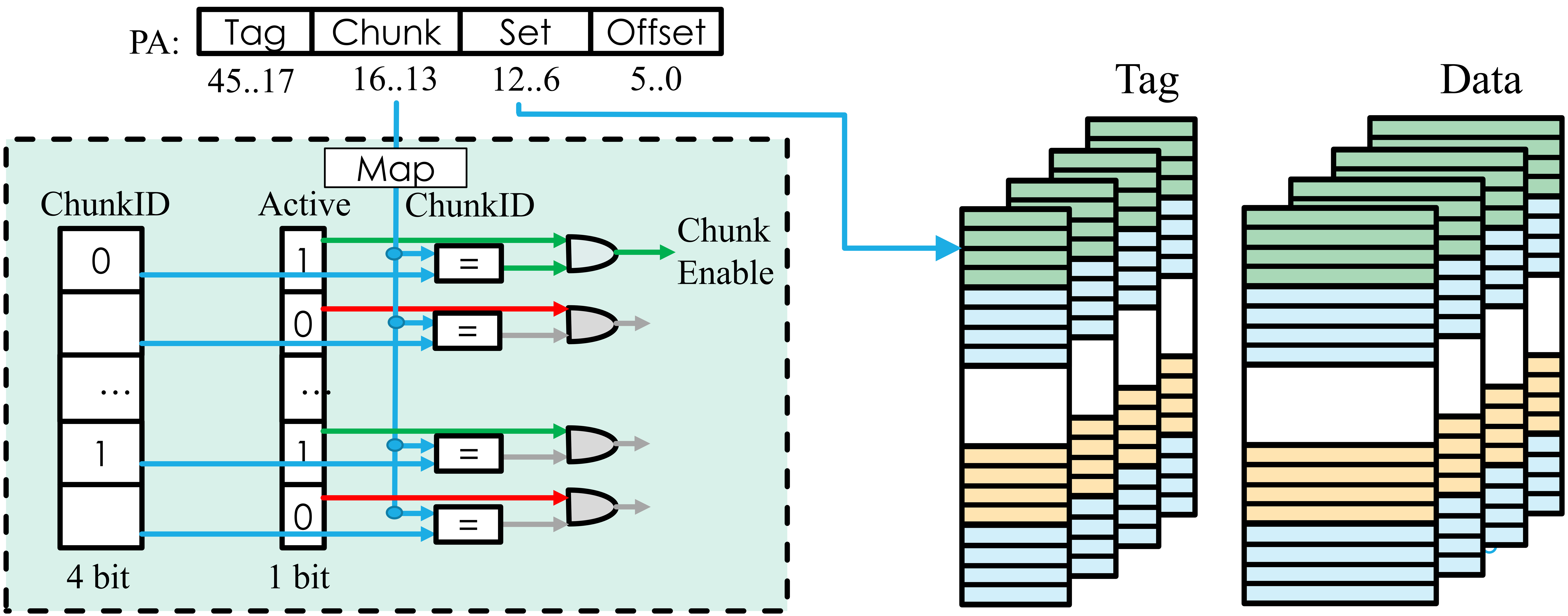


Accessing a Function's Tile

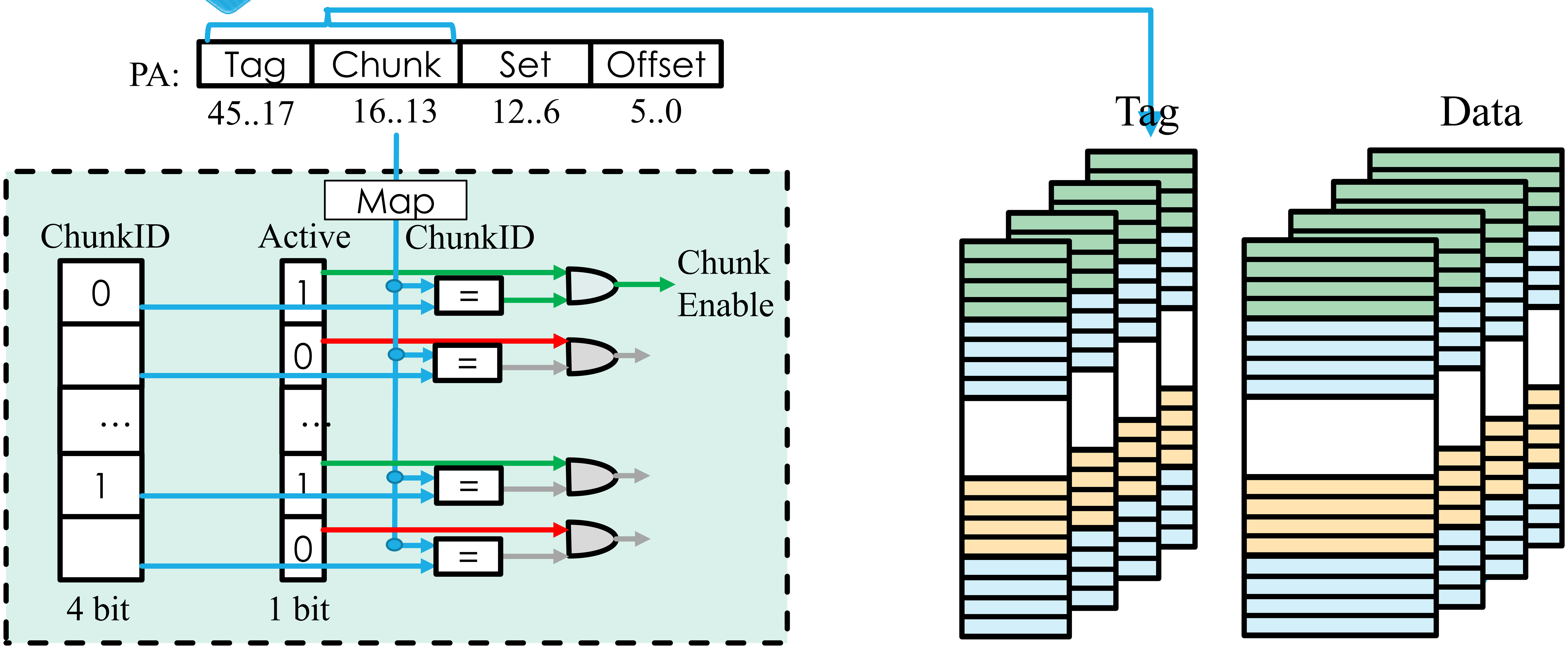




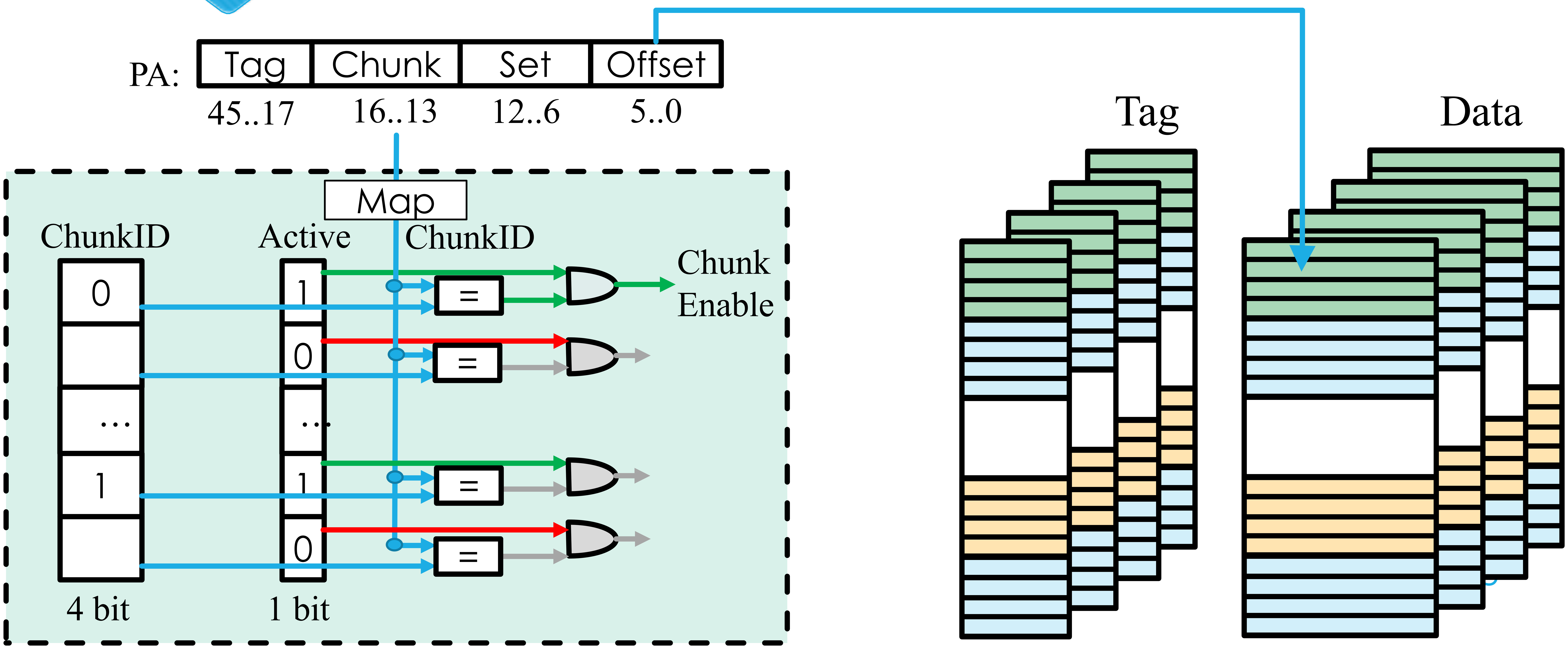
Accessing a Function's Tile



Accessing a Function's Tile



Accessing a Function's Tile

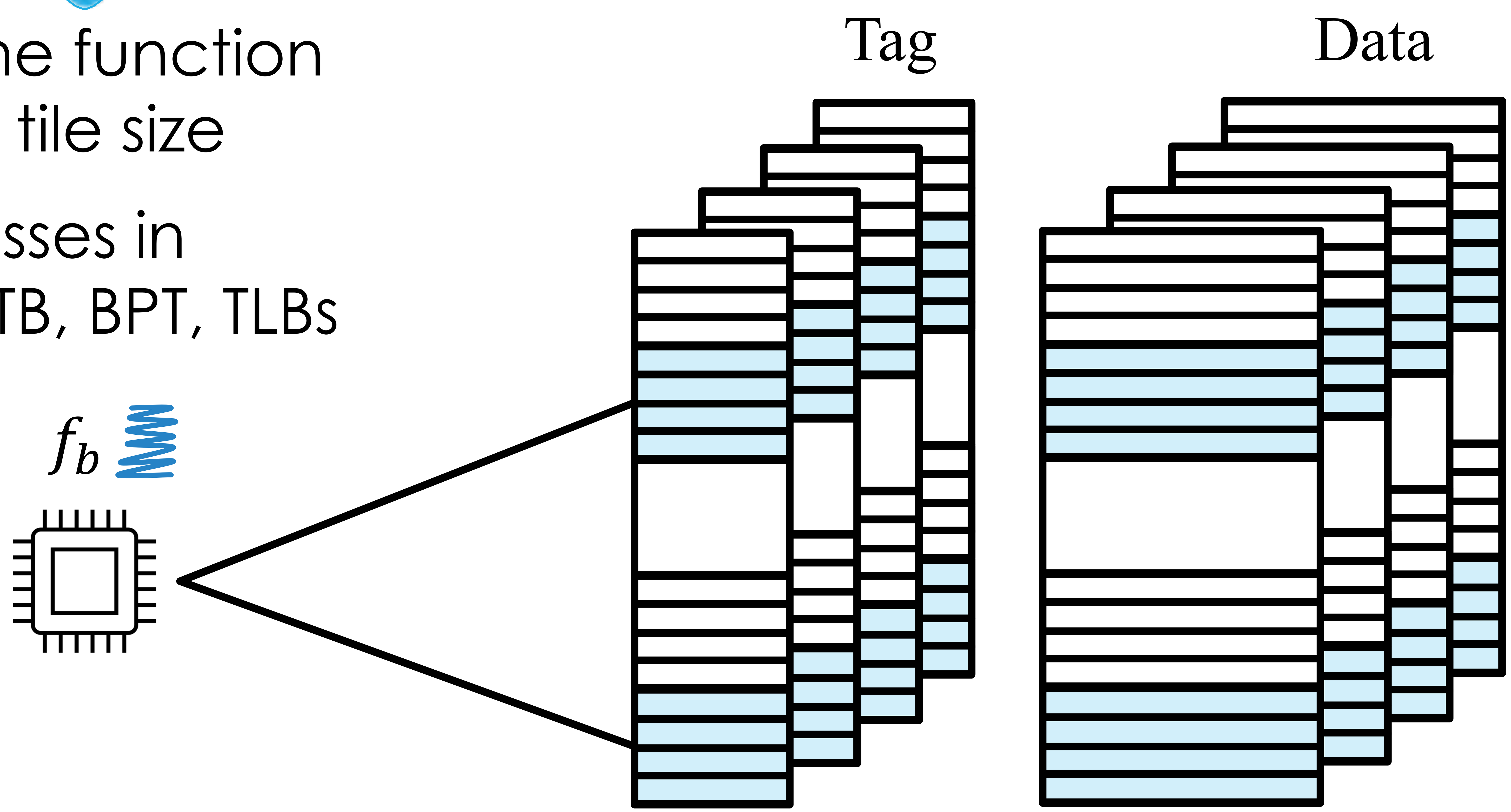


Mosaic: An Architecture for FaaS Environments

- 4 main principles:
  - Partition stateful structures into per-function tiles
  - Size per-function tiles based on individual function needs
  - **Performance-modeling to predict optimal tile size**

Performance Modeling for Optimal Tile Size

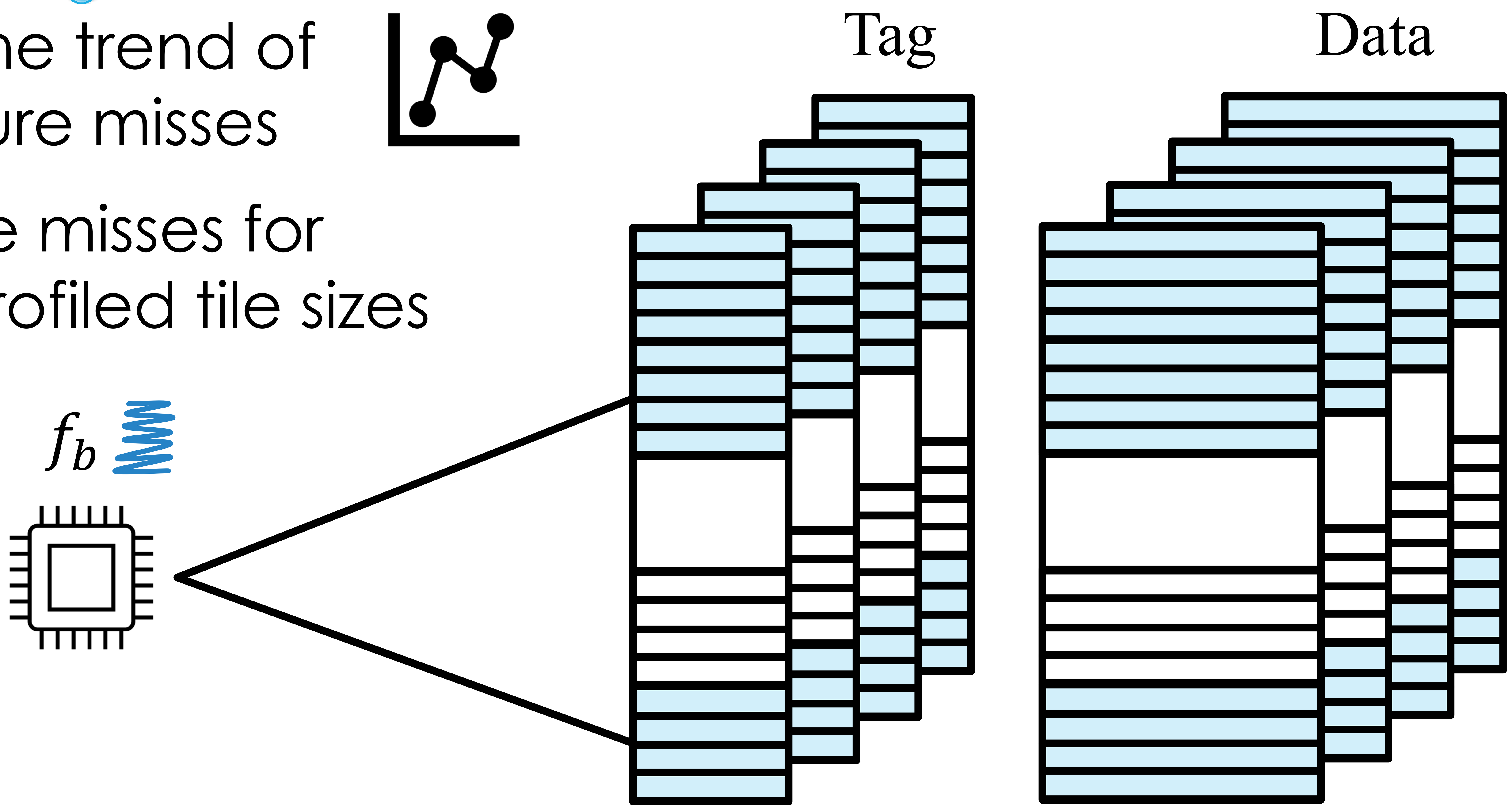
- Execute the function with some tile size
- Record misses in caches, BTB, BPT, TLBs



Performance Modeling for Optimal Tile Size

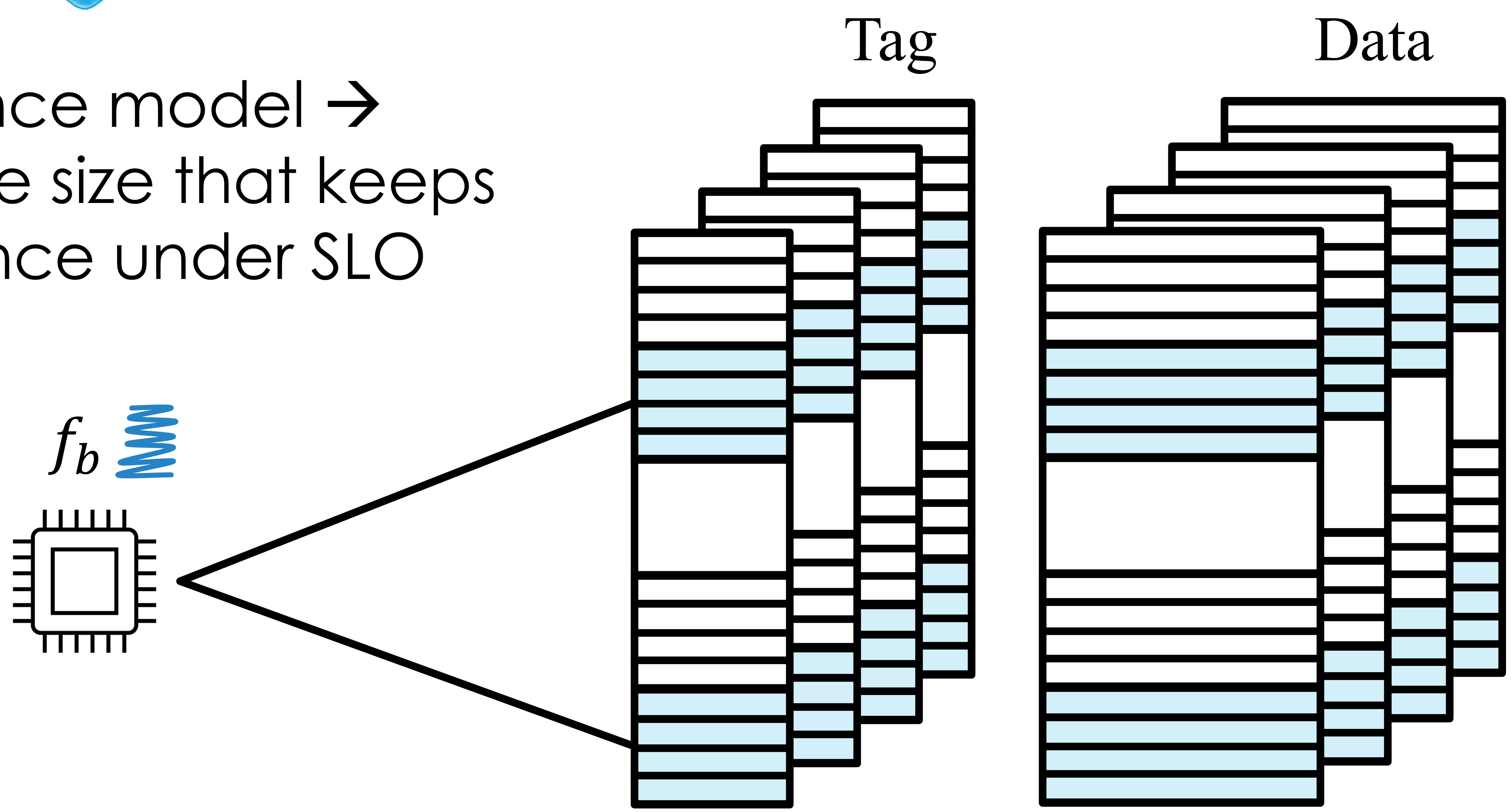
- Deduce the trend of per-structure misses

- Predict the misses for the non-profiled tile sizes



Performance Modeling for Optimal Tile Size

- Performance model → minimal tile size that keeps performance under SLO



Transfer Learning for Lower Profiling Overheads

Database of already profiled functions

Func Char.	Optimal Tile Size
[IPC, Data and Inst. Footprint, Num. Branch]	[L2$, L2TLB, LLC, BTB, BPT]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]

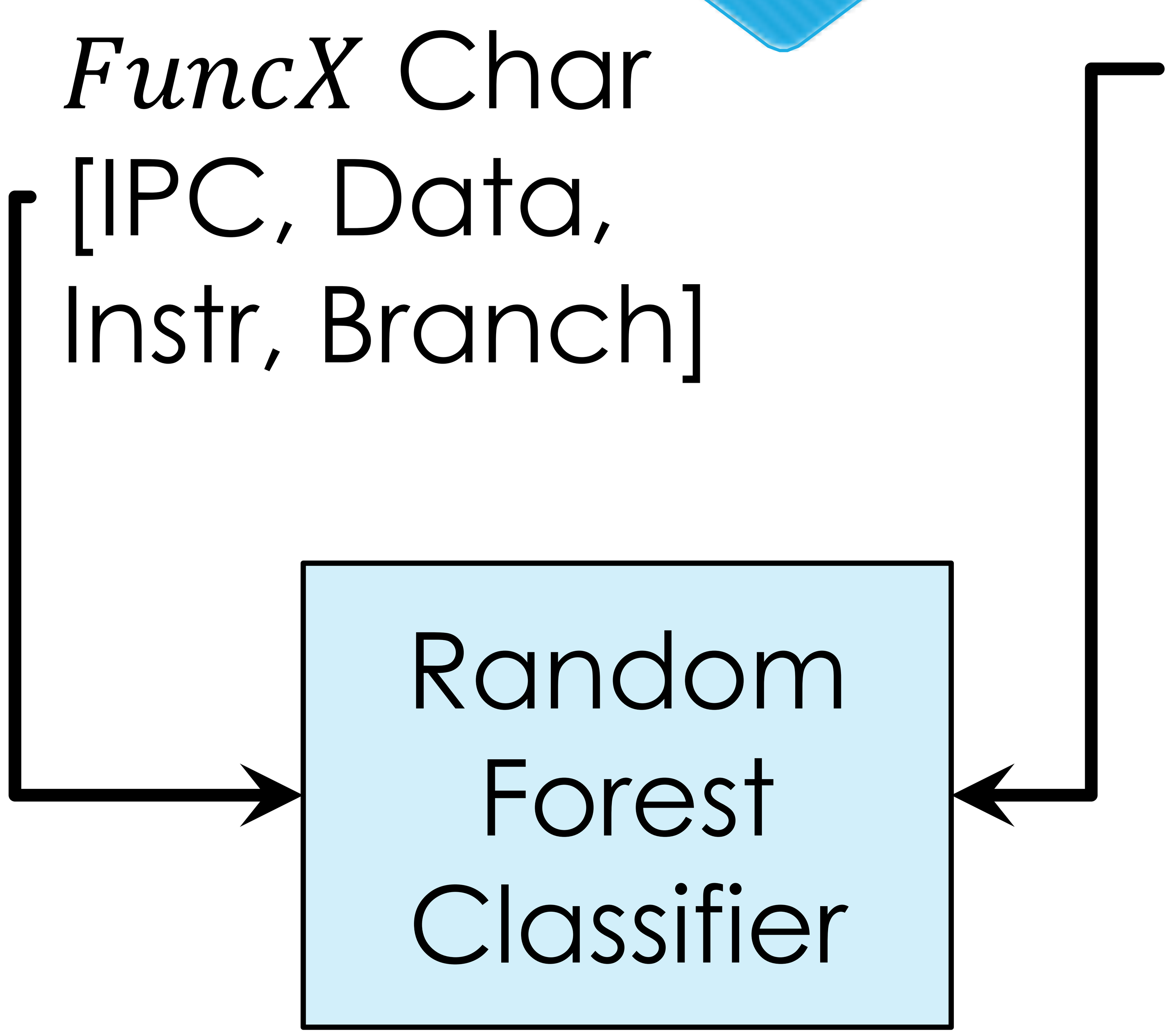
Transfer Learning for Lower Profiling Overheads

$FuncX$ Char
[IPC, Data,
Instr, Branch]

Database of already profiled functions

Func Char.	Optimal Tile Size
[IPC, Data and Inst. Footprint, Num. Branch]	[L2$, L2TLB, LLC, BTB, BPT]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]

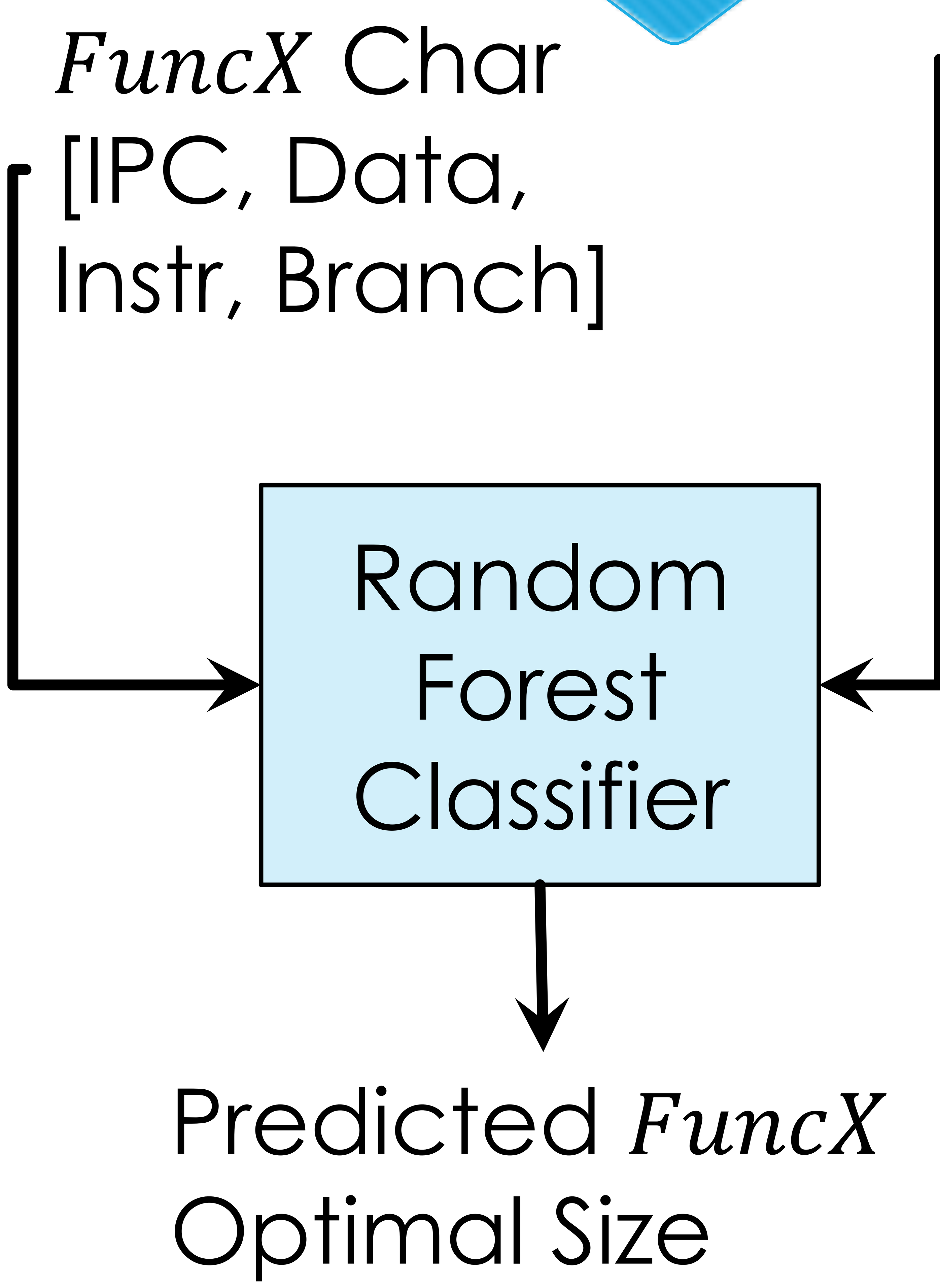
Transfer Learning for Lower Profiling Overheads



Database of already profiled functions

Func Char.	Optimal Tile Size
[IPC, Data and Inst. Footprint, Num. Branch]	[L2$, L2TLB, LLC, BTB, BPT]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]

Transfer Learning for Lower Profiling Overheads



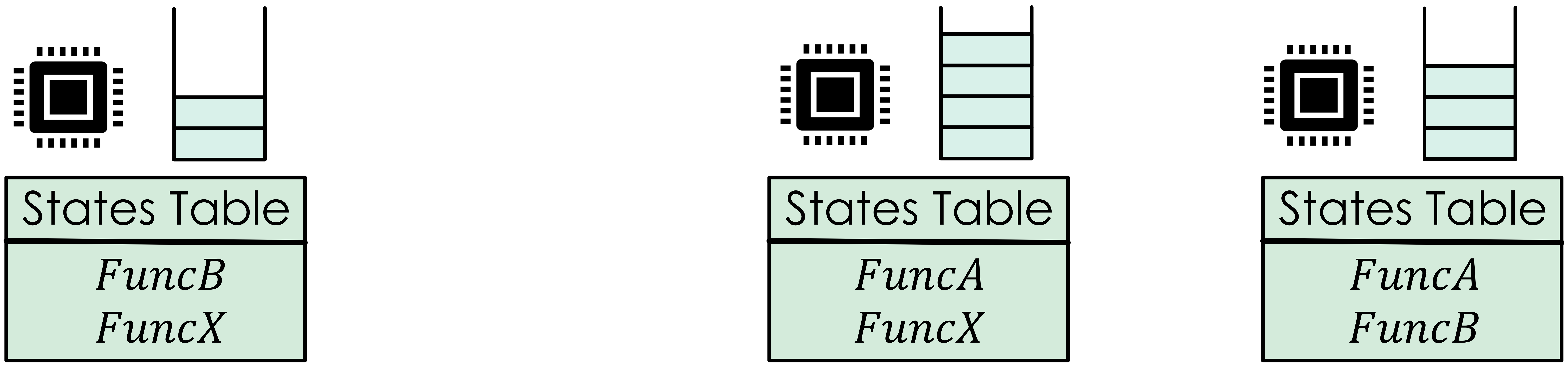
Database of already profiled functions

Func Char.	Optimal Tile Size
[IPC, Data and Inst. Footprint, Num. Branch]	[L2$, L2TLB, LLC, BTB, BPT]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]
[.., .., .., ..]	[.., .., .., .., .., .., .., .., ..]

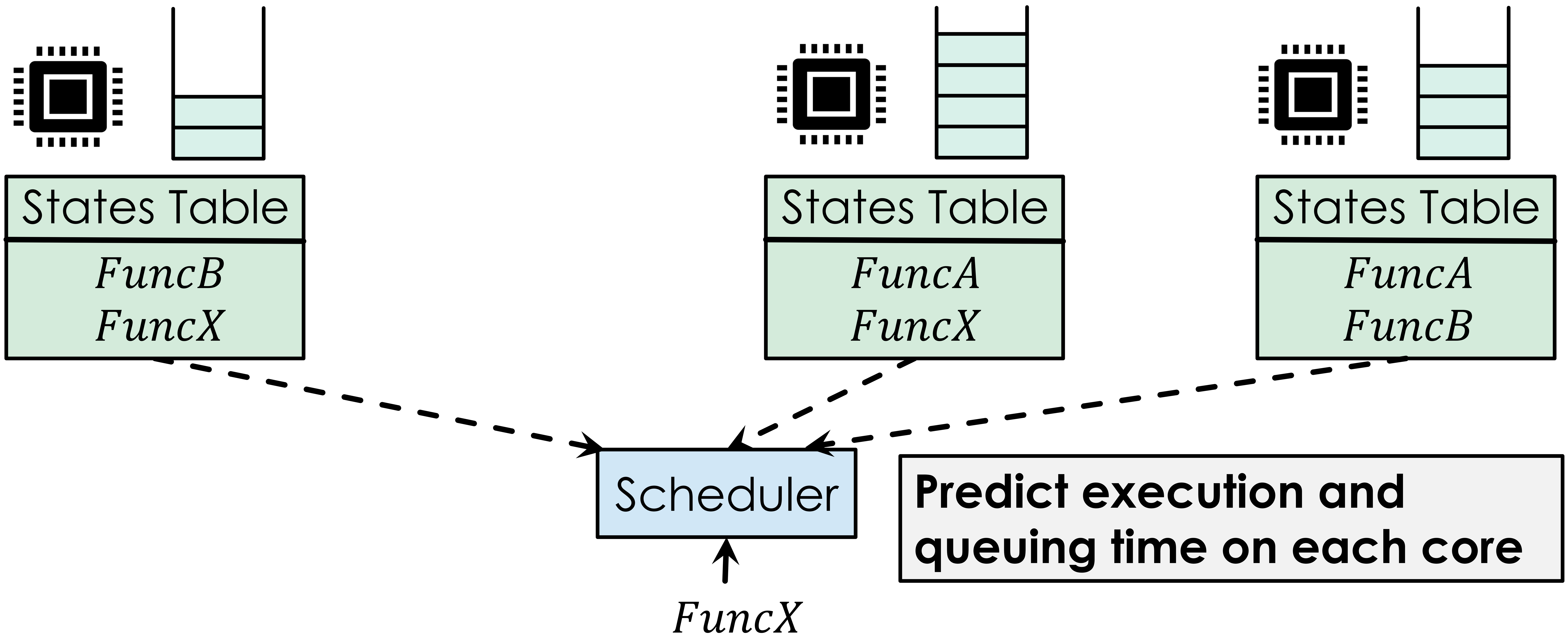
Mosaic: An Architecture for FaaS Environments

- 4 main principles:
  - Partition stateful structures into per-function tiles
  - Size per-function tiles based on individual function needs
  - Performance-modeling to predict optimal tile size
  - **Tight coupling of hardware and software**

Optimized Software Stack: Online Micro-architectural Aware Scheduling

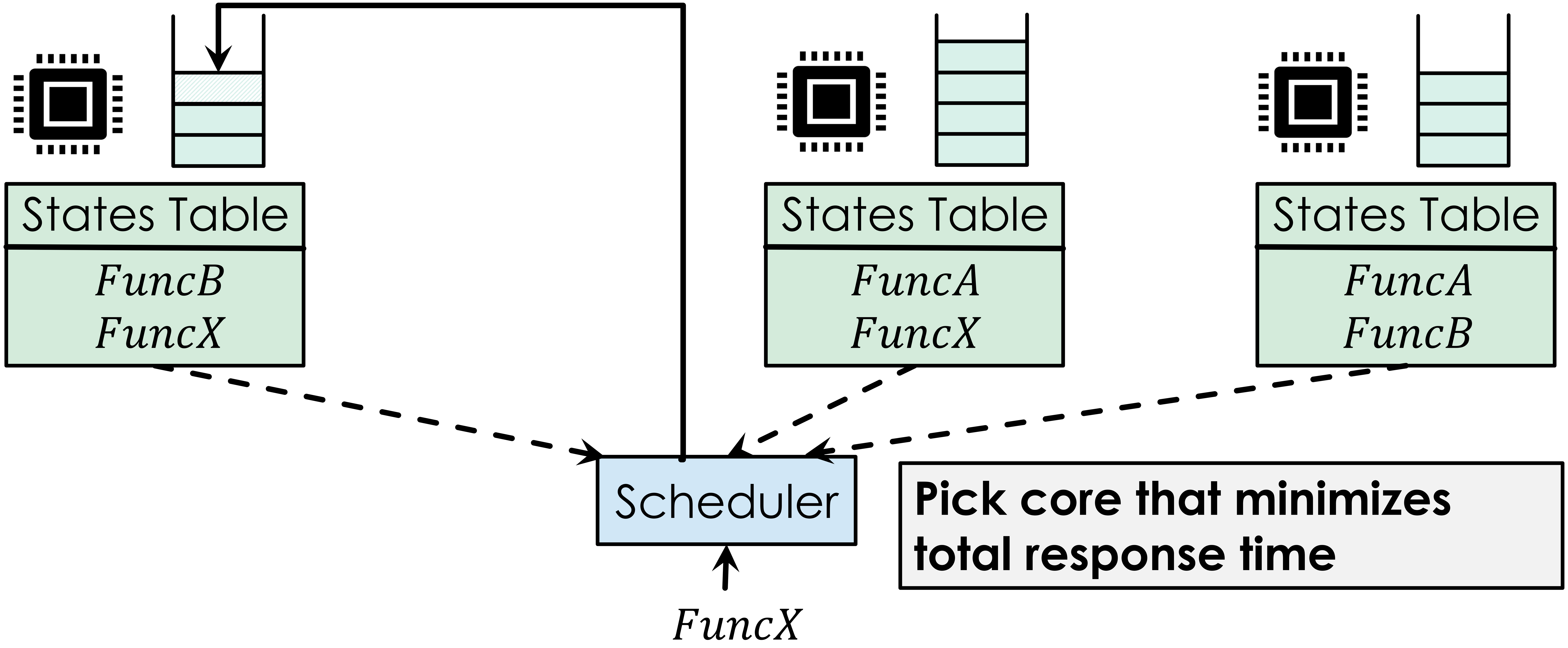


Optimized Software Stack: Online Micro-architectural Aware Scheduling



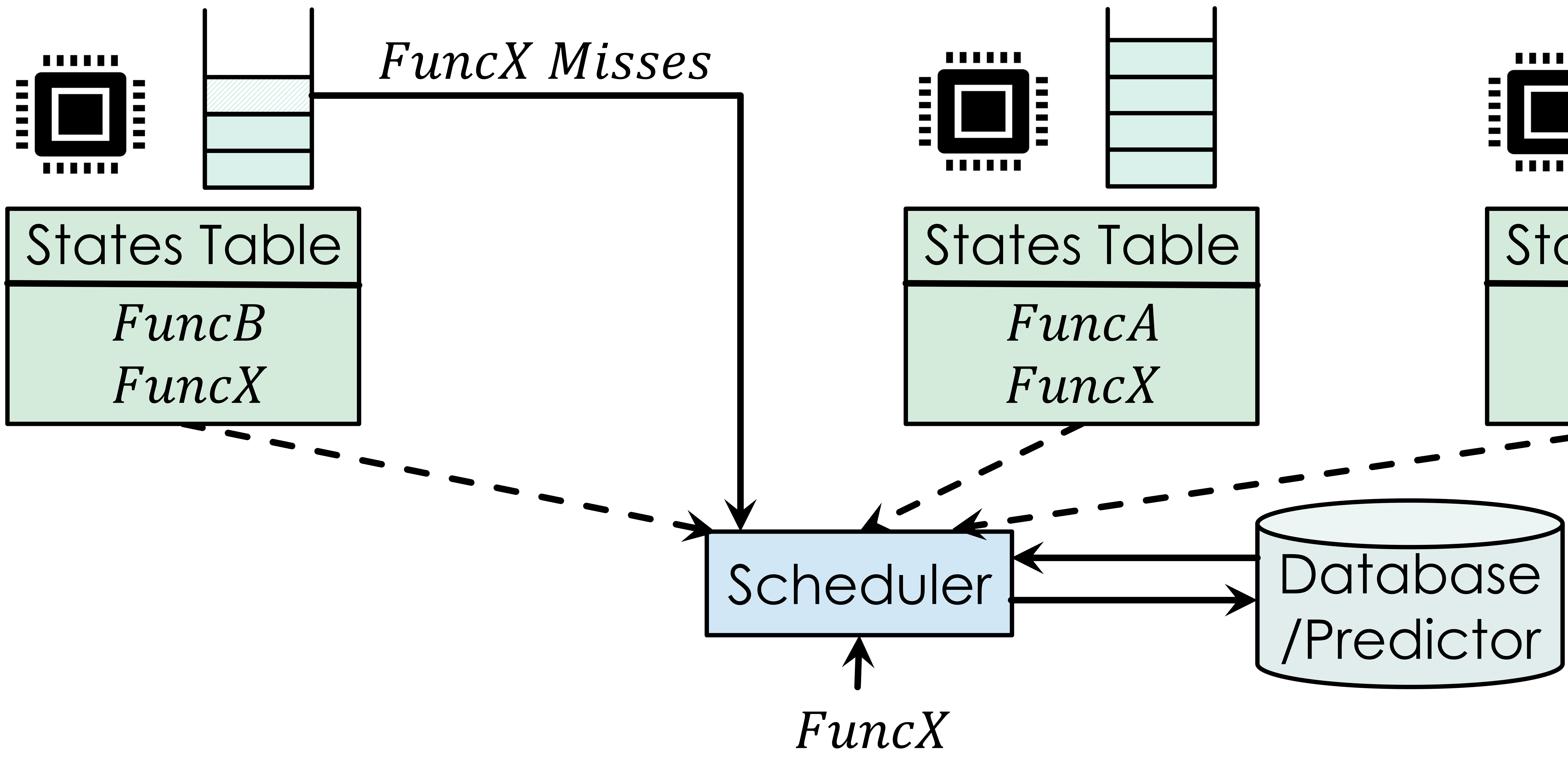
Predict execution and queuing time on each core

Optimized Software Stack: Online Micro-architectural Aware Scheduling

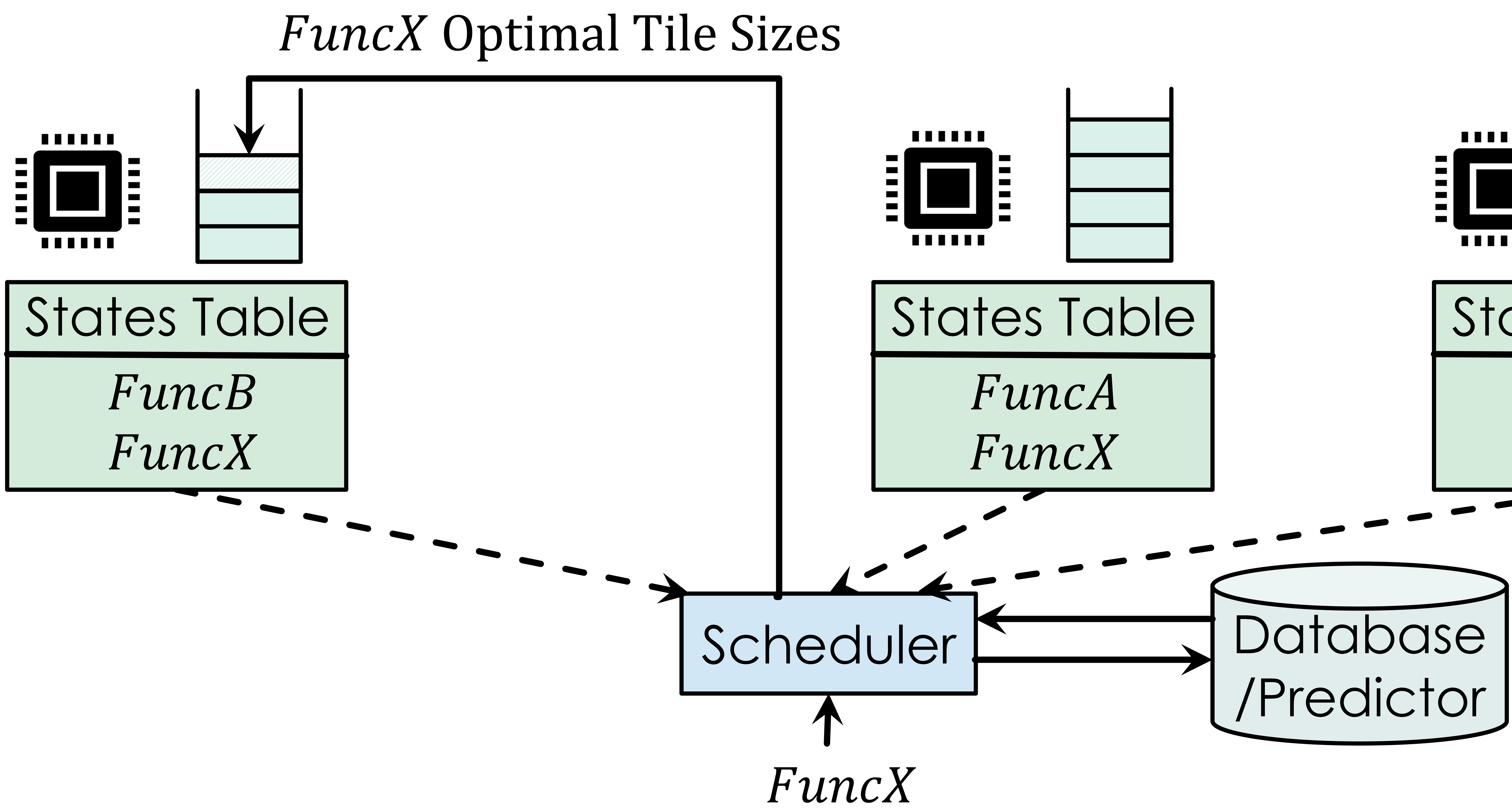


Pick core that minimizes total response time

Optimized Software Stack: Offline Performance Modeling



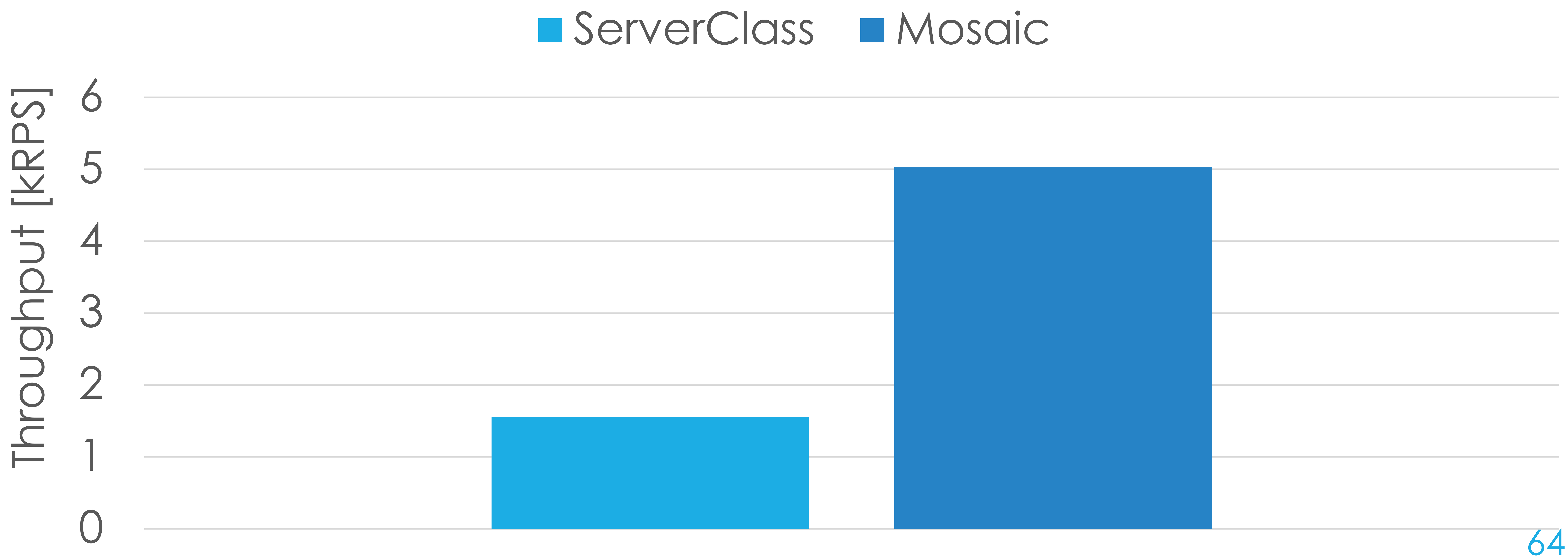
Optimized Software Stack: Offline Performance Modeling



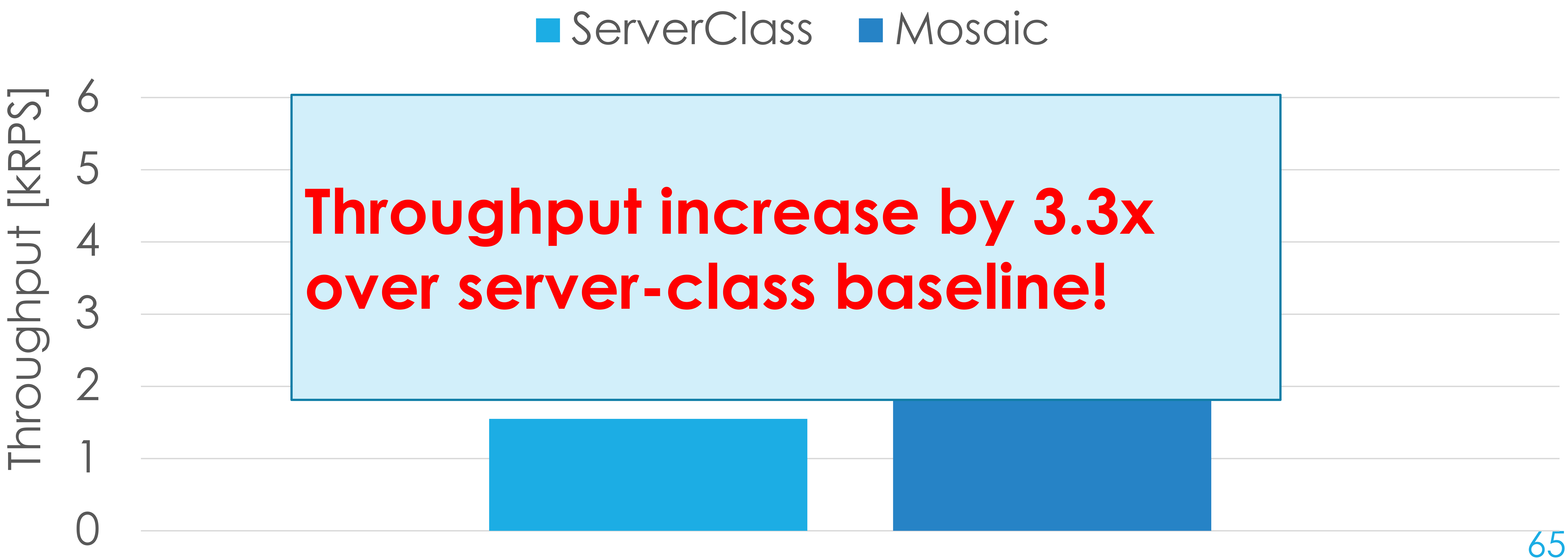
Evaluation Methodology

- Full-system simulations: QEMU + SST + DRAM-Sim2
- 16-core server modeled after Golden Cove in SPR
- McPAT + CACTI for power and area estimates
- Open-source functions with Azure production invocation traces

Mosaic Significantly Boosts Throughput



Mosaic Significantly Boosts Throughput



Throughput increase by 3.3x over server-class baseline!

Conclusion

- Imbalance between current processors and serverless environments
- Mosaic – an architecture for serverless environments
  - Extends current processors optimized for monolithic applications
- Mosaic delivers high performance for serverless workloads
  - Tail latency reduced by 75%
  - Throughput improved 3.3x
  - Average power reduced by 22%

Questions?

Mosaic: Harnessing the *Micro*-architectural Resources of Servers in Serverless Environments

MICRO 2024

Jovan Stojkovic, Esha Choukse*, Enrique Saurez*, Íñigo Goiri*, Josep Torrellas
University of Illinois at Urbana-Champaign, *Microsoft Azure Research Systems