

Weeding out Front-End Stalls with Uneven Block Size Instruction Cache

Roman K. Brunner and Rakesh Kumar

Norwegian University of Science and Technology



NTNU

Department of Computer Science
Computer Architecture Lab

What to remember

- 
- Problem: Frontend is a critical bottleneck in servers
- Stalls for up to 49% (Google) and 37% (Meta) of execution time
 - Instruction cache (L1-I) misses are a major contributor to the bottleneck

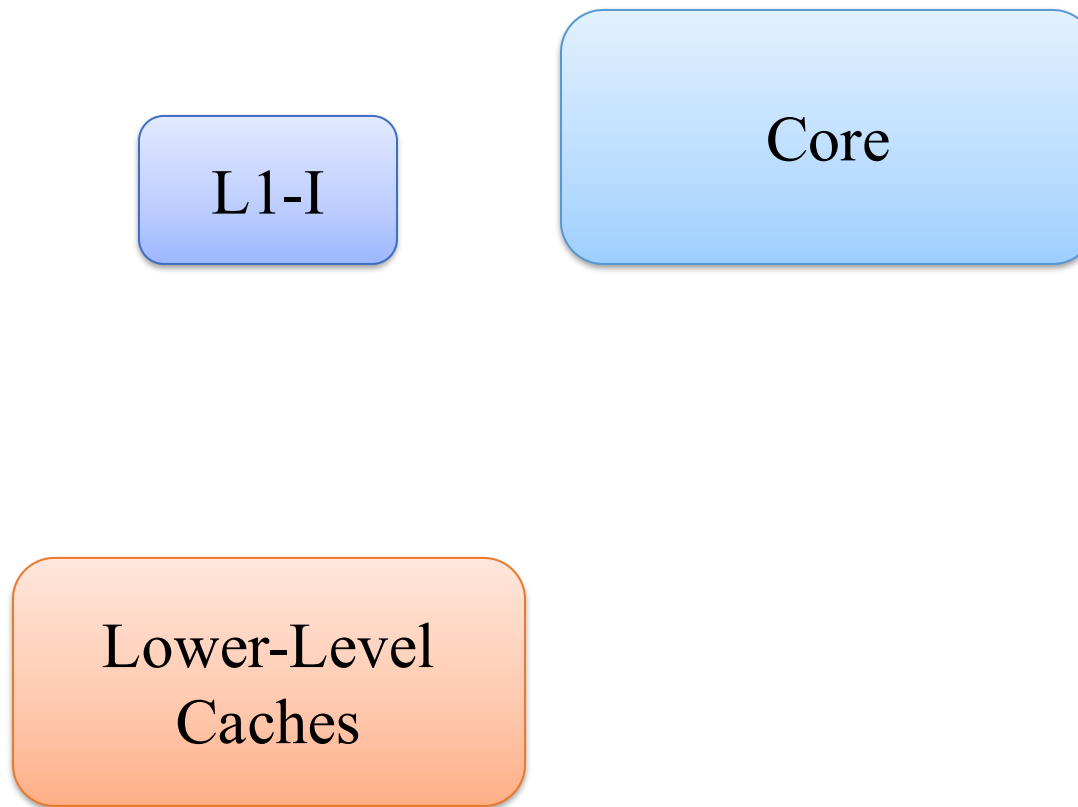
- 
- Observation: L1-I storage utilization is poor
- Large variance in spatial locality in the instruction stream
 - Fixed L1-I cache block sizes

- 
- Idea: Spatial variance aware L1-I design
- Introduce Uneven Block Sizes (UBS) in L1-I

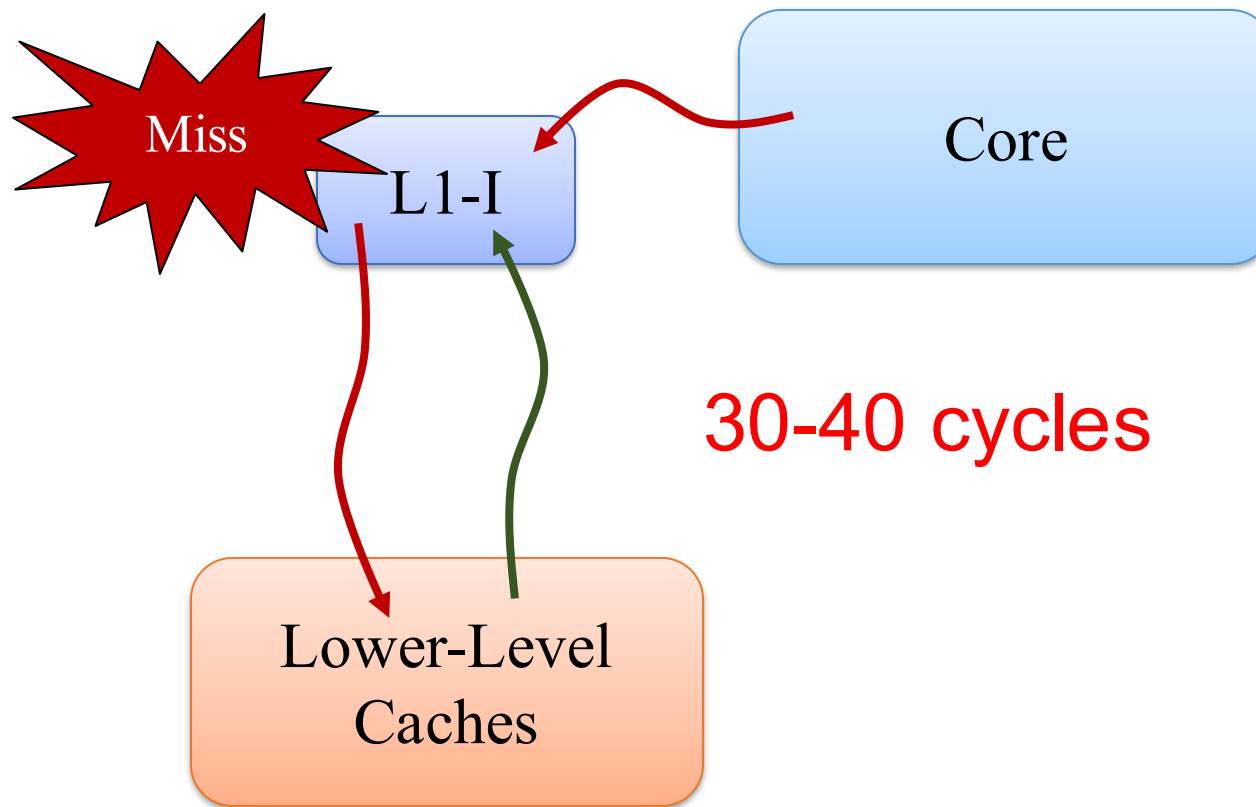
UBS approaches the performance
of a L1-I 2x the size



Front-End Problem: L1-I Misses



Front-End Problem: L1-I Misses



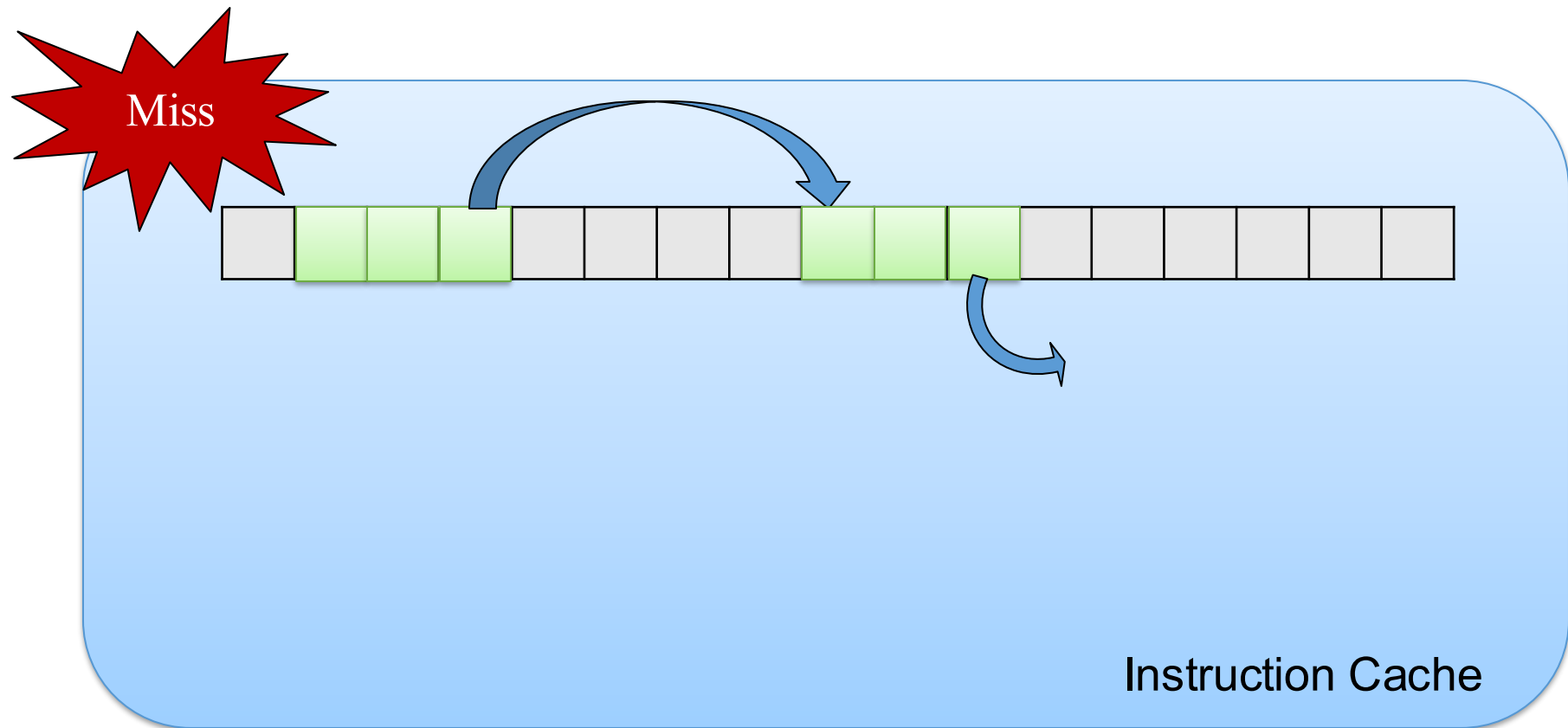
Front-End Problem: L1-I Misses

We analyse how effectively
L1-I storage is utilized

30-40 cycles

Lower-Level
Caches

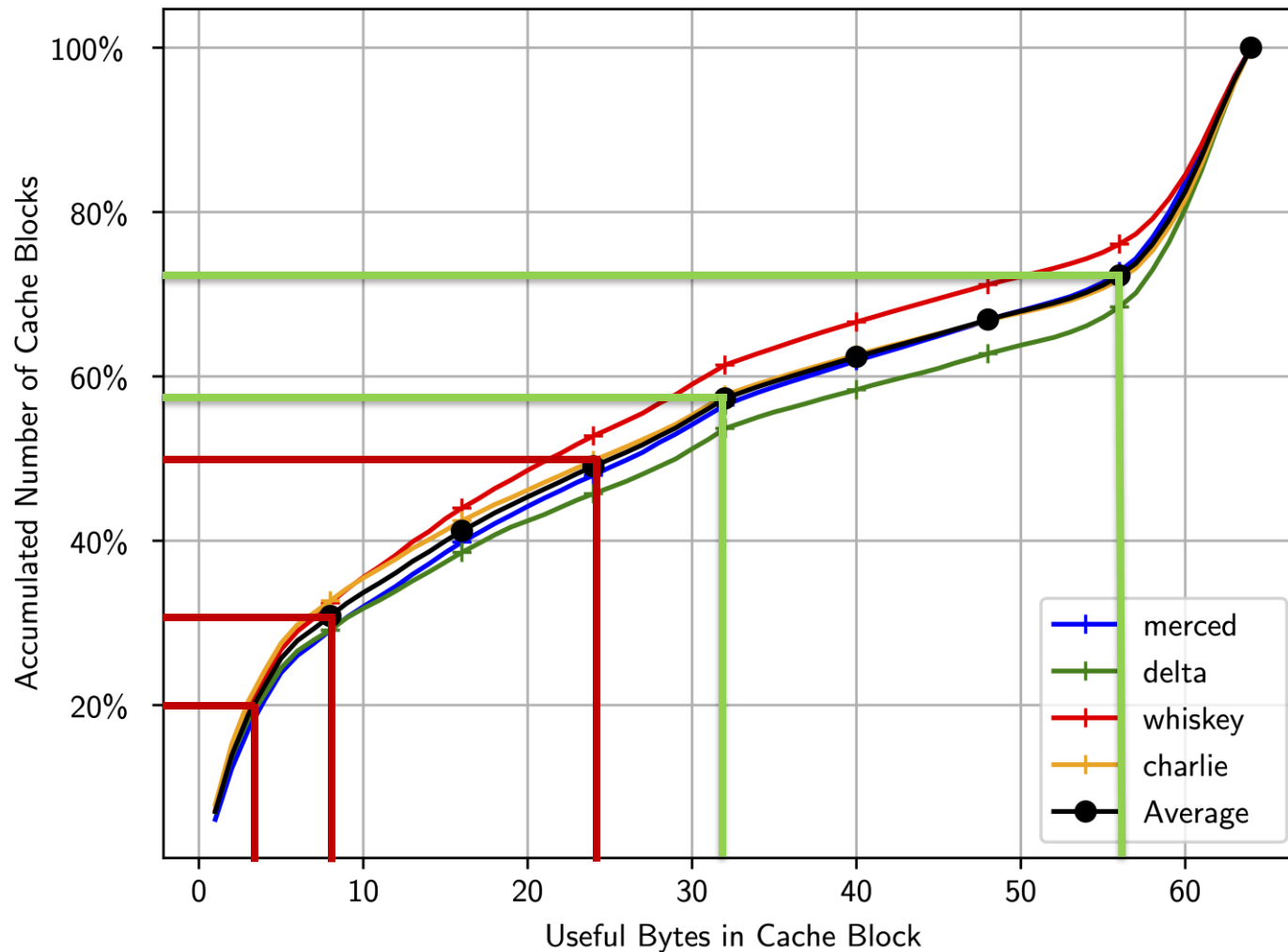
Lifecycle of a L1-I cache block



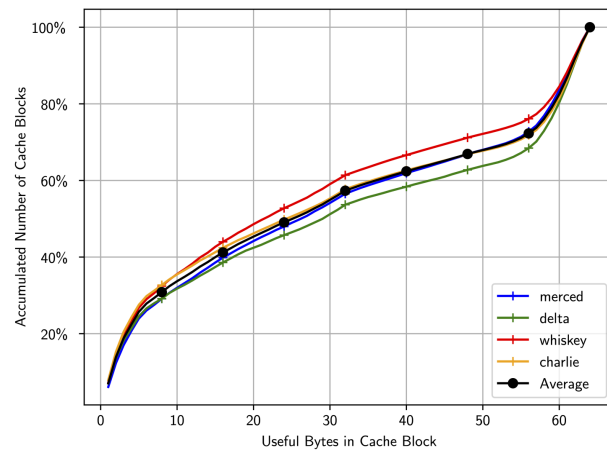
Lifecycle of a L1-I cache block



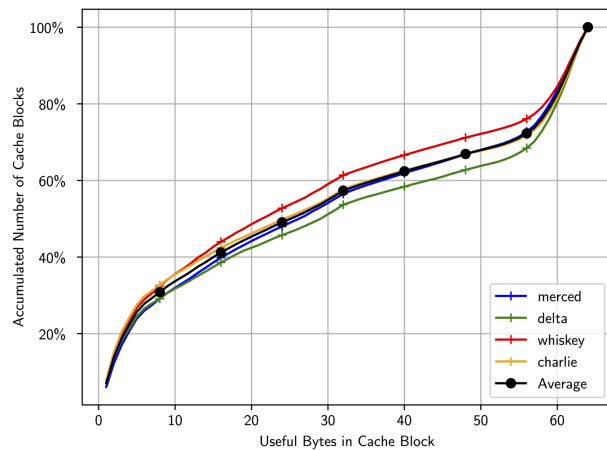
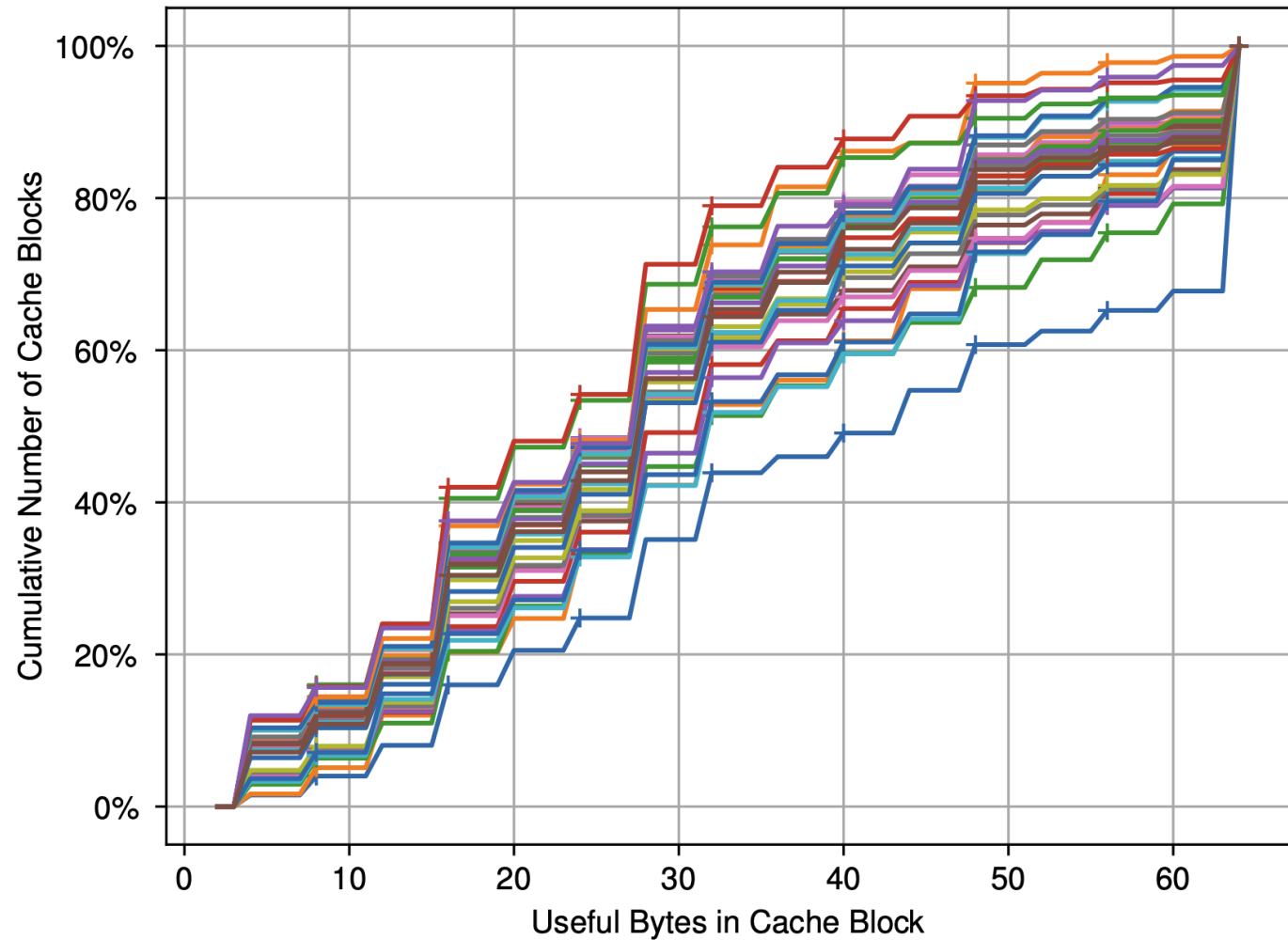
Quantifying Useful Bytes in L1-I Blocks



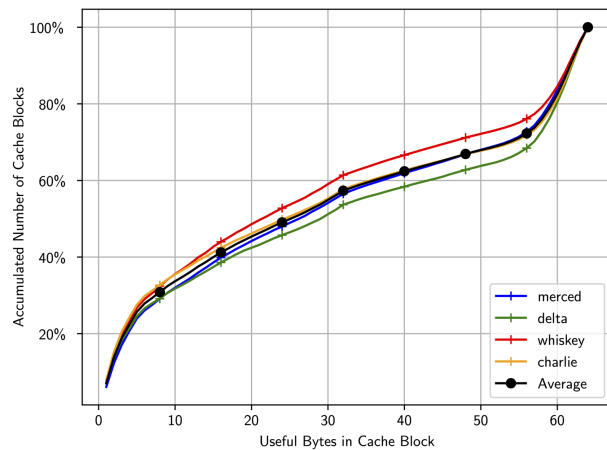
Quantifying Useful Bytes in L1-I Blocks



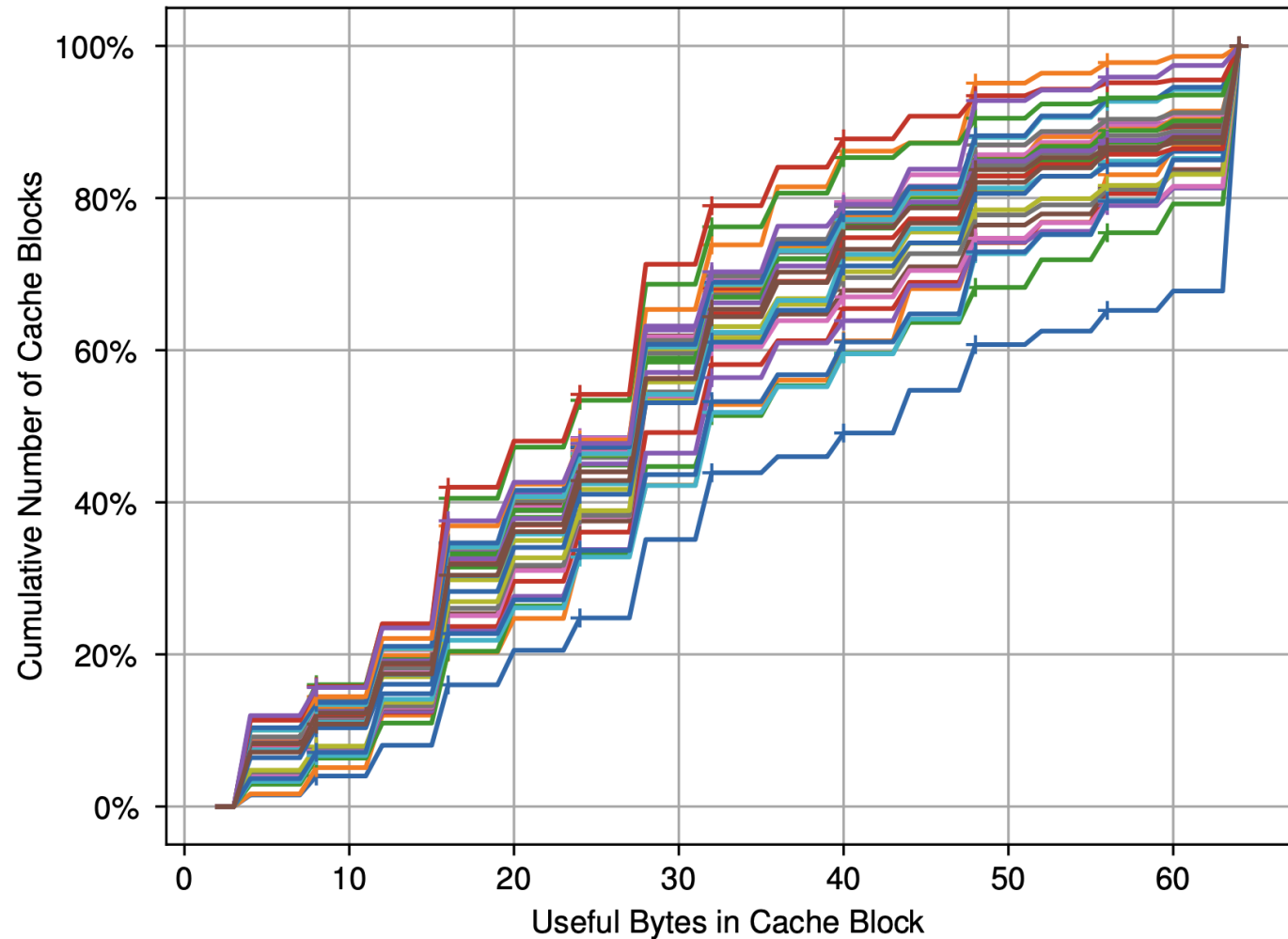
Quantifying Useful Bytes in L1-I Blocks



Quantifying Useful Bytes in L1-I Blocks

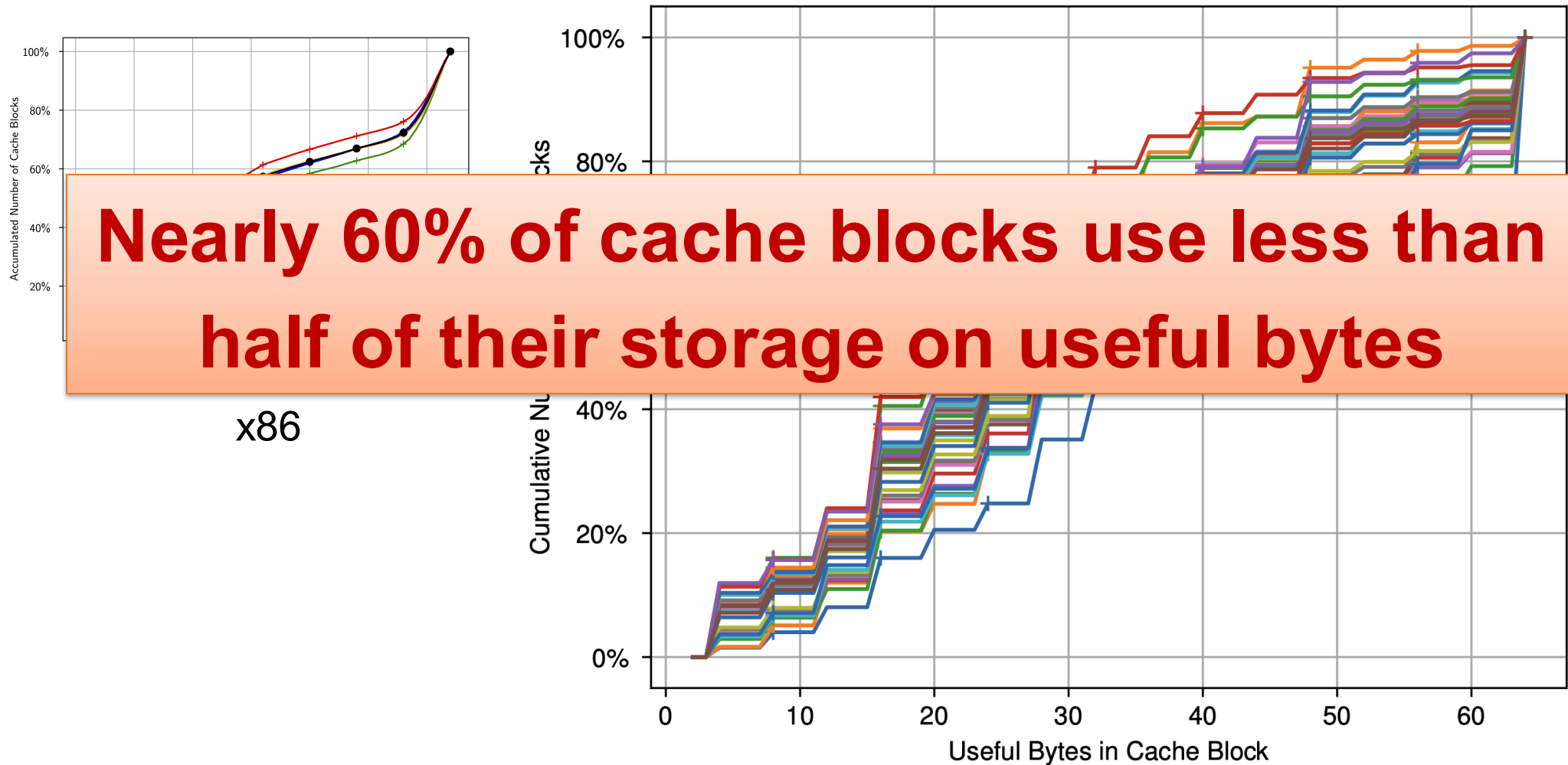


x86

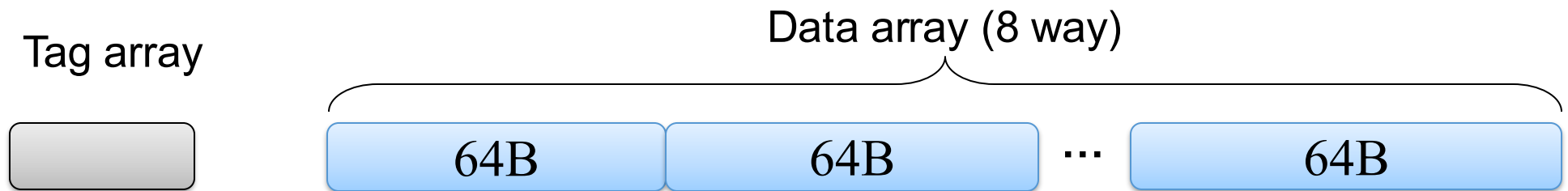


ARM

Quantifying Useful Bytes in L1-I Blocks



Source of Low Storage Efficiency in L1-I

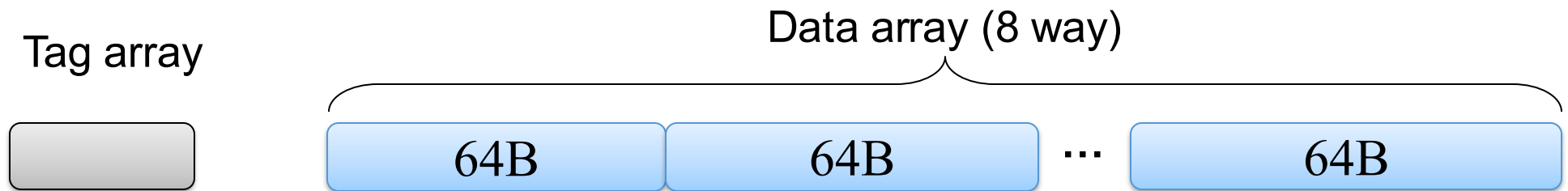


Block-Sizes are fixed

& Large variation in spatial locality

= Large fraction of bytes are not used

Source of Low Storage Efficiency in L1-I



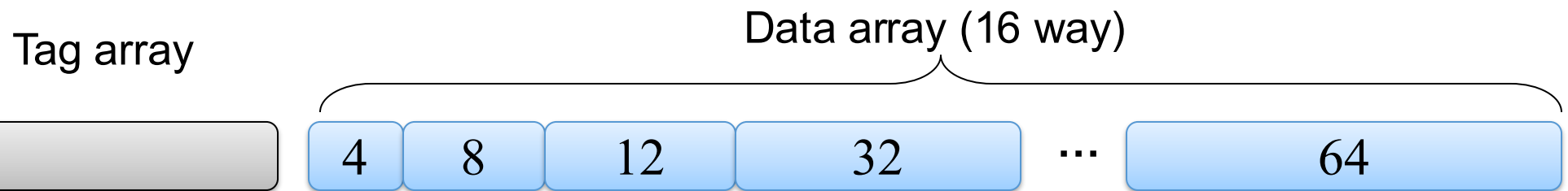
Block-Sizes are fixed

& Large variation in spatial locality

= Large fraction of bytes are not used

**From cache design perspective:
Fixed block sizes are the problem**

Solution: Uneven Block Size (UBS) Cache

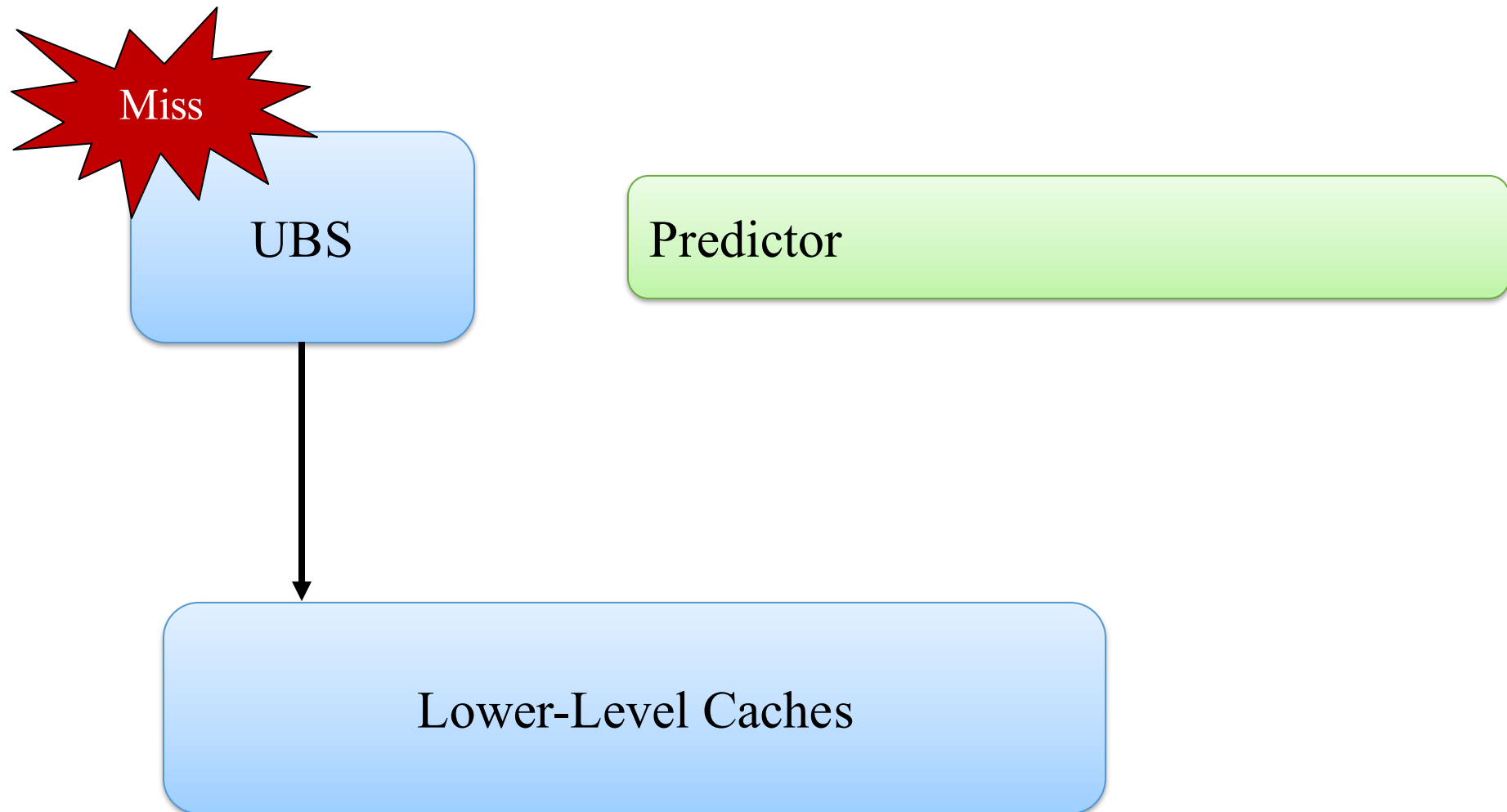


- Features differently sized ways
- Captures the variance in spatial locality
- Improves storage efficiency

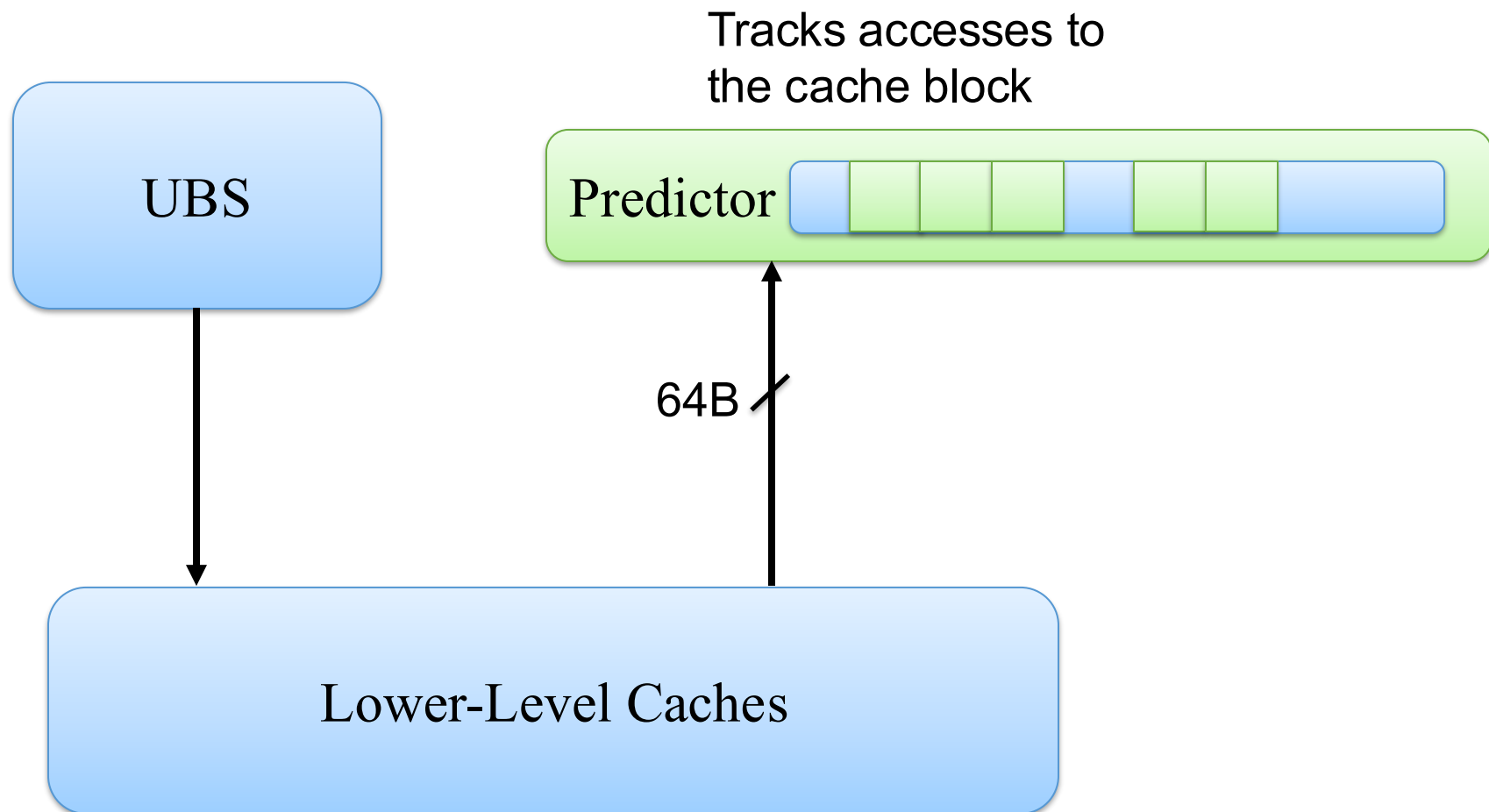
Requirements

- A mechanism to predict useful bytes
- An updated hit detection mechanism

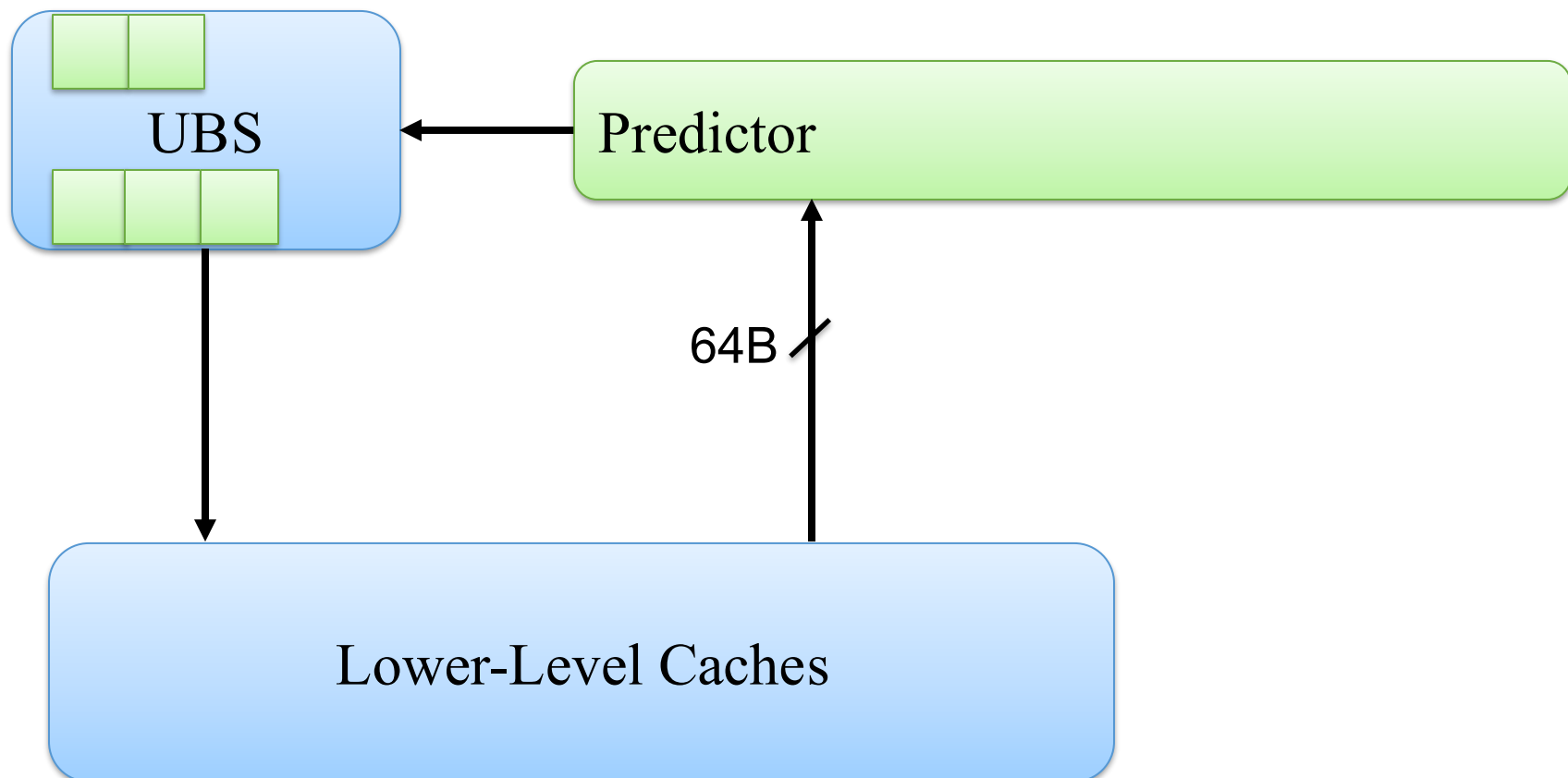
What Bytes to Insert in UBS Cache?



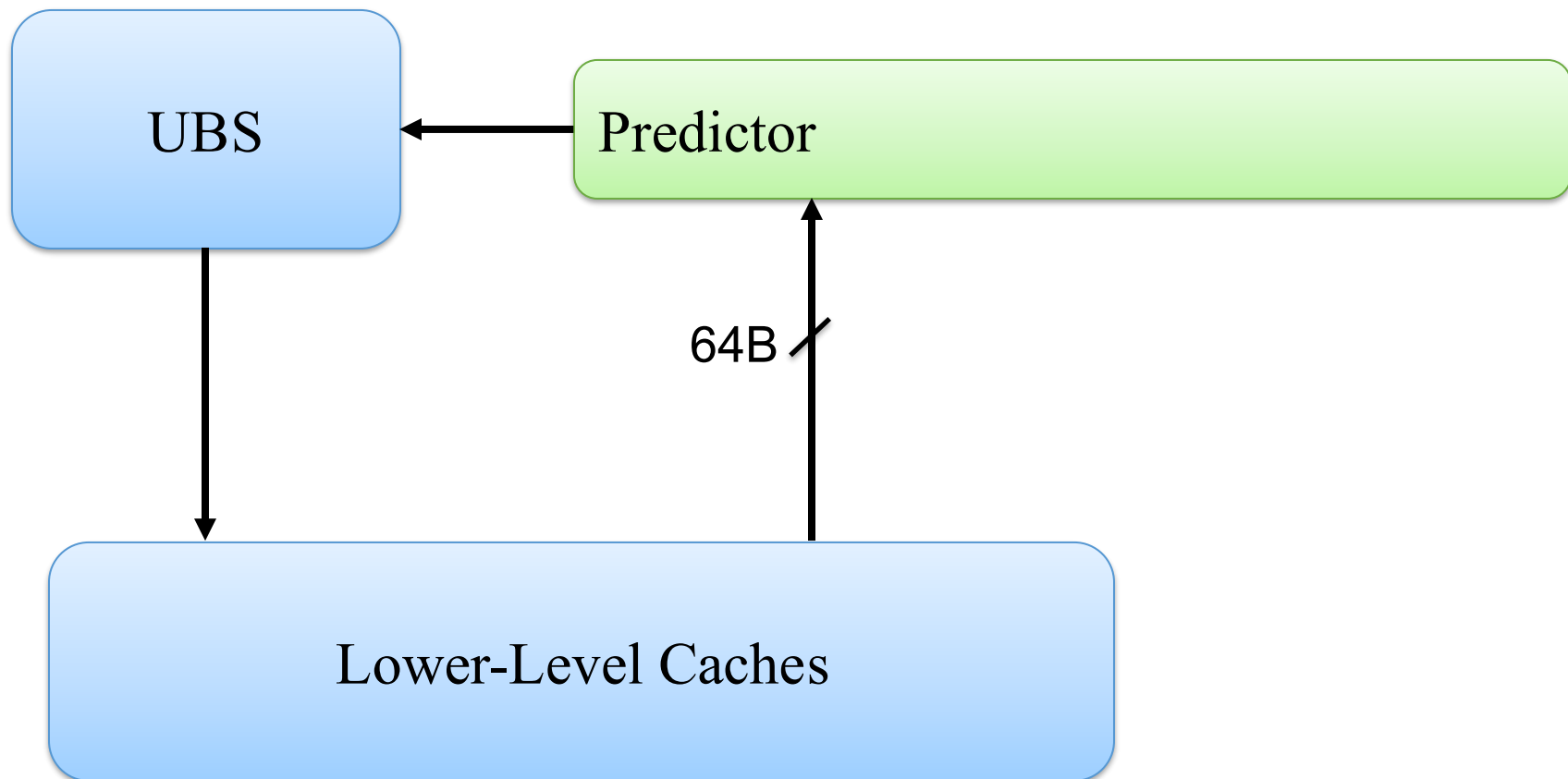
What Bytes to Insert in UBS Cache?



What Bytes to Insert in UBS Cache?

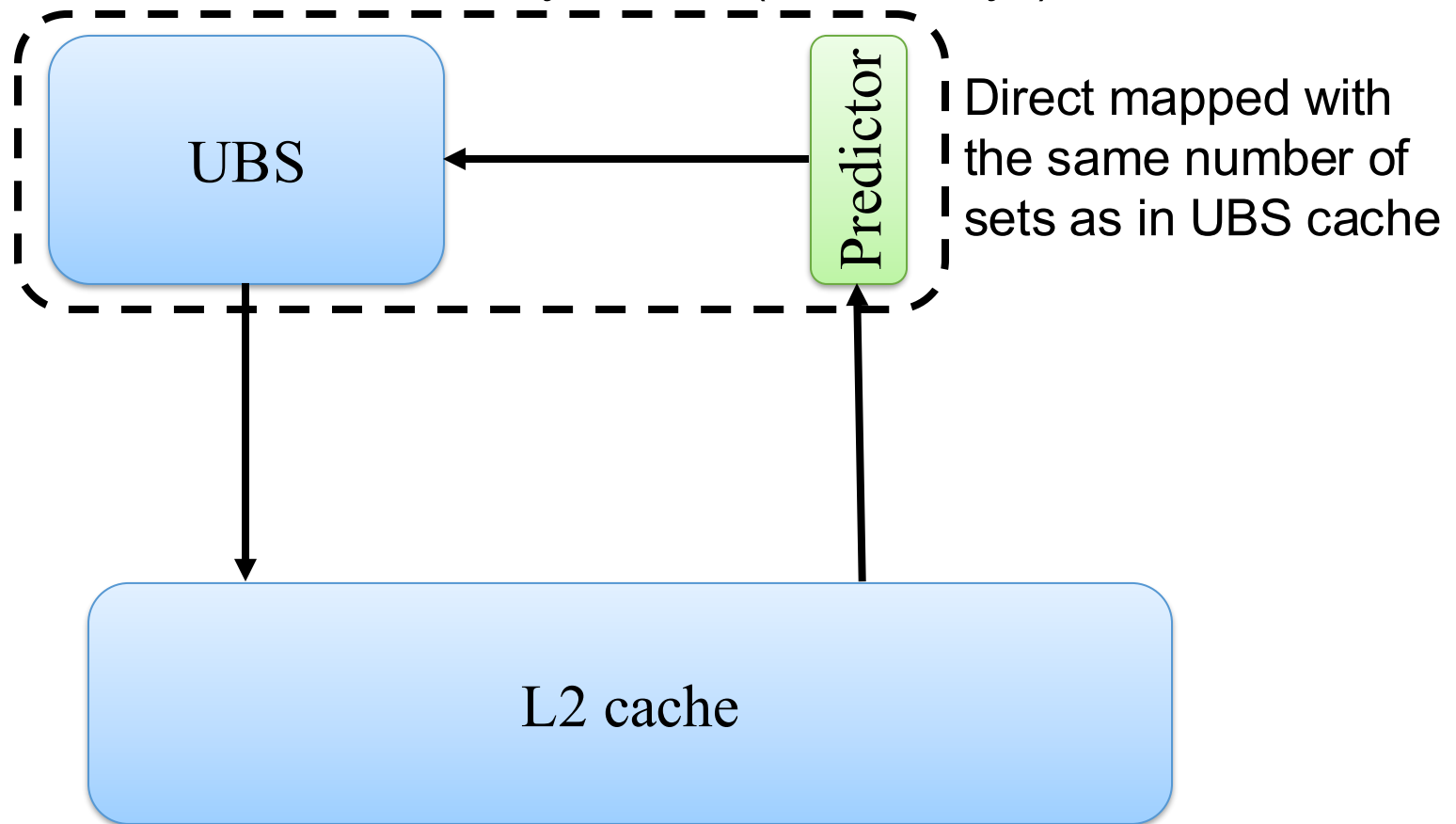


What Bytes to Insert in UBS Cache?

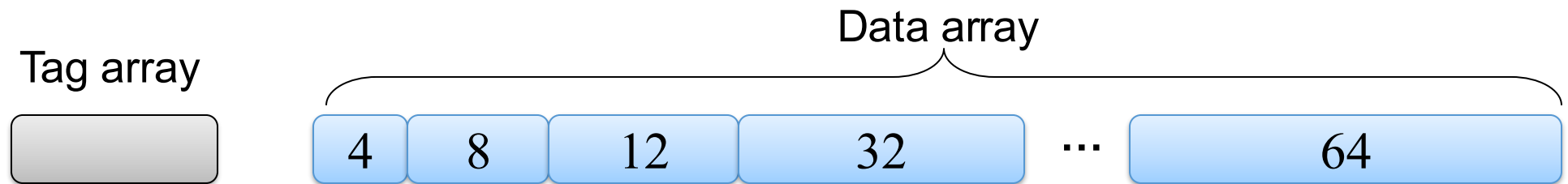


What Bytes to Insert in UBS Cache?

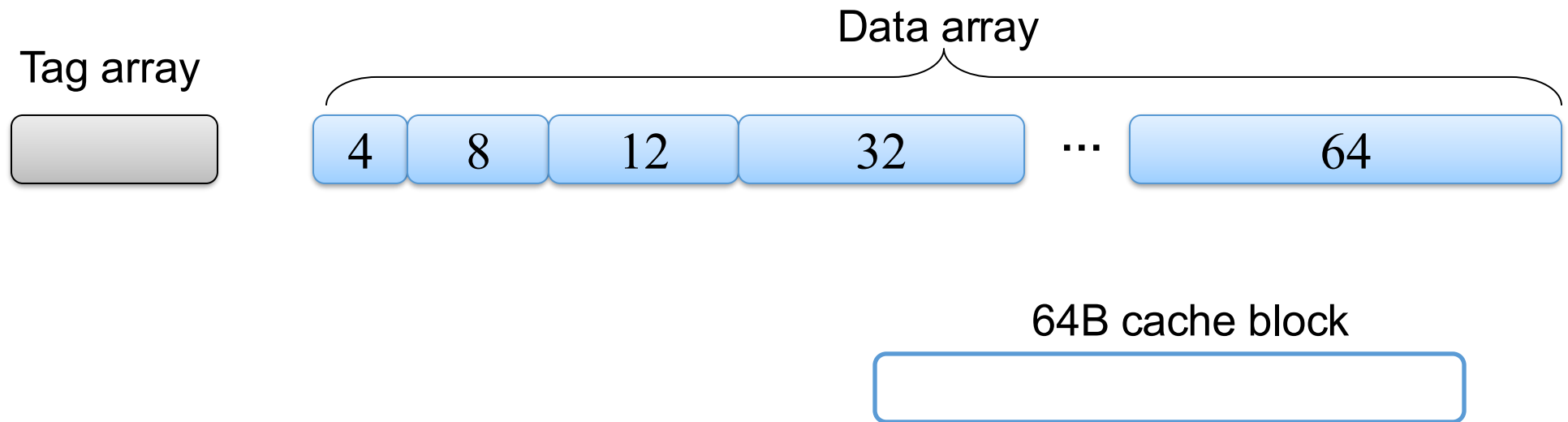
Predictor is an additional way in UBS (16 + 1 ways)



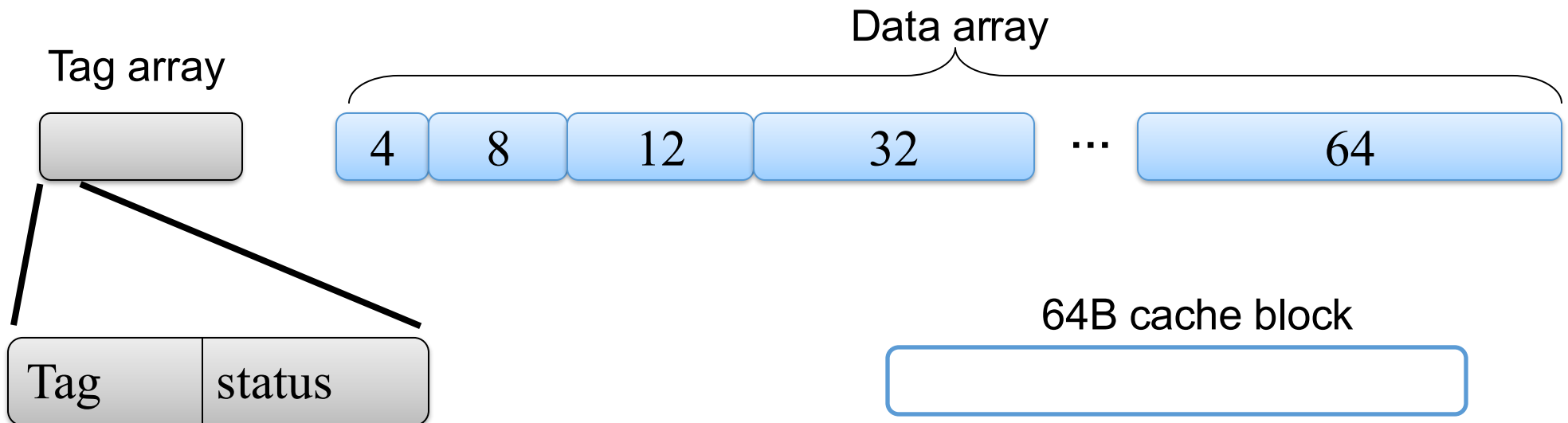
Detecting Hits in UBS Cache



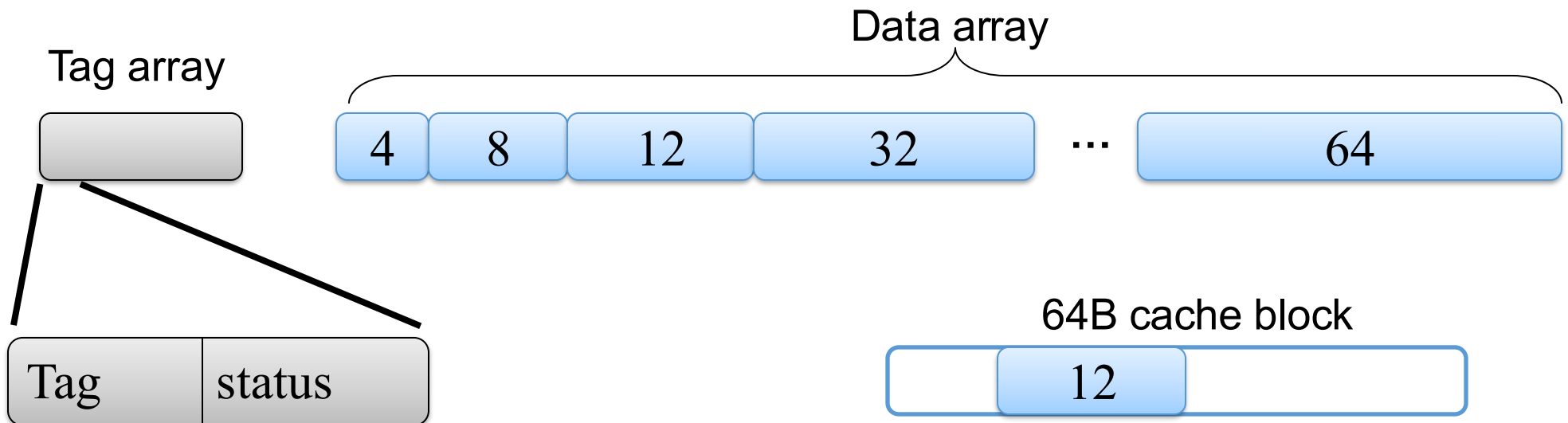
Detecting Hits in UBS Cache



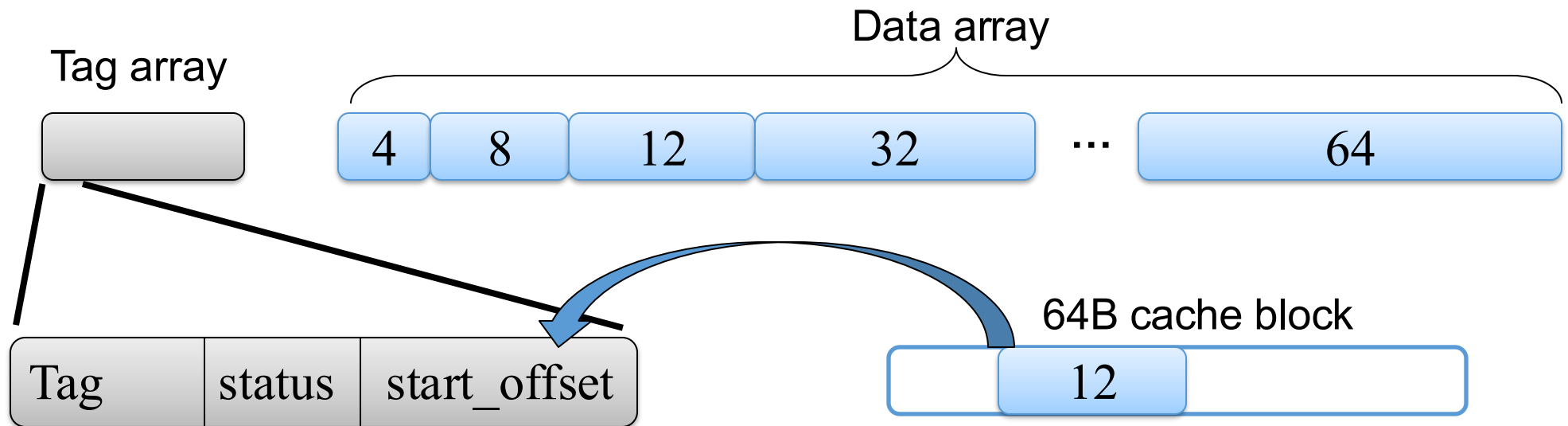
Detecting Hits in UBS Cache



Detecting Hits in UBS Cache



Detecting Hits in UBS Cache

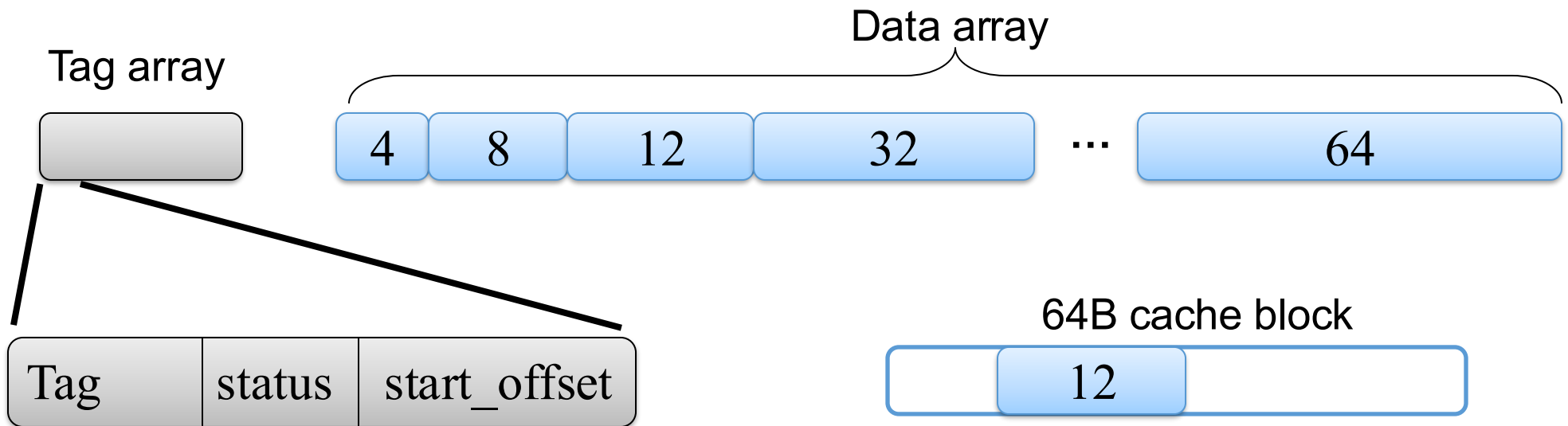


Detecting Hits in UBS Cache



No need to store the block size as
it is implied in the way number ✓

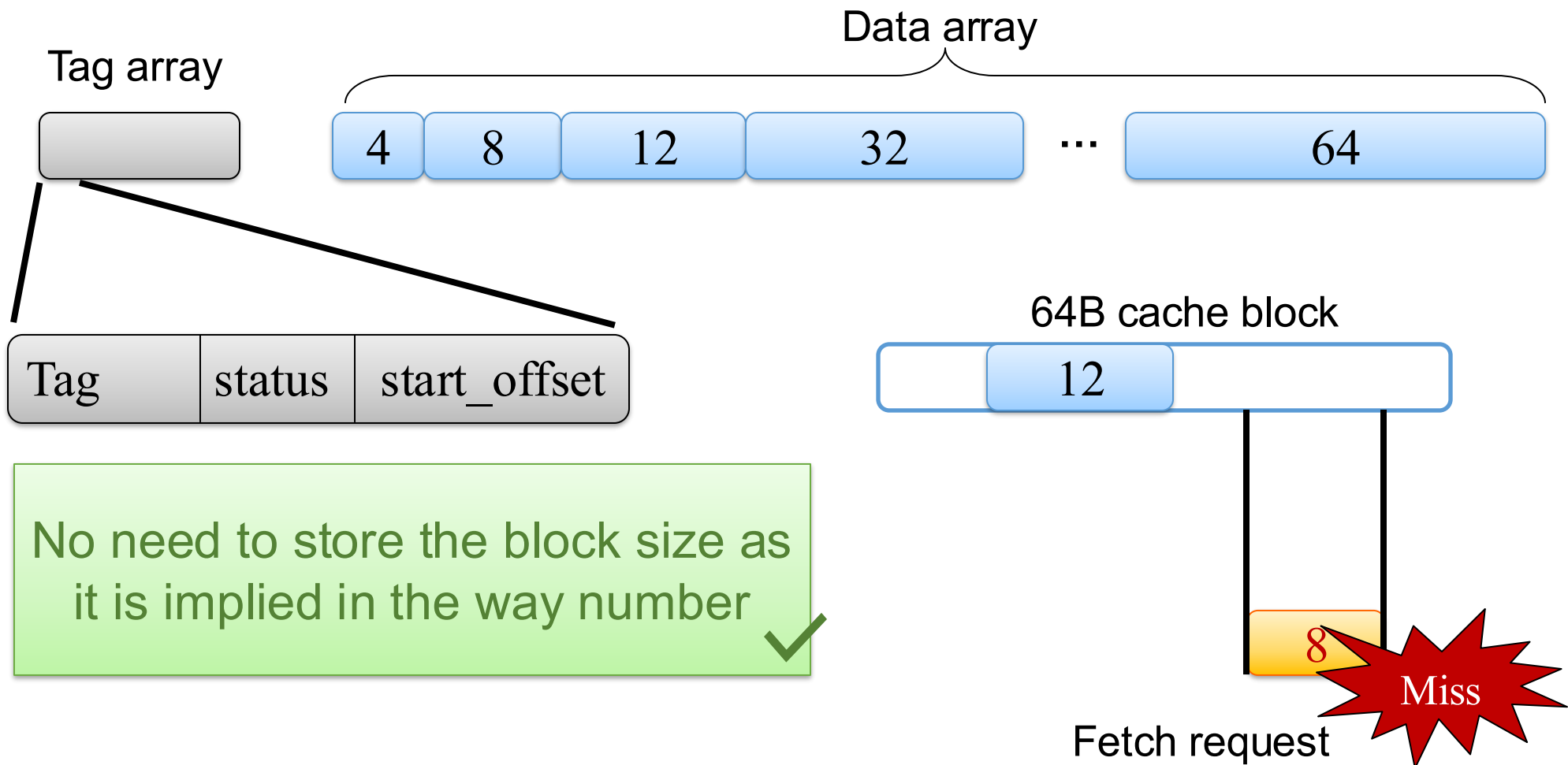
Detecting Hits in UBS Cache



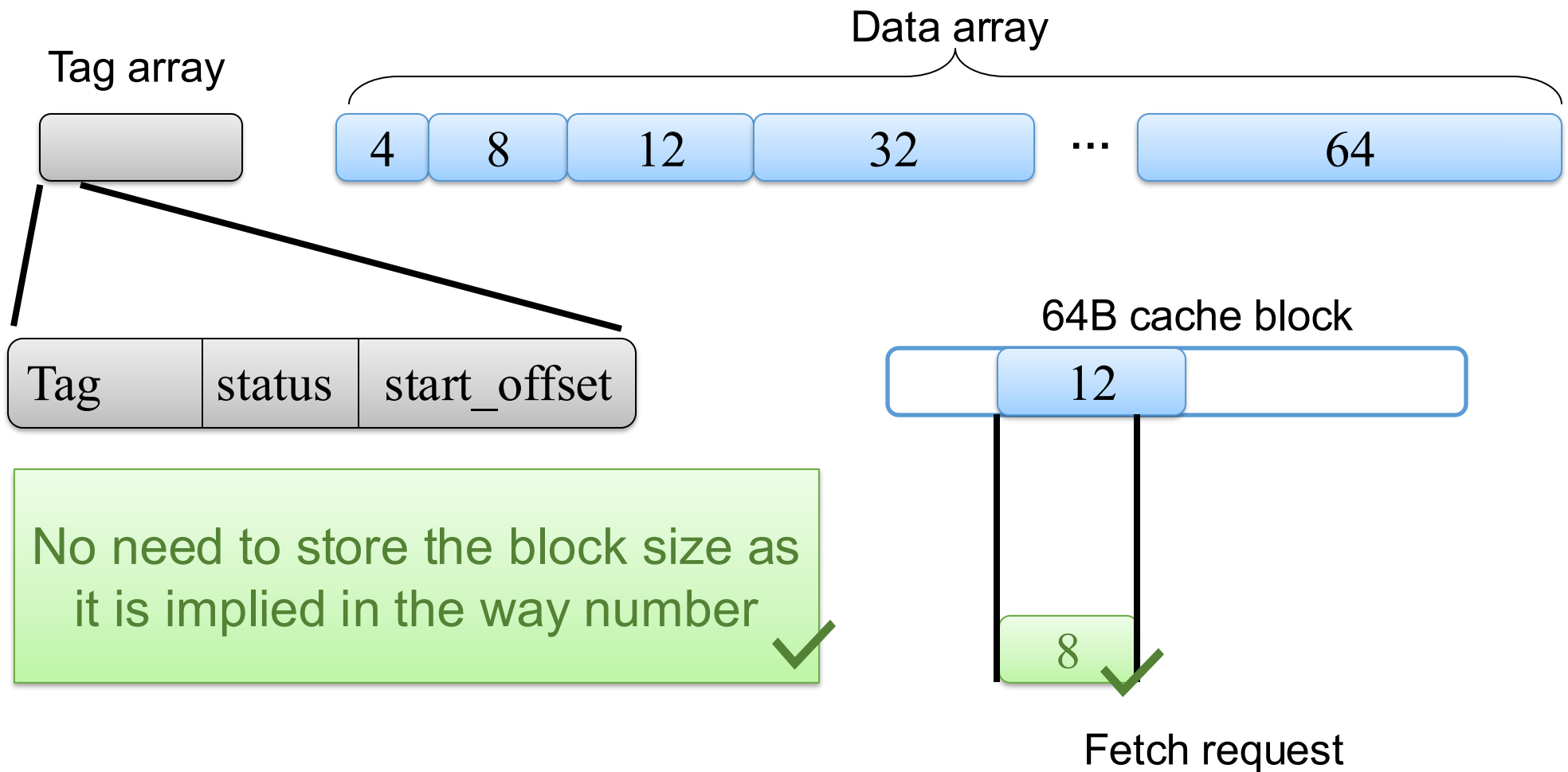
No need to store the block size as
it is implied in the way number ✓

Fetch request

Detecting Hits in UBS Cache



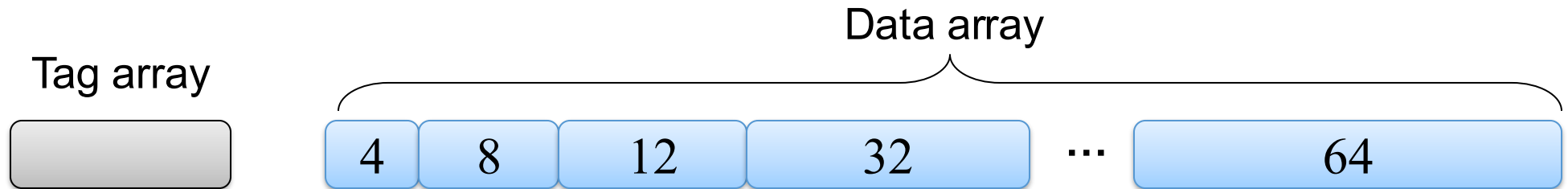
Detecting Hits in UBS Cache



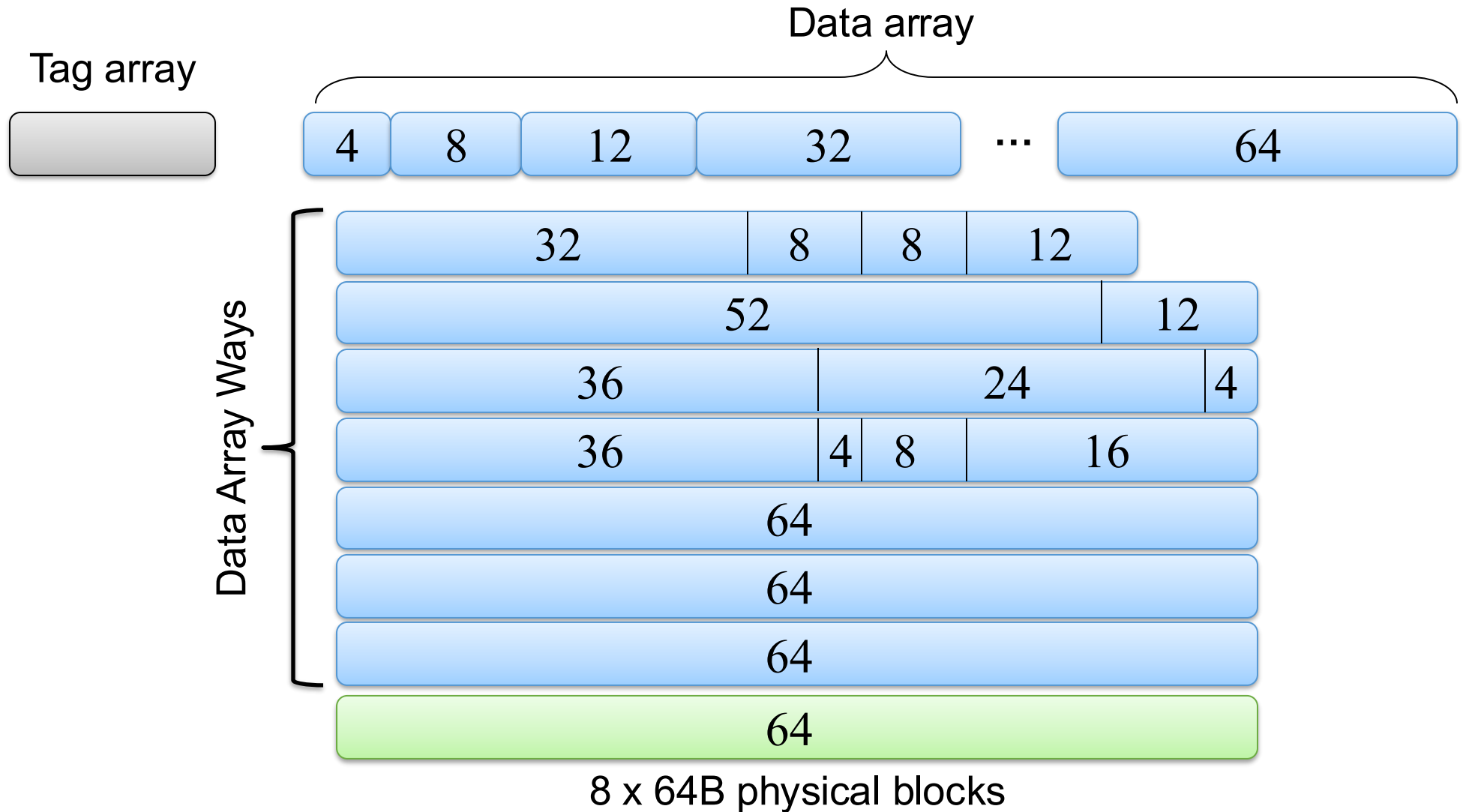
Latency

#ways	#sets	Block size		Tag-array latency	Data-array latency
8	64	64		0.09ns	0.77ns
17	64	64		0.12ns	1.71ns

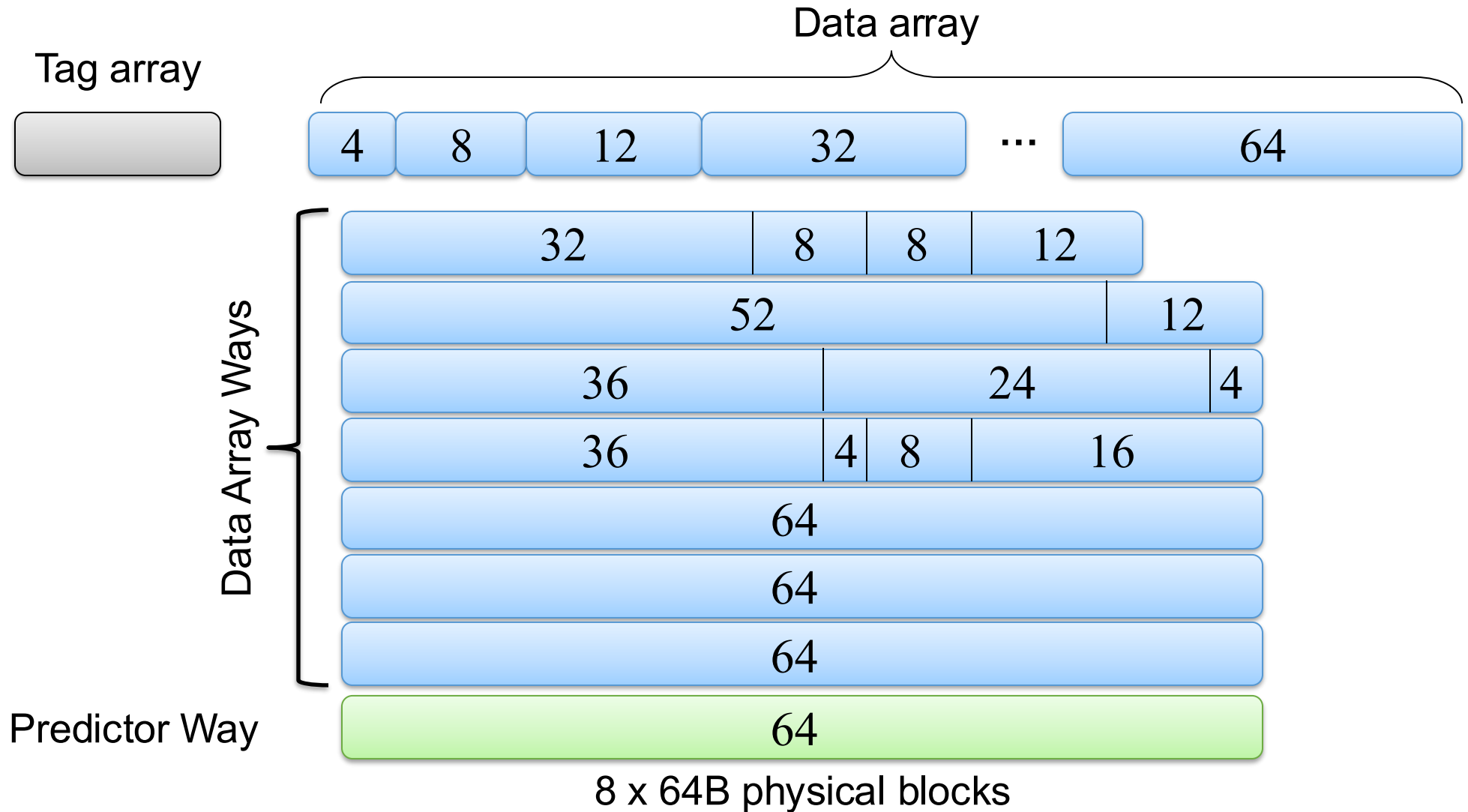
Consolidating UBS ways



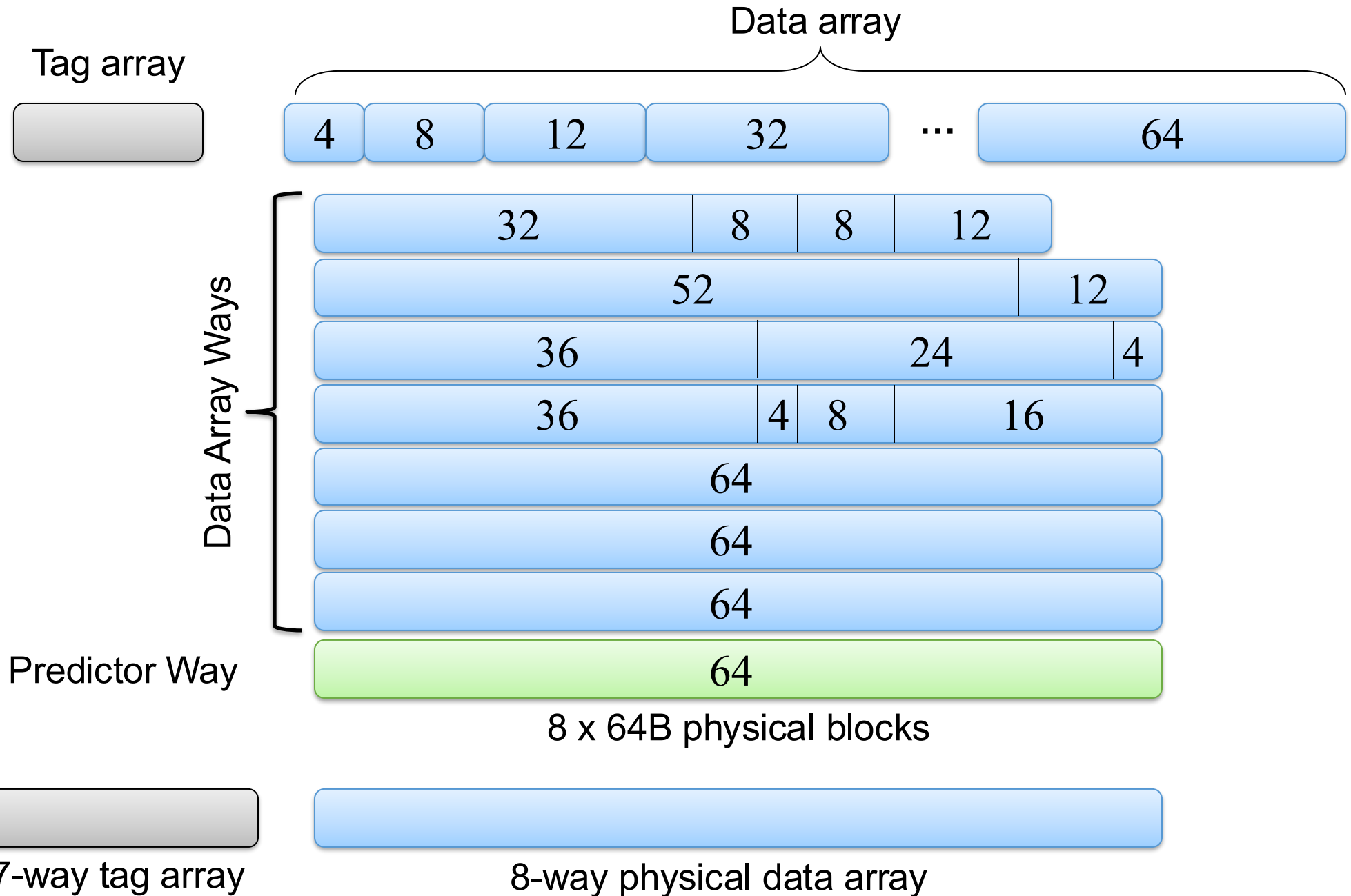
Consolidating UBS ways



Consolidating UBS ways

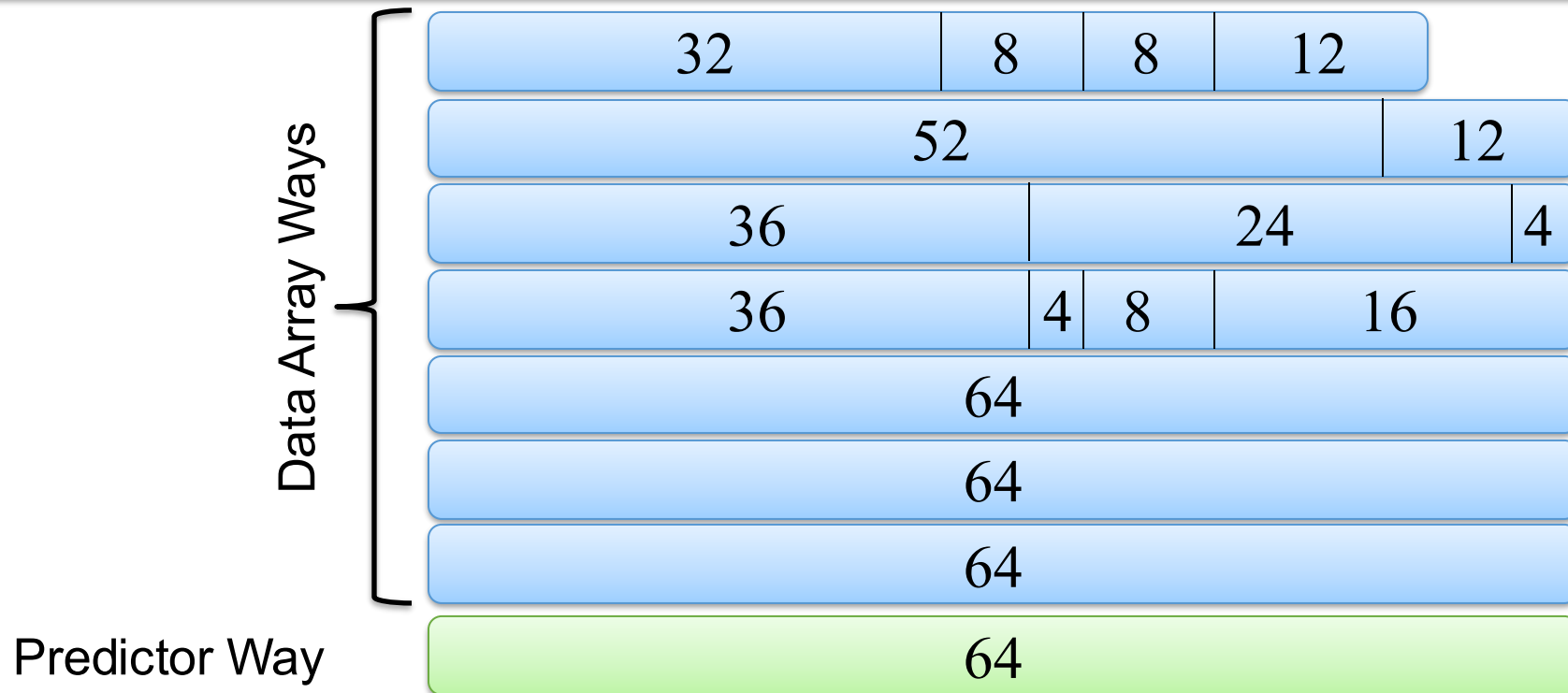


Consolidating UBS ways



Consolidating UBS ways

Data array latency stays the same, Tag array latency is hidden behind the data array latency.



8 x 64B physical blocks



17-way tag array



8-way physical data array

Storage Requirements

Conv-L1I: Conventional L1-I

	32KB Conv-L1I	32KB UBS	UBS Overhead	64KB Conv-L1I
Tag				
Start Offsets				
Predictor Bitvector				
Data				
TOTAL				

Storage Requirements

Conv-L1I: Conventional L1-I

	32KB Conv-L1I	32KB UBS	UBS Overhead	64KB Conv-L1I
Tag	1.875KB			
Start Offsets	0B			
Predictor Bitvector	0B			
Data	32KB			
TOTAL	33.875KB			

Storage Requirements

Conv-L1I: Conventional L1-I

	32KB Conv-L1I	32KB UBS	UBS Overhead	64KB Conv-L1I
Tag	1.875KB	4.086KB		
Start Offsets	0B	384B		
Predictor Bitvector	0B	128B		
Data	32KB	31.75KB		
TOTAL	33.875KB	36.34KB		

Storage Requirements

Conv-L1I: Conventional L1-I

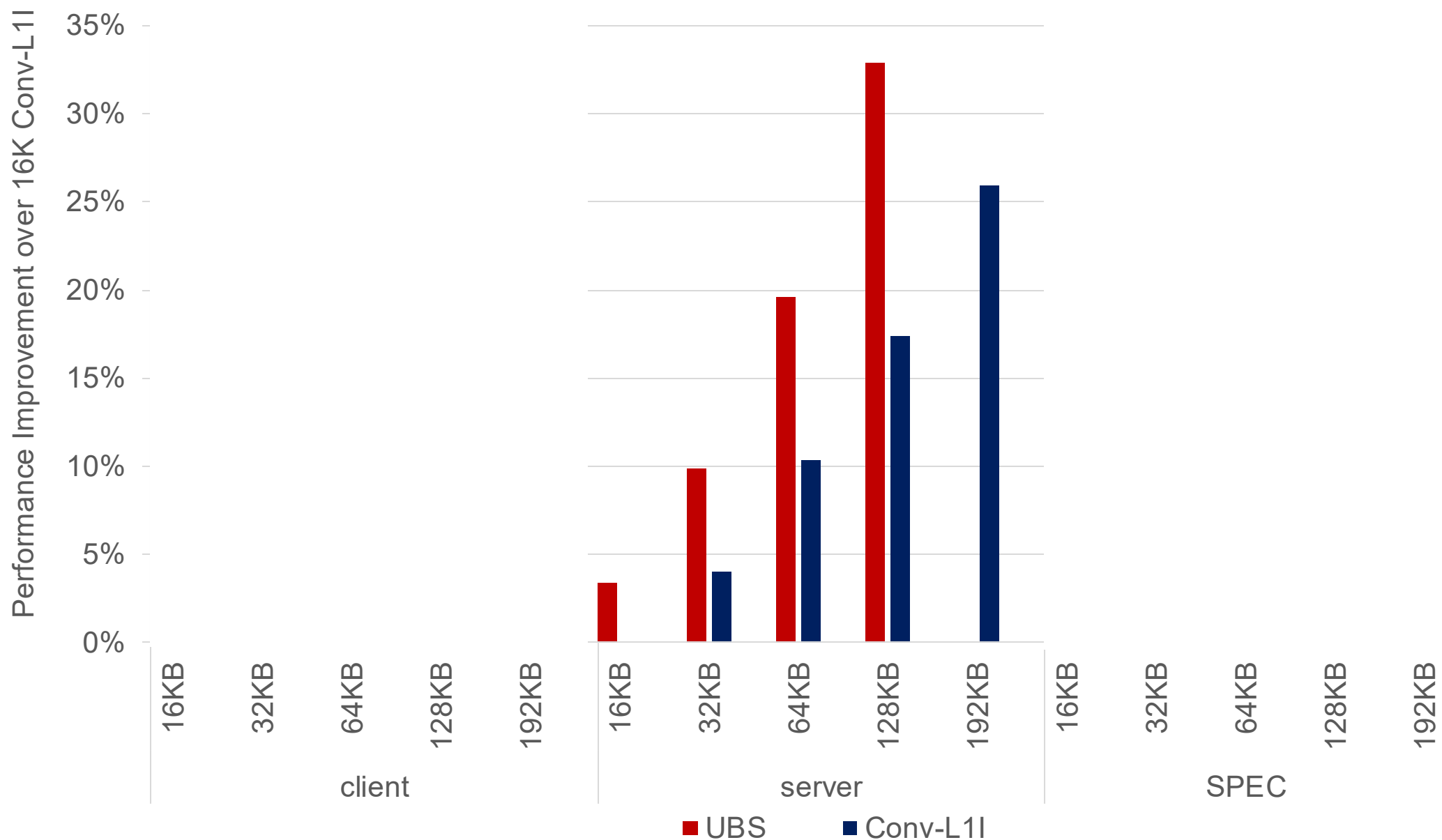
	32KB Conv-L1I	32KB UBS	UBS Overhead	64KB Conv-L1I
Tag	1.875KB	4.086KB	+2.21KB	
Start Offsets	0B	384B	+384B	
Predictor Bitvector	0B	128B	+128B	
Data	32KB	31.75KB	-0.25KB	
TOTAL	33.875KB	36.34KB	+2.46KB	

Storage Requirements

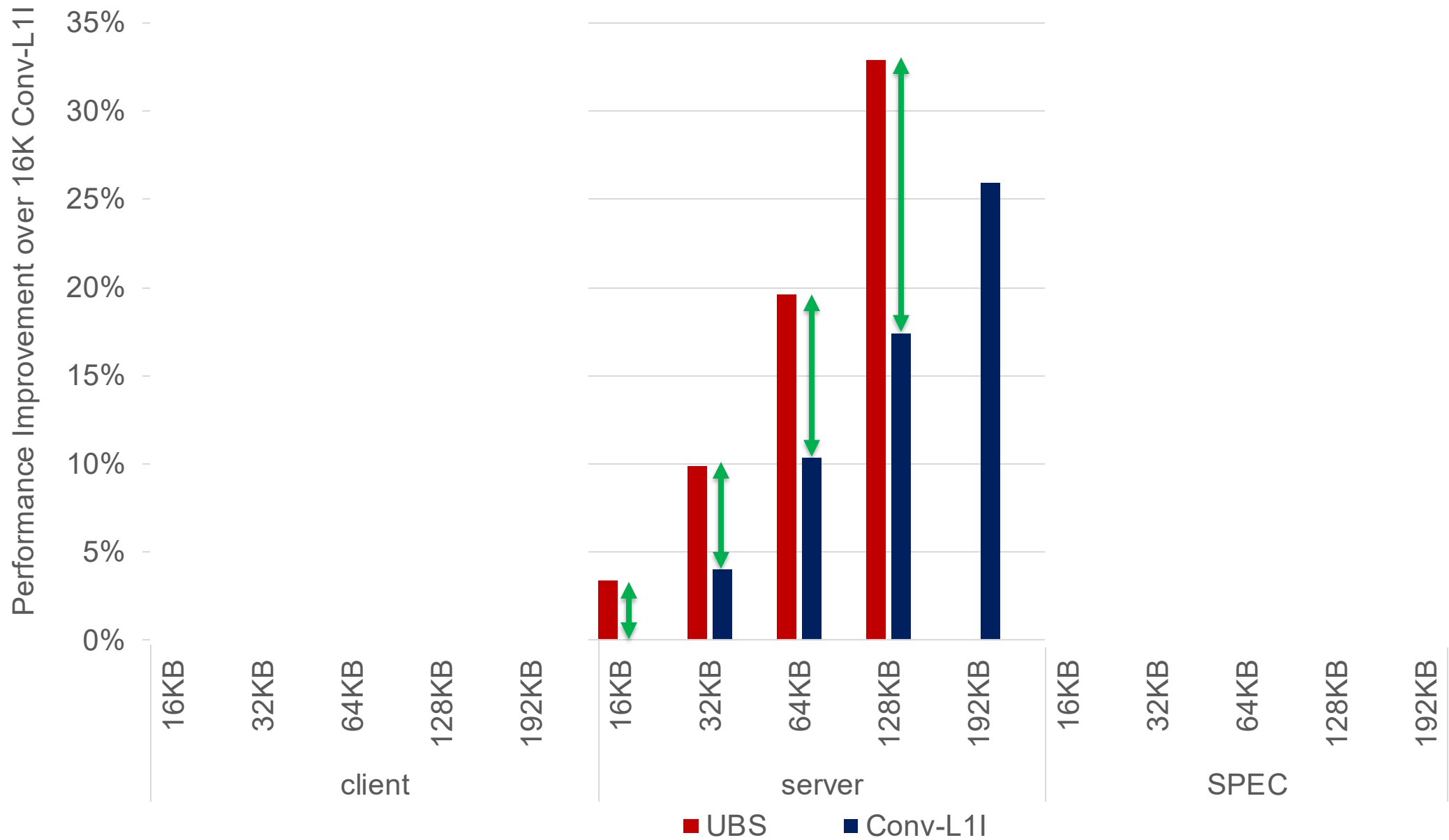
Conv-L1I: Conventional L1-I

	32KB Conv-L1I	32KB UBS	UBS Overhead	64KB Conv-L1I
Tag	1.875KB	4.086KB	+2.21KB	3.75KB
Start Offsets	0B	384B	+384B	0B
Predictor Bitvector	0B	128B	+128B	0B
Data	32KB	31.75KB	-0.25KB	64KB
TOTAL	33.875KB	36.34KB	+2.46KB	67.75KB

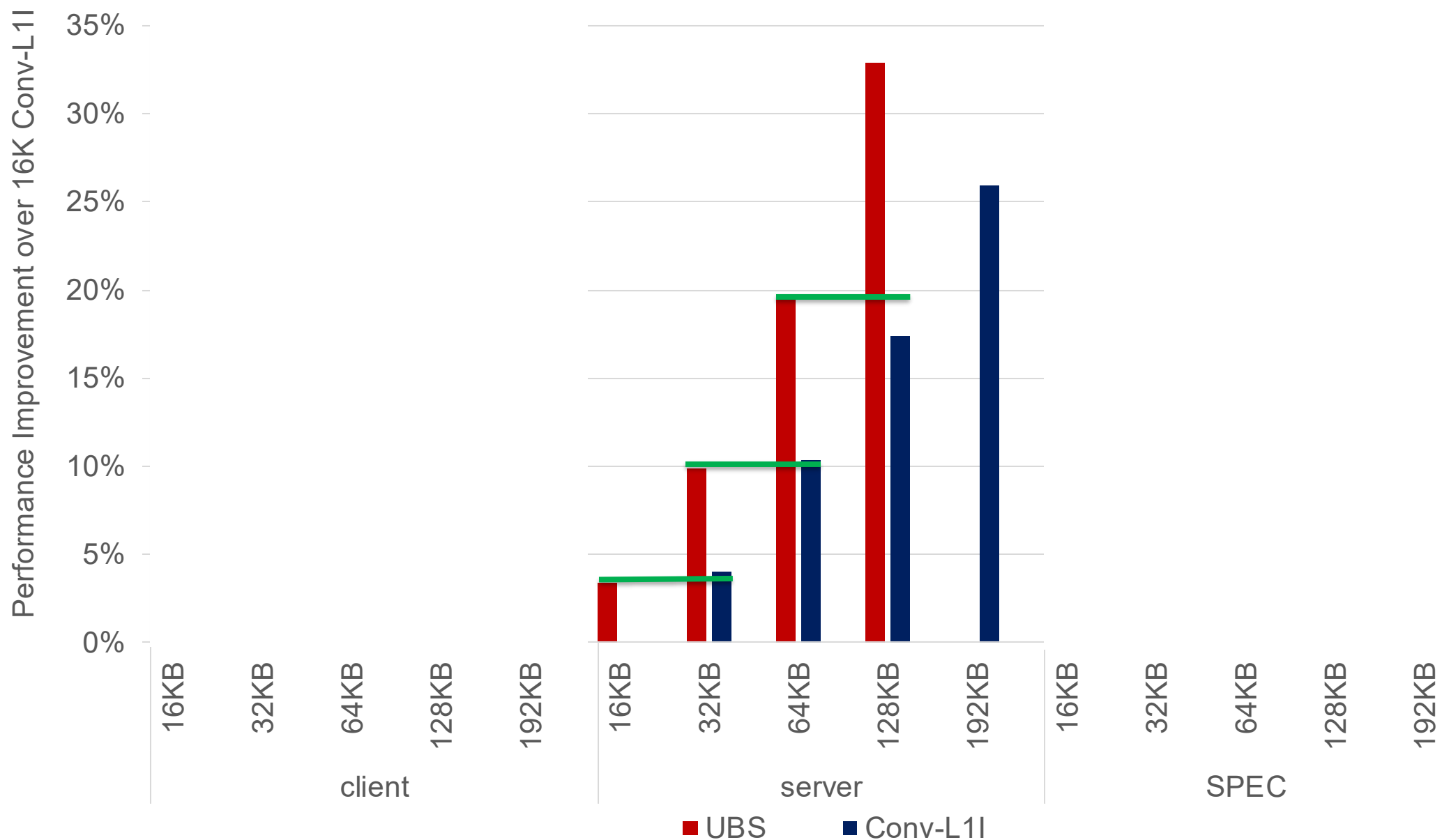
Performance



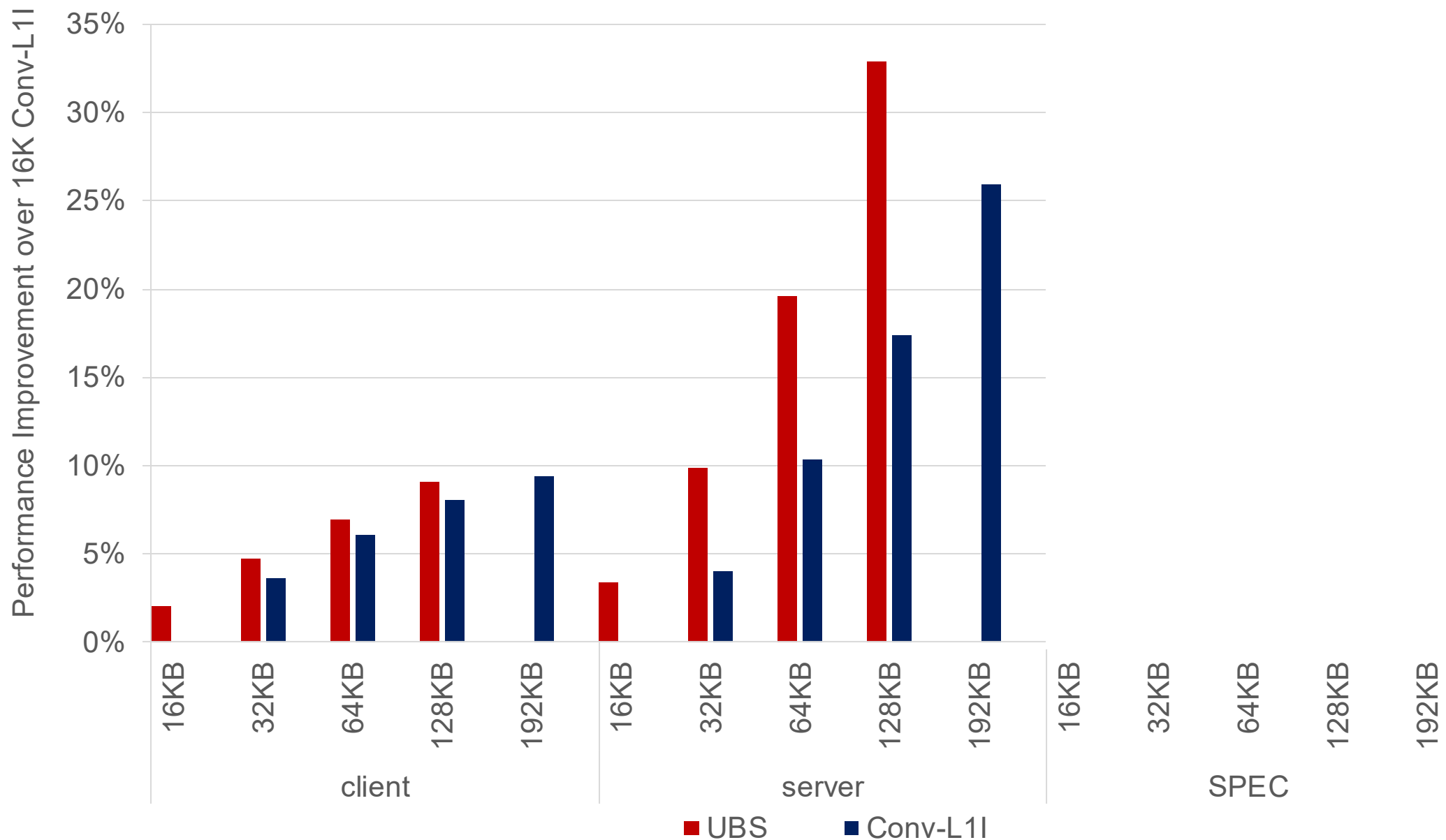
Performance



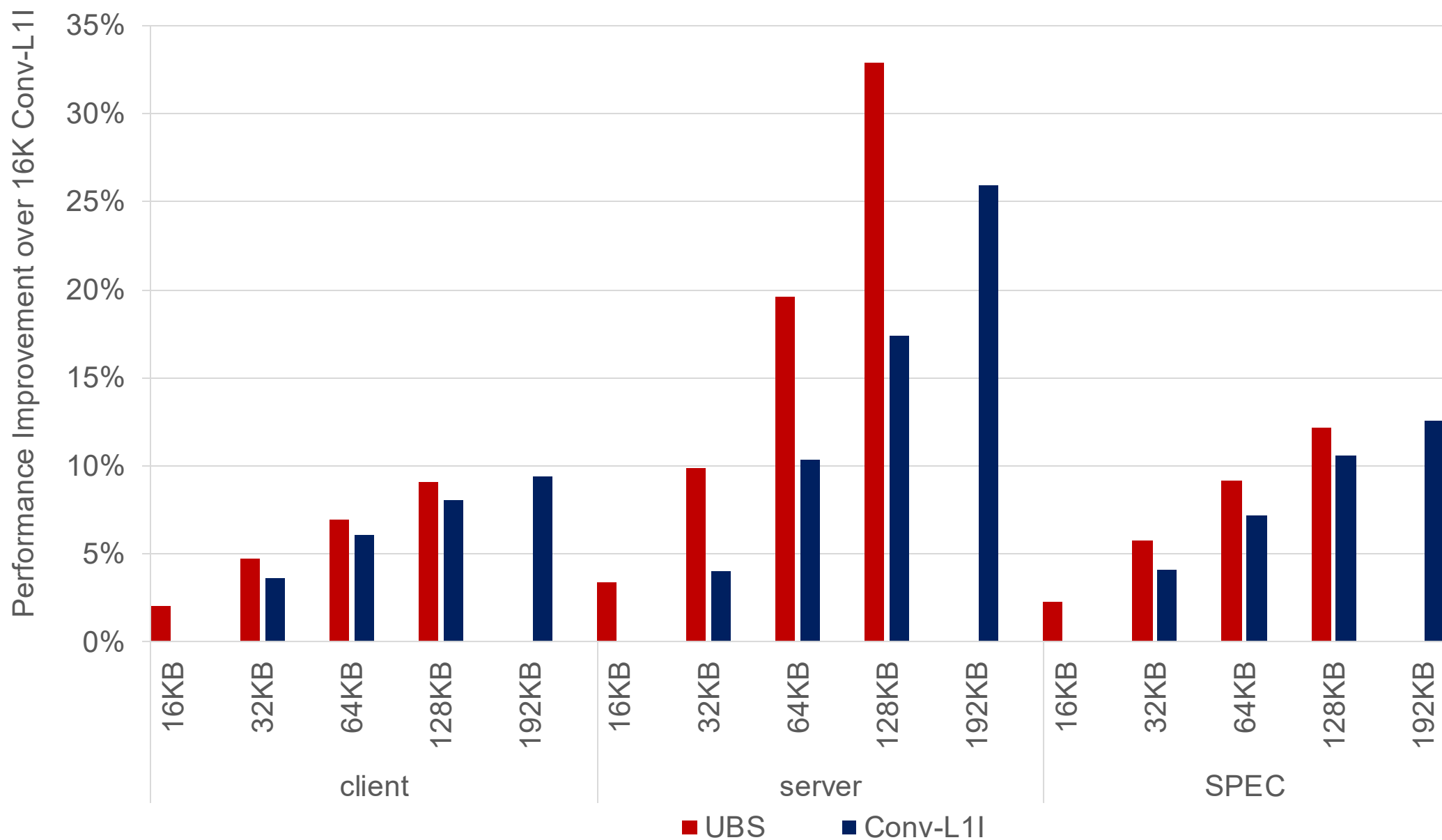
Performance



Performance



Performance



Summary

- Front-end bottleneck remains critical in servers
 - Lots of research on prefetching and replacement policies
- Fixed block sizes cause poor storage efficiency in L1-I
 - High variance in spatial locality in the instruction stream
 - 60% of the blocks see less than half of their bytes used
 - 20% of the blocks see more than 90% of their bytes used
- Uneven block sizes (UBS) maximize the storage efficiency
 - Accommodates >2x blocks by exploiting this variance
 - Approaches the performance of a conventional L1-I 2x the size

Thanks!

Roman K. Brunner
roman.k.brunner@ntnu.no
NTNU

Rakesh Kumar
rakesh.kumar@ntnu.no
NTNU



Department of Computer Science
Computer Architecture Lab

Backups

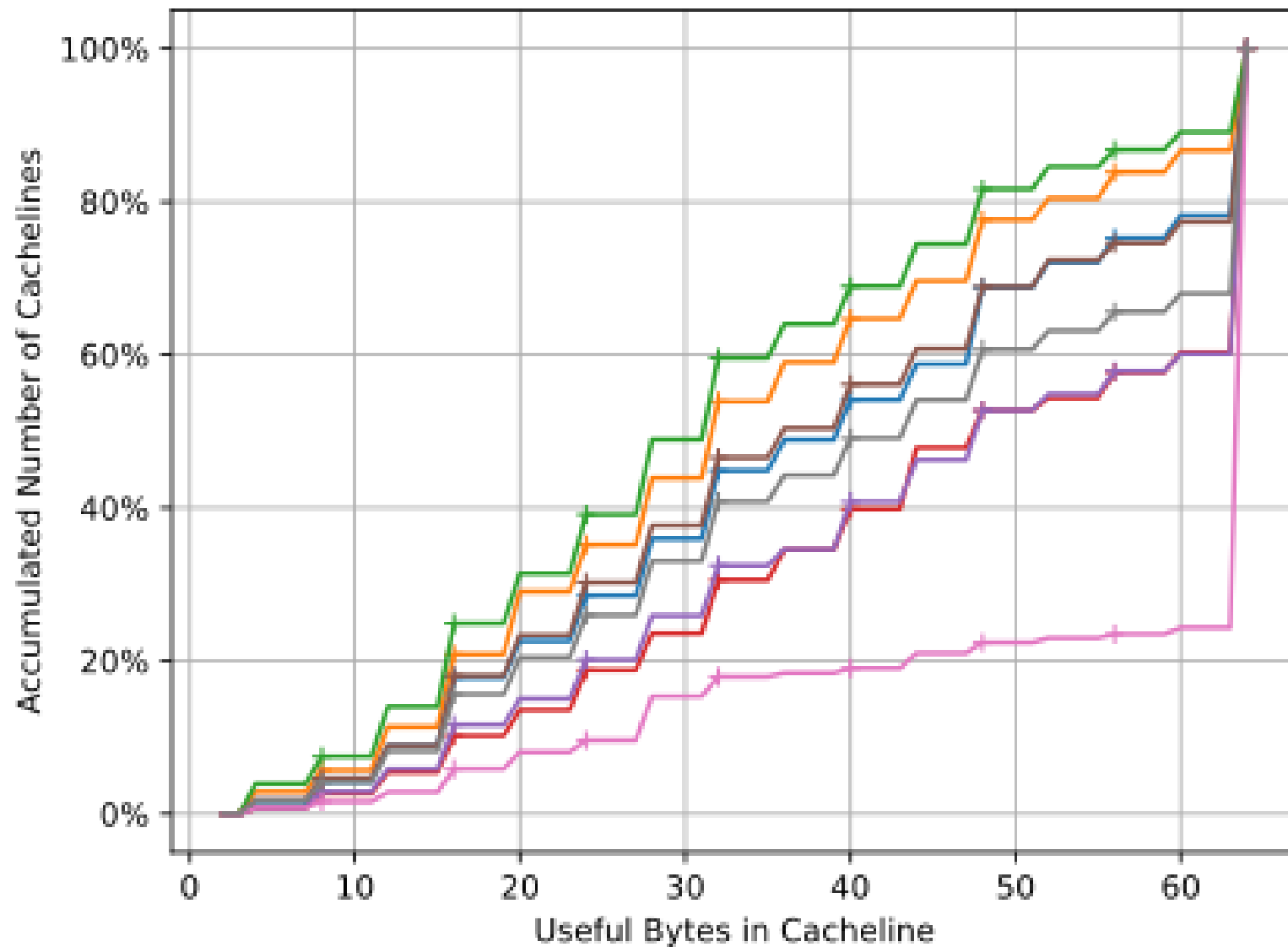
UBS Cache Parameters

Parameter	Value
Predictor	64-sets, direct-mapped
Cache	64-sets, 16-ways
Cache Way Sizes	4, 4, 8, 8, 8, 12, 12, 16, 24, 32, 36, 36, 52, 64, 64, 64
Replacement Policy	Modified LRU
Fetch latency	4 cycles
MSHR	8 entries

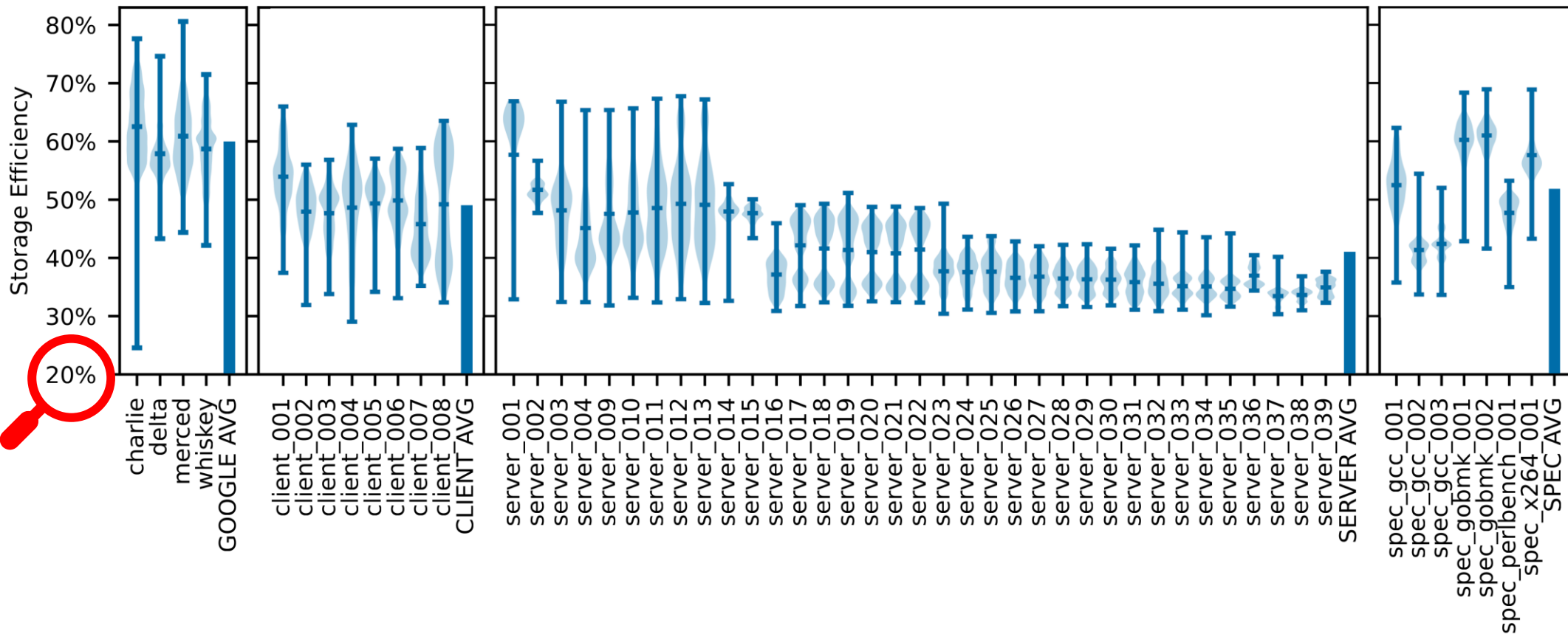
Core Configuration

Parameter	Value
Core	4 wide fetch, decode and commit, 224 entry ROB, 97 entry scheduler, 128 entry load queue, 72 entry store queue
Branch Prediction Unit	4k entry BTB, Hashed Perceptron
Instruction Prefetcher	FDIP, 128 entry Fetch Target Queue
L1-I	32KB, 8 ways, 4 cycle latency, LRU, 8 MSHR
L1-D	48KB, 12 ways, 5 cycle latency, LRU, 16 MSHR
L2	512KB, 8 ways, 12 cycle latency, LRU, 32 MSHR
L3	2MB, 16 ways, 30 cycles latency, LRU, 64 MSHR
DRAM	3200MHz, 1 channel, 1 rank, 8 bankcs, 12.5 (tRP), 12.5 (tRCD), 12.5 (tCAS)

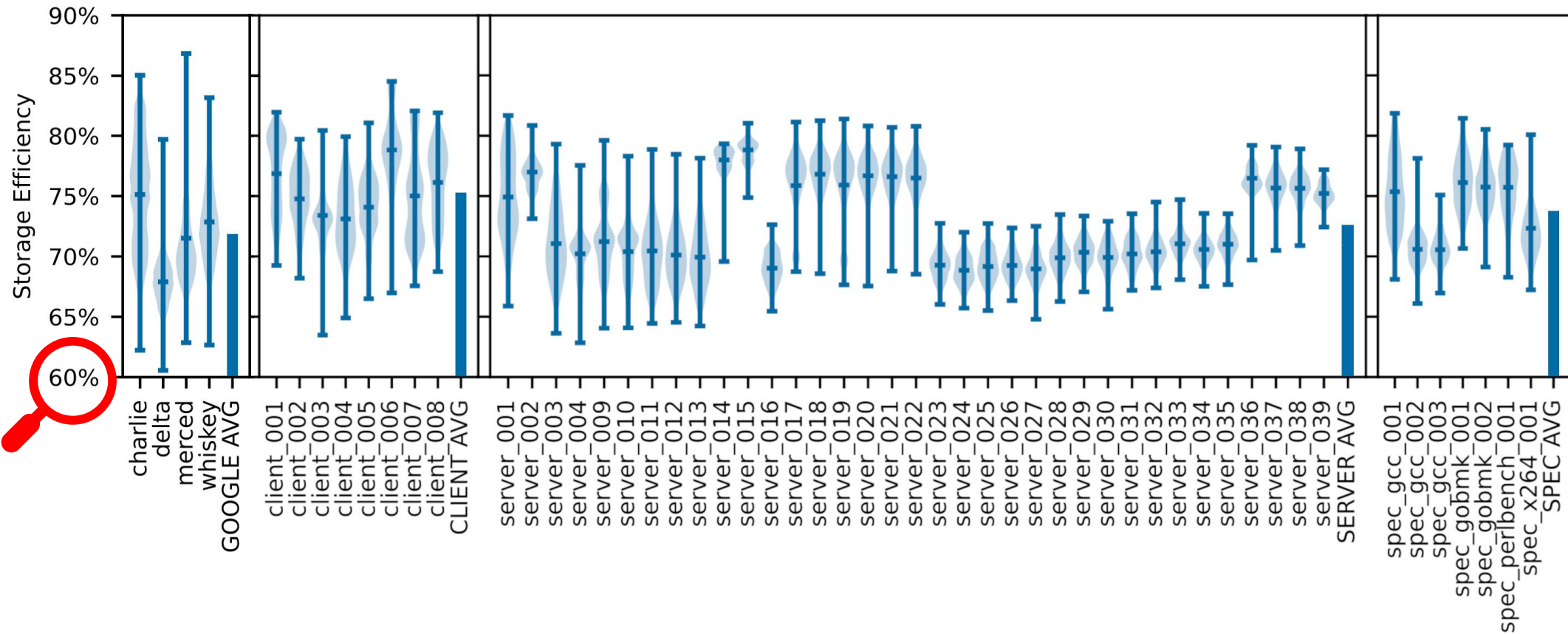
Useful Bytes in L1-I Blocks (Client/Spec)



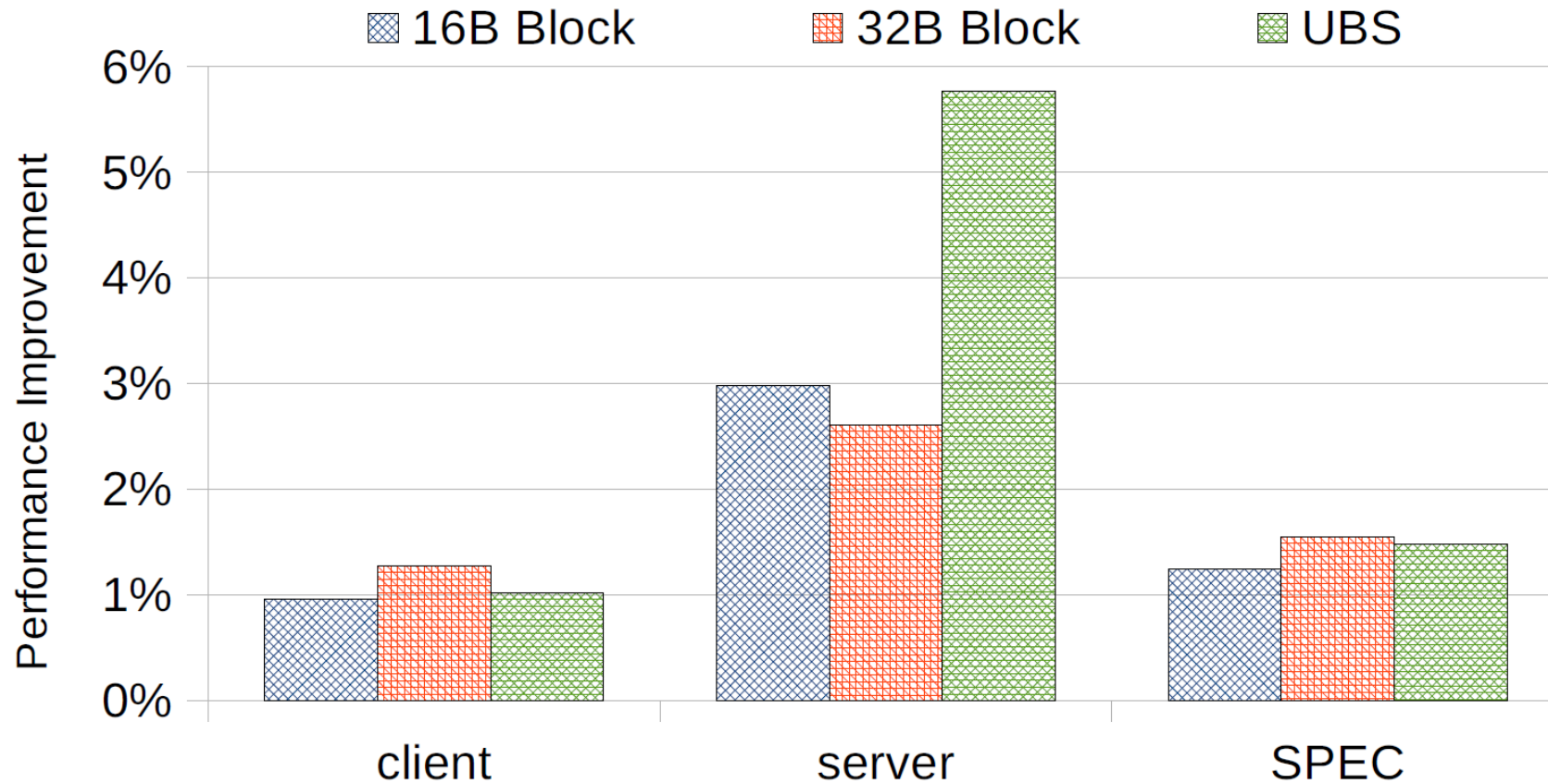
Storage Efficiency: 32k L1-I Baseline



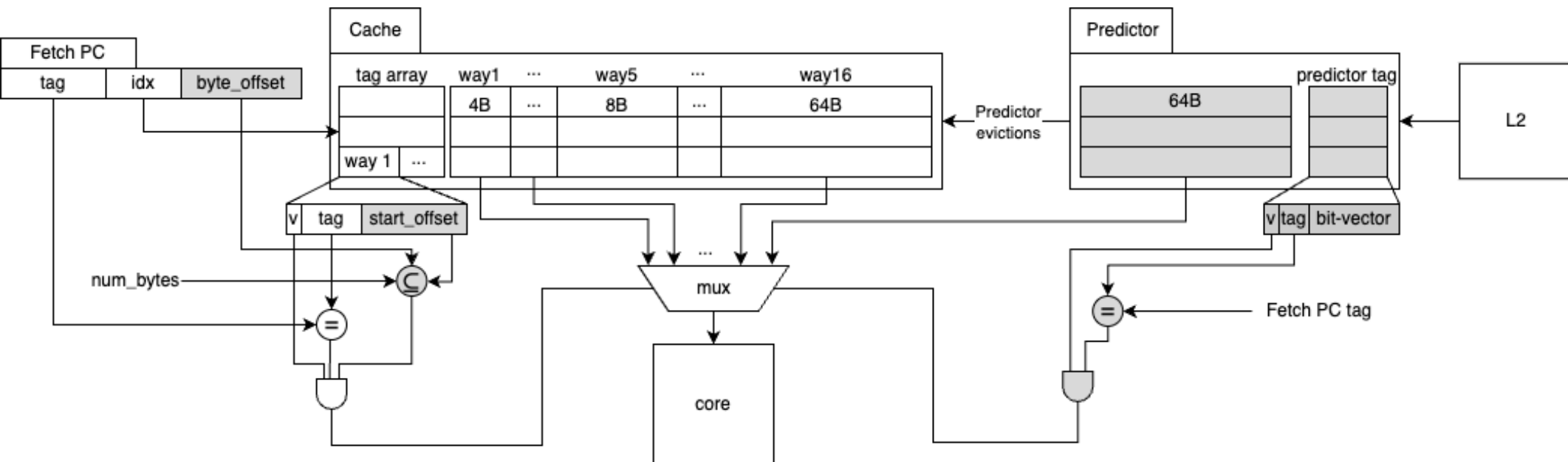
Storage Efficiency: 32k L1-I UBS



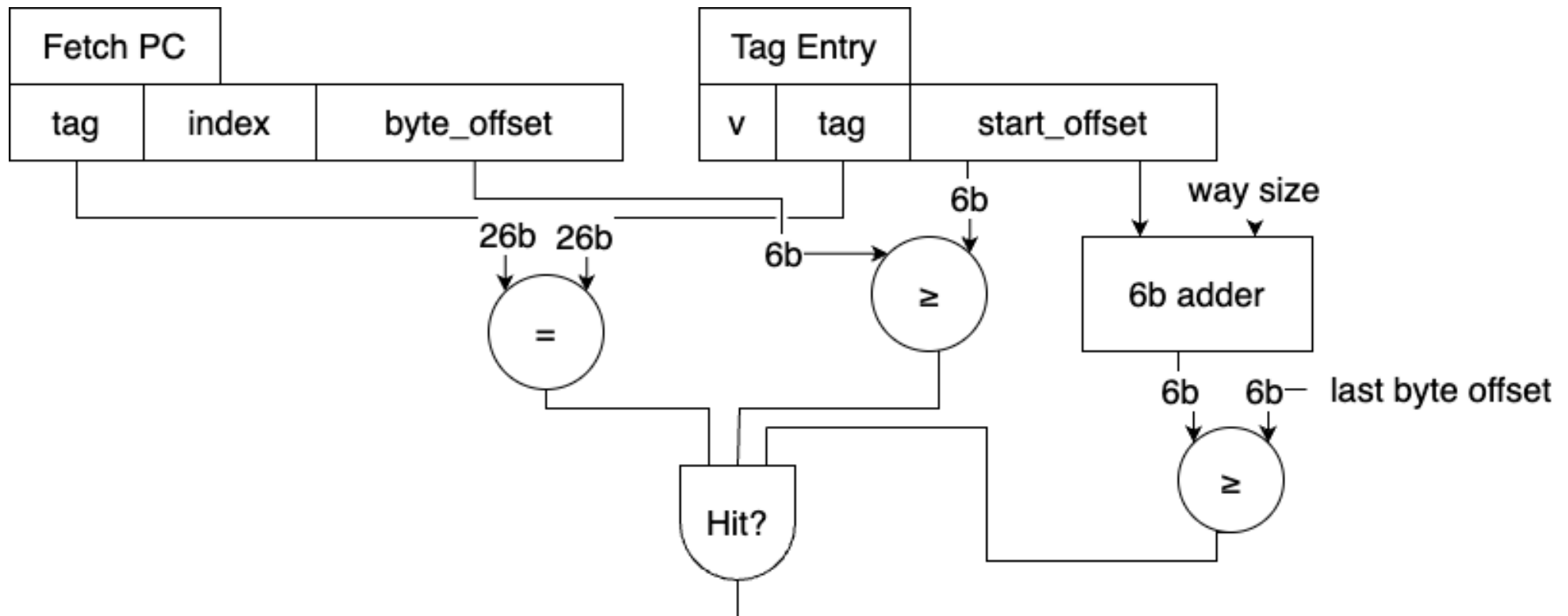
UBS vs Small Cache Blocks



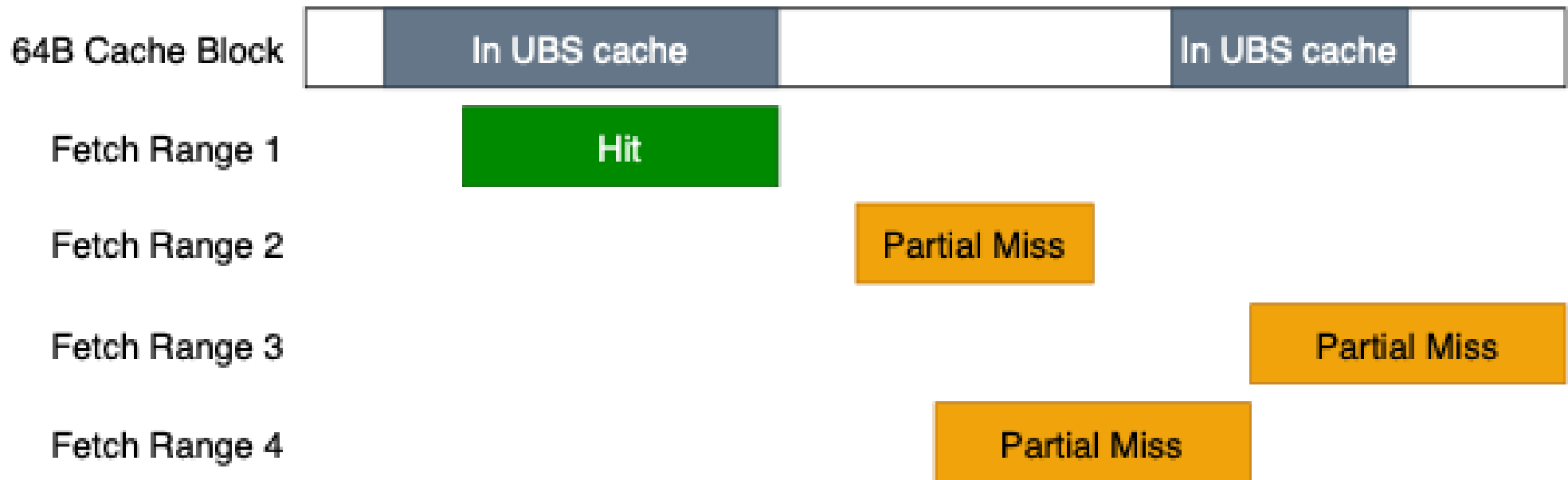
UBS: Full Architecture Overview



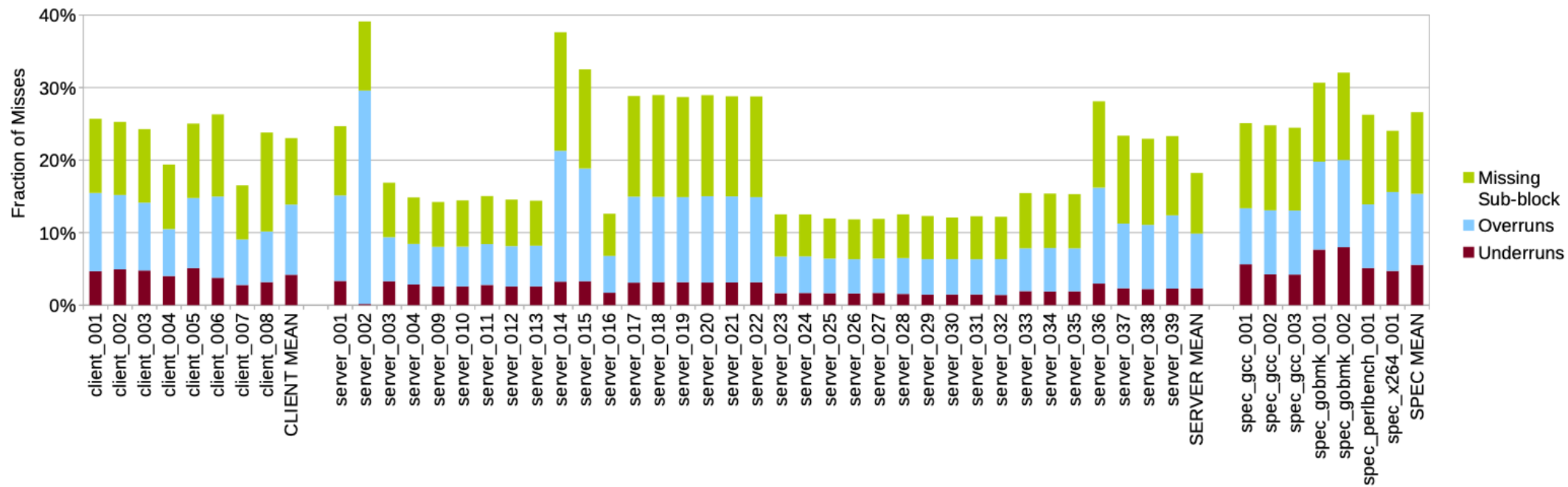
Detecting Hits in UBS Cache



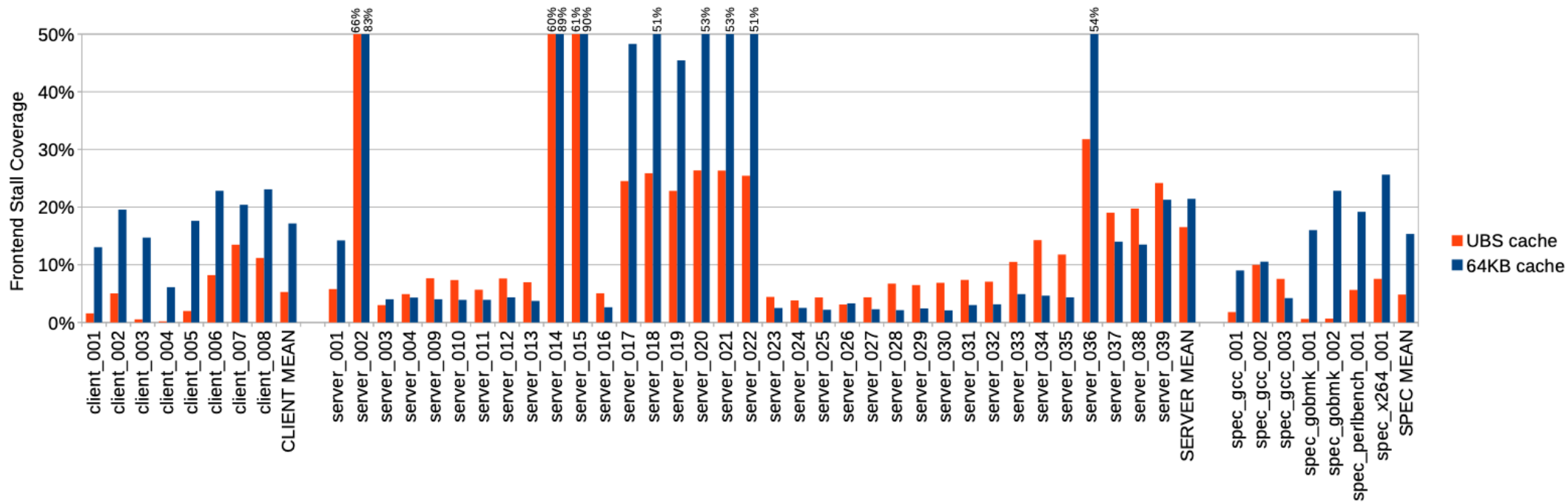
Partial Misses



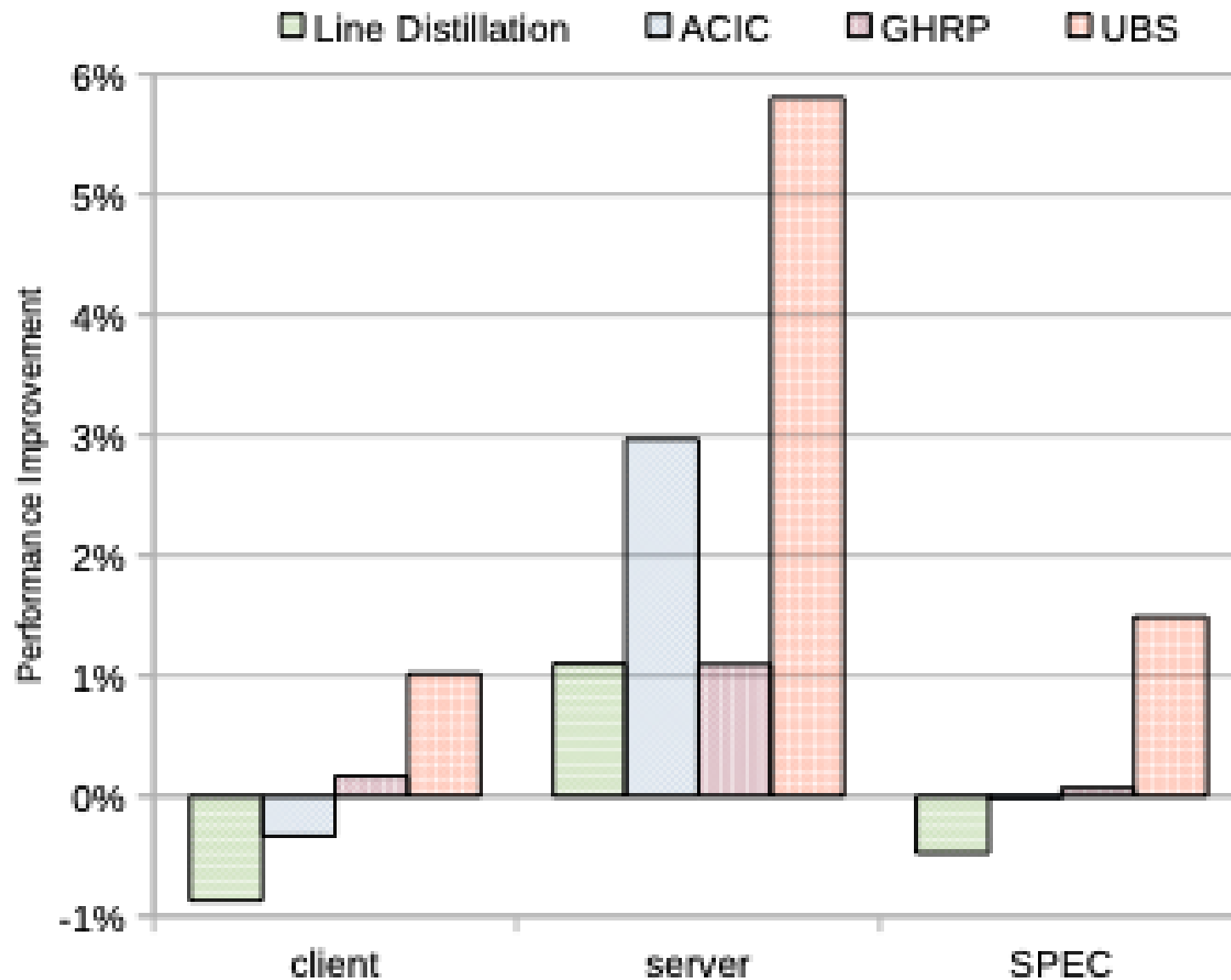
Partial Misses



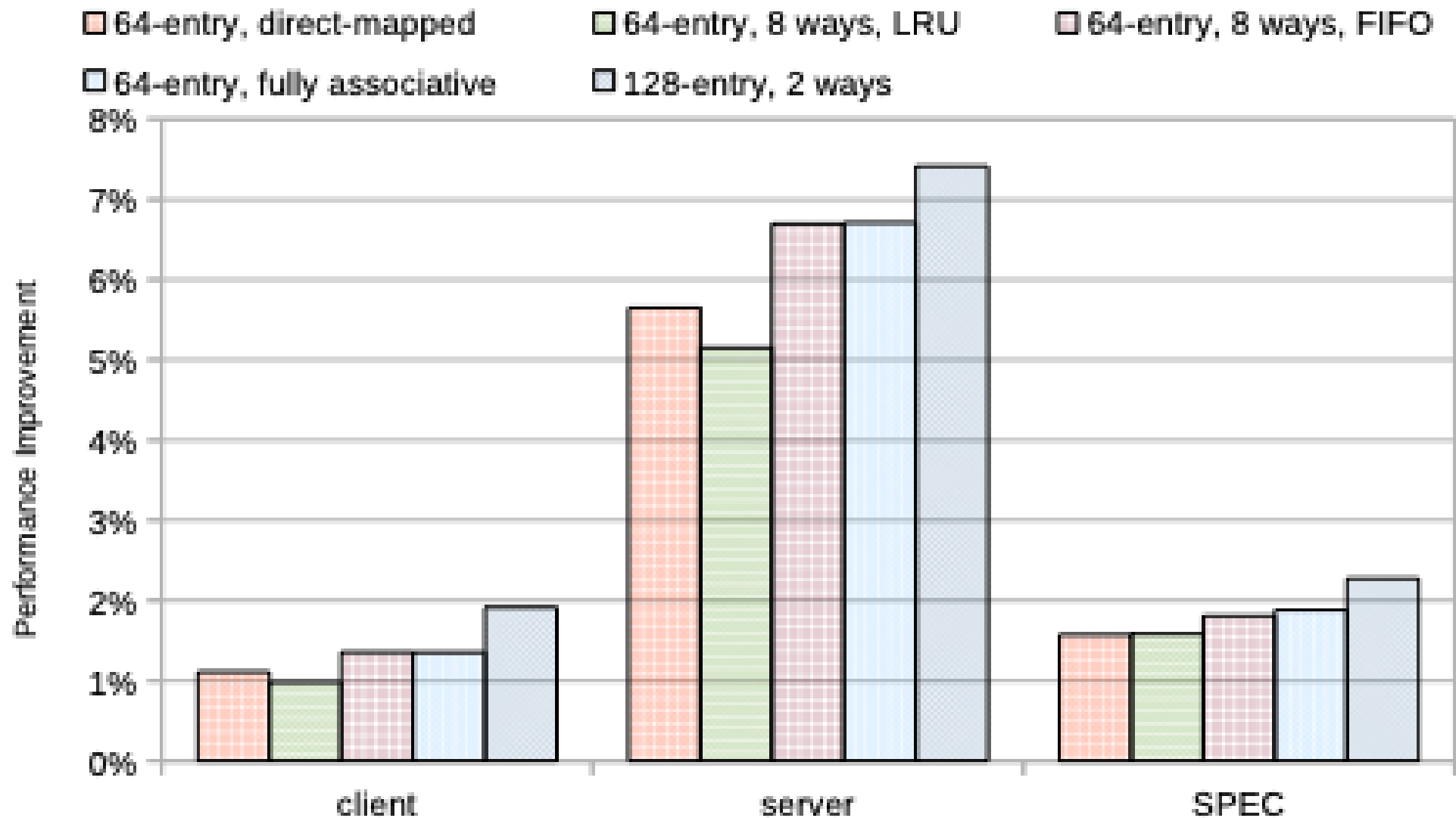
Frontend Stalls Covered



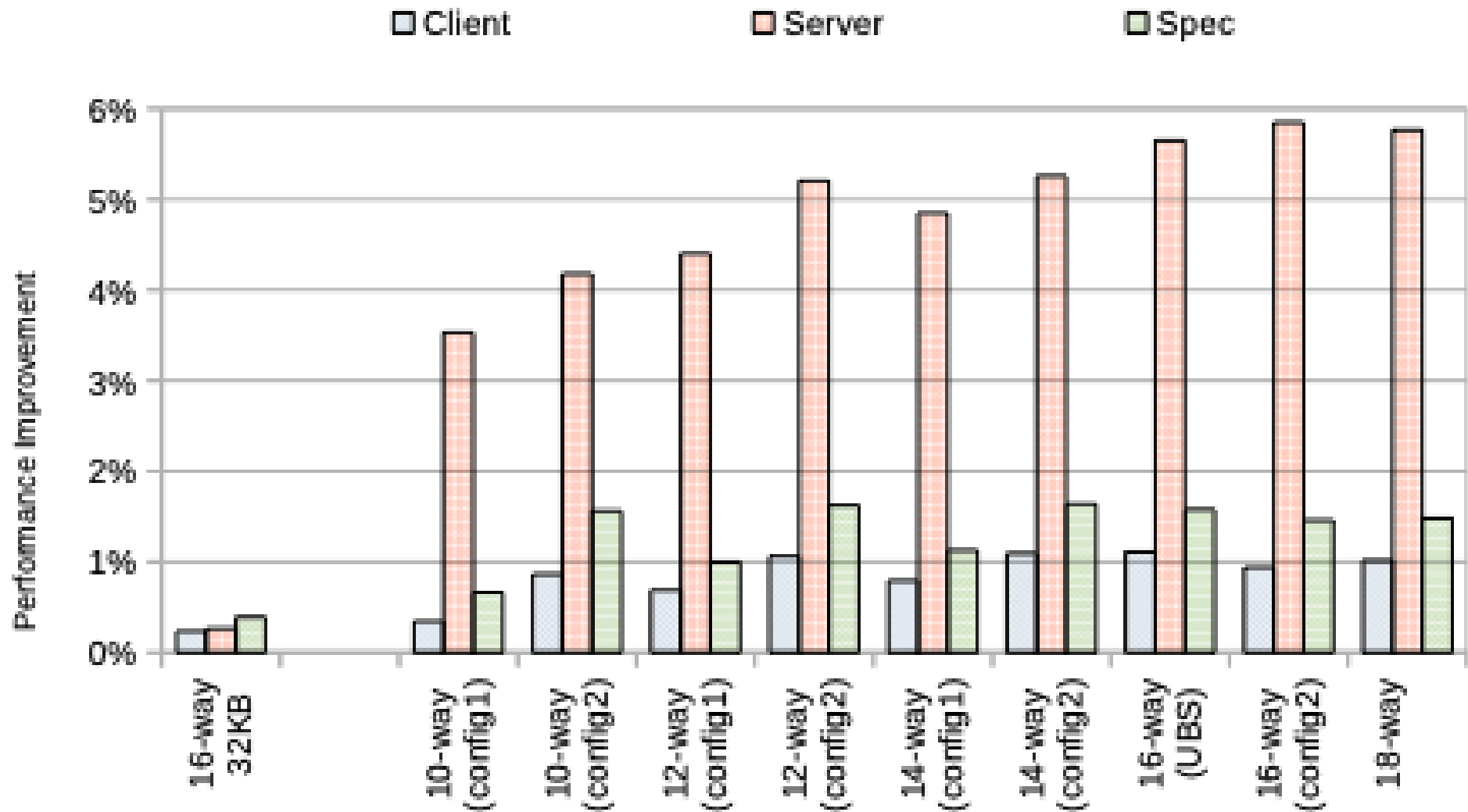
Prior Work Comparison



Predictor Sensitivity



Way Configuration Sensitivity



Fraction of all accessed bytes touched until next “n” misses

