Base-Delta-Immediate Compression: Practical Data Compression for On-Chip Caches

Gennady Pekhimenko
Vivek Seshadri
Onur Mutlu , Todd C. Mowry
Phillip B. Gibbons*
Michael A. Kozuch*

Carnegie Mellon
intel *

Executive Summary

- Off-chip memory latency is high
  - Large caches can help, **but** at significant cost
- Compressing data in cache enables larger cache at low cost
- **Problem**: Decompression is on the execution critical path
- **Goal**: Design a new compression scheme that has

1. low decompression latency, 2. low cost, 3. high compression ratio

- **Observation:** Many cache lines have low dynamic range data
- **Key Idea**: Encode cachelines as a base **+** multiple differences
- **Solution**: Base-Delta-Immediate compression with low decompression latency and high compression ratio
  - Outperforms three state-of-the-art compression mechanisms

Motivation for Cache Compression

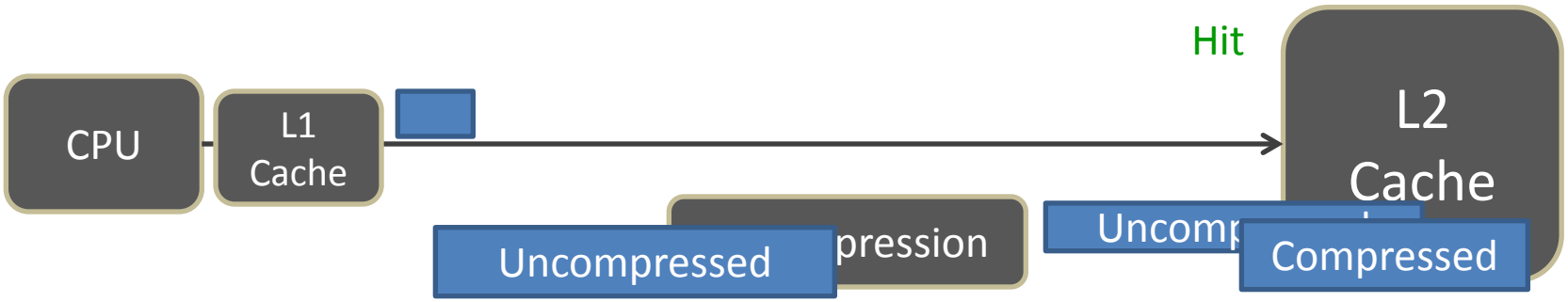
Significant redundancy in data:

0x**00000000**	0x**0000000**B	0x**00000000**3	0x**00000000**4	…

How can we exploit this redundancy?

- **Cache compression** helps
- Provides effect of a larger cache without making it physically larger

Background on Cache Compression



- Key requirements:
  - **Fast** (low decompression latency)
  - **Simple** (avoid complex hardware changes)
  - **Effective** (good compression ratio)

Shortcomings of Prior Work

Compression Mechanisms	Decompression Latency	Complexity	Compression Ratio
Zero	✓	✓	✗

Shortcomings of Prior Work

Compression Mechanisms	Decompression Latency	Complexity	Compression Ratio
Zero	✓	✓	✗
Frequent Value	✗	✗	✓

Shortcomings of Prior Work

Compression Mechanisms	Decompression Latency	Complexity	Compression Ratio
Zero	✓	✓	✗
Frequent Value	✗	✗	✓
Frequent Pattern	✗	✗/✓	✓

Shortcomings of Prior Work

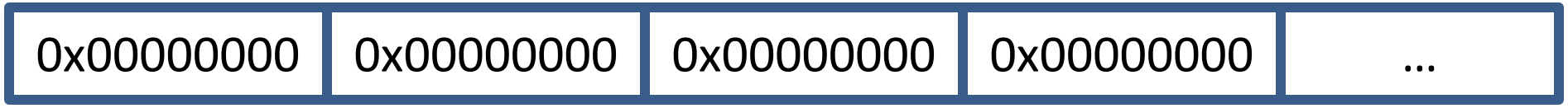
Compression Mechanisms	Decompression Latency	Complexity	Compression Ratio
Zero	✓	✓	✗
Frequent Value	✗	✗	✓
Frequent Pattern	✗	✗ / ✓	✓
Our proposal: BΔI	✓	✓	✓

Outline

- Motivation & Background
- Key Idea & Our Mechanism
- Evaluation
- Conclusion

Key Data Patterns in Real Applications

Zero Values: initialization, sparse matrices, NULL pointers



Repeated Values: common initial values, adjacent pixels

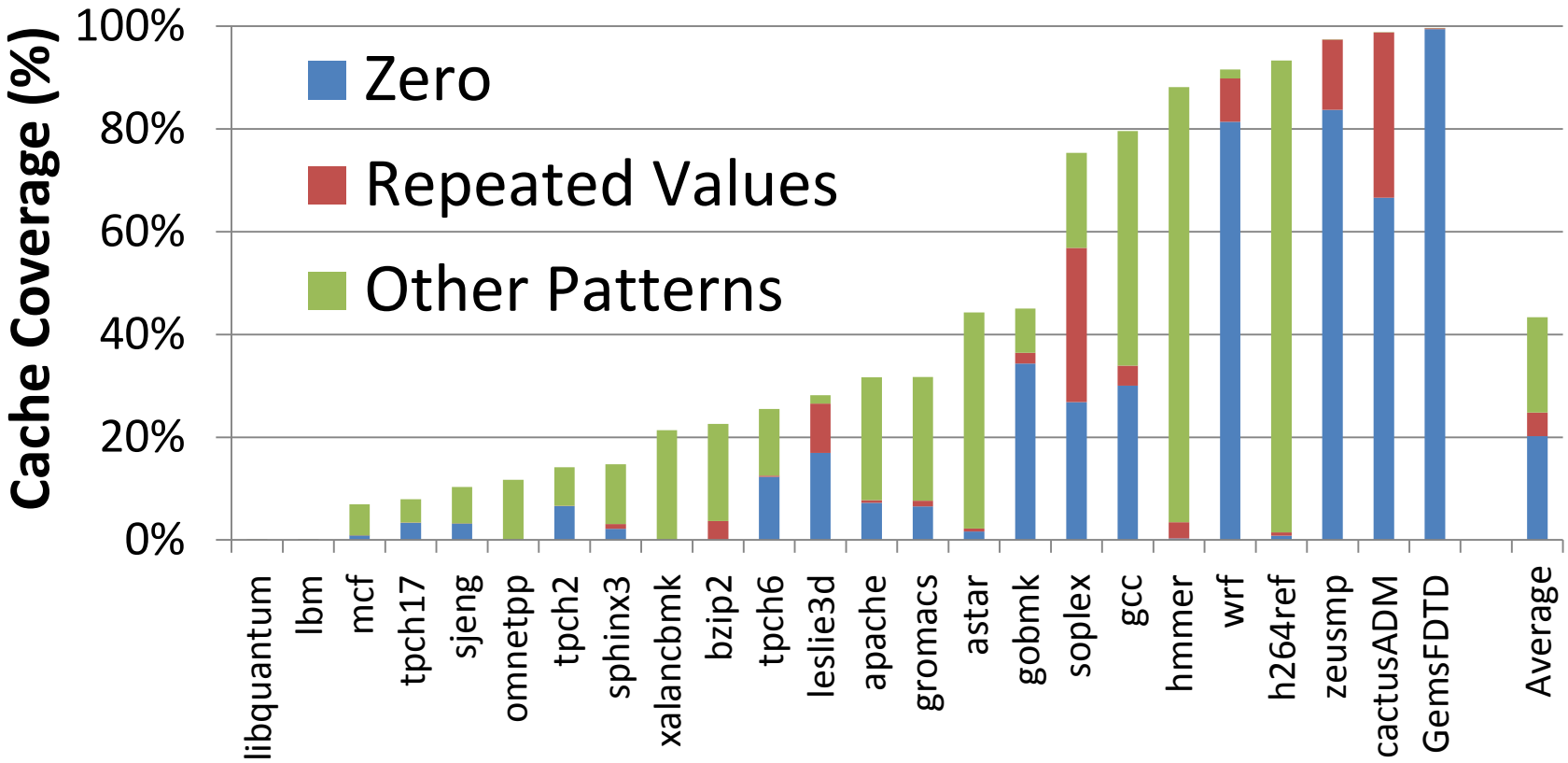
Narrow Values: small values stored in a big data type

Other Patterns: pointers to the same memory region

How Common Are These Patterns?

SPEC2006, databases, web workloads, 2MB L2 cache

"Other Patterns" include Narrow Values



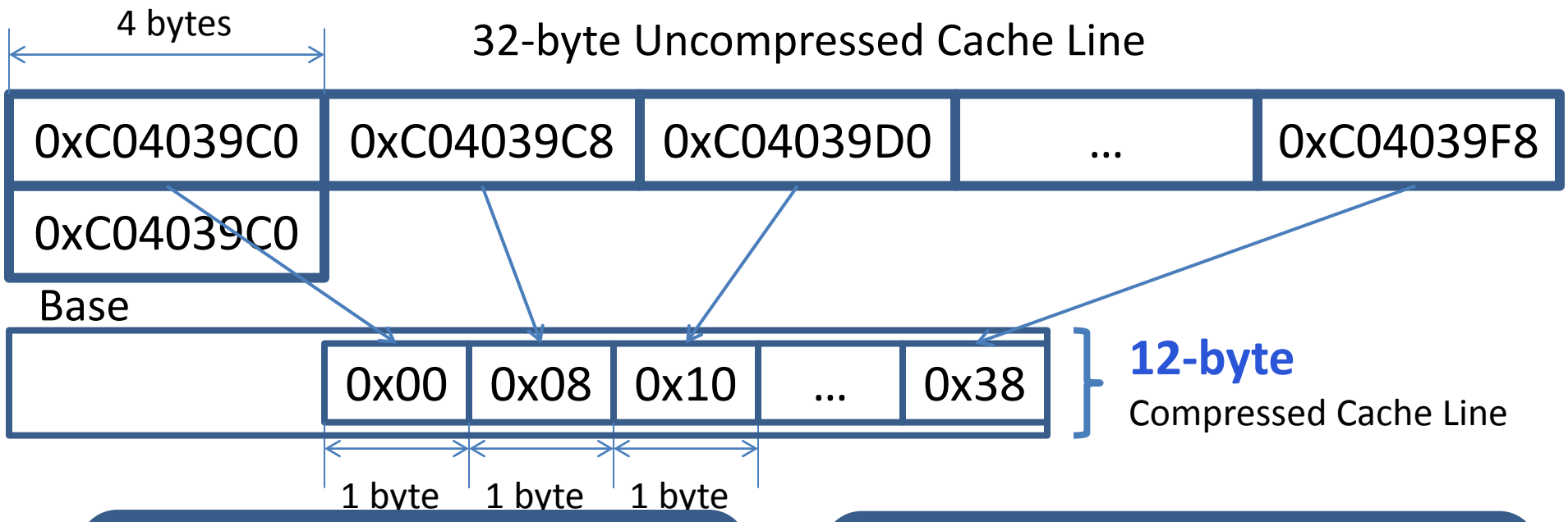
43% of the cache lines belong to key patterns

Key Data Patterns in Real Applications

Low Dynamic Range:

Differences between values are significantly smaller than the values themselves

Key Idea: Base+Delta (B+Δ) Encoding



✓ **Fast Decompression:** vector addition

✓ **Simple Hardware:** arithmetic and comparison

✓ **Effective:** good compression ratio

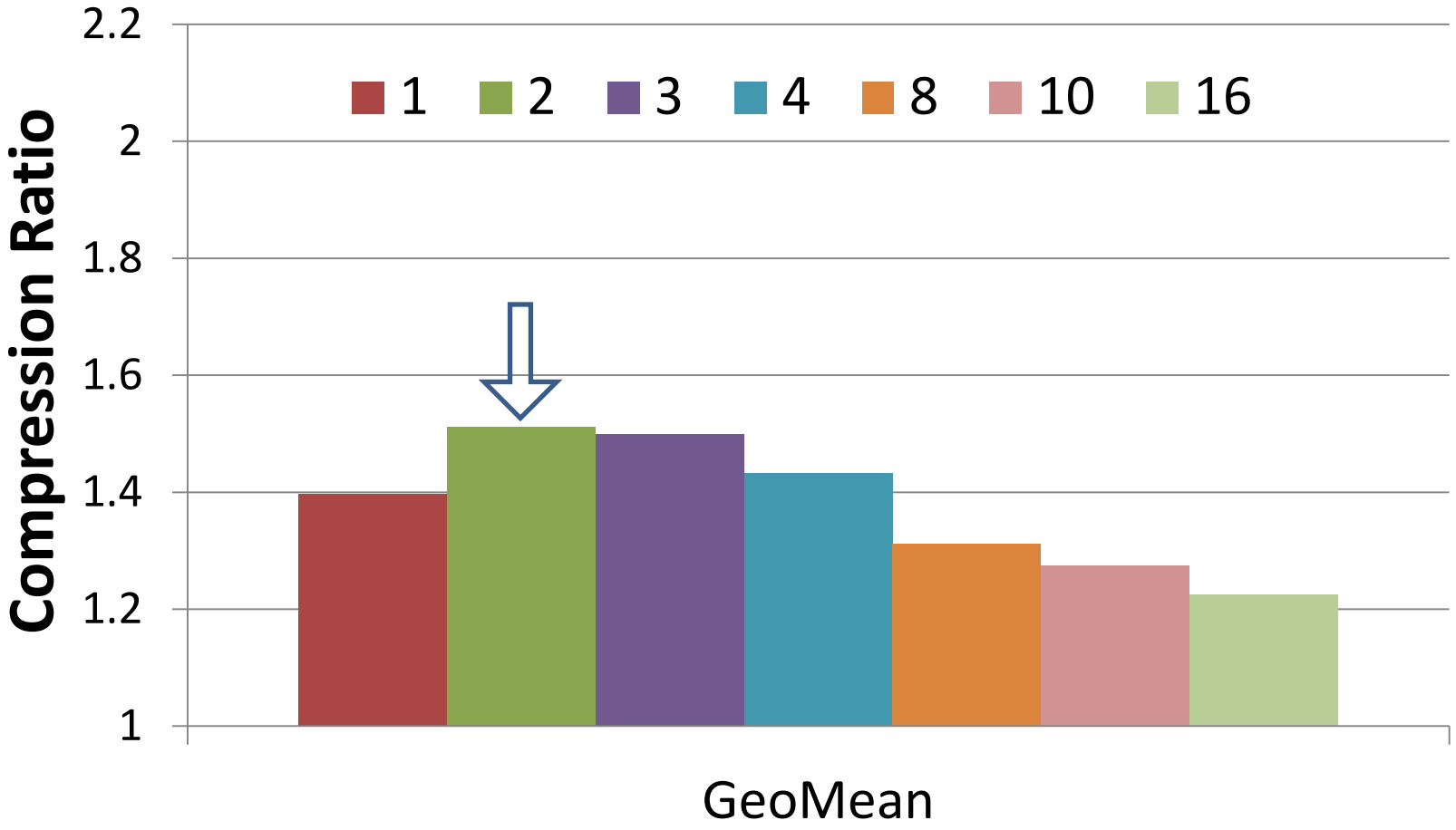
Can We Do Better?

- Uncompressible cache line (with a single base):

0x00000000	**0x09A4**0178	**0x0000000**B	**0x09A4**A838	…

- **Key idea:**
  Use more bases, e.g., two instead of one
- Pro:
  - More cache lines can be compressed
- Cons:
  - Unclear how to find these bases efficiently
  - Higher overhead (due to additional bases)

B+Δ with Multiple Arbitrary Bases



✓ **2 bases –** the best option based on evaluations

How to Find Two Bases Efficiently?

1. **First base - first element** in the cache line

   ✓ **Base+Delta part**

2. **Second base -** implicit base of **0**

   ✓ **Immediate part**

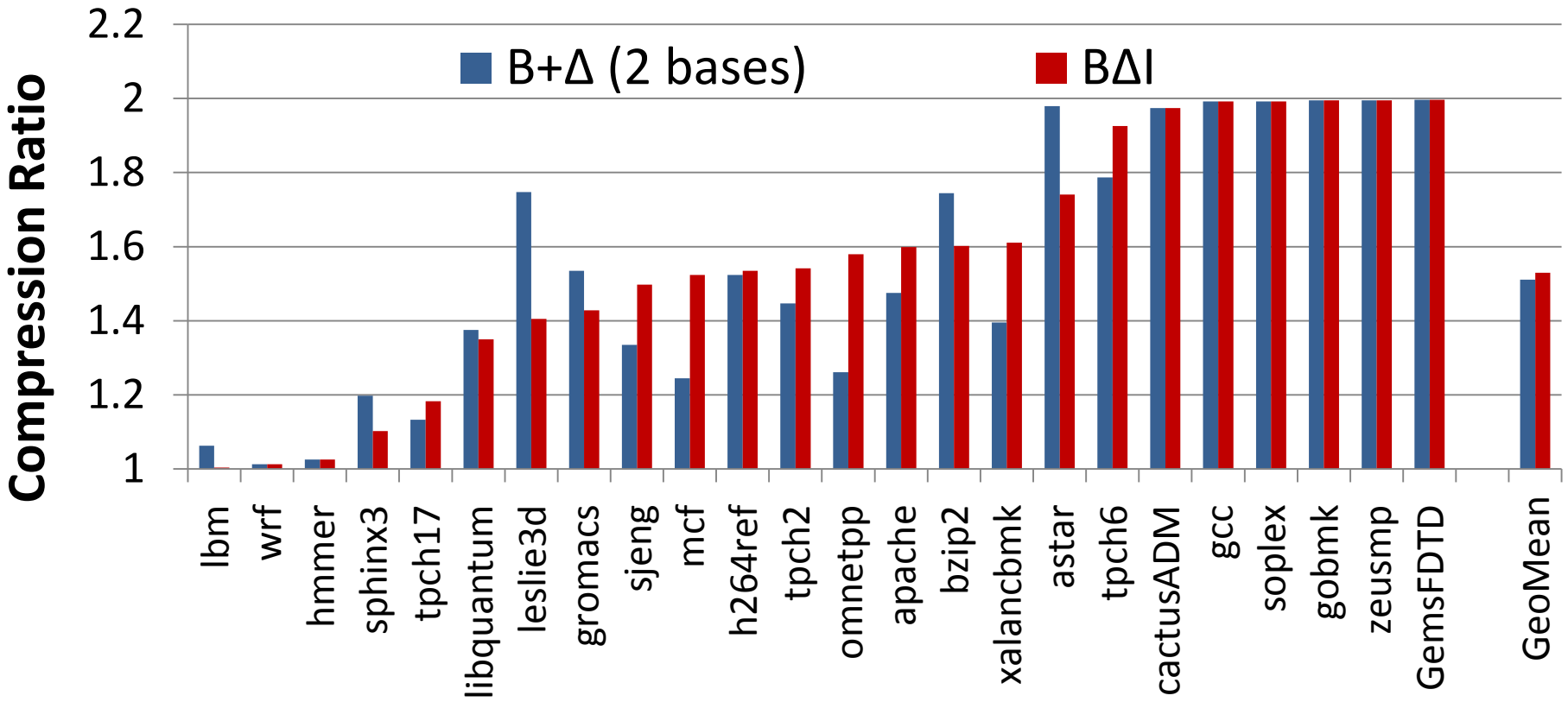
Advantages over 2 arbitrary bases:

– Better compression ratio

– Simpler compression logic

Base-Delta-Immediate (BΔI) Compression

B+Δ (with two arbitrary bases) vs. BΔI



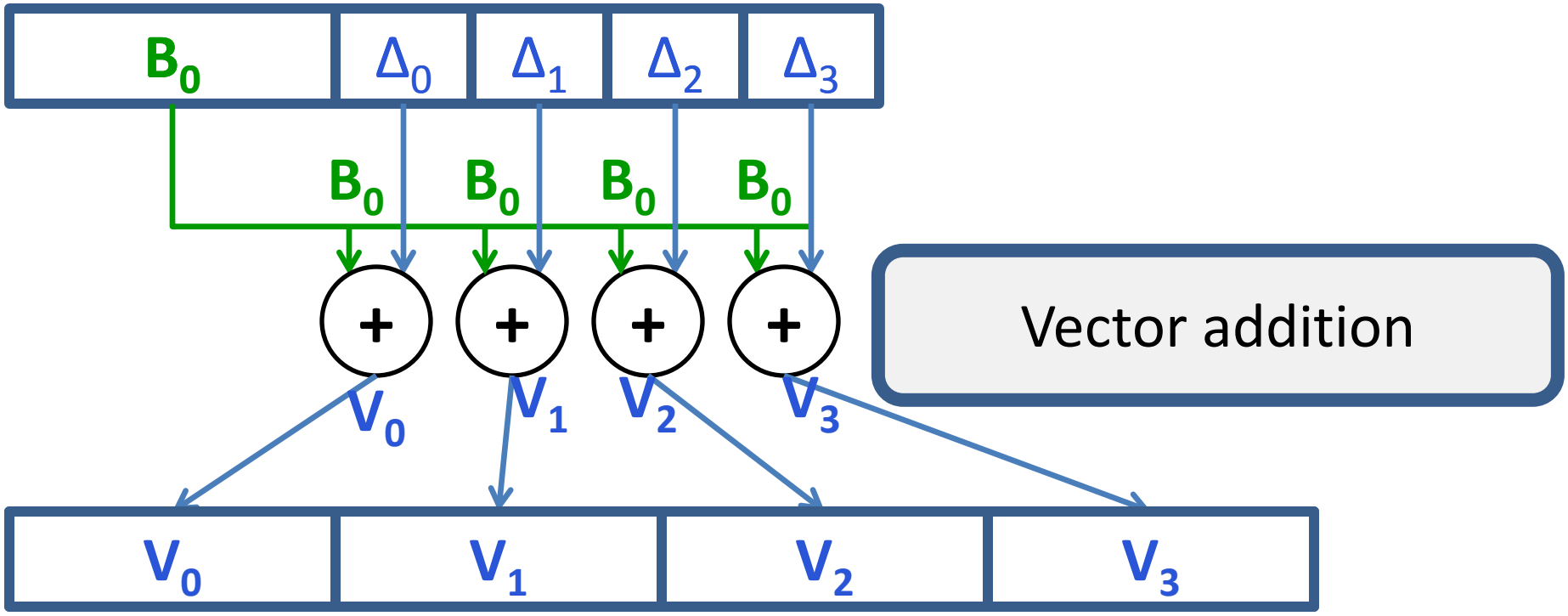
Average compression ratio is close, but **BΔI** is **simpler**

BΔI Implementation

- **Decompressor Design**
  - Low latency
- **Compressor Design**
  - Low cost and complexity
- **BΔI Cache Organization**
  - Modest complexity

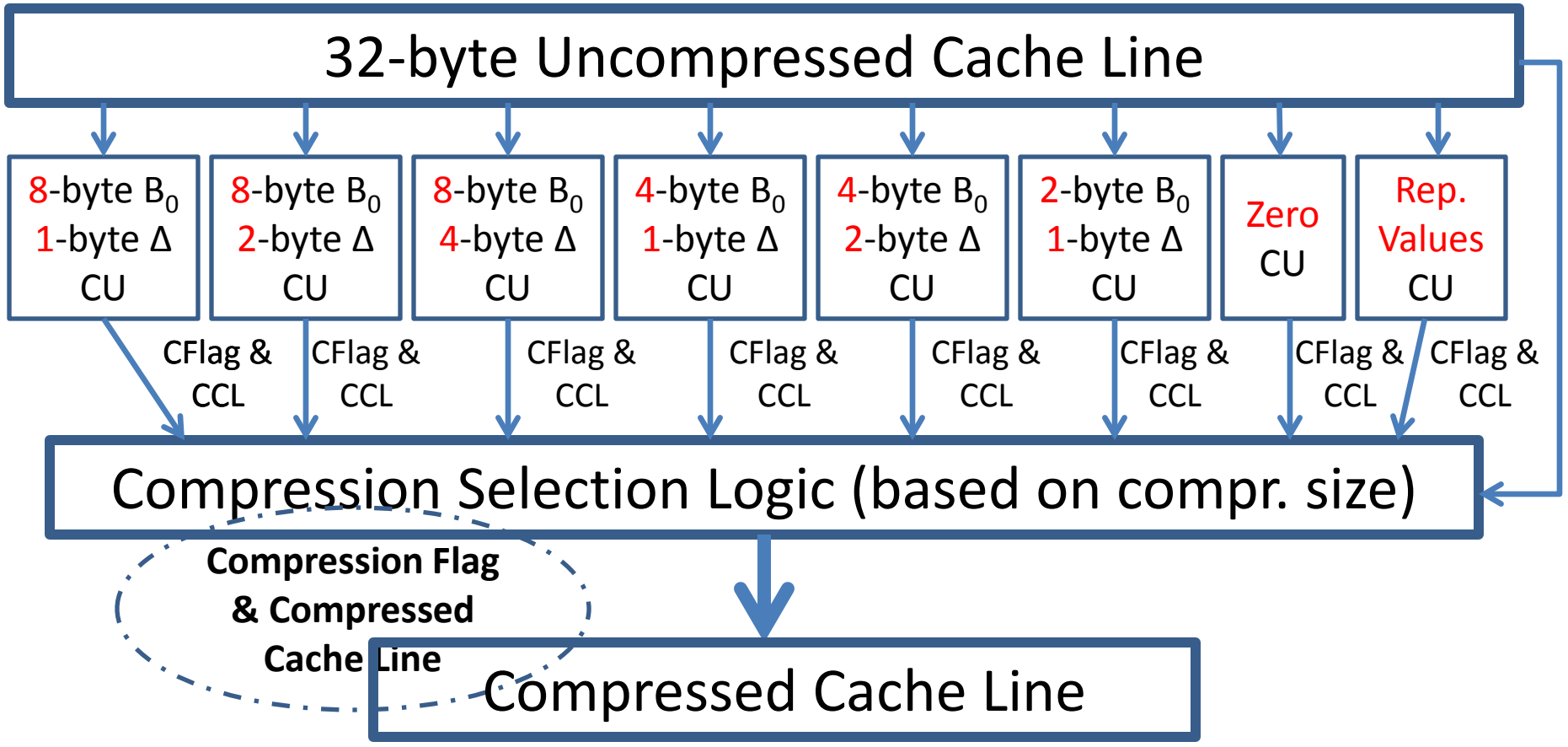
BΔI Decompressor Design

Compressed Cache Line

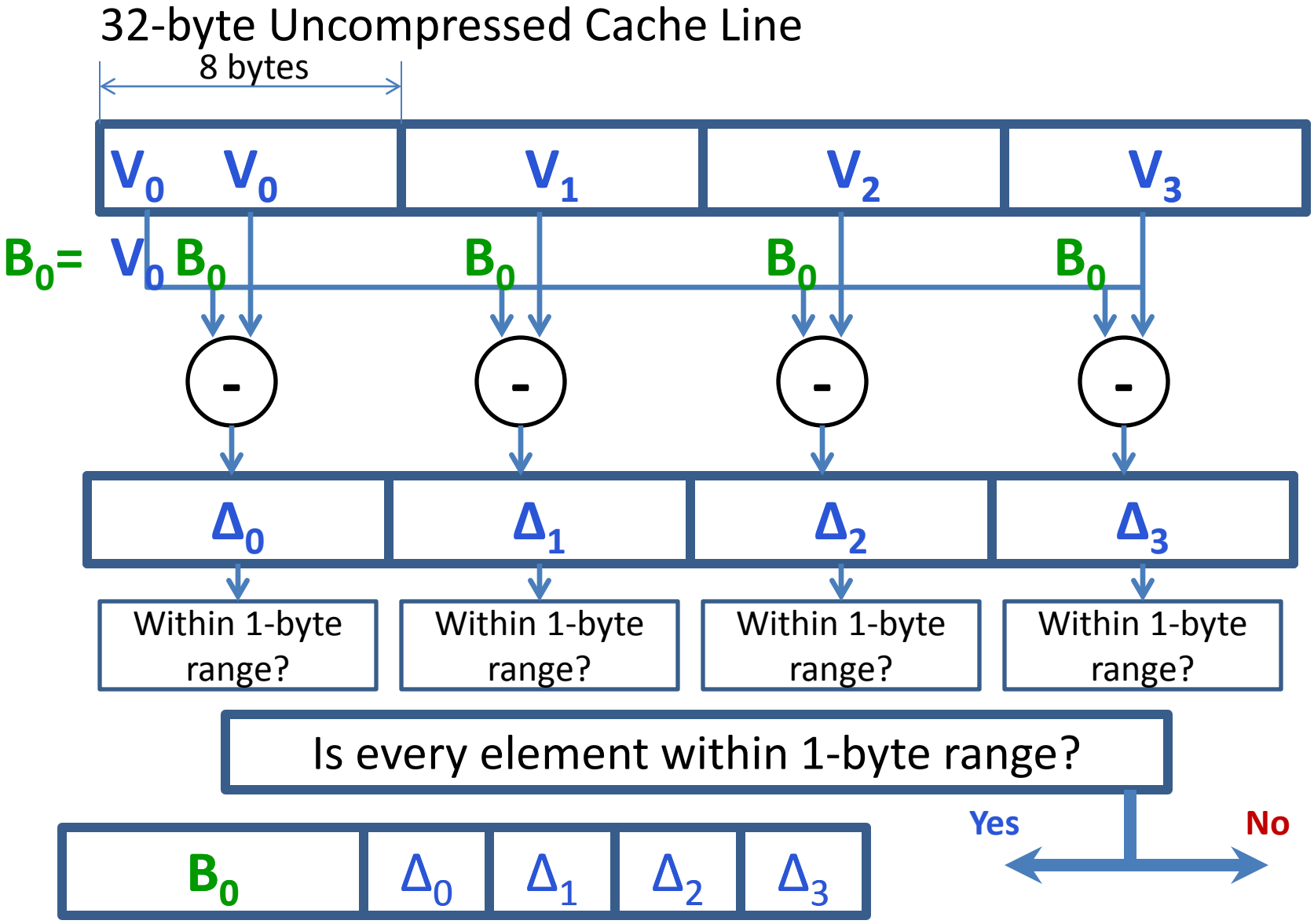


Uncompressed Cache Line

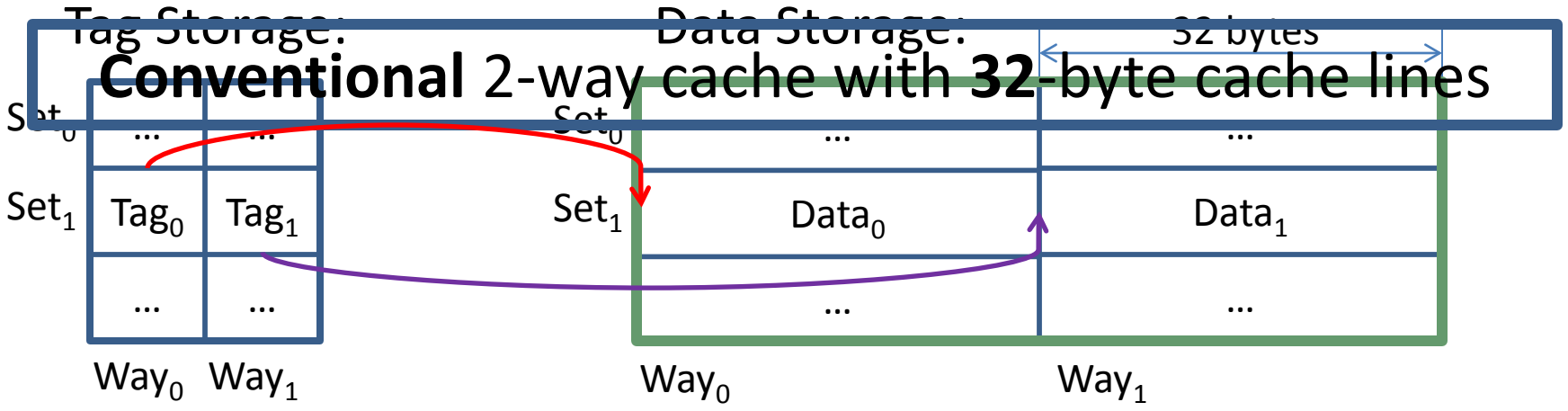
BΔI Compressor Design



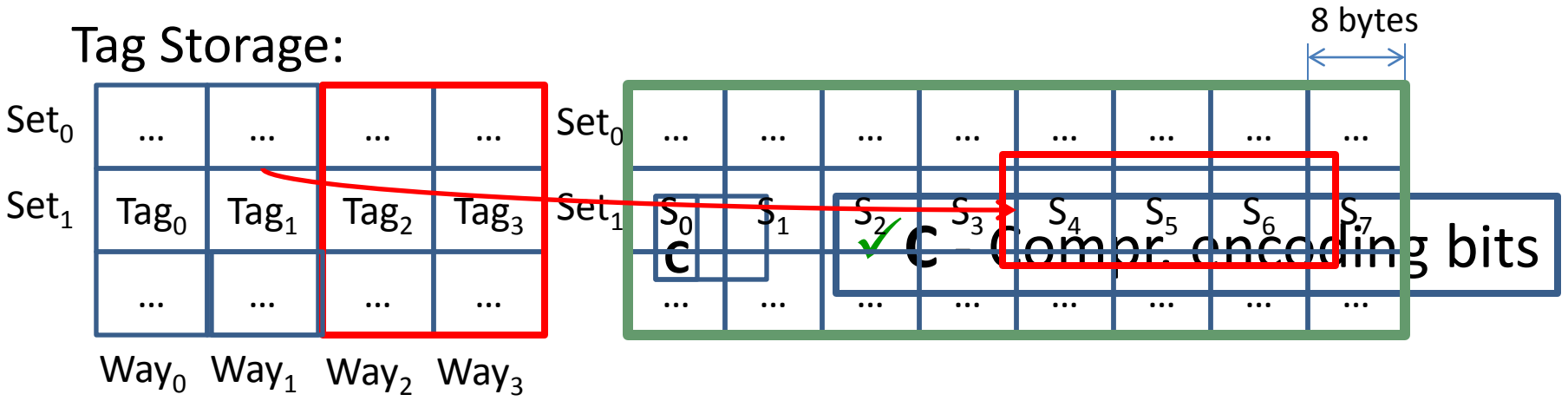
BΔI Compression Unit: 8-byte B_0 1-byte Δ



BΔI Cache Organization



BΔI: 4-way cache with **8**-byte segmented data



✓ Twice as many tags

✓ Tags map to multiple adjacent segments

2.3% overhead for 2 MB cache

Qualitative Comparison with Prior Work

- **Zero-based designs**
  - ZCA *[Dusser+, ICS'09]*: zero-content augmented cache
  - ZVC *[Islam+, PACT'09]*: zero-value cancelling
  - Limited applicability (only zero values)
- **FVC** *[Yang+, MICRO'00]*: frequent value compression
  - High decompression latency and complexity
- **Pattern-based compression designs**
  - FPC *[Alameldeen+, ISCA'04]*: frequent pattern compression
    - High decompression latency (5 cycles) and complexity
  - C-pack *[Chen+, T-VLSI Systems'10]*: practical implementation of FPC-like algorithm
    - High decompression latency (8 cycles)

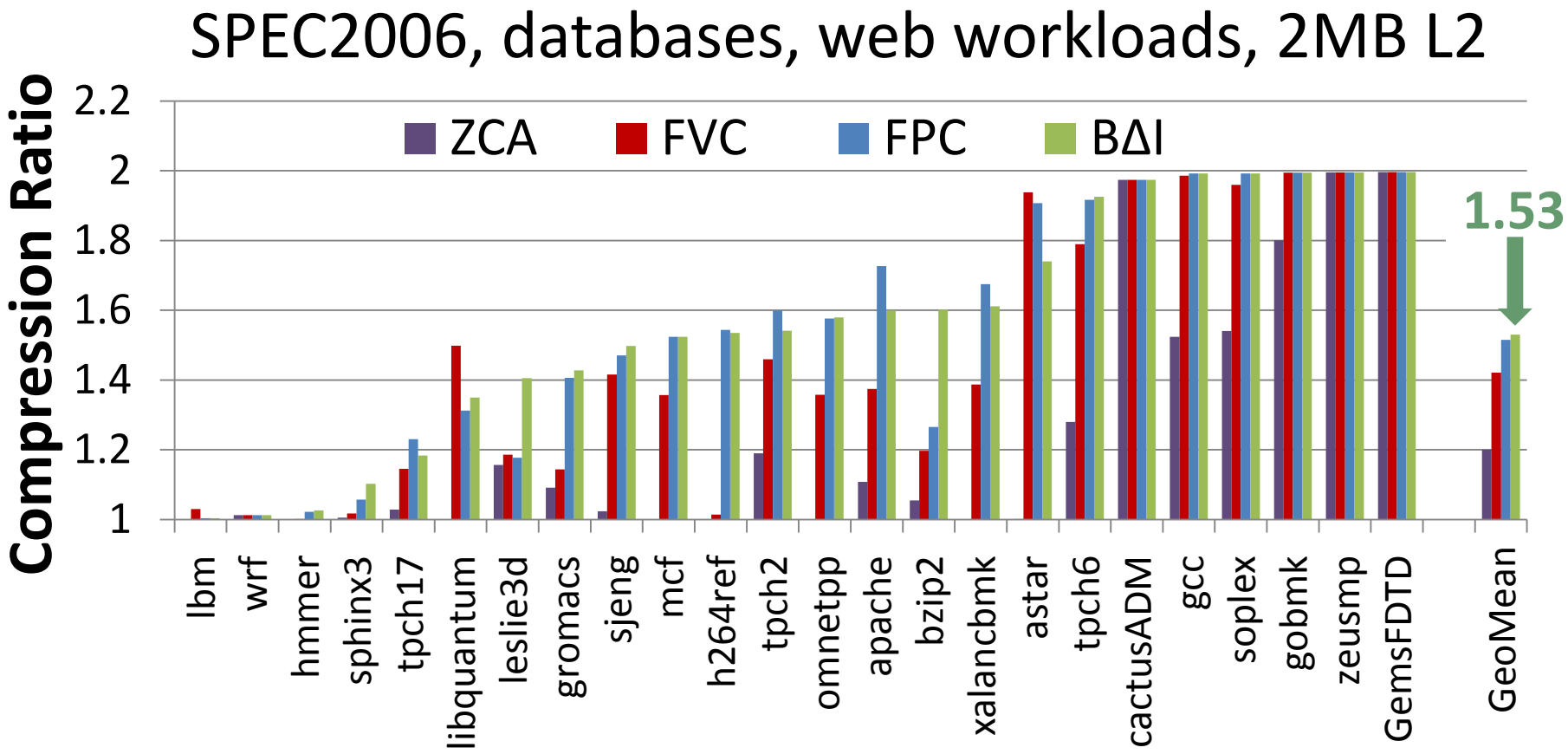
Outline

- Motivation & Background
- Key Idea & Our Mechanism
- Evaluation
- Conclusion

Methodology

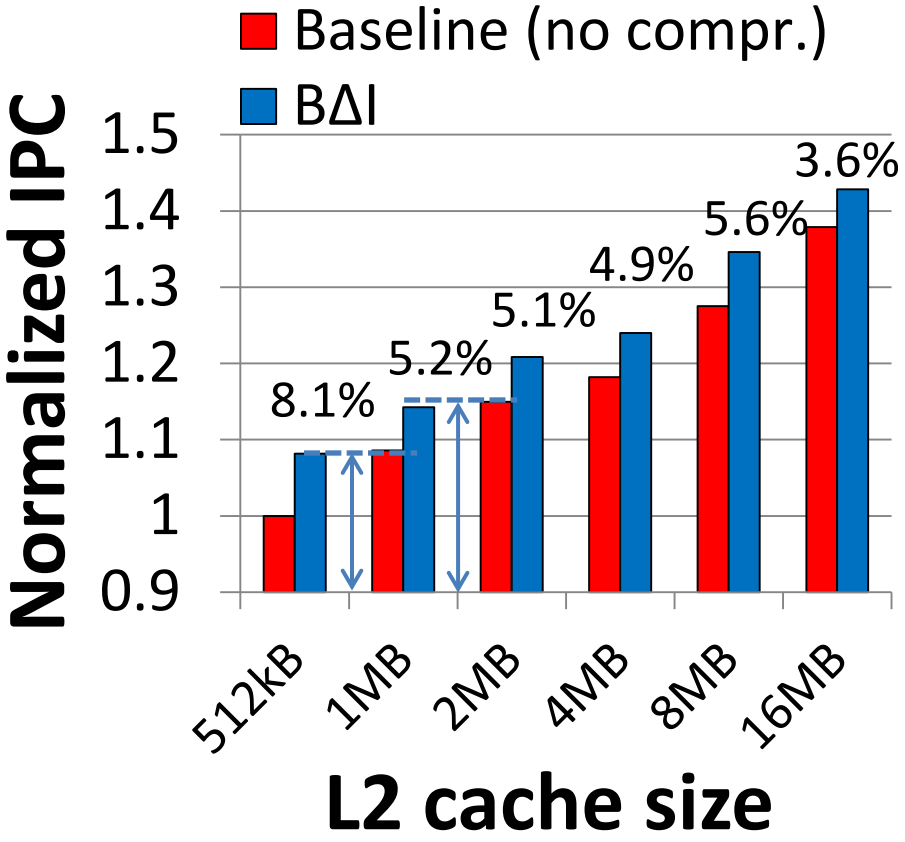
- **Simulator**
  - x86 event-driven simulator based on Simics *[Magnusson+, Computer'02]*
- **Workloads**
  - SPEC2006 benchmarks, TPC, Apache web server
  - 1 – 4 core simulations for 1 billion representative instructions
- **System Parameters**
  - L1/L2/L3 cache latencies from CACTI *[Thoziyoor+, ISCA'08]*
  - 4GHz, x86 in-order core, **512kB - 16MB** L2, simple memory model (**300**-cycle latency for row-misses)

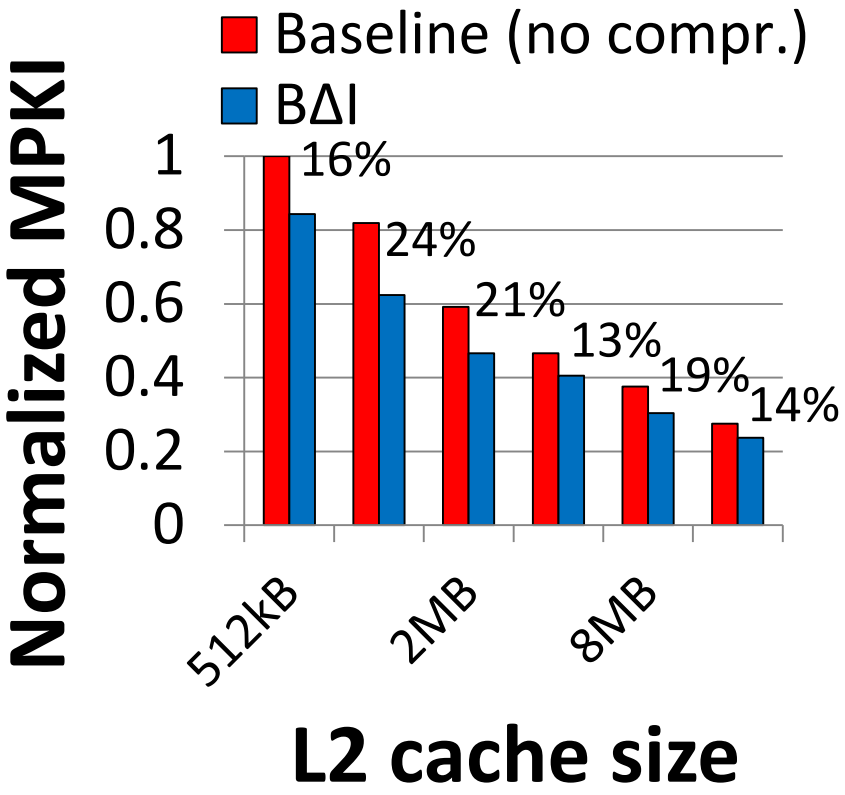
Compression Ratio: BΔI vs. Prior Work



BΔI achieves the highest compression ratio

Single-Core: IPC and MPKI





BΔI achieves the performance of a 2X-size cache

Performance improves due to the decrease in MPKI

Multi-Core Workloads

- Application classification based on

  Compressibility: effective cache size increase

  (Low Compr. (***LC***) < 1.40, High Compr. (***HC***) >= 1.40)

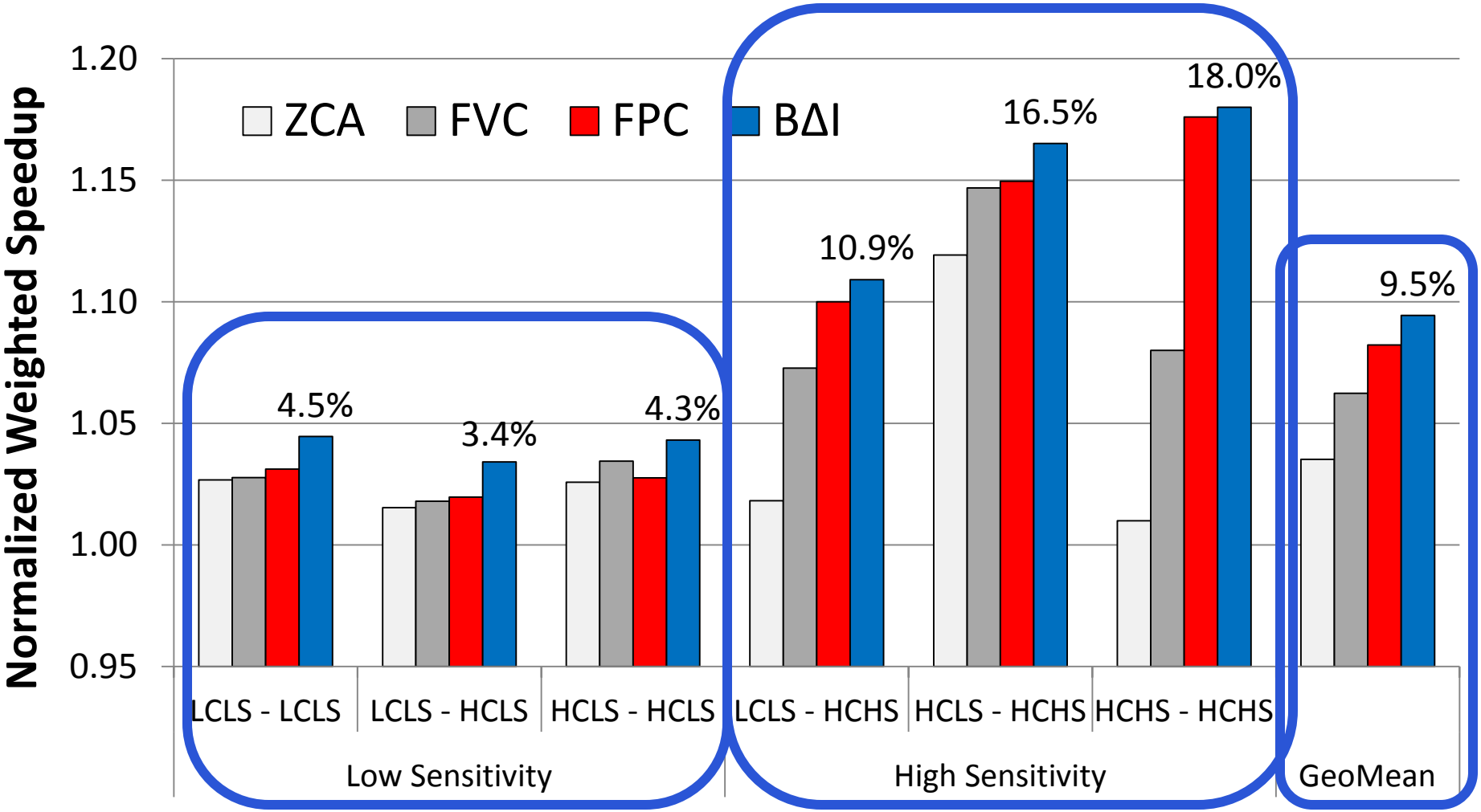
  Sensitivity: performance gain with more cache

  (Low Sens. (***LS***) < 1.10, High Sens. (***HS***) >= 1.10; 512kB -> 2MB)

- Three classes of applications:
  - LCLS, HCLS, HCHS, **no LCHS** applications

- For 2-core - **random** mixes of each possible class pairs (20 each, 120 total workloads)

Multi-Core: Weighted Speedup



If at least one application is sensitive, then the performance improves

BΔI performance improvement is the highest (**9.5%**)

Other Results in Paper

- IPC comparison against **upper** bounds
  - BΔI almost achieves performance of the 2X-size cache
- Sensitivity study of having **more** than 2X tags
  - Up to 1.98 average compression ratio
- Effect on **bandwidth** consumption
  - 2.31X decrease on average
- Detailed quantitative comparison with prior work
- **Cost analysis** of the proposed changes
  - 2.3% L2 cache area increase

Conclusion

- A new **Base-Delta-Immediate** compression mechanism
- Key insight: many cache lines can be efficiently represented using **base + delta encoding**
- Key properties:
  - **Low** latency decompression
  - **Simple** hardware implementation
  - **High compression ratio** with high coverage
- **Improves** *cache hit ratio* and *performance* of both single-core and multi-core workloads
  - Outperforms state-of-the-art cache compression techniques: FVC and FPC

Base-Delta-Immediate Compression: Practical Data Compression for On-Chip Caches

Gennady Pekhimenko,
Vivek Seshadri ,
Onur Mutlu , Todd C. Mowry

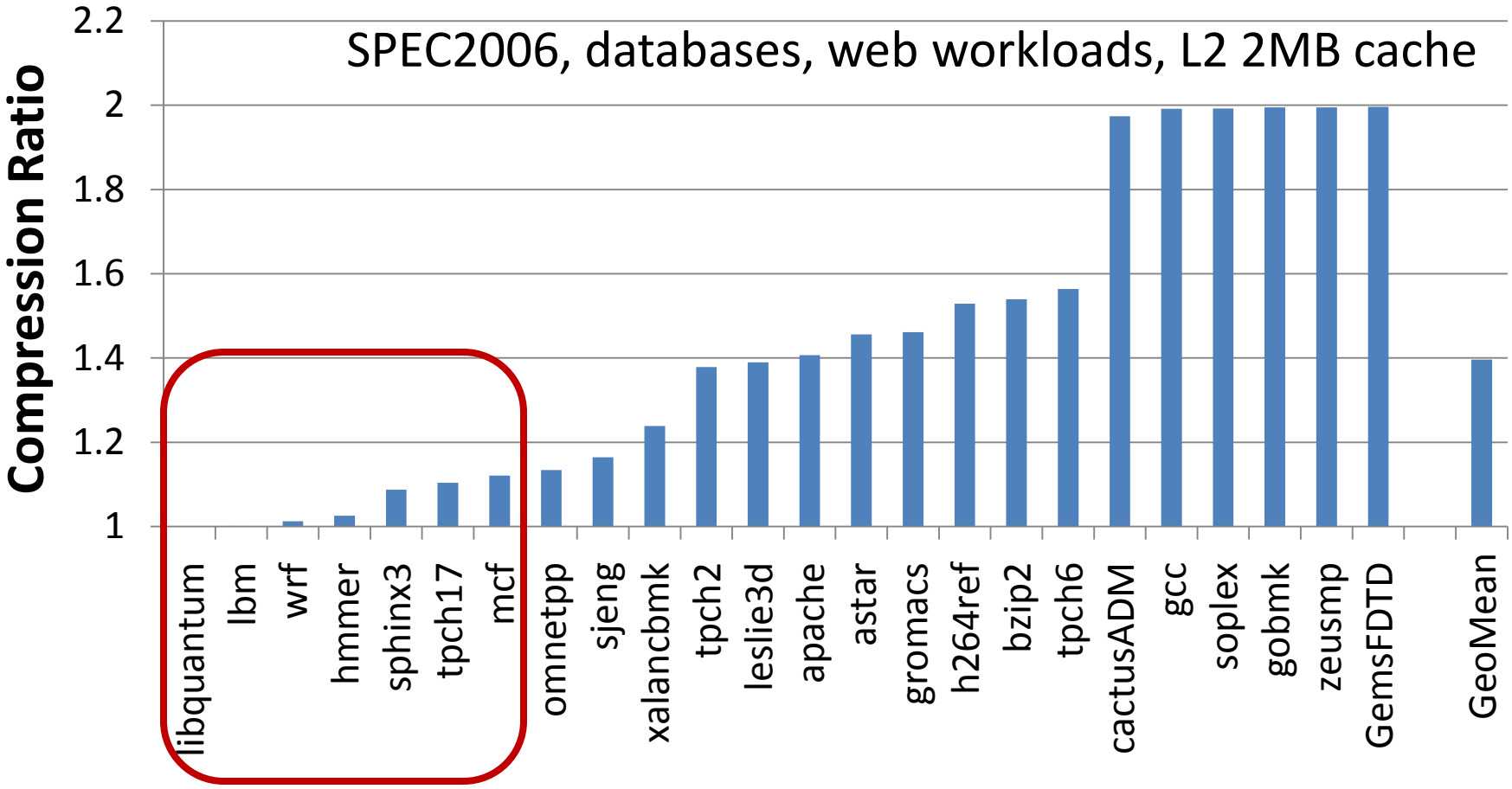
Phillip B. Gibbons*,
Michael A. Kozuch*

Carnegie Mellon

intel *

Backup Slides

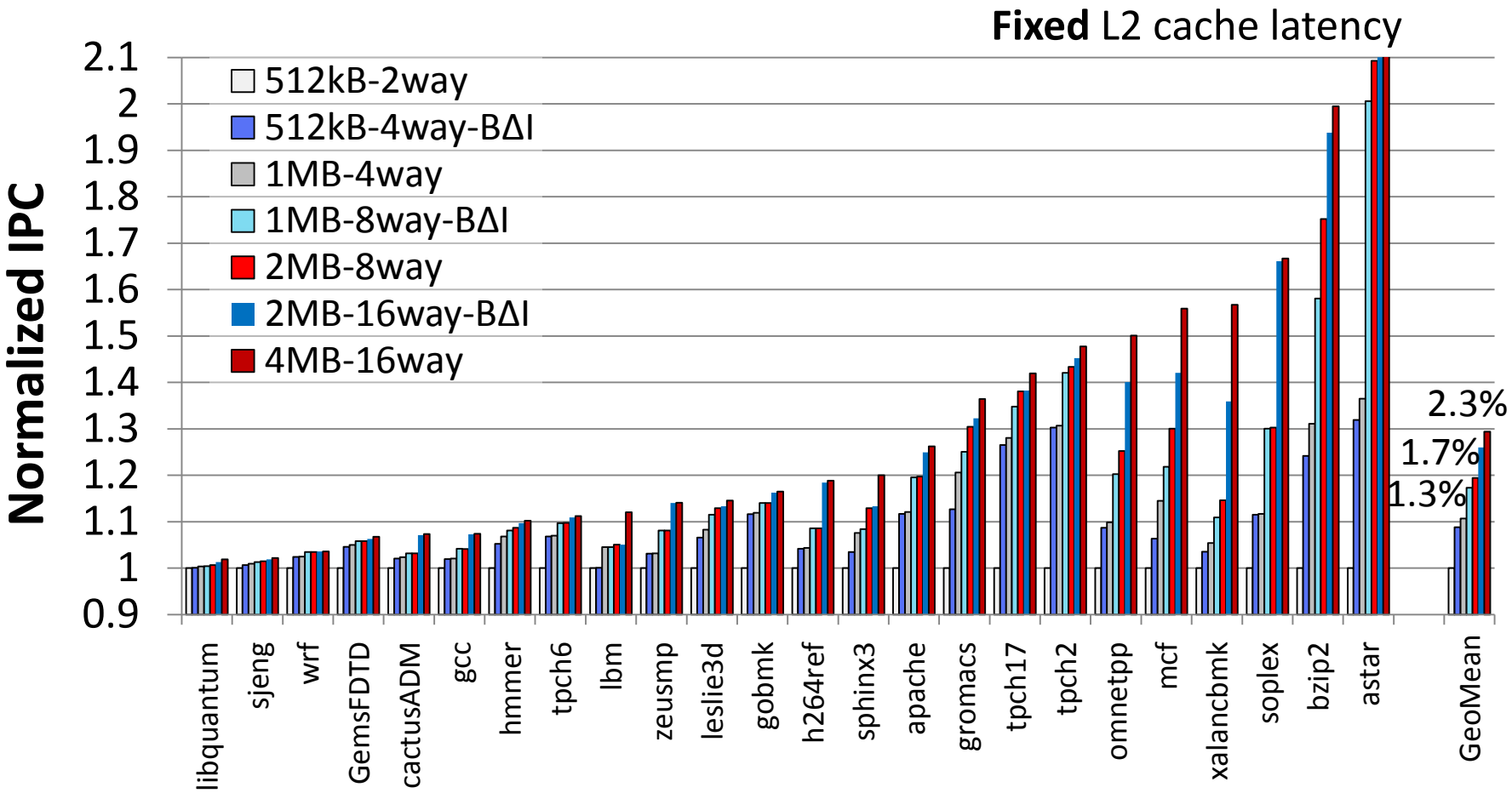
B+Δ: Compression Ratio



Good average compression ratio (**1.40**)

But some benchmarks have low compression ratio

Single-Core: Effect on Cache Capacity



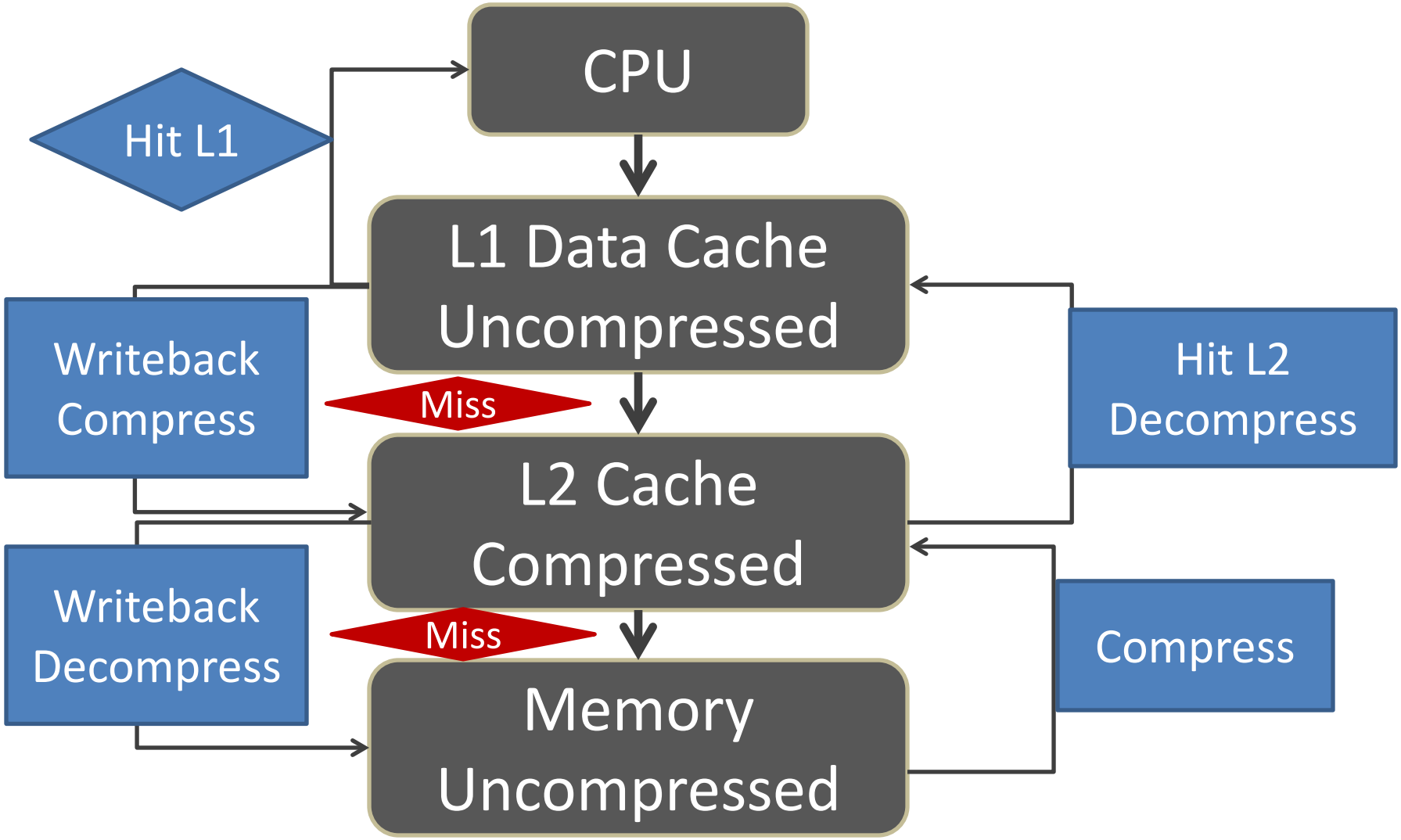
BΔI achieves performance close to the upper bound

Multiprogrammed Workloads - I

Cat.	Name	C. Ratio	Sens.	Name	C. Ratio	Sens.	Name	C. Ratio	Sens.
LCLS	gromacs	1.43 / L	L	hmmer	1.03 / L	L	lbm	1.00 / L	L
	sphinx	1.10 / L	L	tpch17	1.18 / L	L	wrf	1.01 / L	L
HCLS	apache	1.60 / H	L	zeusmp	1.99 / H	L	gcc	1.99 / H	L
	sjeng	1.50 / H	L	tpch2	1.54 / H	L	tpch6	1.93 / H	L
HCHS	astar	1.74 / H	H	bzip2	1.60 / H	H	mcf	1.52 / H	H
	soplex	1.99 / H	H	h264ref	1.52 / H	H			

Cat.	Name	C. Ratio	Sens.	Name	C. Ratio	Sens.
LCLS	libquantum	1.25 / L	L	leslie3d	1.41 / L	L
HCLS	GemsFDTD	1.99 / H	L	gobmk	1.99 / H	L
	cactusADM	1.97 / H	L			
HCHS	xalancbmk	1.61 / H	H	omnetpp	1.58 / H	H

Cache Compression Flow



Example of Base+Delta Compression

- Narrow values (taken from h264ref):