HyComp: A Hybrid Cache Compression Method for Selection of Data-Type-Specific Compression Methods

Angelos Arelakis,
Fredrik Dahlgren & Per Stenstrom

Chalmers University of Technology
Gothenburg, Sweden

Background

- Caches are essential:
  - \+ Bridge the speed gap between processors and memory
  - – Occupy 20-40% of chip's real estate
- Increasing the cache size:
  - \+ Reduces Avg. Memory Access Time (AMAT), but
  - – More area to occupy
  - – Higher cache hit time
  - – Dissipate more power

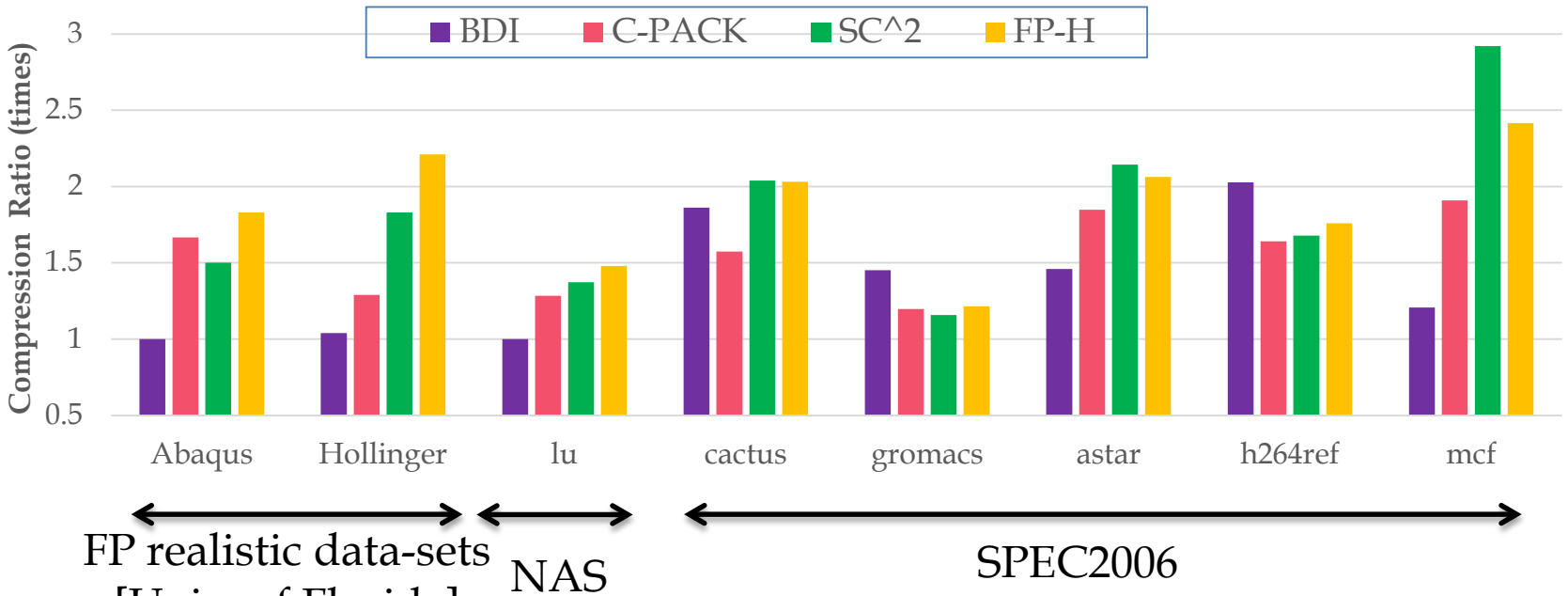
Cache compression: Alternative to extend cache capacity without physically enlarging the cache

Motivation

- Challenges in cache compression:
  - Decompression adds latency to the cache access → must be **fast**!
  - Data of diverse types: exhibit different value locality properties

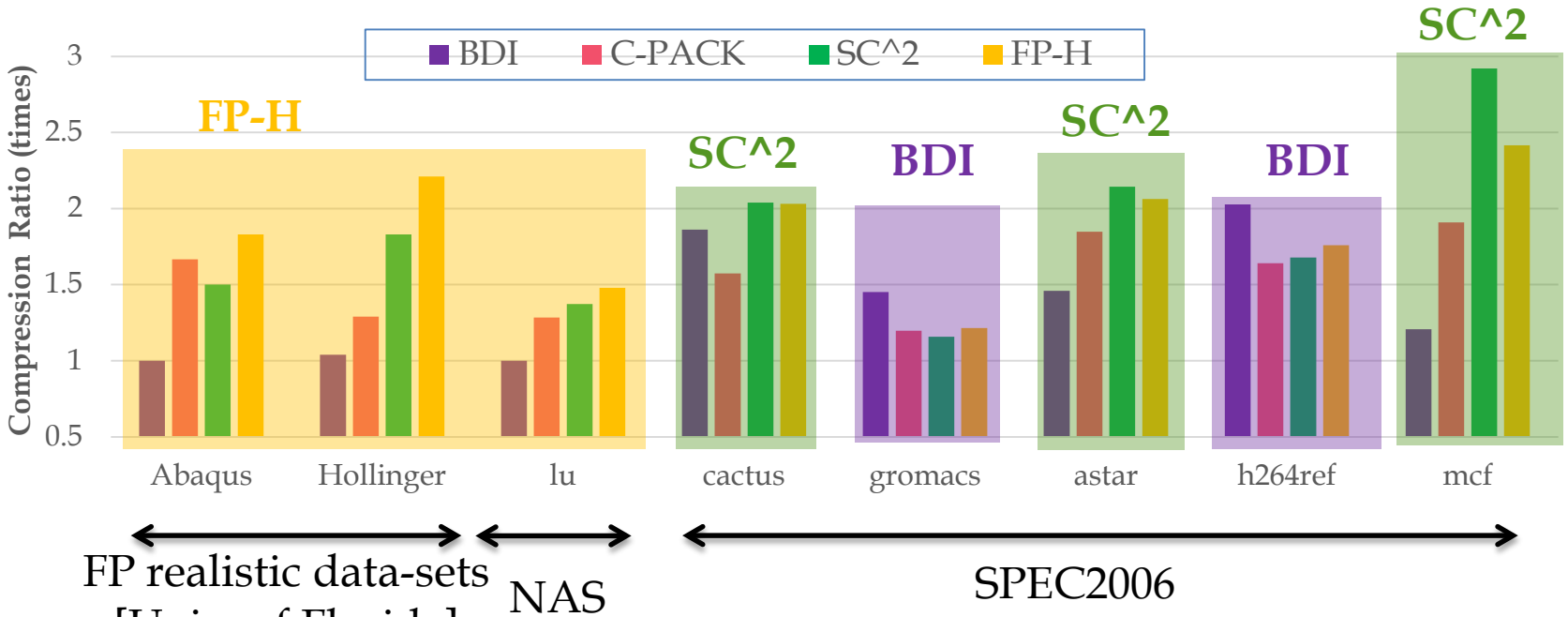
Motivation

- Base-Delta Immediate compression (**BDI**): Exploits clustered value locality
- **C-Pack:** Significance and Dictionary-based compression
- Statistical cache compression (**SC^2**): Exploits Huffman encoding
- **FP-H**: Semantic bit-field compression (Discussed later)



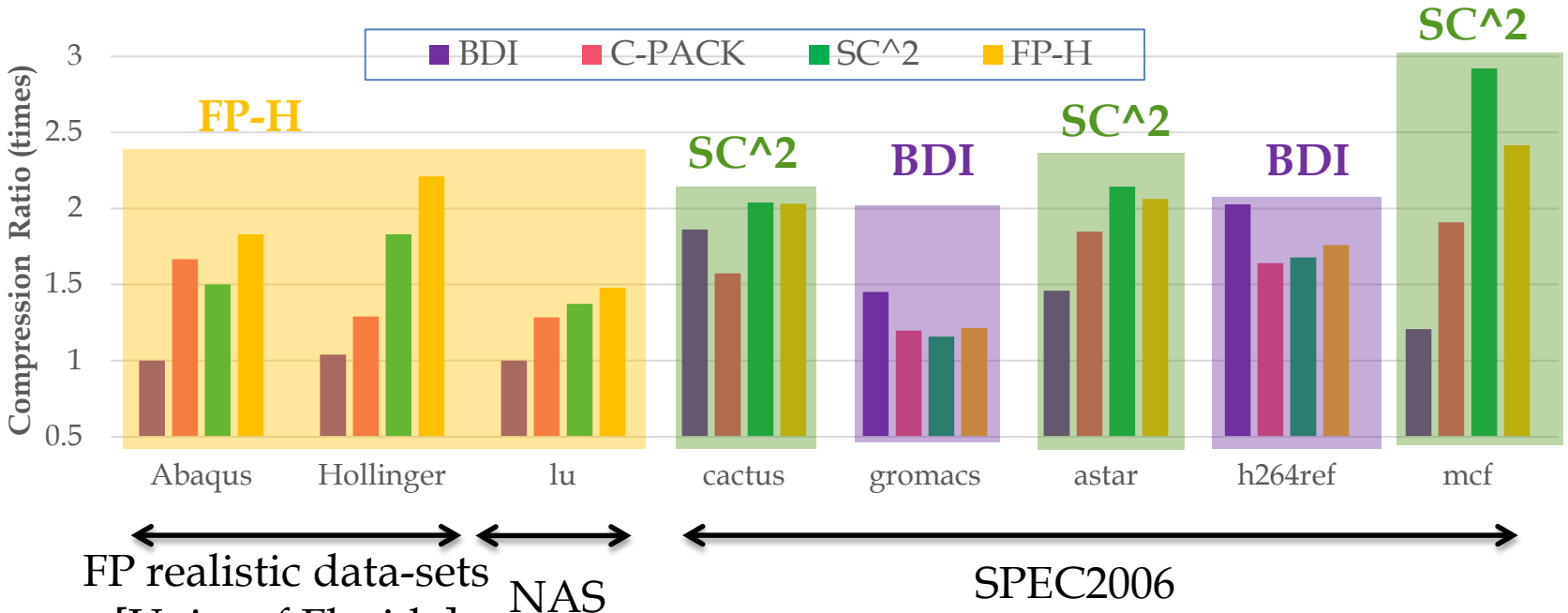
Motivation

One method does **not always** compress the best!



Motivation

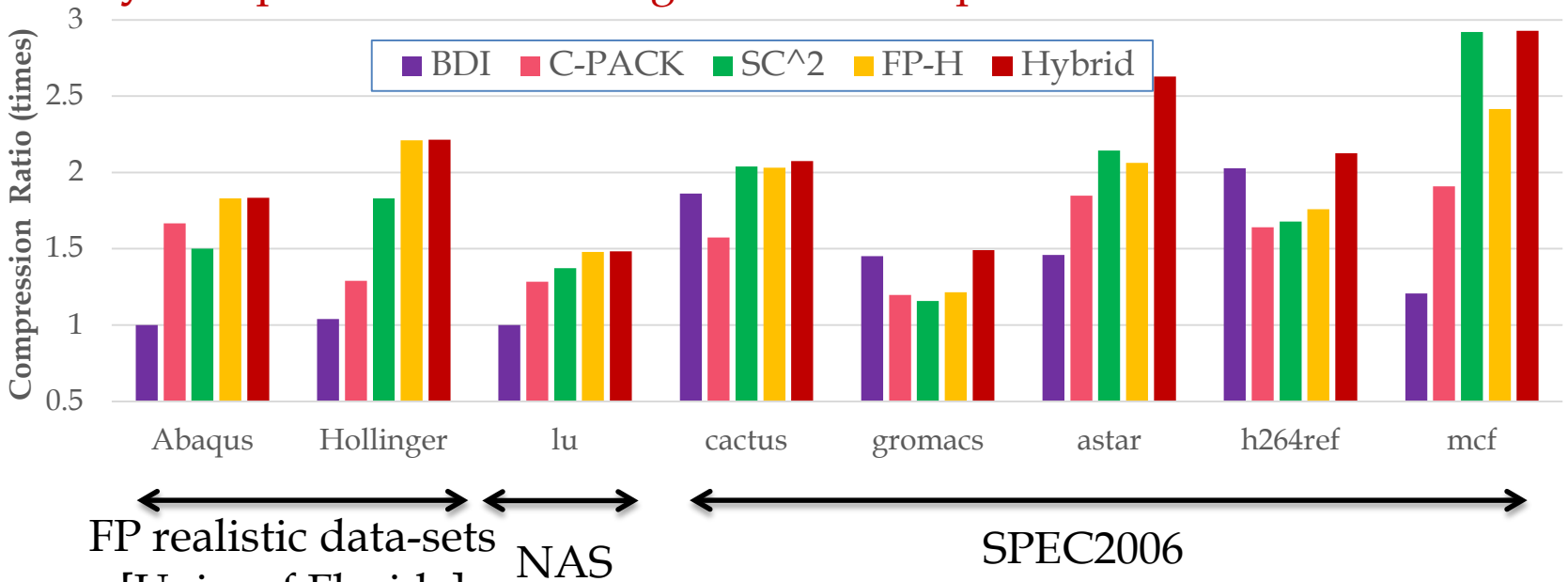
One method does **not always** compress the best!
Why?
Each method targets specific data-types



Motivation

One method does **not always** compress the best!
Why?
Each method targets specific data-types

- **Hybrid**: picks the best among different compression methods



Contributions

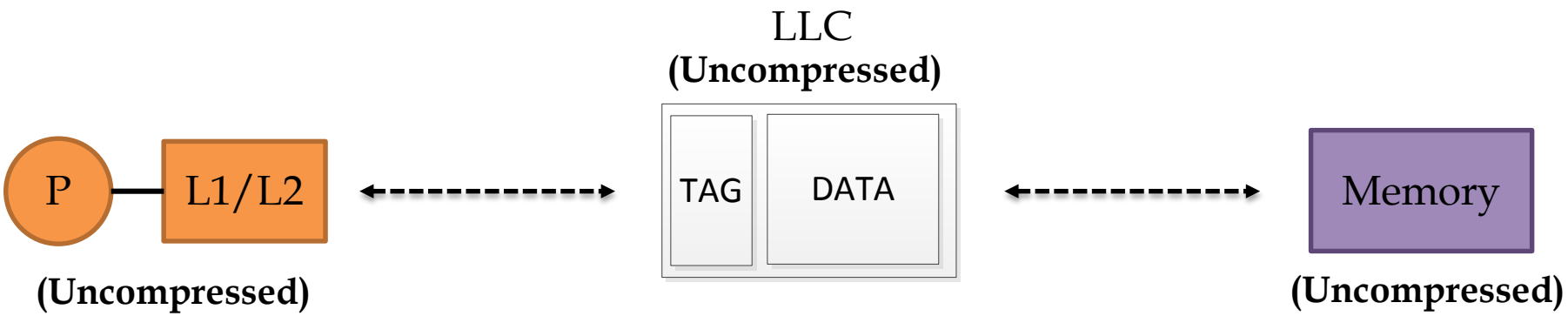
- **HyComp**: Hybrid compression method for caches
  - Runtime selection of the best compression method based on type prediction with 82% accuracy on average
  - Robust compression and speedup across applications with diverse data types
  - No impact on decompression latency
- **FP-H**: Semantic bit-field compression of Floating-point data, using Huffman, for caches
  - Value locality in the mantissa, if partitioned

Outline

- Background
- Motivation
- Contributions
- HyComp: Hybrid compression in cache
- FP-H: Semantic bit-field compression applied to floating-point numbers
- Evaluation
- Conclusions and future work

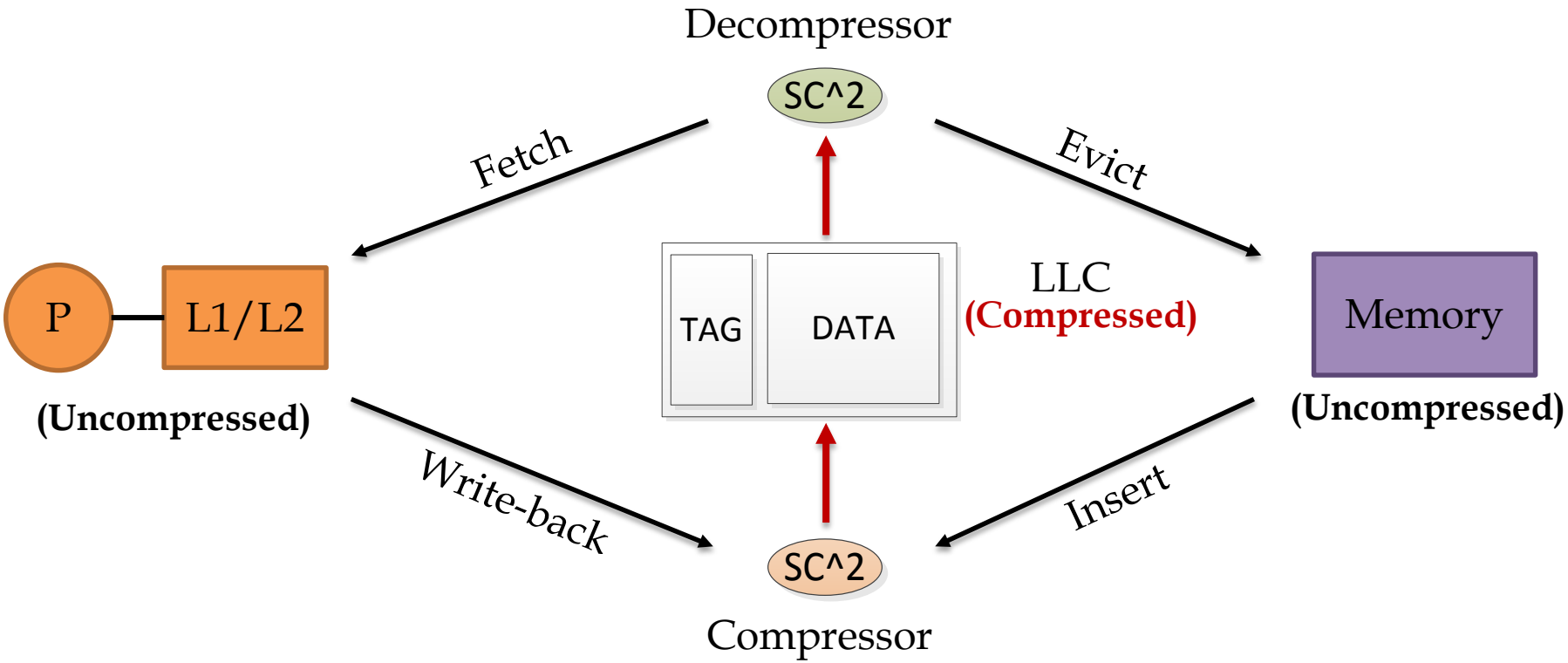
Baseline

- **Framework:** Multi-core with a 3-level cache hierarchy
- Compress the LLC to reduce the off-chip accesses



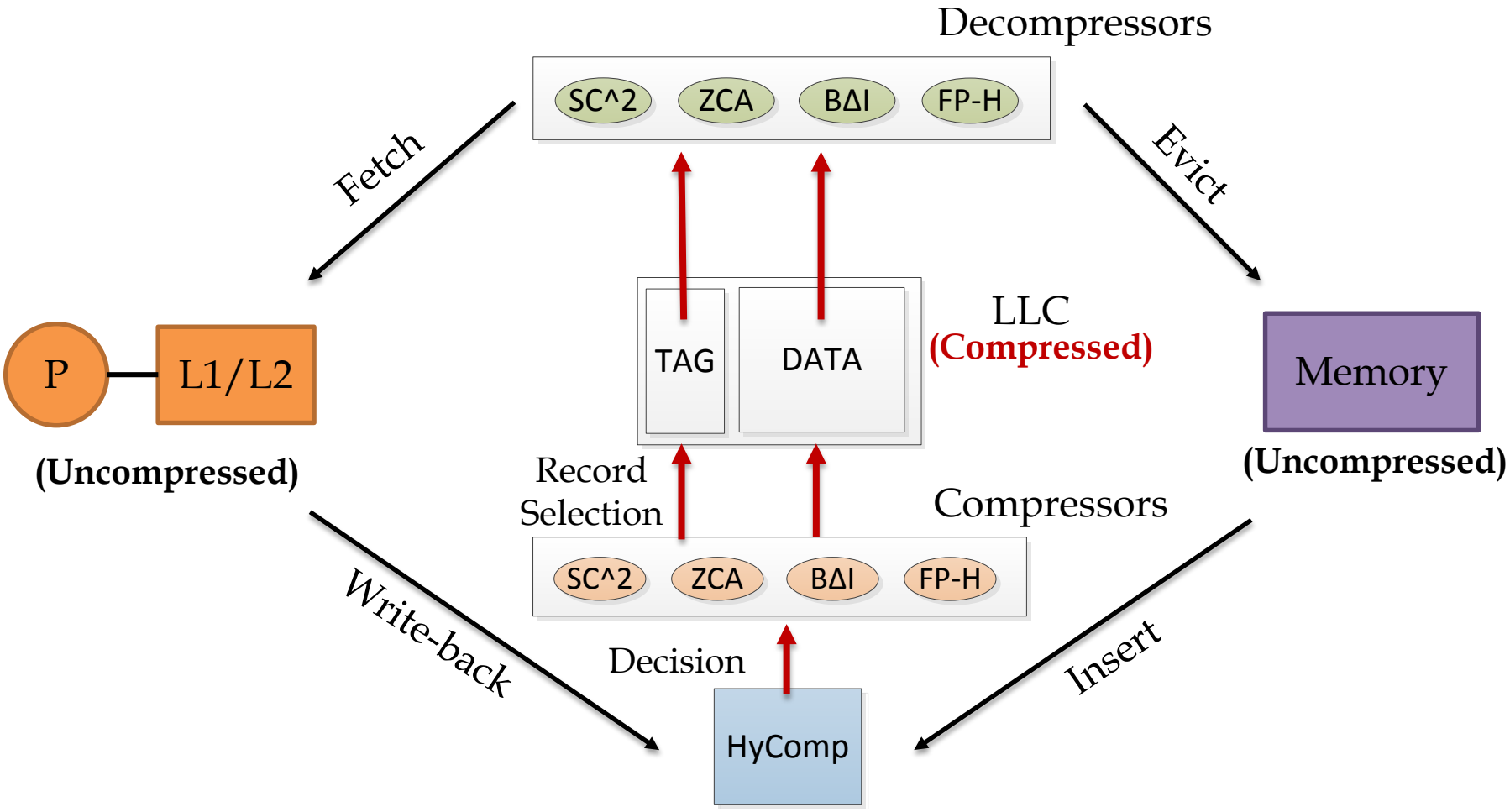
Compression with one Scheme

- Compressed LLC with SC^2



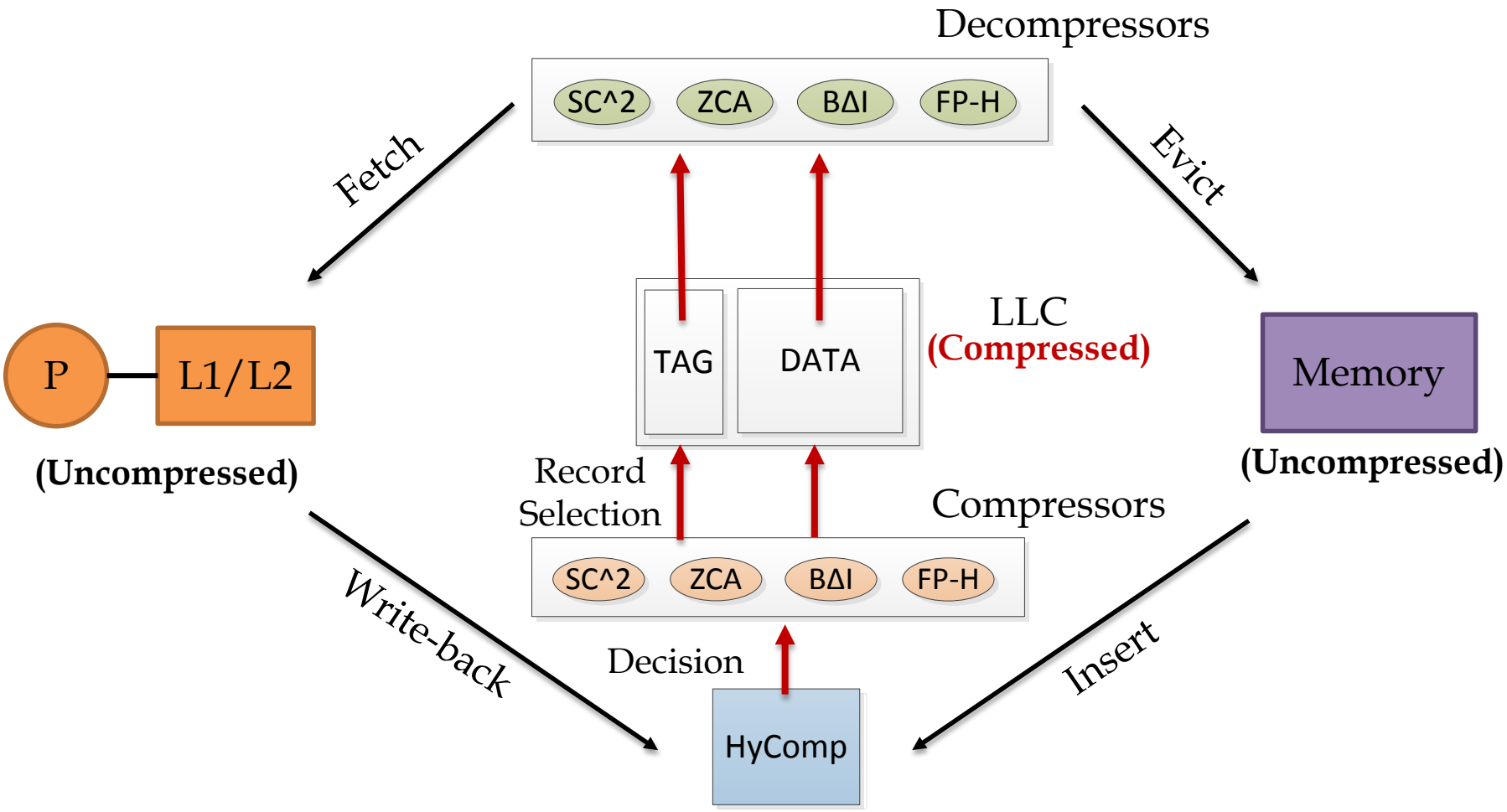
Compression with HyComp

- Combine various methods together
- HyComp: Select the "best-of-breed" compression method



Compression with HyComp

- **Zero impact on decompression latency!**

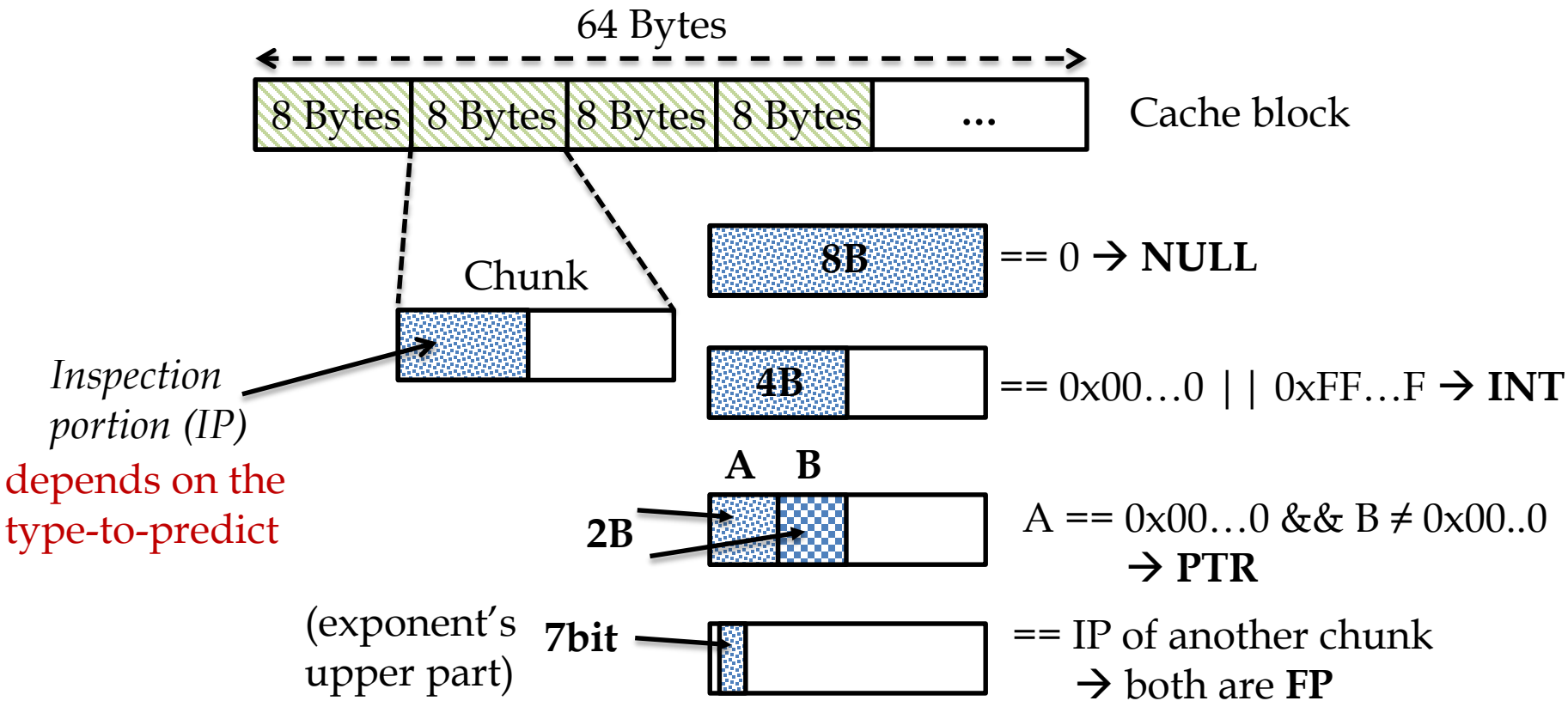


HyComp: Compression

- Two phase approach (heuristic-based):
  1. Block inspection → Prediction of content-types
  2. Method selection and compression
- Supported types:
  - Integer **SC^2 [Arelakis, Stenstrom, ISCA '14]**
  - Pointer **BDI [Pekhimenko et al., PACT' 12]**
  - Floating-point **FP-H [Arelakis et al., discussed later]**
  - Null block **ZCA [Dusser et al., ICS '09]**

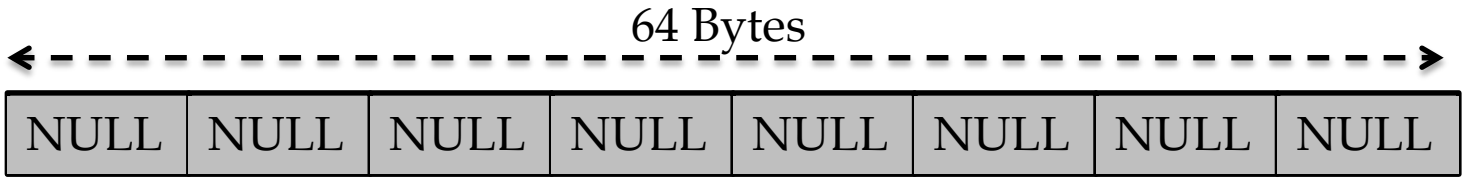
HyComp: Heuristic

- Phase I: Inspection and type-characterization

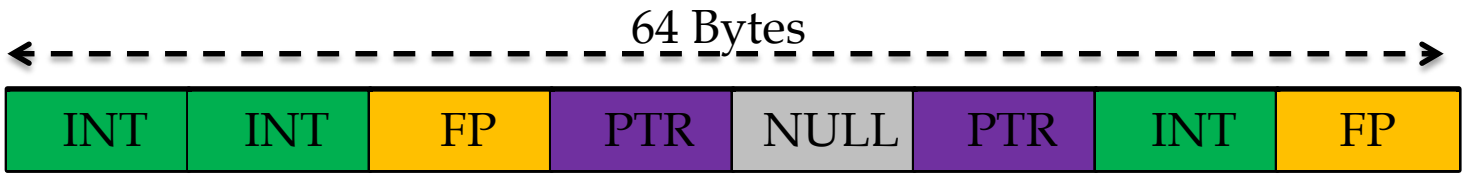


HyComp: Heuristic

- Phase II: Decision and compression



- Compress the block with **ZCA**



- Compress the block with **SC^2**

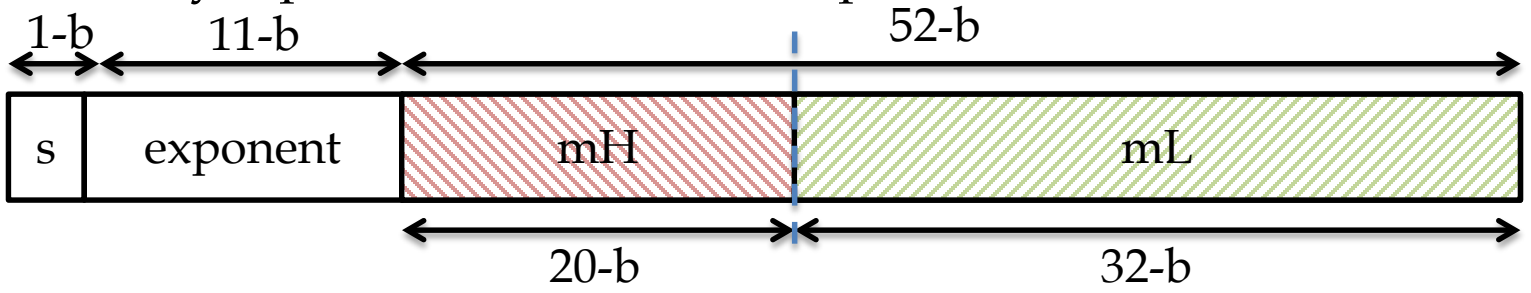
➢ Block's predominant type is the main selection criterion
➢ No known type found → Do not compress (not common)

Outline

- Background
- Motivation
- Contributions
- HyComp: Hybrid compression in cache
- FP-H: Semantic bit-field compression applied to floating-point numbers
- Evaluation
- Conclusions and future work

FP-H: Semantic bit-field compression for Floating-point

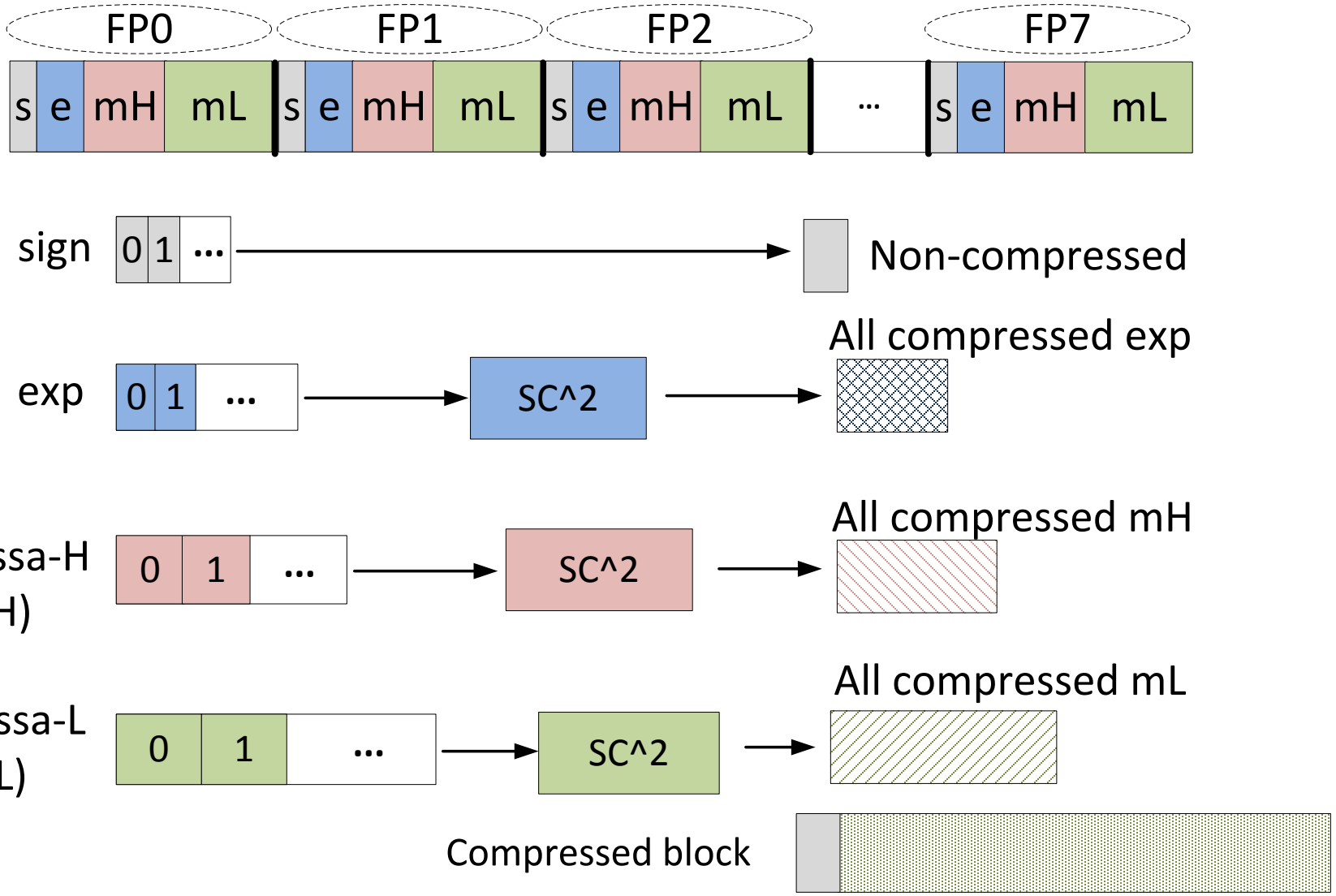
- IEEE-754 standard:
  - Binary representation and 64-bit precision



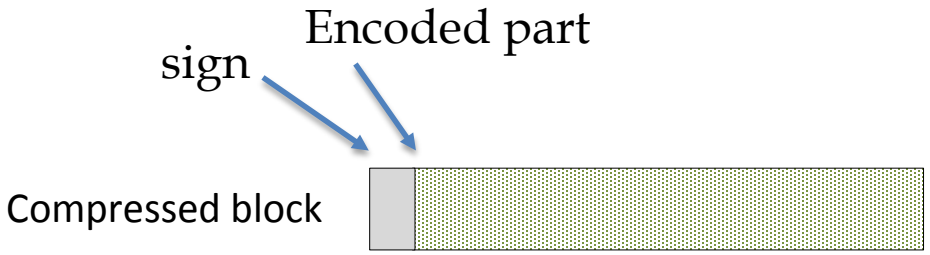
- Prior work: Only exponent exhibits value locality
- **Our observation:** Mantissa exhibits value locality, if partitioned into **≥ 2 subfields**

Idea: compress the semantic bit-fields in isolation using Huffman encoding

FP-H: compression

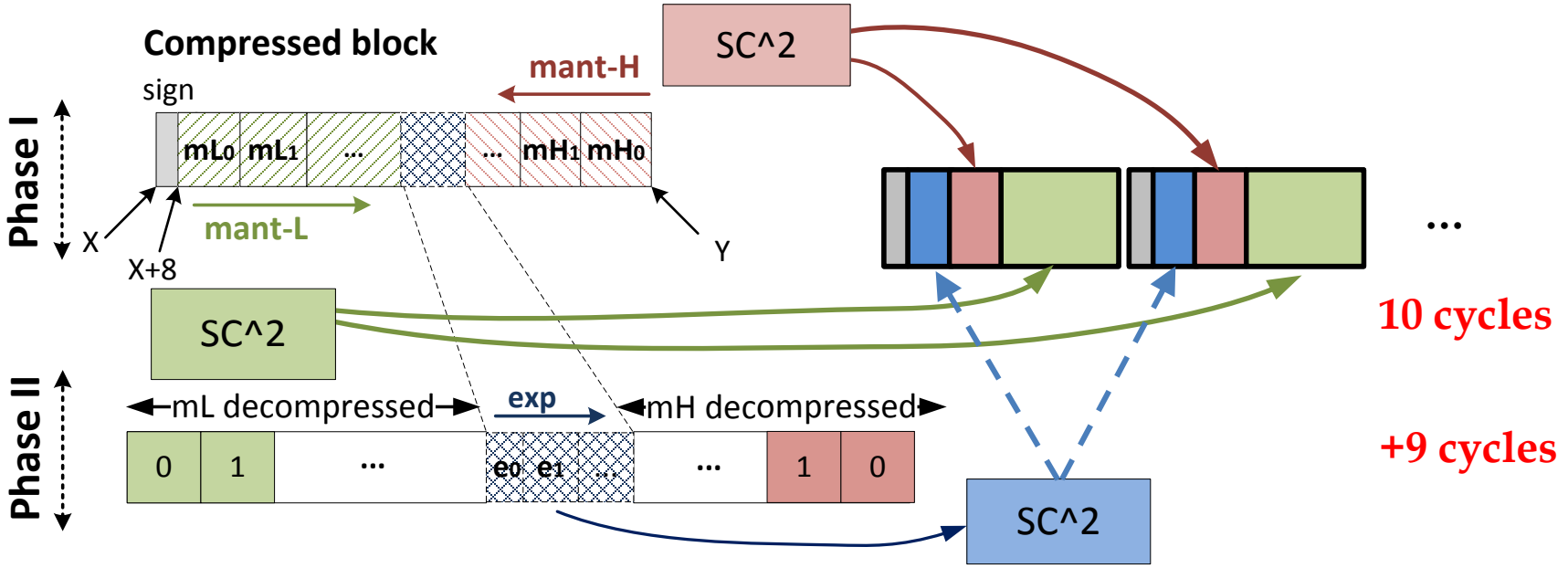


FP-H: Decompression



SC^2: ~10 cycles to decompress 8 values
→ **30 cycles** to decompress exp, mH and mL

Instead: (I) decompress mH and mL in parallel, then (II) decompress exp



Outline

- Background
- Motivation
- Contributions
- HyComp: Hybrid compression in cache
- FP-H: Semantic bit-field compression applied to floating-point numbers
- Evaluation
- Conclusions and future work

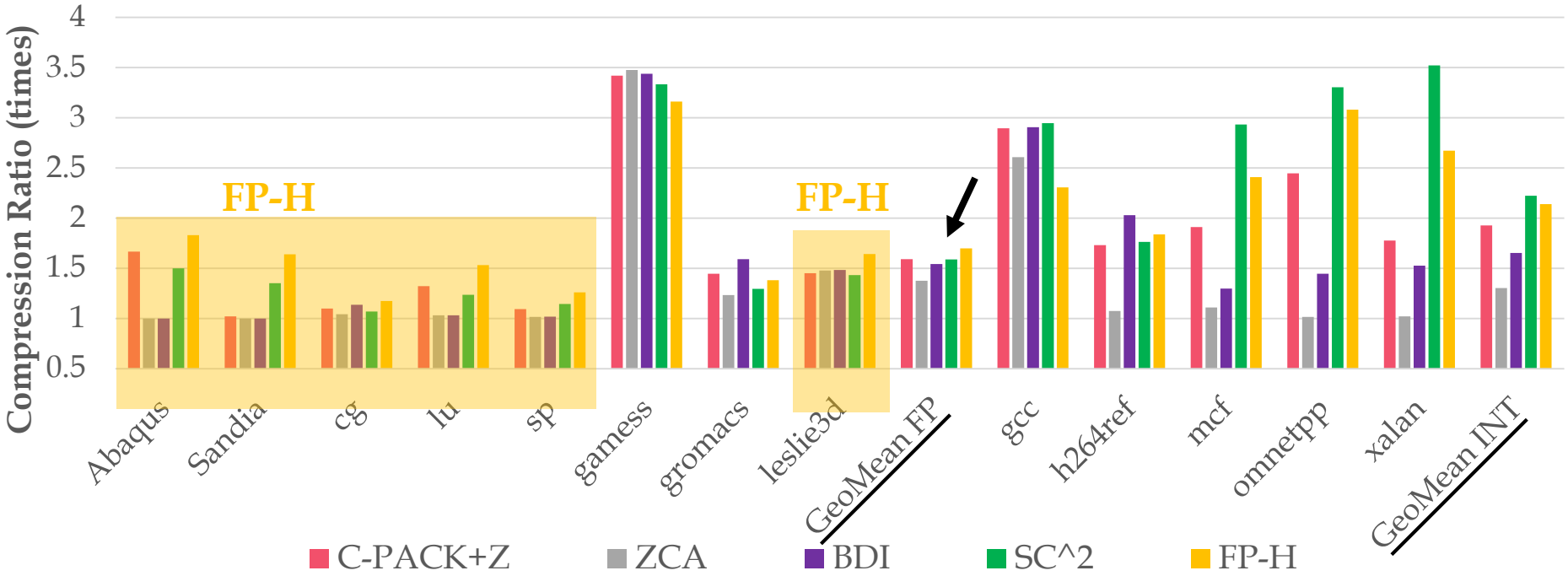
Experimental Setup

- Multicore: 8 cores (3-GHz), X86-64, OOO, 4-way issue
- 3-level cache hierarchy
  - L3 (LLC) shared: 4MB, 8 banks, 20-cycle latency
- HyComp and other schemes → LLC (4x tags)
- GEM5 cycle-accurate simulator
- Area, power, access times (32nm technology):
  - Caches: CACTI 6.5
  - Compression schemes: Synopsys Design Compiler
  - Simpoints: 10-15; Interval=250M inst. each
- Applications:
  - 28 benchmarks: SPEC2006, NAS
  - Multicore mixes of 8 apps

Heuristics: **+2 cycles** in compression **only** → not in the critical path

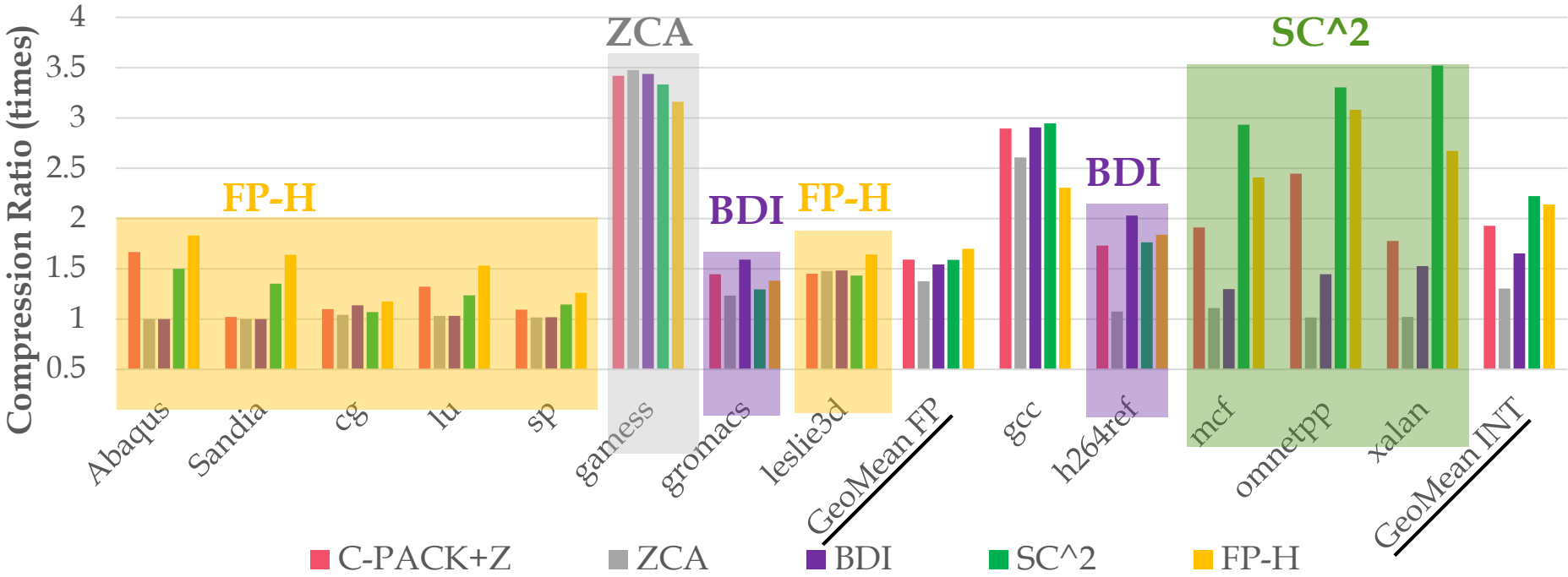
Latency (cycl)	CPACK+Z	ZCA	SC^2	BDI	FP-H	HyComp
Compr.	16	1	9	2	7	2+*
Decompr.	9	0	10	1	20	*

Compressibility



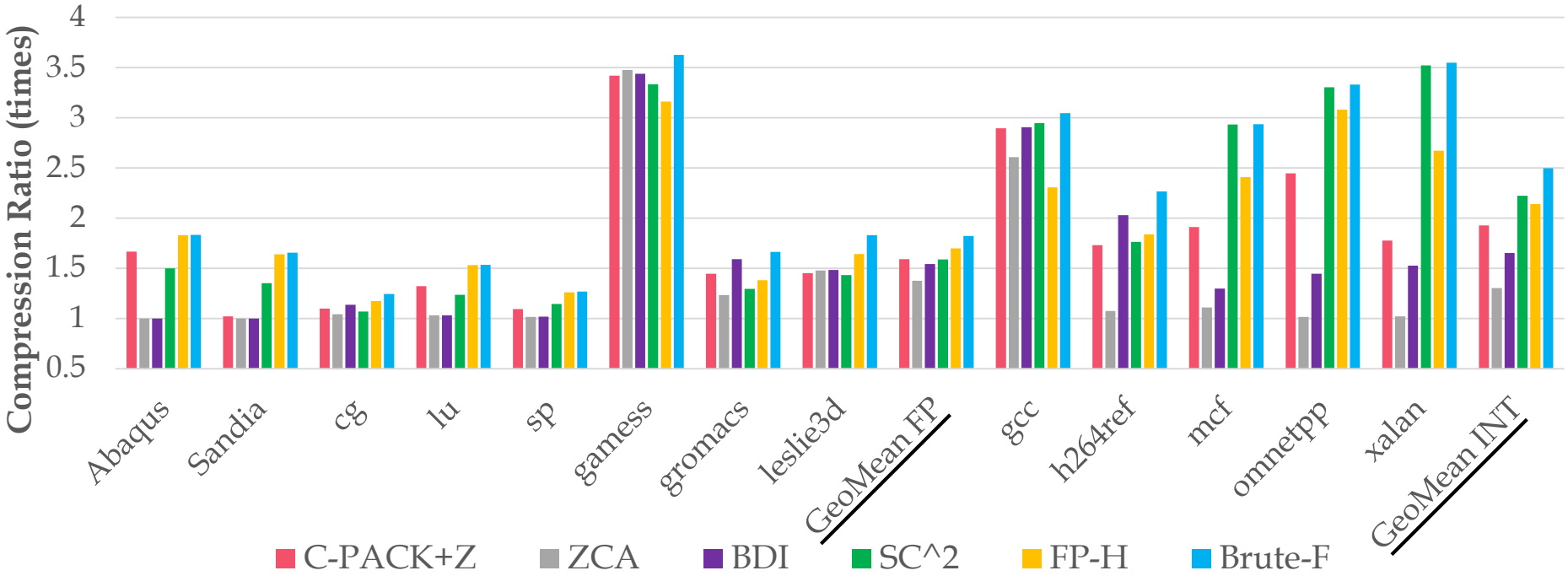
- **FP-H**: higher compression than other standalone compression methods for most FP workloads

Compressibility



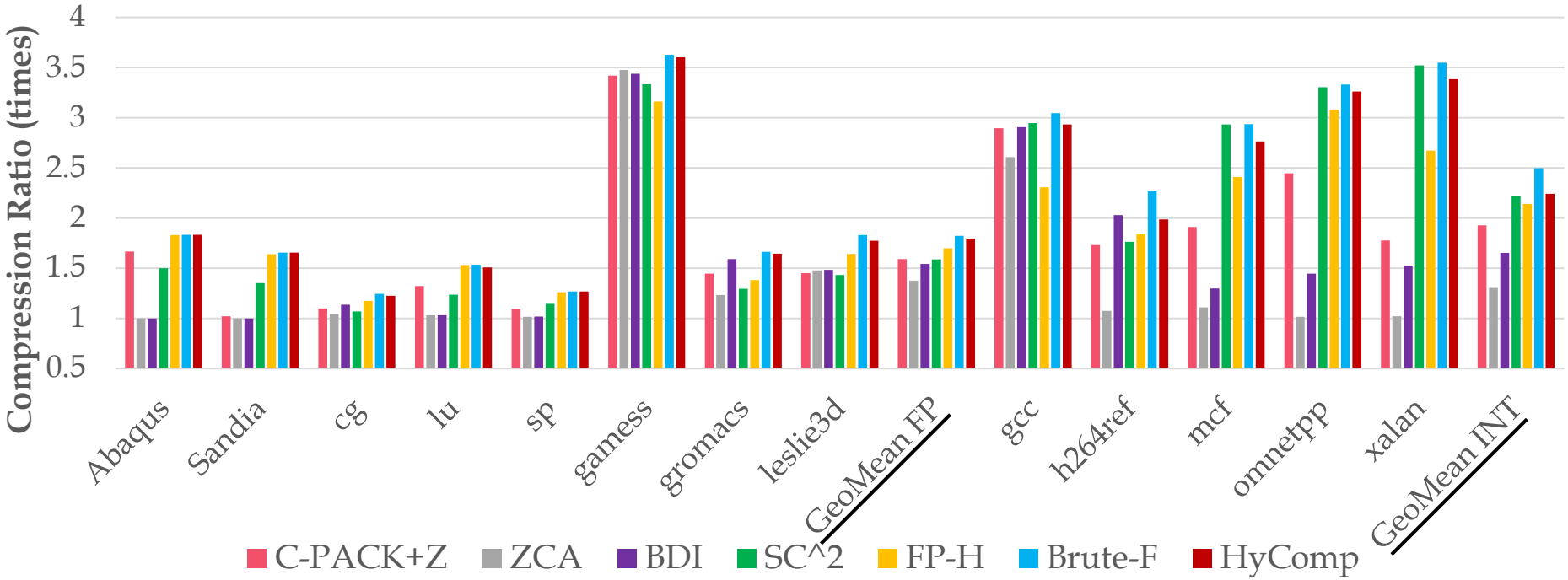
- **FP-H**: higher compression than other standalone compression methods for most FP workloads

Compressibility



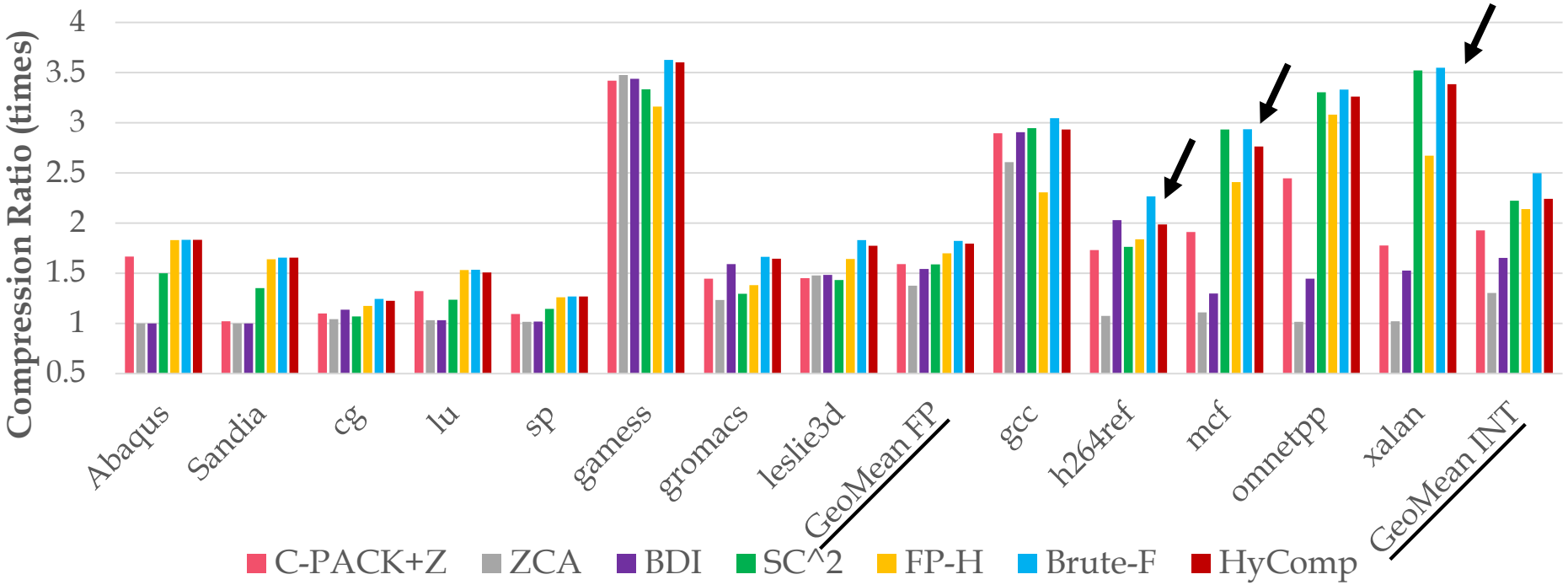
- **FP-H**: higher compression than other standalone compression methods for most FP workloads
- Ideal hybrid (**Brute-F**): always better compression

Compressibility



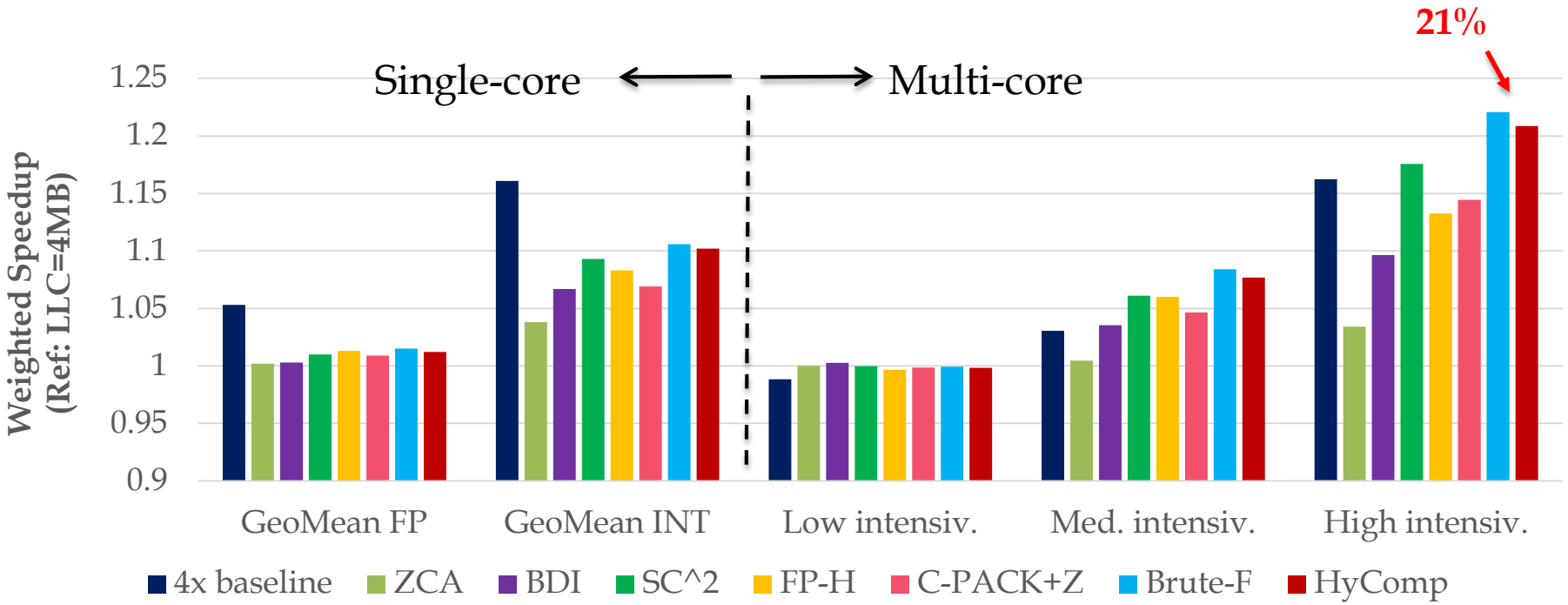
- **FP-H**: higher compression than other standalone compression methods for most FP workloads
- Ideal hybrid (**Brute-F**): always better compression
- **HyComp**: very close to Brute-F

Compressibility



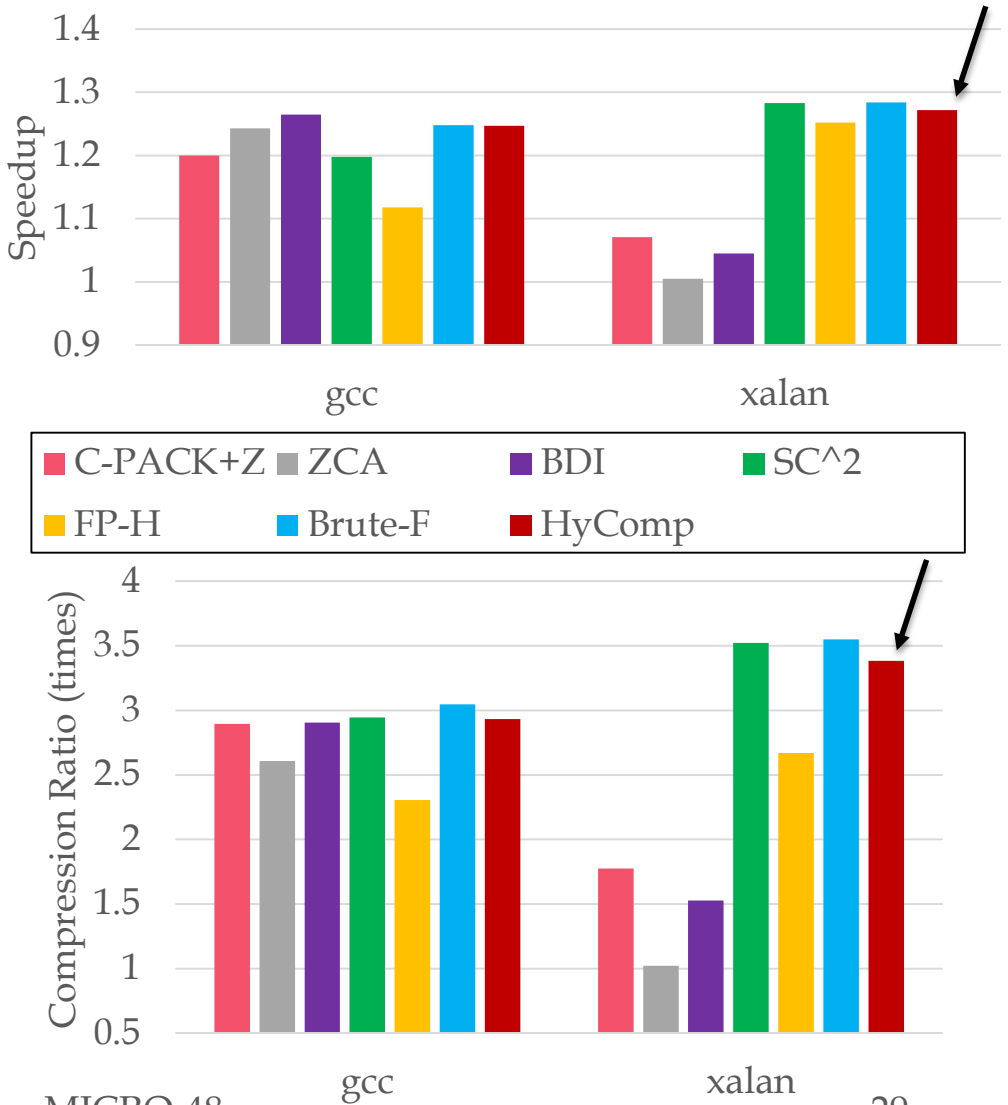
- **FP-H**: higher compression than other standalone compression methods for most FP workloads
- Ideal hybrid (**Brute-F**): always better compression
- **HyComp**: very close to Brute-F
  - Deviation in few cases is attributed to heuristics inaccuracy

Speedup

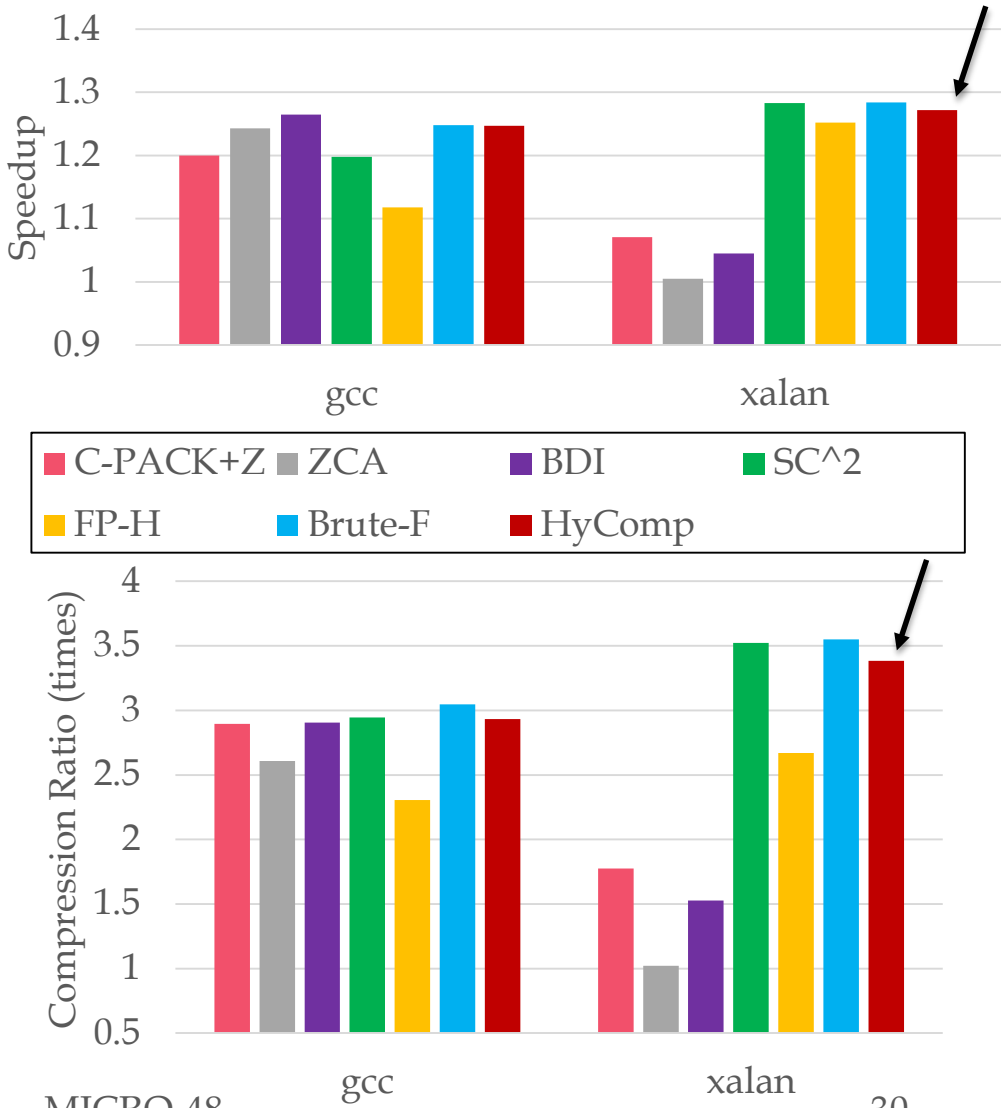


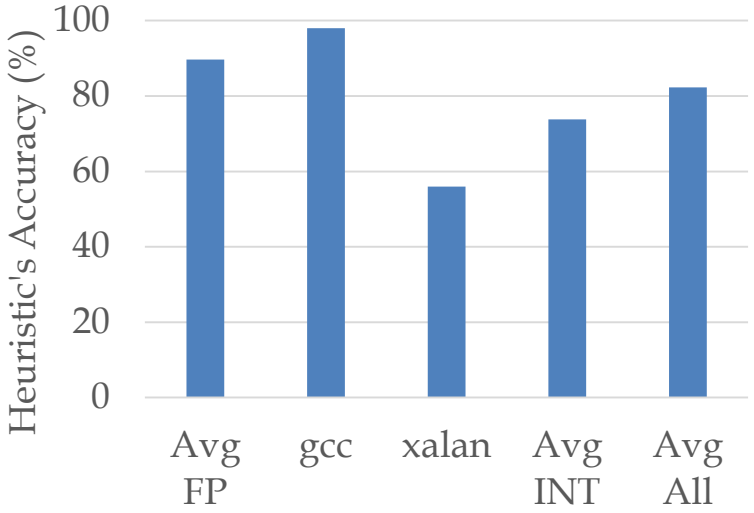
- Multicore: Multiprogram mixes based on cache intensity and compressibility
- HyComp:
  - Consistently higher speedup than individual schemes
  - Close to performance of a 4x larger cache

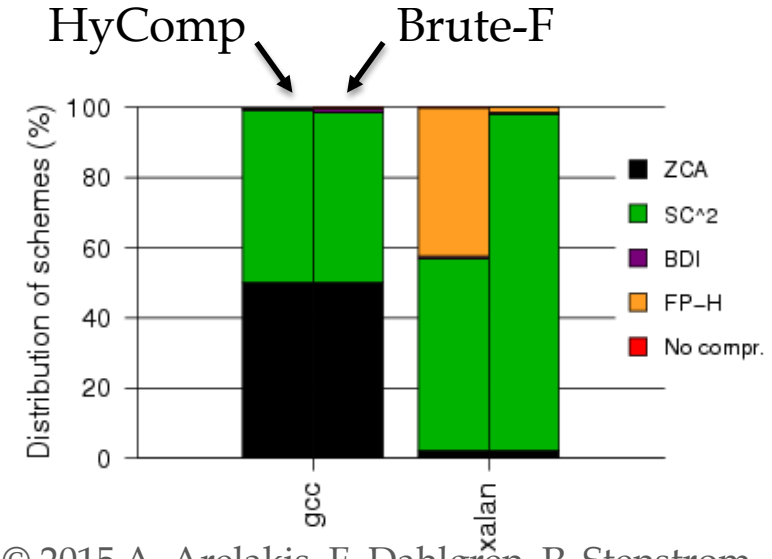
Exceptions



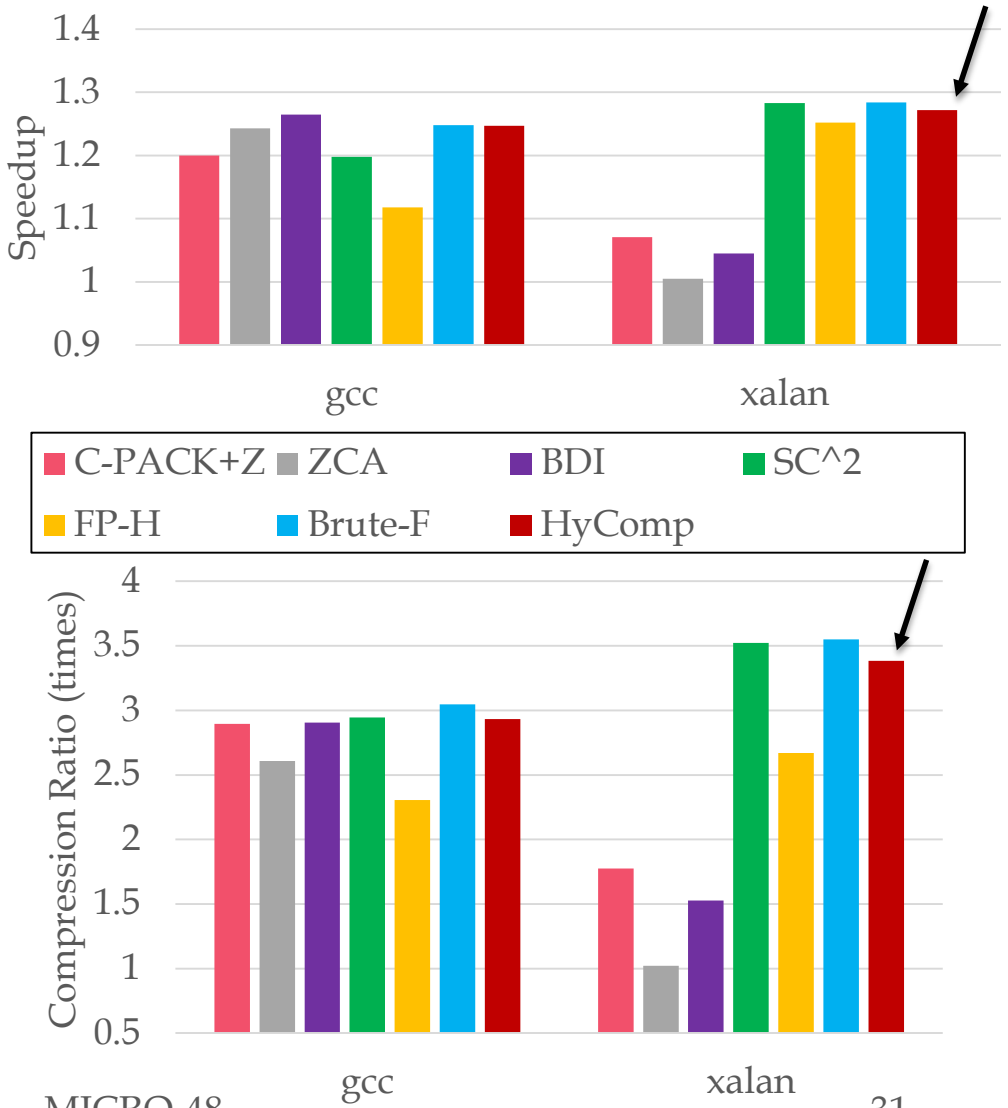
Exceptions

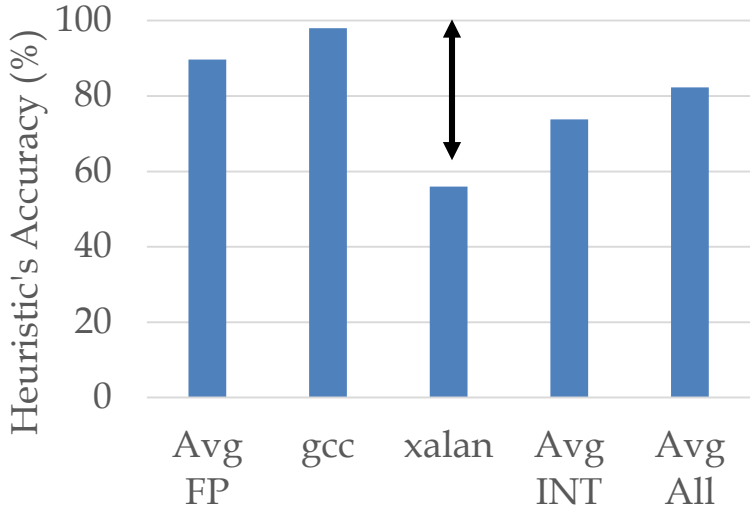


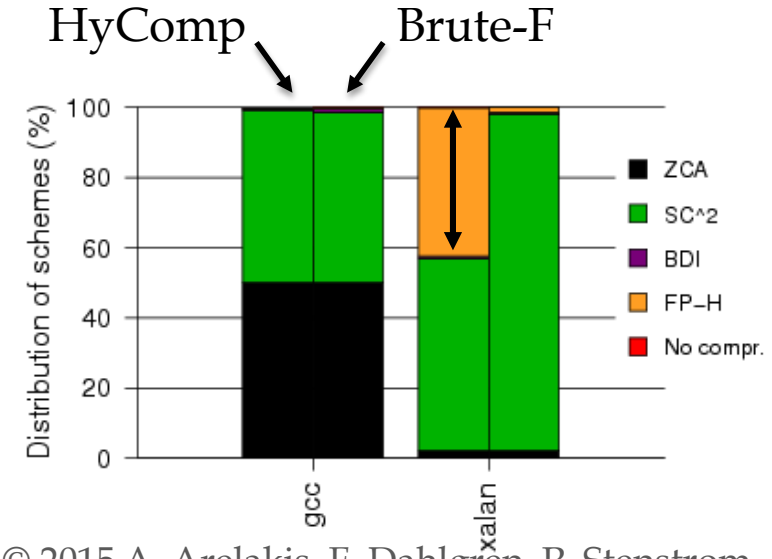




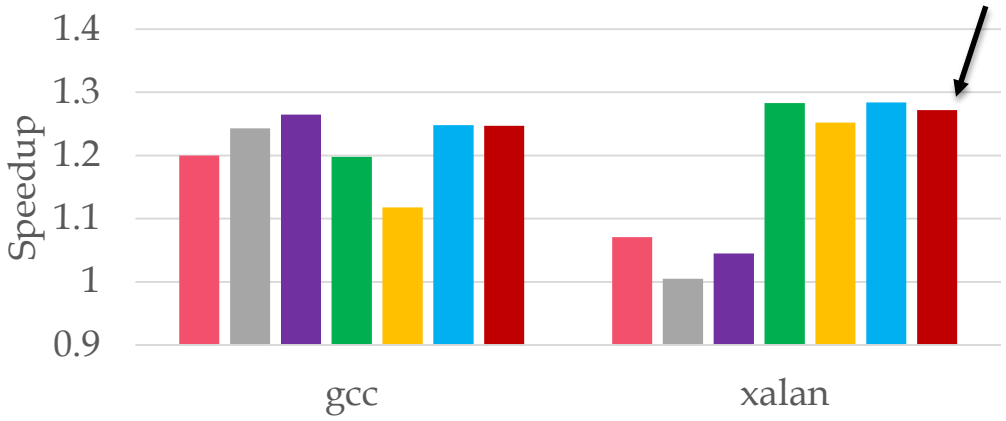
Exceptions

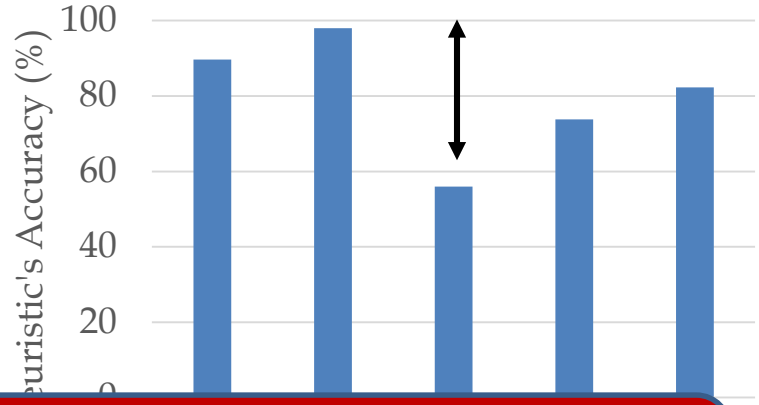




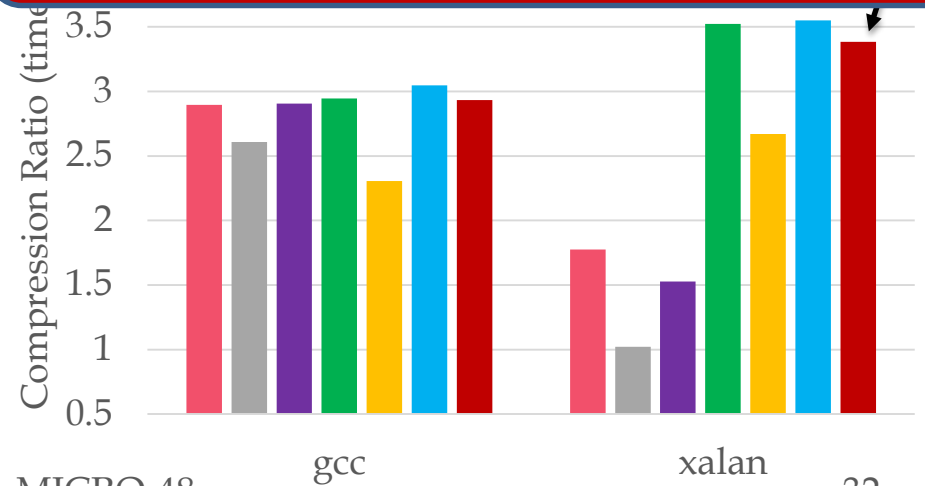


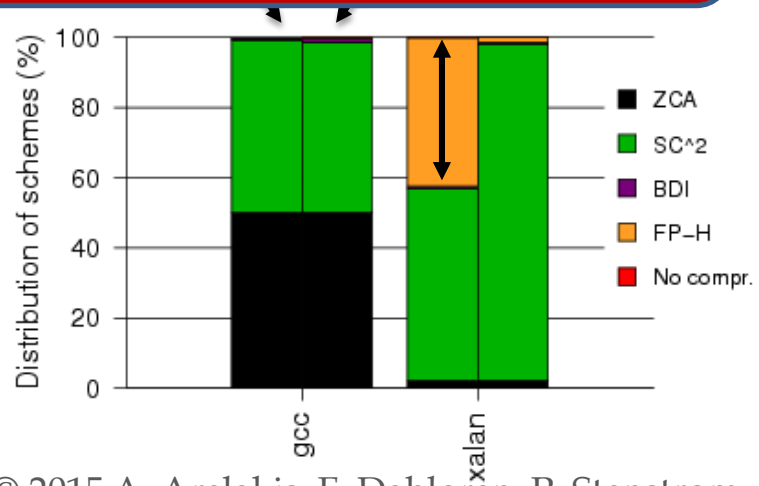
Exceptions



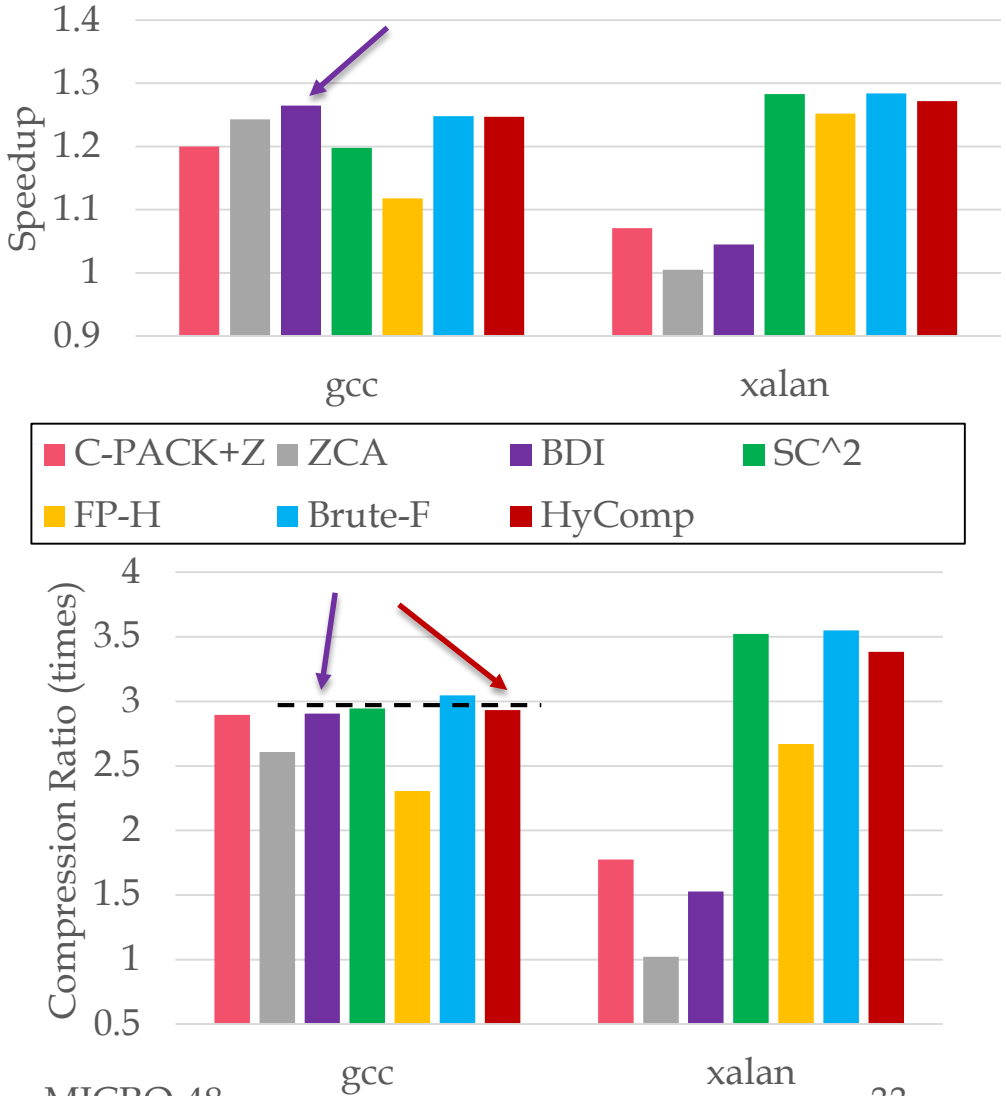


HyComp: Marginally lower speedup than an individual method in few cases (e.g., xalan) due to heuristic inaccuracy

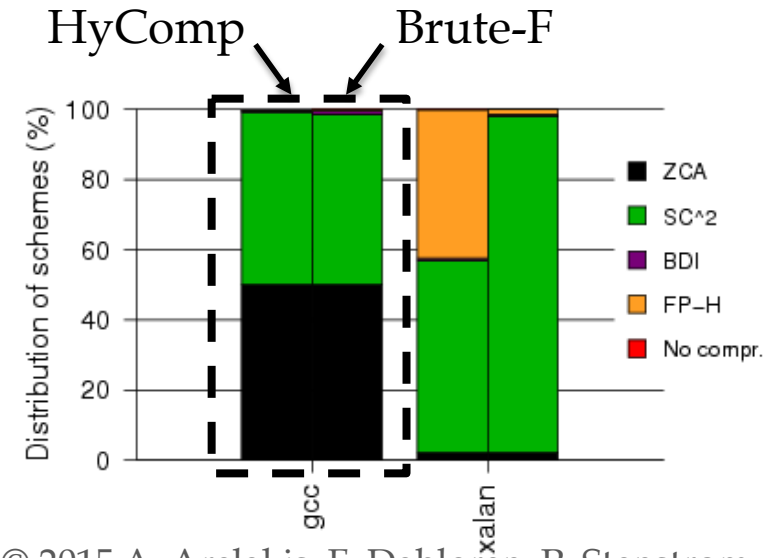




Exceptions



- **gcc**: HyComp correctly picks SC^2, but BDI has better speedup:
  - SC^2: slightly better CR than BDI, but
  - BDI: higher speedup due to smaller decompr. lat



Conclusions

Two contributions:

- HyComp:
  - Robust compression for diverse data-types: each and across applications, with
  - Higher speedup than standalone schemes
  - No impact on decompression latency

- FP-H:
  - FP data: Better compression when semantic information is utilized
  - Mantissa exhibits value locality if partitioned
  - Parallel decompression of partitioned mantissa

Future Work

- Explore semantic bit-field compression in other types
- HyComp:
  - Support other types (e.g., text)
  - Include other decision parameters (e.g., decompression latency)
  - Improve heuristics accuracy