Linearly Compressed Pages: A Main Memory Compression Framework with Low Complexity and Low Latency

Gennady Pekhimenko,
Vivek Seshadri , Yoongu Kim,
Hongyi Xin, Onur Mutlu,
Todd C. Mowry

Phillip B. Gibbons,
Michael A. Kozuch

Carnegie Mellon University
SAFARI

intel

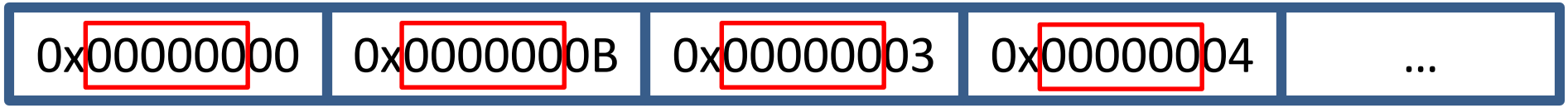
Executive Summary

- Main memory is a limited shared resource
- **Observation**: Significant data redundancy
- **Idea**: Compress data in main memory
- **Problem**: How to avoid inefficiency in address computation?
- **Solution**: Linearly Compressed Pages (LCP): **fixed-size** cache line granularity compression

1. Increases memory capacity (**62%** on average)
2. Decreases memory bandwidth consumption (**24%**)
3. Decreases memory energy consumption (**4.9%**)
4. Improves overall performance (**13.9%**)

Potential for Data Compression

Significant redundancy in in-memory data:



How can we exploit this redundancy?

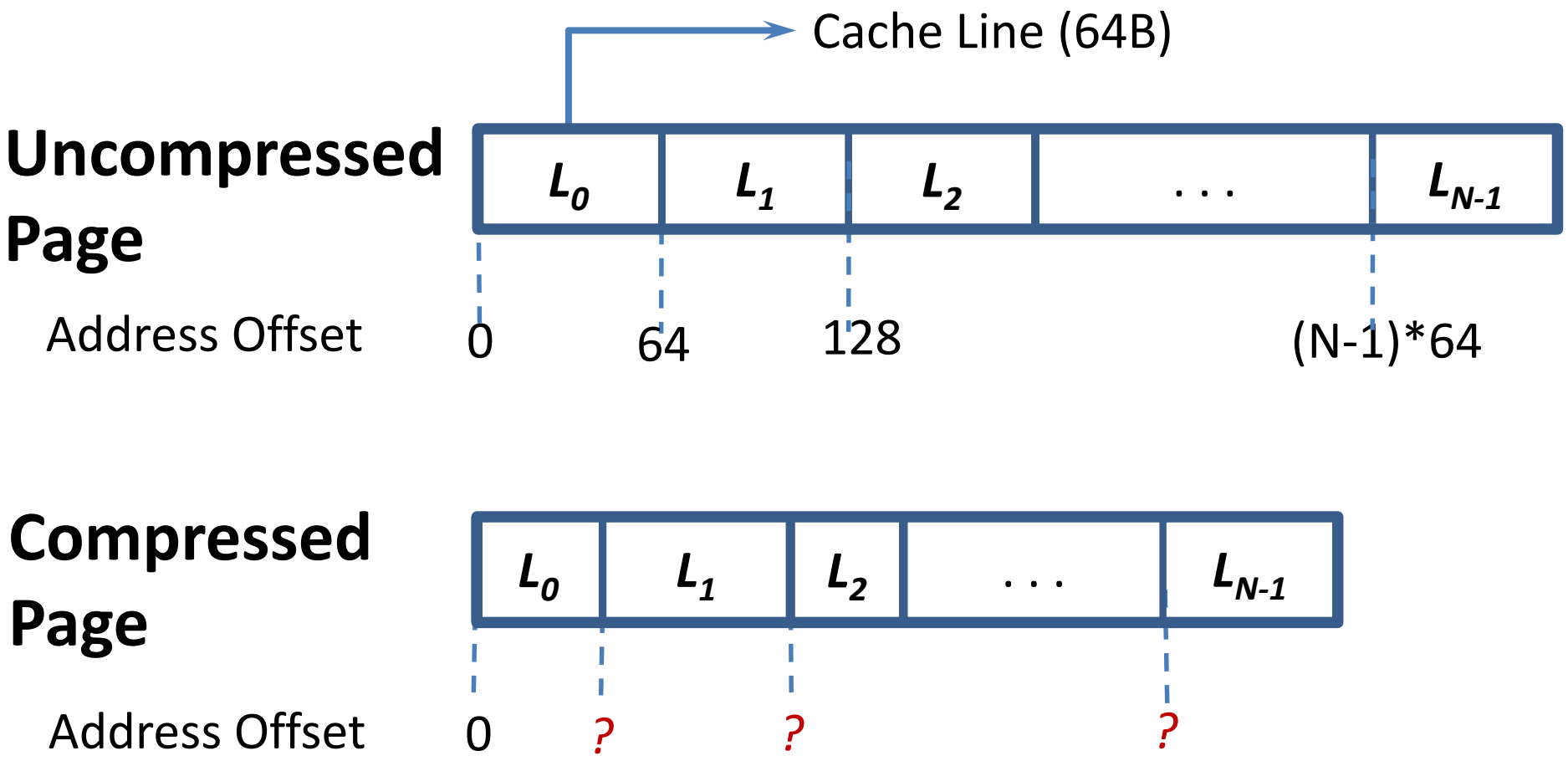
- **Main memory compression** helps
- Provides effect of a larger memory without making it physically larger

Challenges in Main Memory Compression

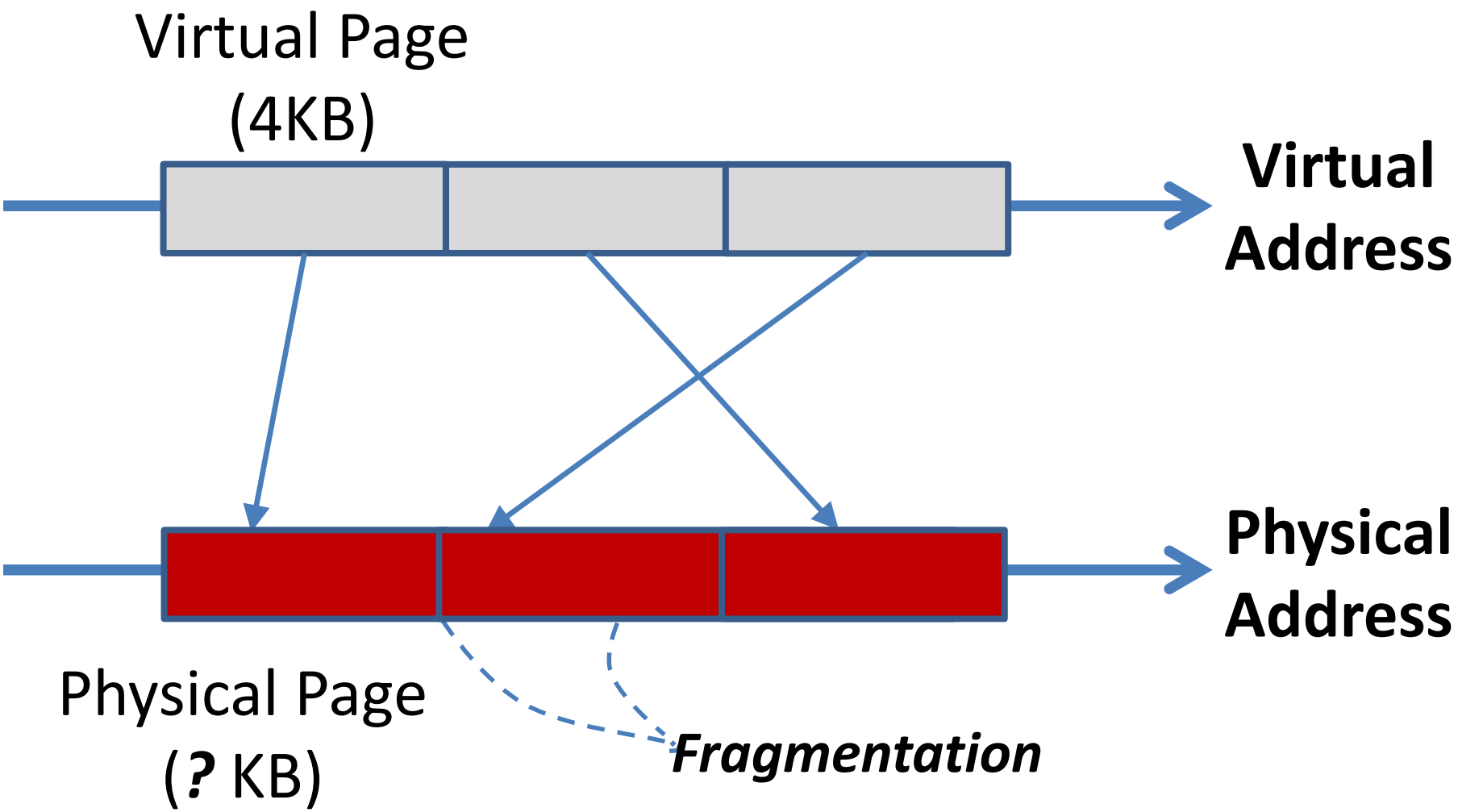
1. Address Computation

2. Mapping and Fragmentation

Challenge 1: Address Computation



Challenge 2: Mapping & Fragmentation



Outline

- Motivation & Challenges
- Shortcomings of Prior Work
- LCP: Key Idea
- LCP: Implementation
- Evaluation
- Conclusion and Future Work

Key Parameters in Memory Compression

Compression Ratio	Address Comp. Latency	Decompression Latency	Complexity and Cost

Shortcomings of Prior Work

Compression Mechanisms	Compression Ratio	Address Comp. Latency	Decompression Latency	Complexity and Cost
IBM MXT *[IBM J.R.D. '01]*	✓✓	✗ 2X	✗ 64 cycles	✗

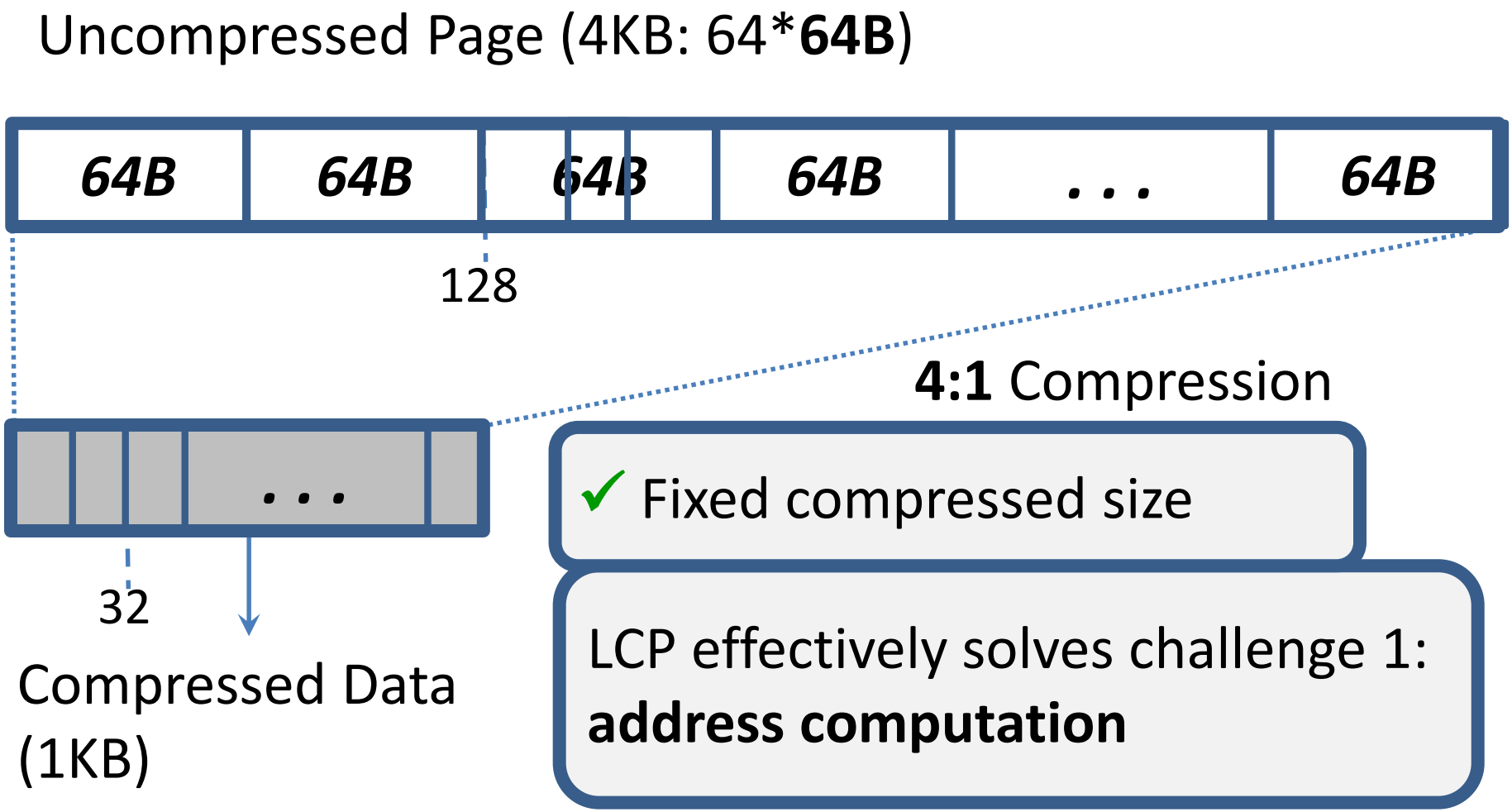
Shortcomings of Prior Work (2)

Compression Mechanisms	Compression Ratio	Address Comp. Latency	Decompression Latency	Complexity And Cost
IBM MXT *[IBM J.R.D. '01]*	✓✓	✗	✗	✗
Robust Main Memory Compression *[ISCA'05]*	✓	✗	✓	✗

Shortcomings of Prior Work (3)

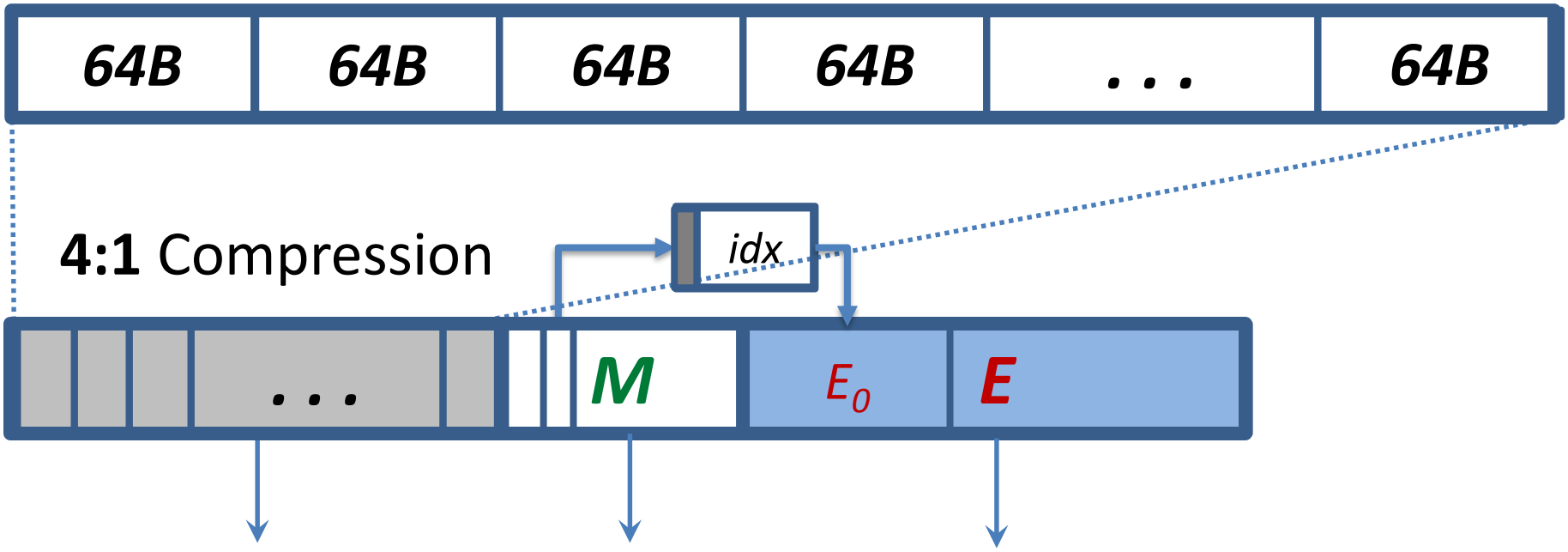
Compression Mechanisms	Compression Ratio	Address Comp. Latency	Decompression Latency	Complexity And Cost
IBM MXT *[IBM J.R.D. '01]*	✓✓	✗	✗	✗
Robust Main Memory Compression *[ISCA'05]*	✓	✗	✓	✗
LCP: Our Proposal	✓	✓	✓	✓

Linearly Compressed Pages (LCP): Key Idea



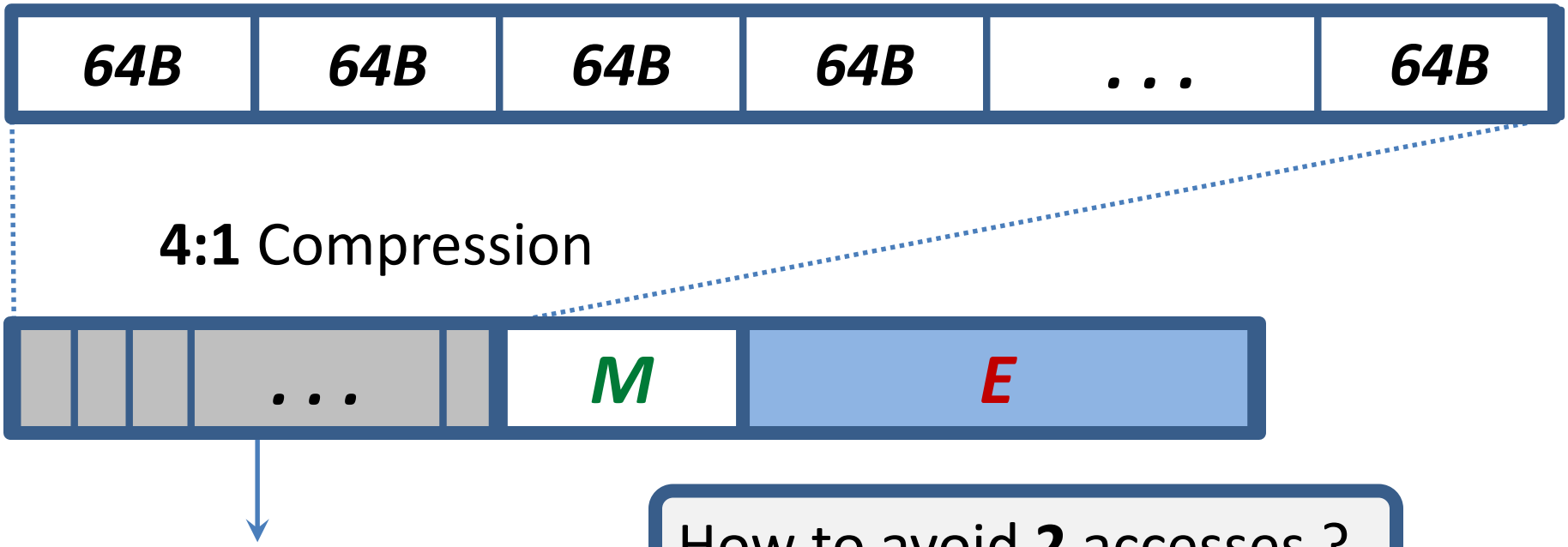
LCP: Key Idea (2)

Uncompressed Page (4KB: 64*__64B__)



But, wait …

Uncompressed Page (4KB: 64*__64B__)



How to avoid **2** accesses ?

✓ Metadata (MD) cache

Key Ideas: Summary

✓ Fixed compressed size per cache line

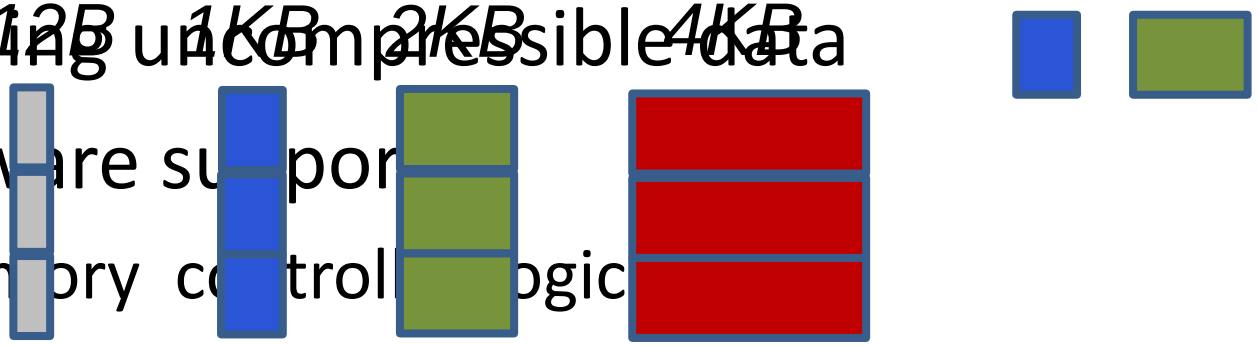
✓ Metadata (MD) cache

Outline

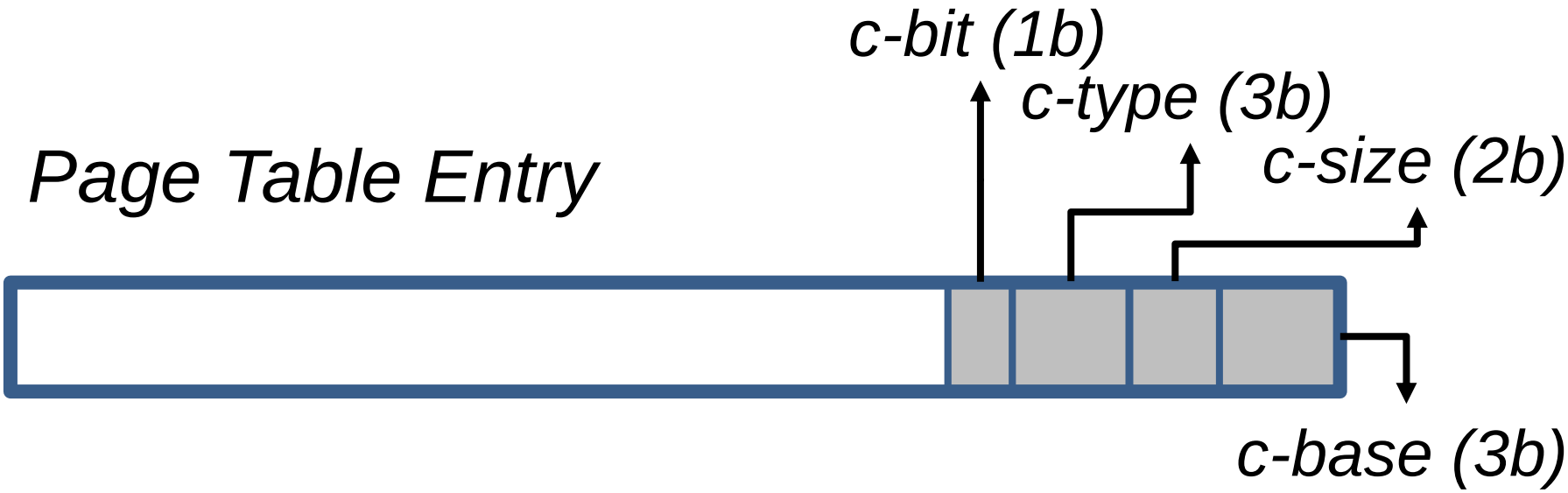
- Motivation & Challenges
- Shortcomings of Prior Work
- LCP: Key Idea
- LCP: Implementation
- Evaluation
- Conclusion and Future Work

LCP Overview

- Page Table entry extension
  - compression type and size (fixed encoding)
- OS support for multiple page sizes
  - **4** memory pools (512B, 1KB, 2KB, 4KB)
- Handling uncompressible data
- Hardware support
  - memory controller logic
  - **metadata (MD) cache**

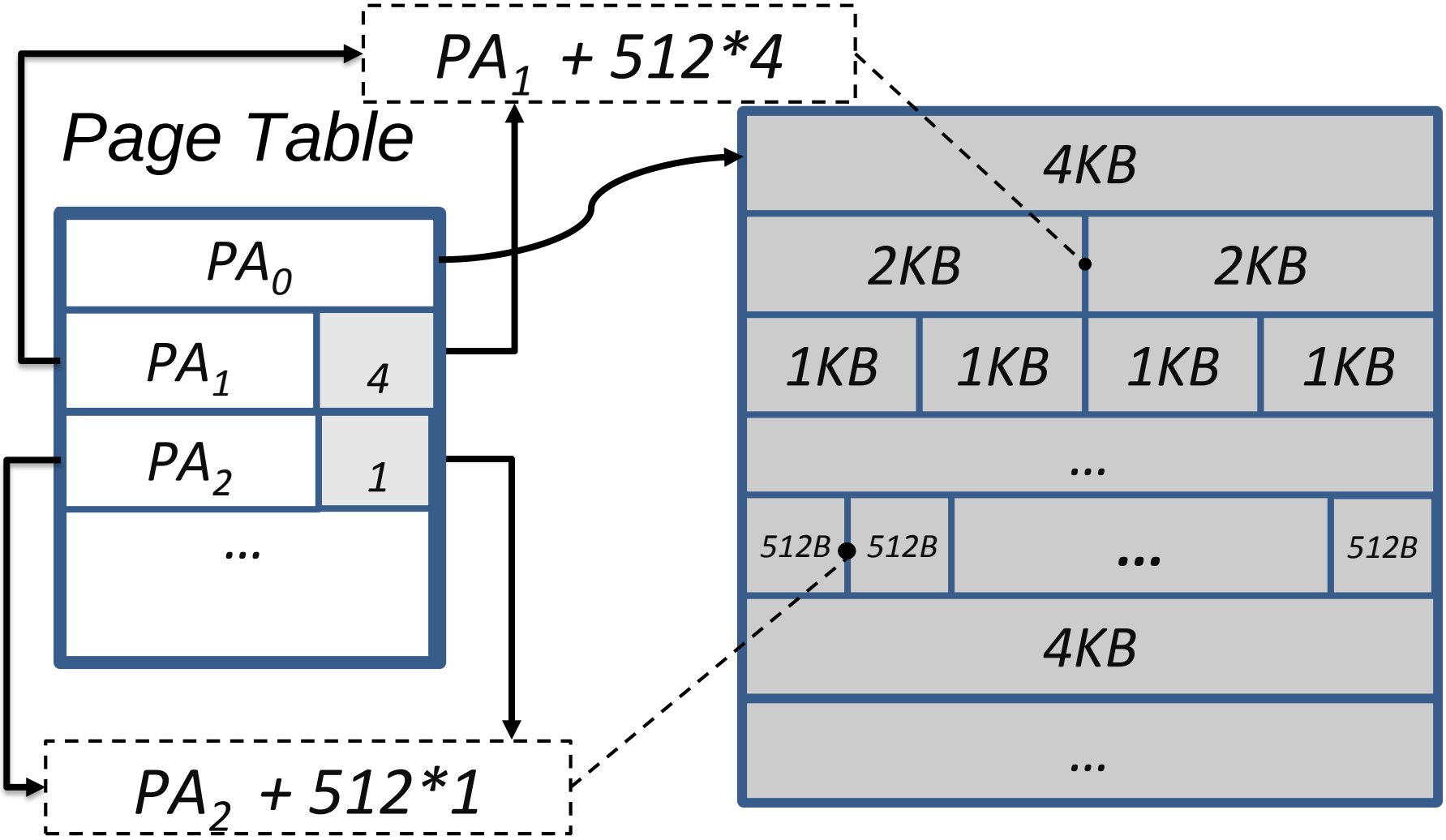


Page Table Entry Extension



- *c-bit (1b)* – compressed or uncompressed page
- *c-type (3b)* – compression encoding used
- *c-size (2b)* – LCP size (e.g., 1KB)
- *c-base (3b)* – offset within a page

Physical Memory Layout



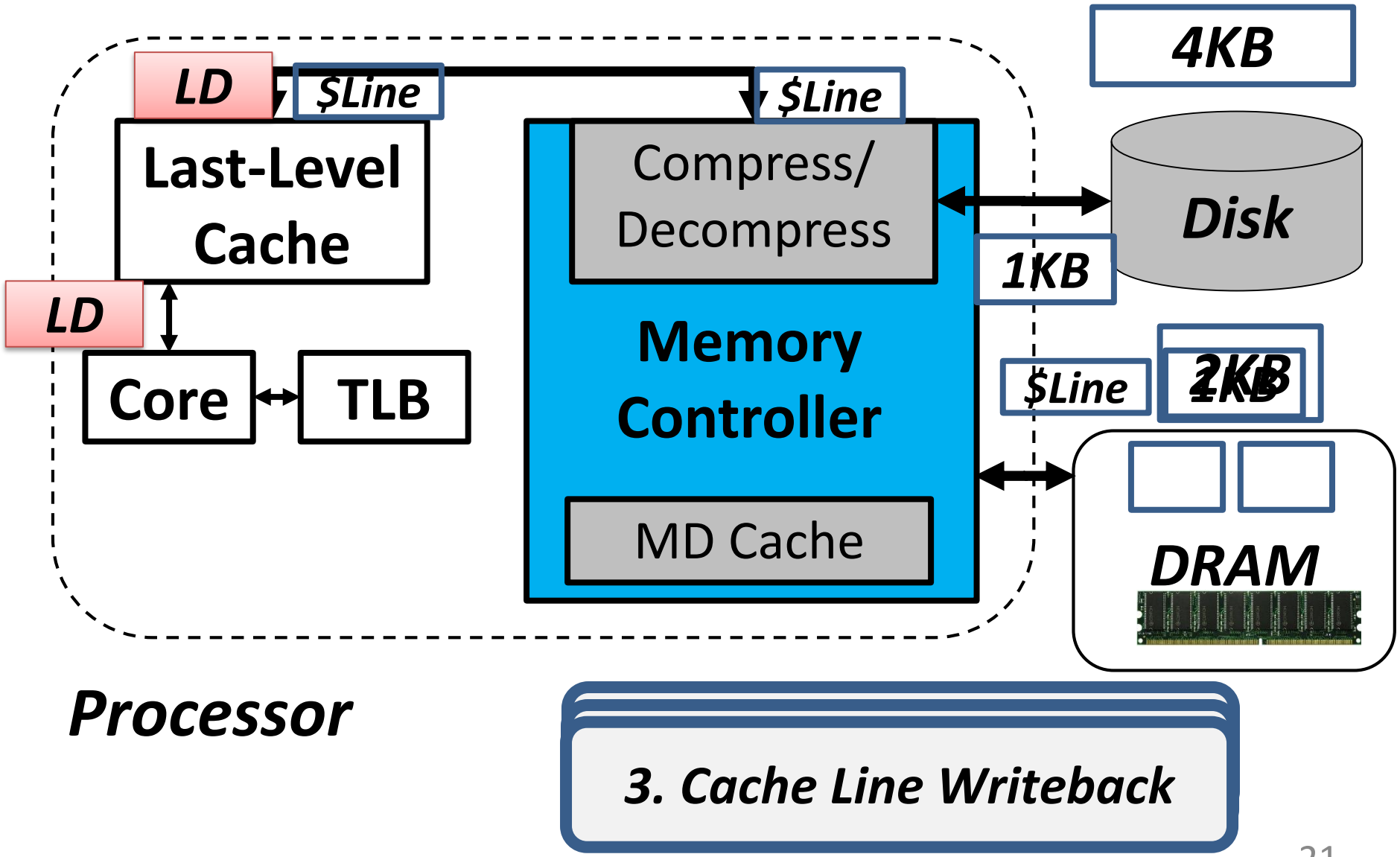
Memory Request Flow

1. *Initial Page Compression*

2. *Cache Line Read*

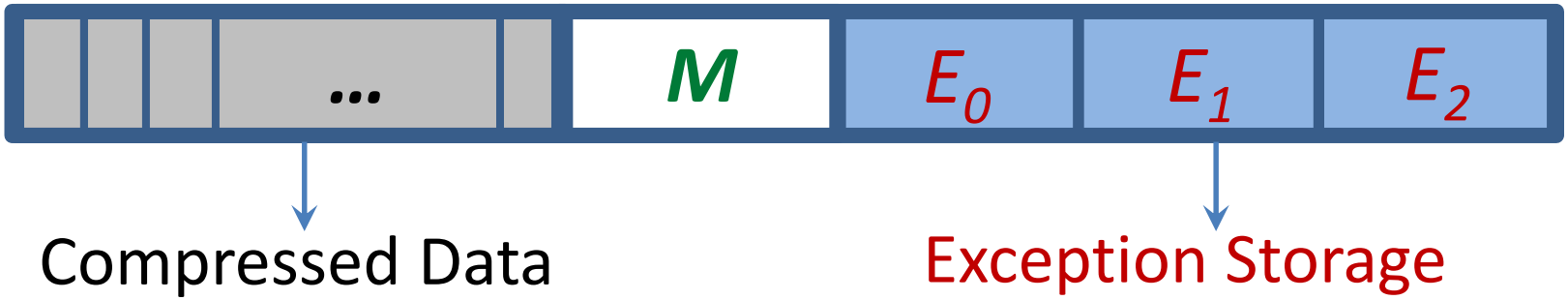
3. *Cache Line Writeback*

Main Memory Compression (1/3)



Handling Page Overflows

- Happens after writebacks, when all slots in the exception storage are already taken



- Two possible scenarios:
  - Type-1 overflow: requires allocating a new physical page (e.g., 4KB page size)
  - Type-2 overflow: requires decompression and full uncompressed physical page (e.g., 4KB)

$ line

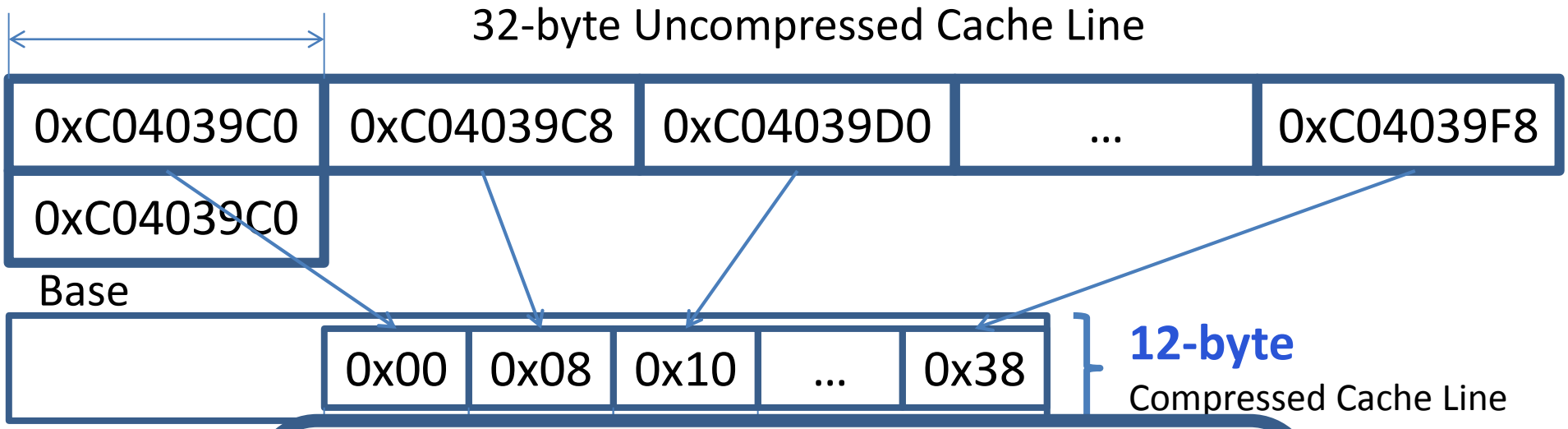
Happens infrequently - once per ~2M instructions

Compression Algorithms

- Key requirements:
  - Low hardware complexity
  - Low decompression latency
  - High effective compression ratio

- Frequent Pattern Compression *[ISCA'04]*
  - Uses simplified dictionary-based compression
- Base-Delta-Immediate Compression *[PACT'12]*
  - Uses low-dynamic range in the data

Base-Delta Encoding *[PACT'12]*



BDI *[PACT'12]* **has two bases:**

1. **zero base (for narrow values)**
2. **arbitrary base (first non-zero value in the cache line)**

✓ Fas … e:

ve … mparison

✓ Effective: good compression ratio

LCP-Enabled Optimizations

- Memory bandwidth reduction:

1 transfer
instead of 4

- Zero pages and zero cache lines
  - Handled separately in TLB (1-bit) and in metadata (1-bit per cache line)

Outline

- Motivation & Challenges
- Shortcomings of Prior Work
- LCP: Key Idea
- LCP: Implementation
- Evaluation
- Conclusion and Future Work

Methodology

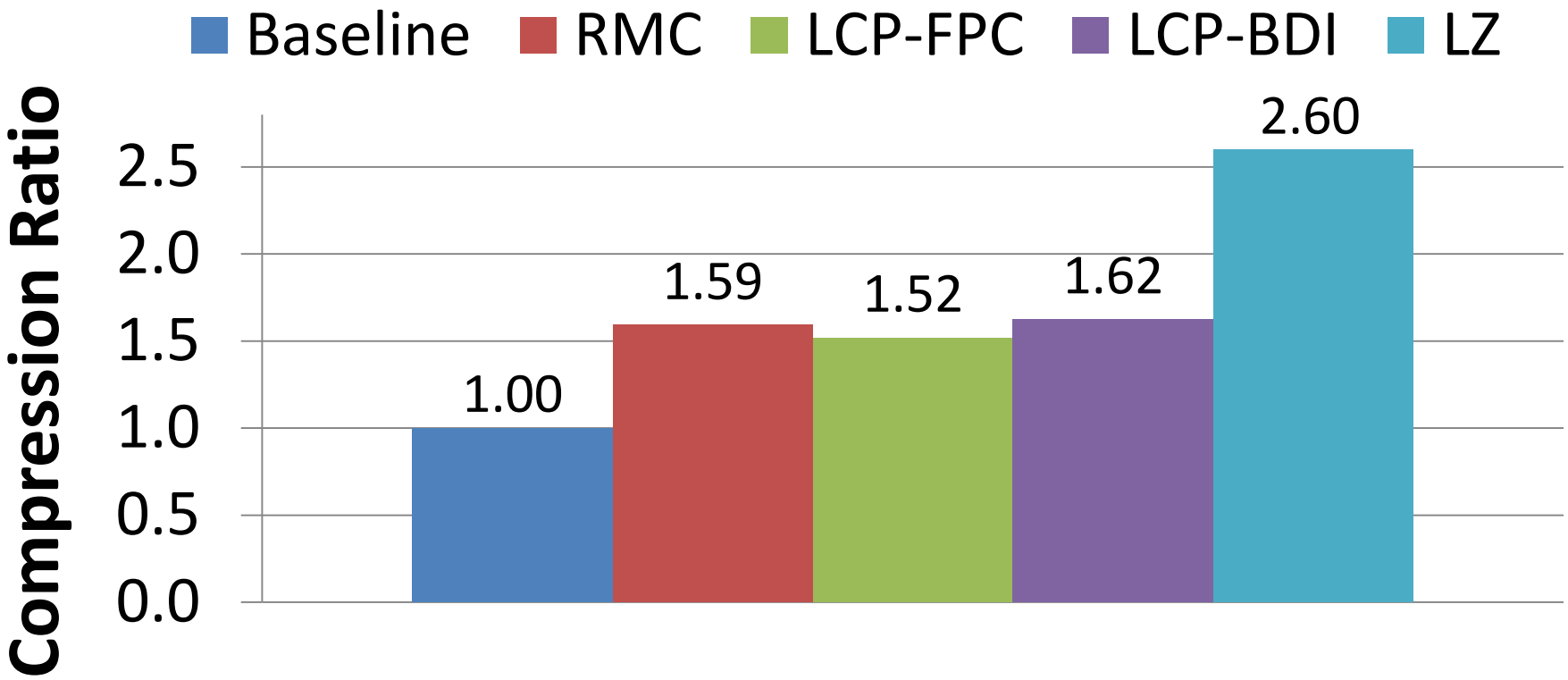
- Simulator: x86 event-driven based on Simics
- Workloads (32 applications)
  - SPEC2006 benchmarks, TPC, Apache web server
- System Parameters
  - L1/L2/L3 cache latencies from CACTI *[Thoziyoor+, ISCA'08]*
  - 512kB - 16MB L2 caches
  - DDR3-1066, 1 memory channel
- Metrics
  - Performance: Instructions per cycle, weighted speedup
  - Capacity: Effective compression ratio
  - Bandwidth: Bytes per kilo-instruction (BPKI)
  - Energy: Memory subsystem energy

Evaluated Designs

Design	Description
Baseline	Baseline (no compression)
RMC	Robust main memory compression *[ISCA'05]* (RMC) and FPC *[ISCA'04]*
LCP-FPC	LCP framework with FPC
LCP-BDI	LCP framework with BDI *[PACT'12]*
LZ	Lempel-Ziv compression (per page)

Effect on Memory Capacity

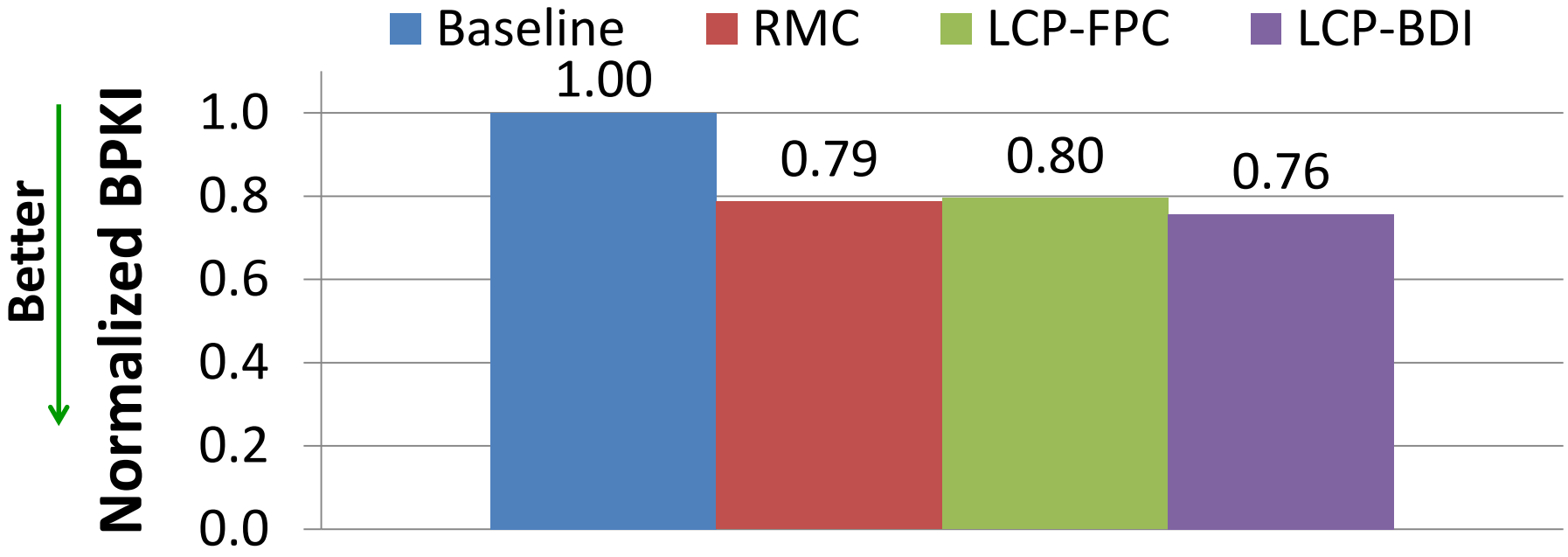
32 SPEC2006, databases, web workloads, 2MB L2 cache



LCP-based designs achieve competitive average compression ratios with prior work

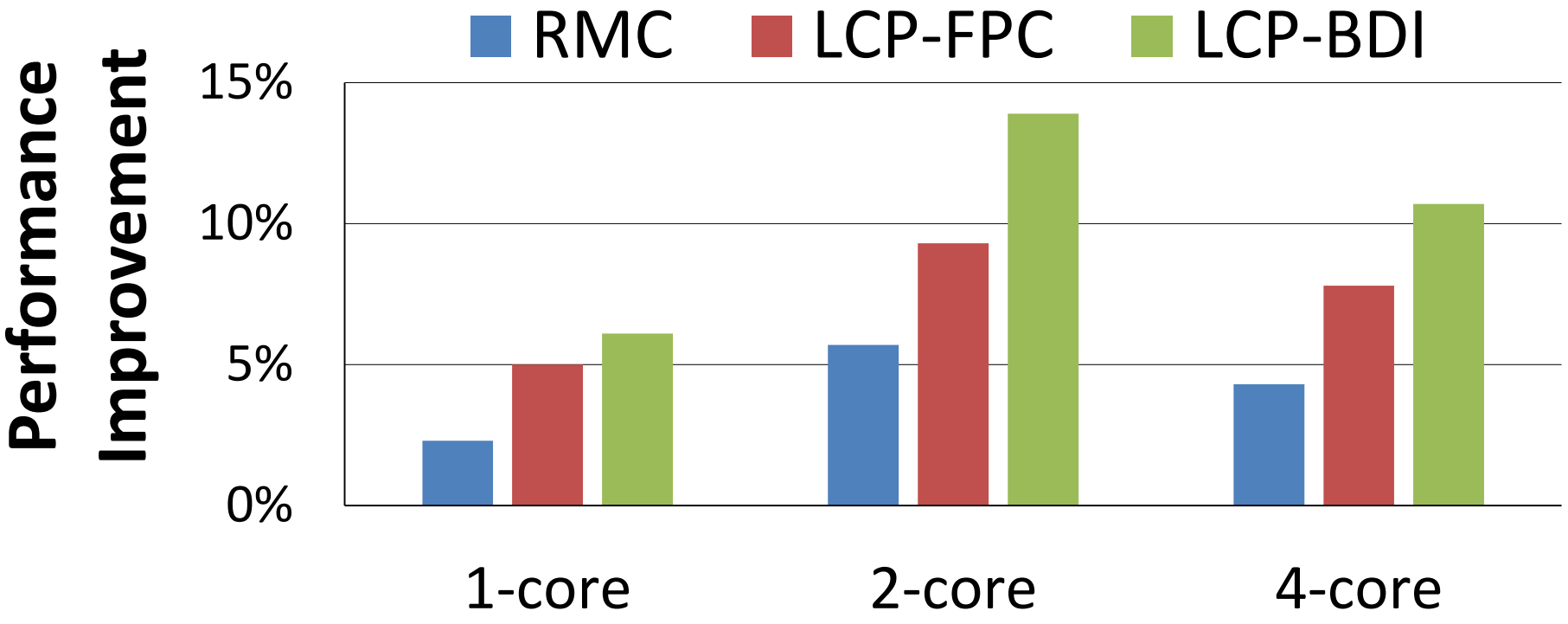
Effect on Bus Bandwidth

32 SPEC2006, databases, web workloads, 2MB L2 cache



LCP-based designs significantly reduce bandwidth (**24**%) (due to data compression)

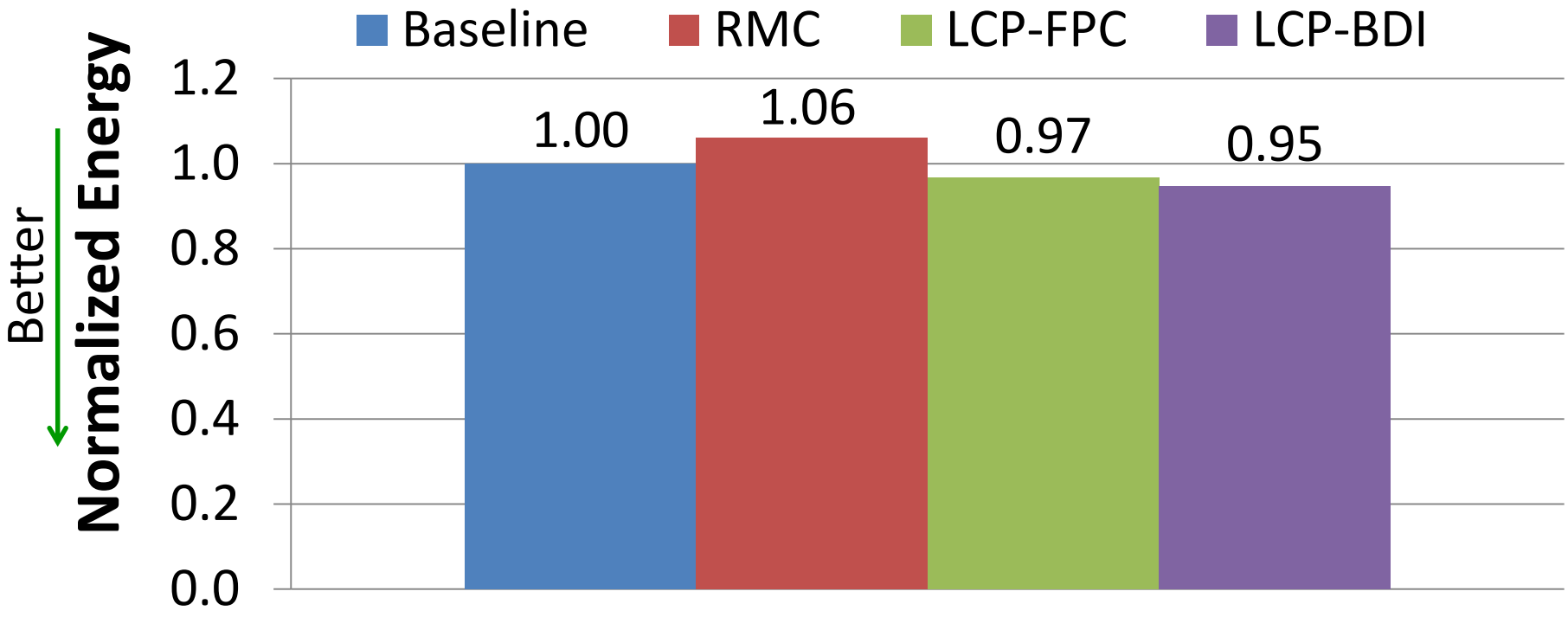
Effect on Performance



LCP-based designs significantly improve performance over RMC

Effect on Memory Subsystem Energy

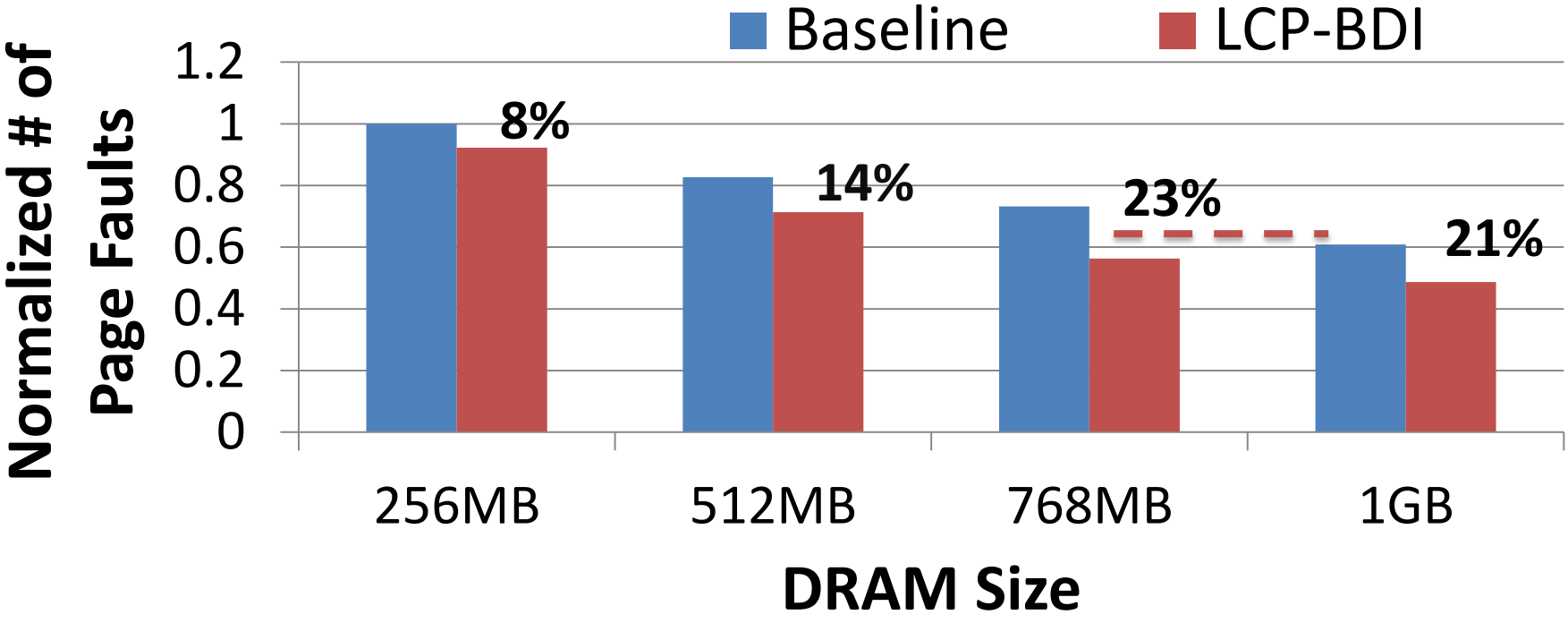
32 SPEC2006, databases, web workloads, 2MB L2 cache



LCP framework is more energy efficient than RMC

Effect on Page Faults

32 SPEC2006, databases, web workloads, 2MB L2 cache



LCP framework significantly decreases the number of page faults (up to **23%** on average for 768MB)

Other Results and Analyses in the Paper

- Analysis of page overflows
- Compressed page size distribution
- Compression ratio over time
- Number of exceptions (per page)
- Detailed single-/multicore evaluation
- Comparison with stride prefetching
  - performance and bandwidth

Conclusion

- Old Idea: Compress data in main memory
- **Problem**: How to avoid inefficiency in address computation?
- **Solution**: A new main memory compression framework called **LCP (Linearly Compressed Pages)**
  - **Key idea: fixed-size** for compressed cache lines within a page
- Evaluation:
  1. Increases memory capacity (**62%** on average)
  2. Decreases bandwidth consumption (**24%**)
  3. Decreases memory energy consumption (**4.9%**)
  4. Improves overall performance (**13.9%**)

Linearly Compressed Pages: A Main Memory Compression Framework with Low Complexity and Low Latency

Gennady Pekhimenko,
Vivek Seshadri , Yoongu Kim,
Hongyi Xin, Onur Mutlu,
Todd C. Mowry

Phillip B. Gibbons,
Michael A. Kozuch

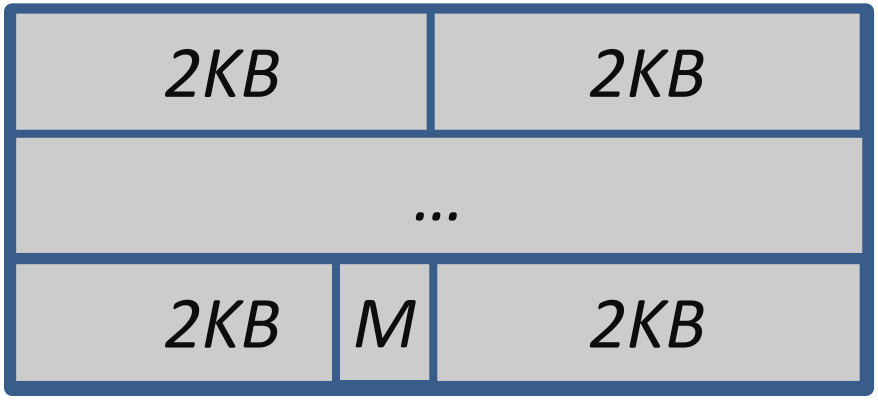
Carnegie Mellon University
SAFARI

intel

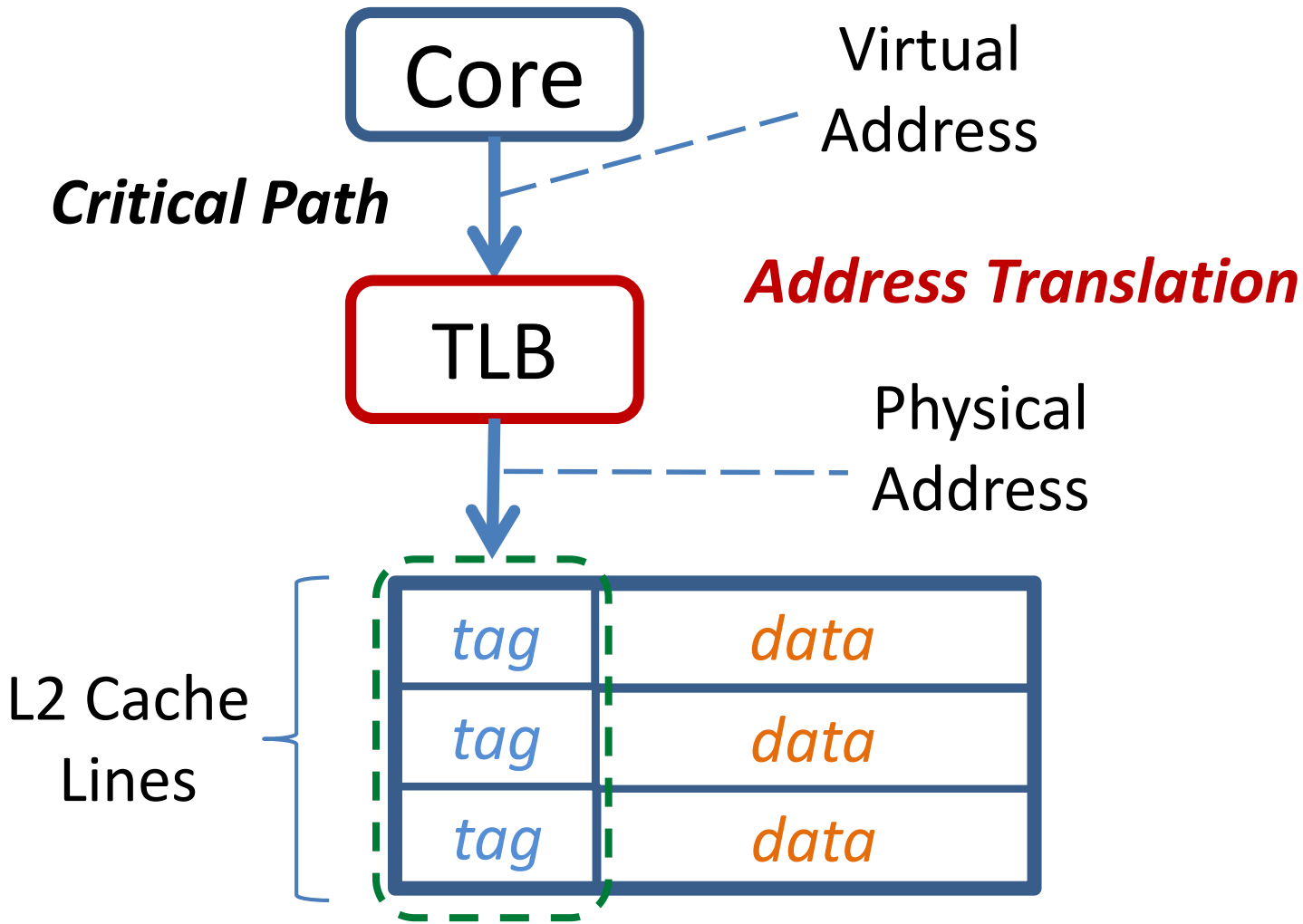
Backup Slides

Large Pages (e.g., 2MB or 1GB)

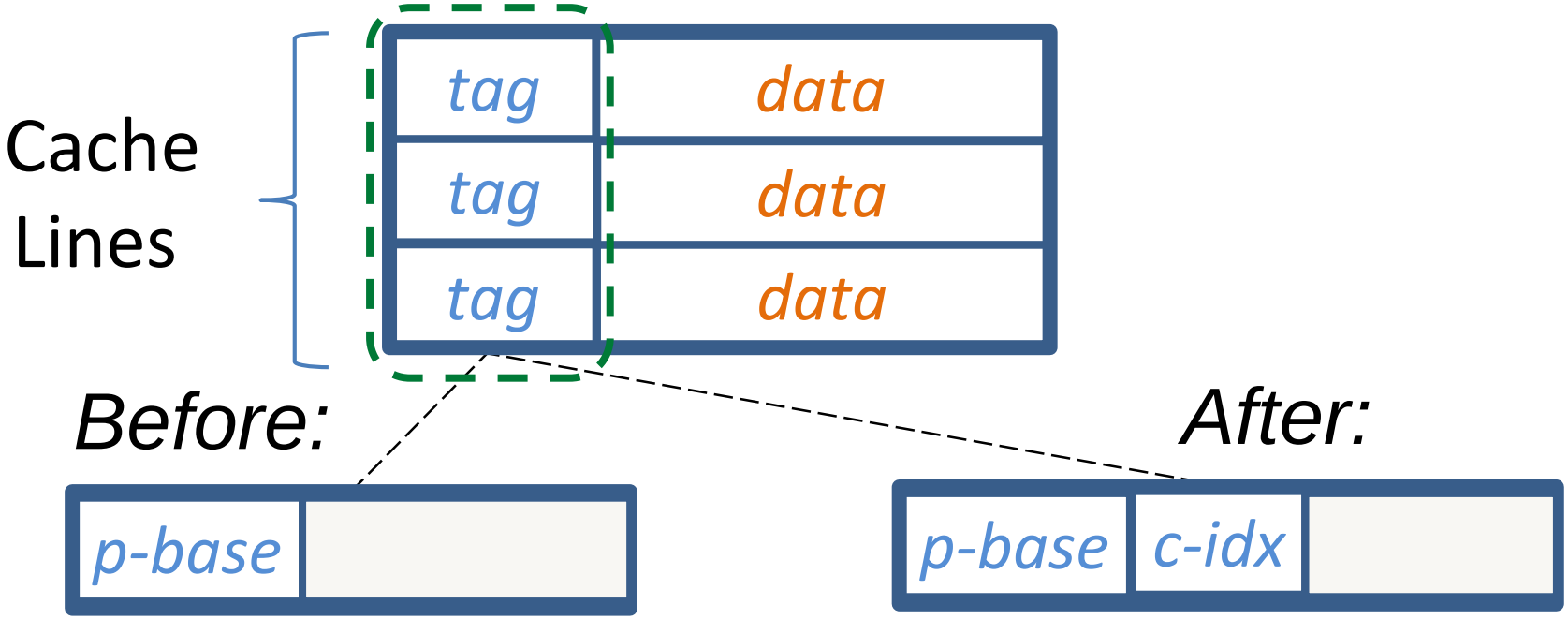
- Splitting large pages into smaller 4KB sub-pages (compressed individually)
- 64-byte metadata chunks for every sub-page



Physically Tagged Caches

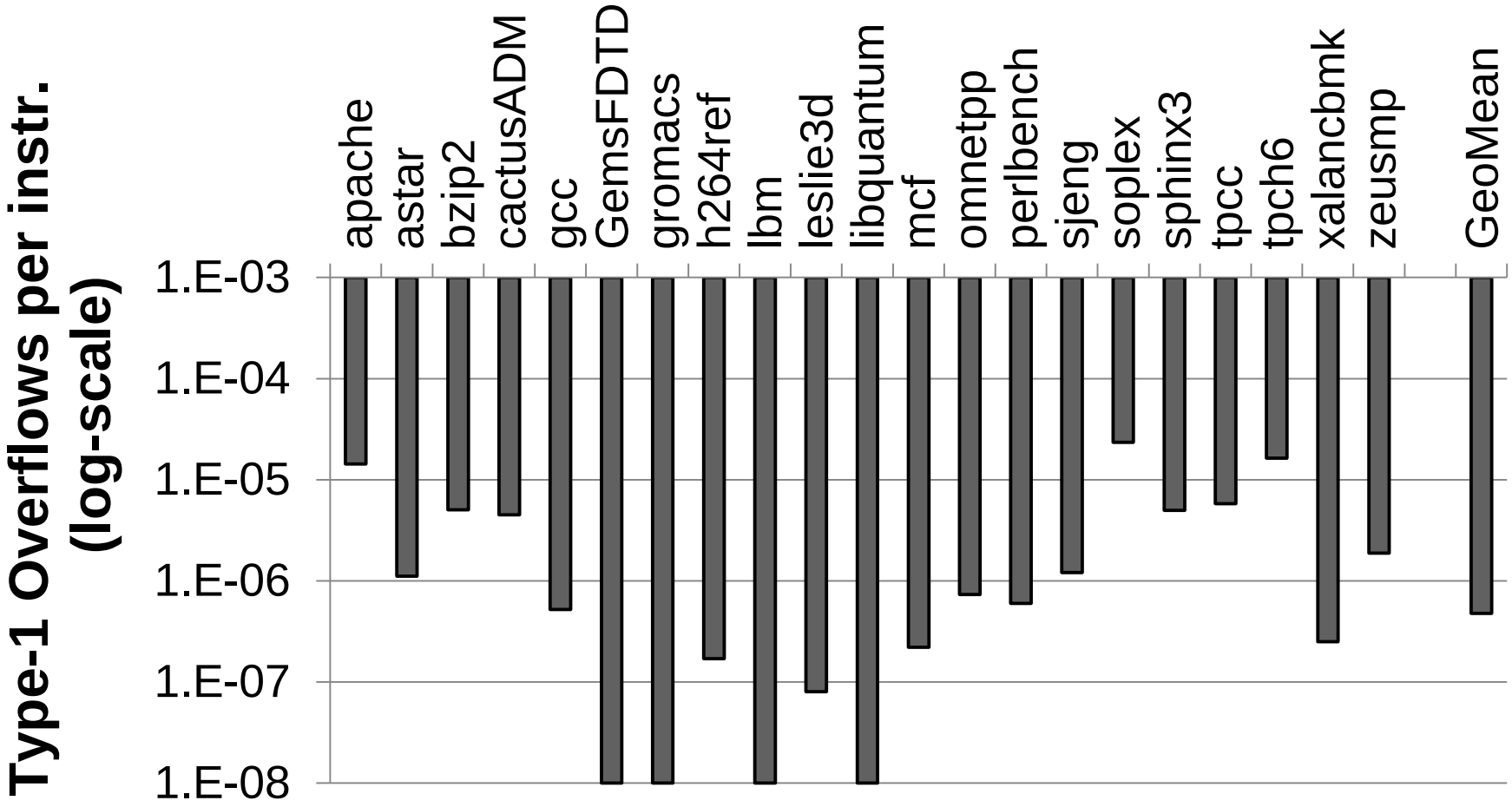


Changes to Cache Tagging Logic



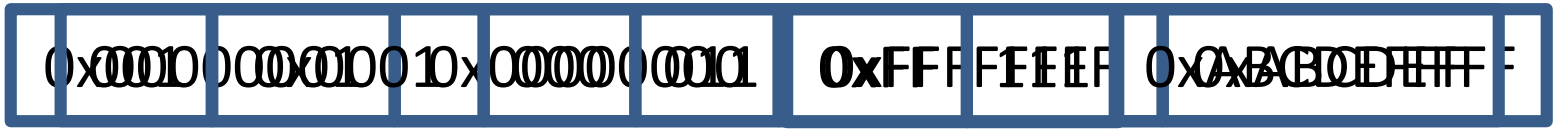
- *p-base* – physical page base address
- *c-idx* – cache line index within the page

Analysis of Page Overflows



Frequent Pattern Compression

Idea: encode cache lines based on frequently occurring patterns, e.g., first half of a word is zero



Frequent Patterns:

000 – All zeros
001 – First half zeros
010 – Second half zeros
011 – Repeated bytes
100 – All ones
…
111 – Not a frequent pattern

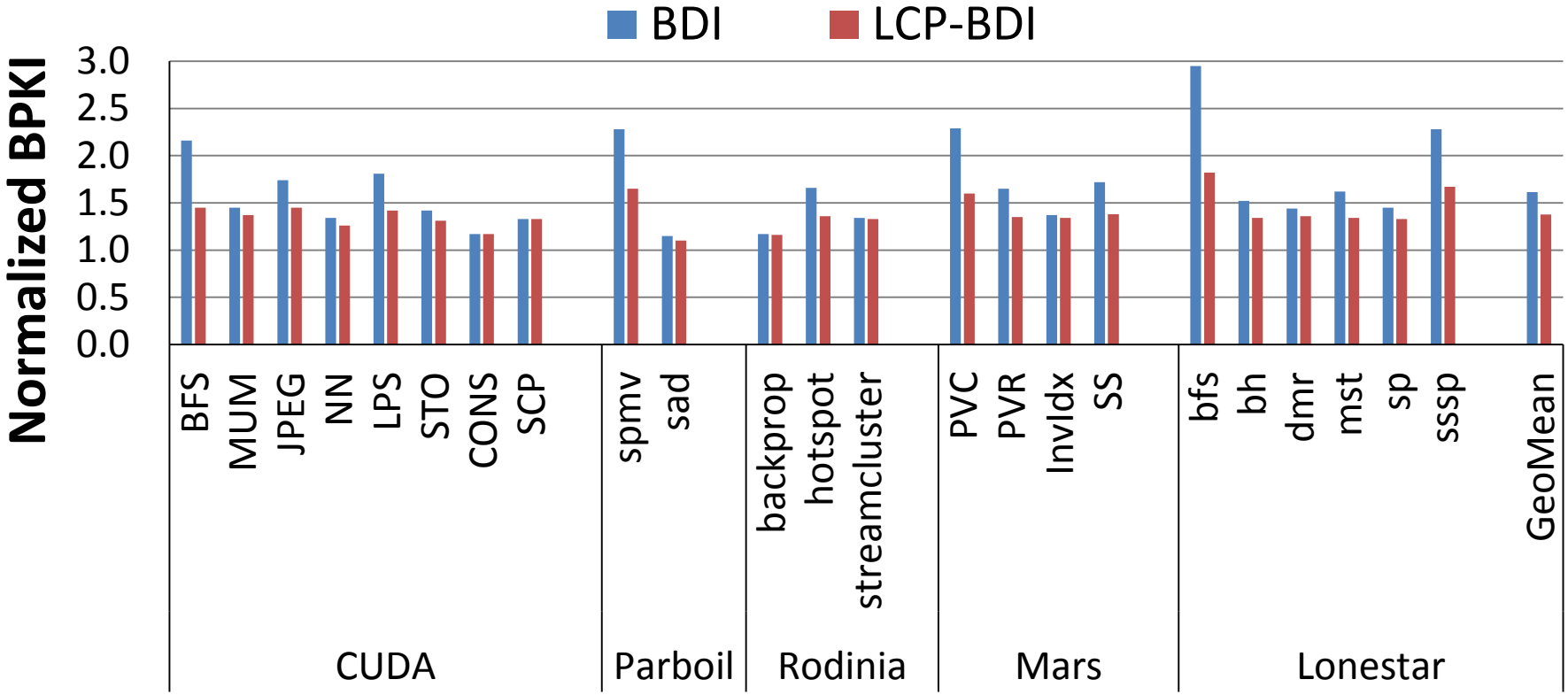
0x00000001
0x00000000
0xFFFFFFFF
0xABCDEFFF

001	0x0001
000	
011	0xFF
111	0xABCDEFFF

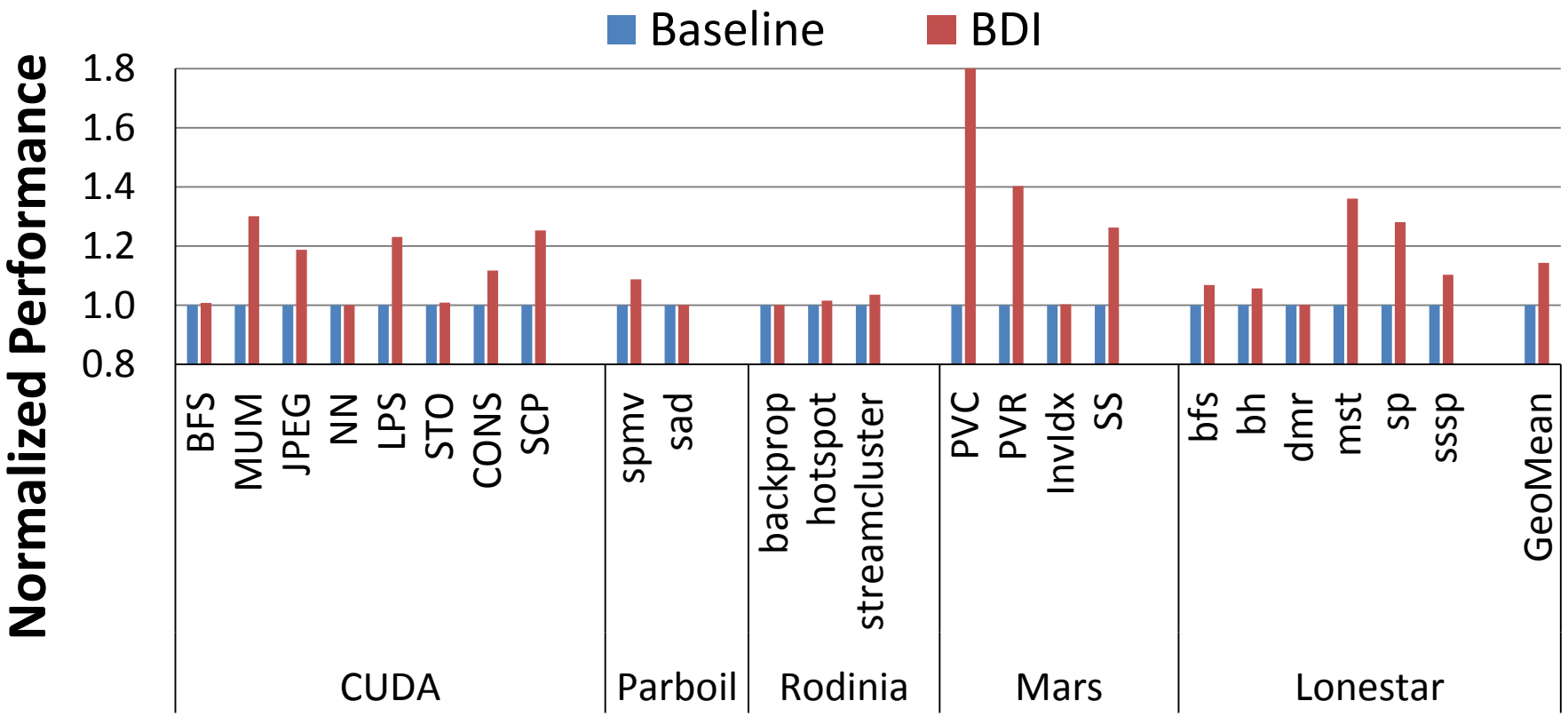
GPGPU Evaluation

- Gpgpu-sim v3.x
- Card: NVIDIA GeForce GTX 480 (Fermi)
- Caches:
  - DL1: 16 KB with 128B lines
  - L2: 786 KB with 128B lines
- Memory: GDDR5

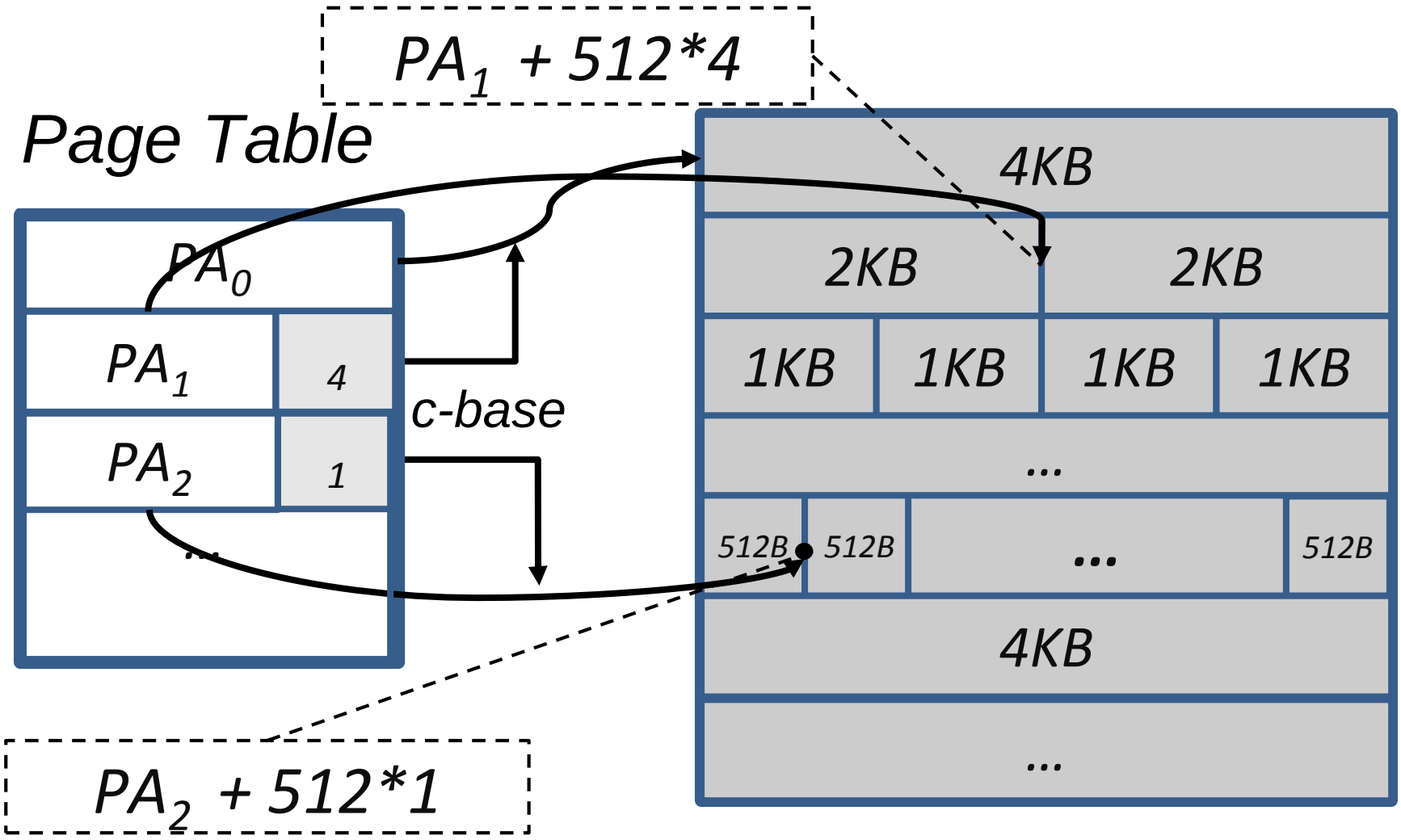
Effect on Bandwidth Consumption



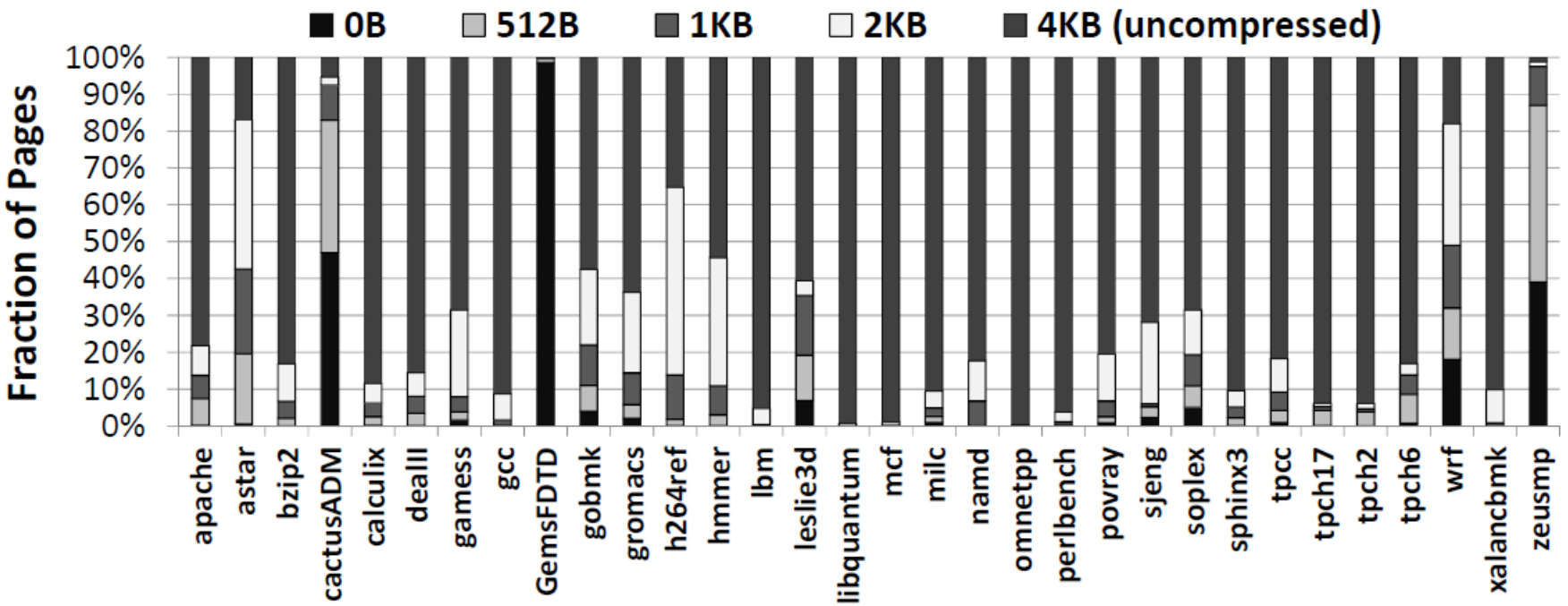
Effect on Throughput



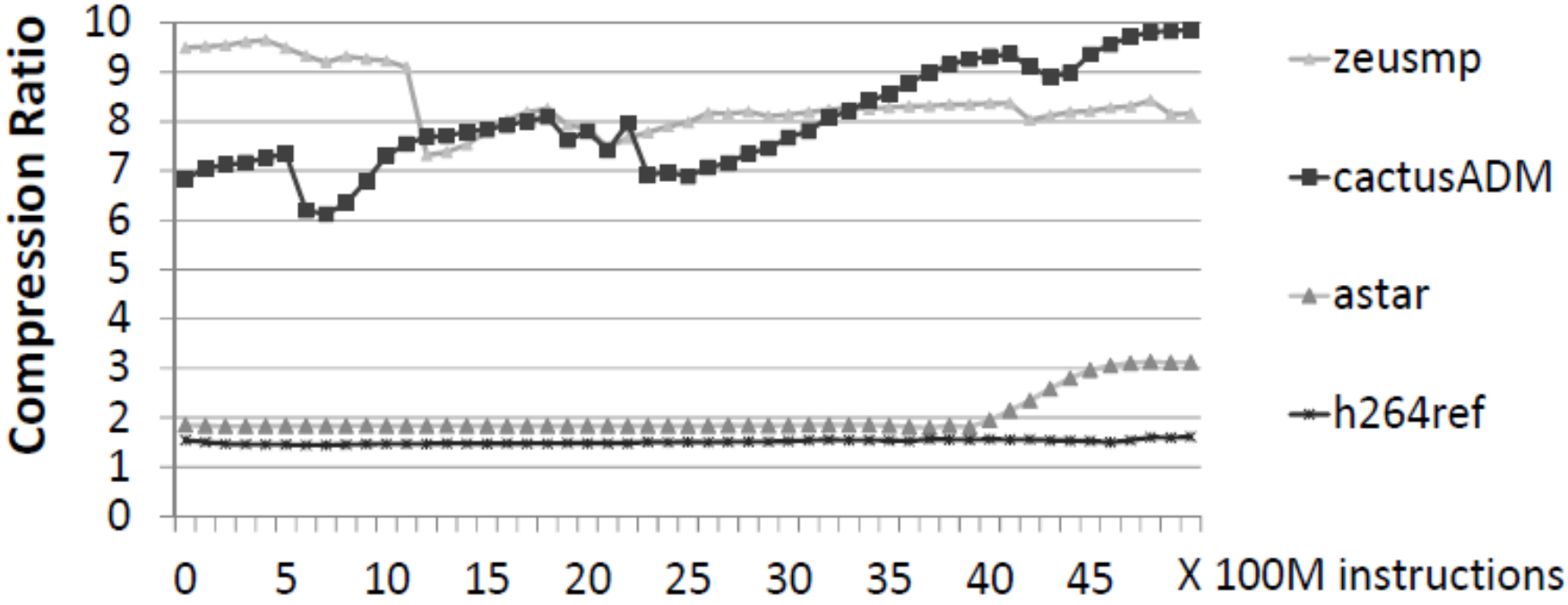
Physical Memory Layout



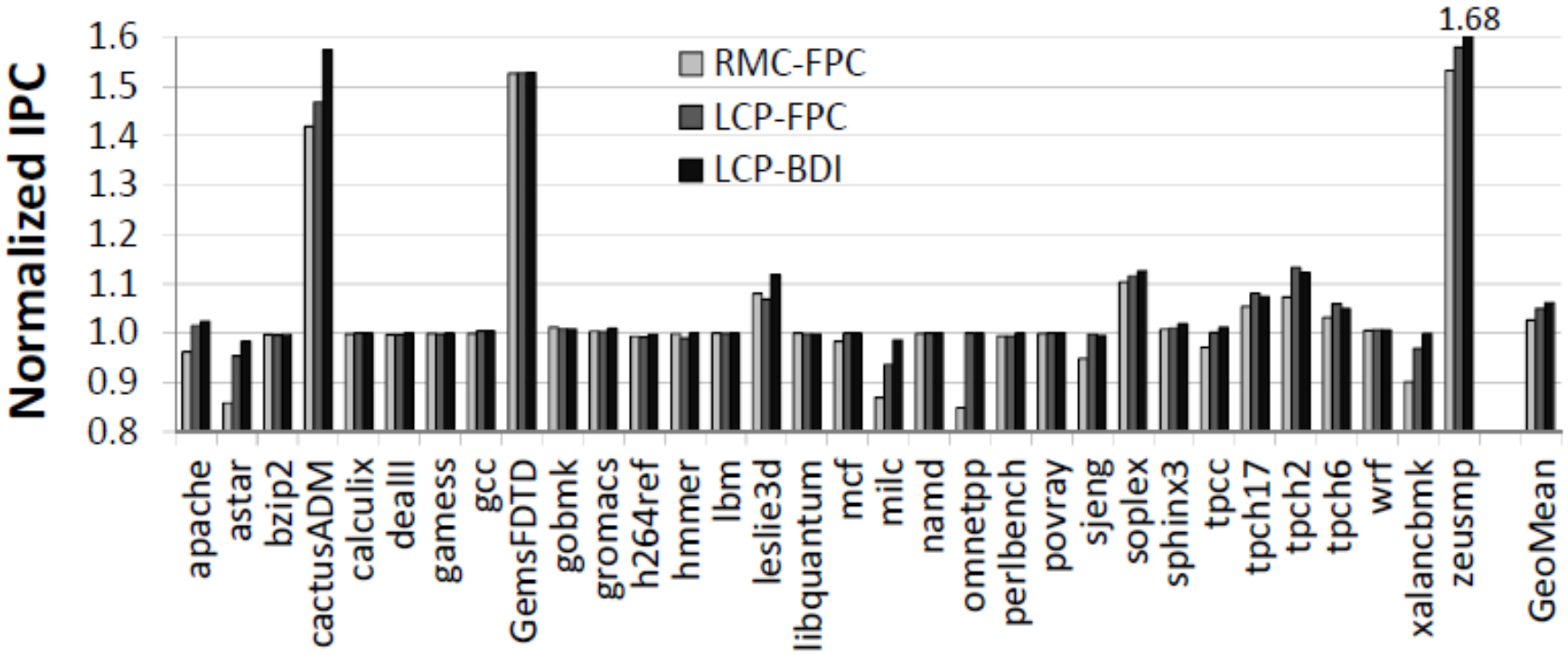
Page Size Distribution



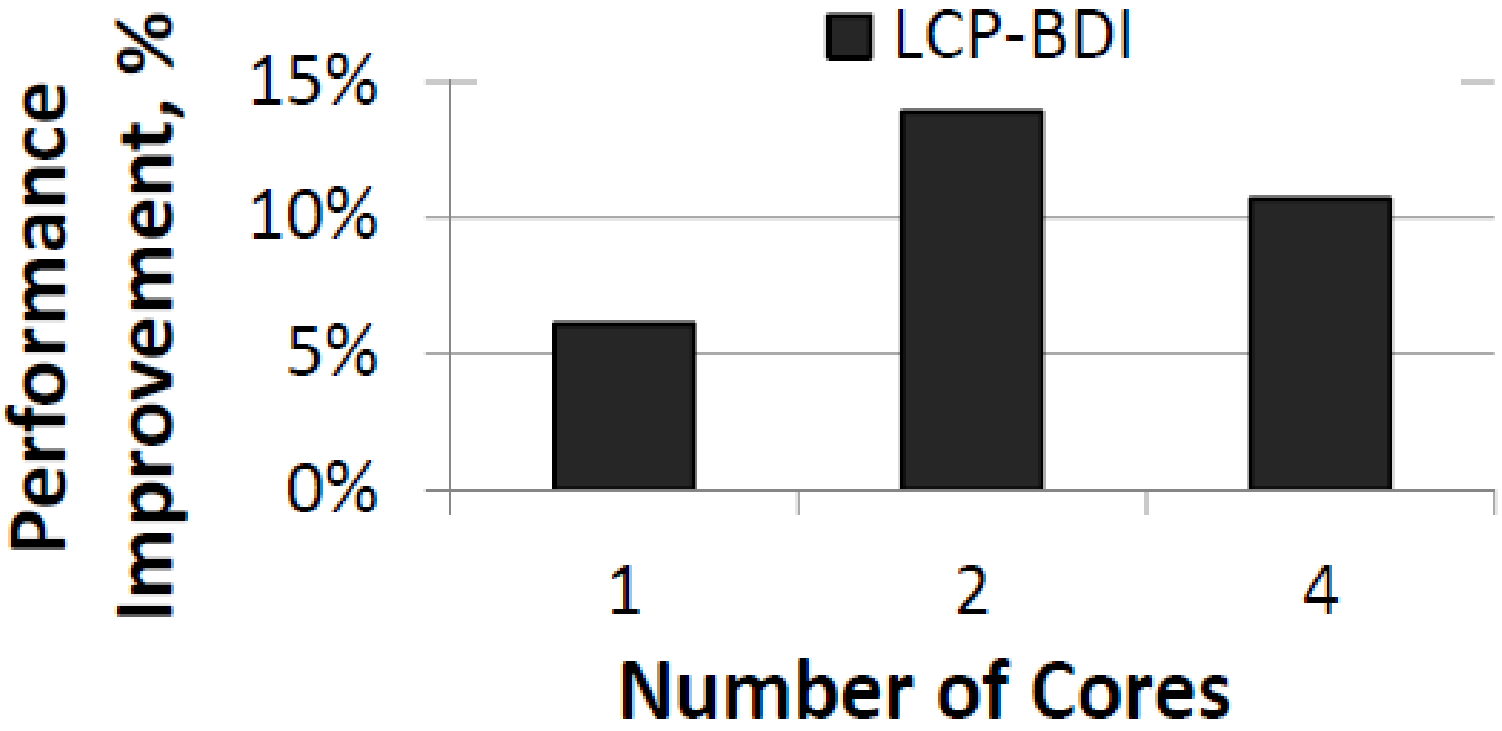
Compression Ratio Over Time



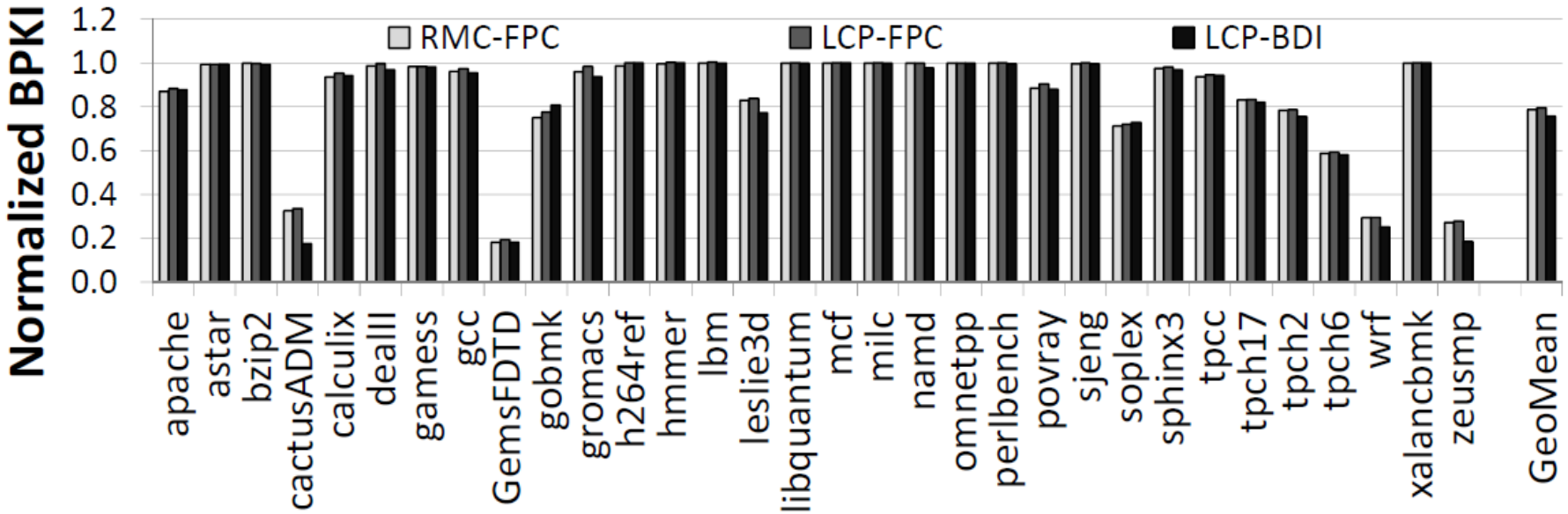
IPC (1-core)


Normalized IPC
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
RMC-FPC
LCP-FPC
LCP-BDI
1.68
apache
astar
bzip2
cactusADM
calculix
dealII
gamess
gcc
GemsFDTD
gobmk
gromacs
h264ref
hmmer
lbm
leslie3d
libquantum
mcf
milc
namd
omnetpp
perlbench
povray
sjeng
soplex
sphinx3
tpcc
tpch17
tpch2
tpch6
wrf
xalancbmk
zeusmp
GeoMean

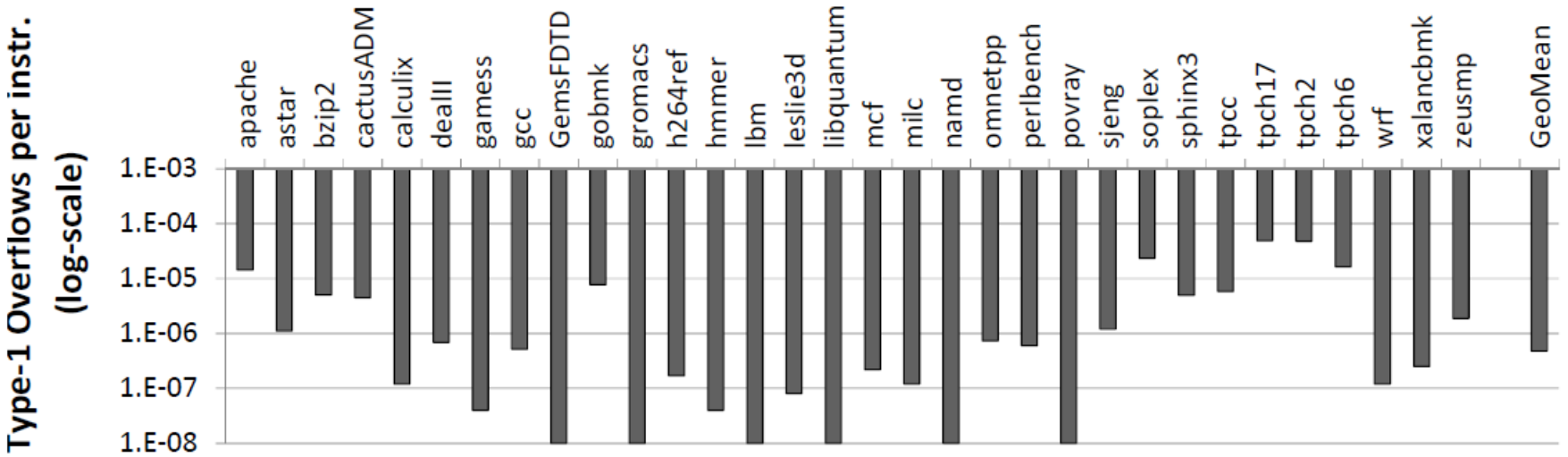
Weighted Speedup



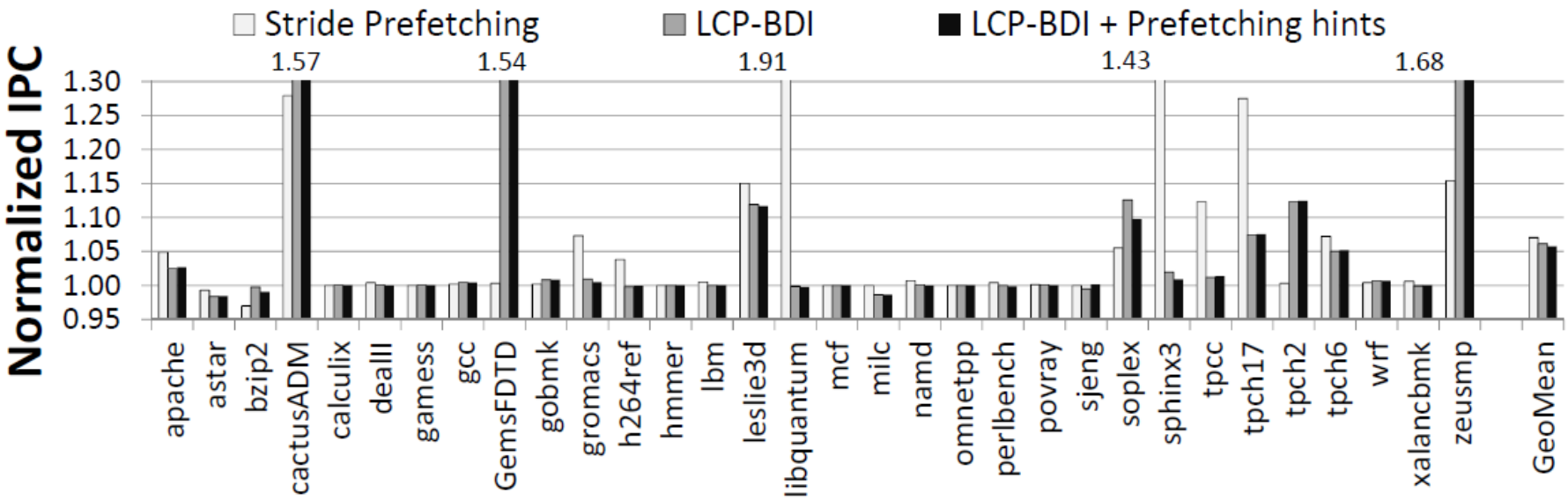
Bandwidth Consumption



Page Overflows



Stride Prefetching - IPC



Stride Prefetching - Bandwidth