Compression and SSDs: Where and How?

Aviad Zuck, Sivan Toledo, Dmitry Sotnikov, Danny Harnik

outline

- Introduction
- Compression and SSDs
  - Where and how
  - Typical use case
- Evaluation
- Conclusion

why compress data

Databases, HPC, desktop apps, virtual machines…

data

less data transferred to the device via interconnect

Reduced latency, improved throughput

More capacity
(or) free space
beneficial to COW and log-structured systems (e.g. SSDs)

For SSDs less writes equals less wear
Extended lifetime

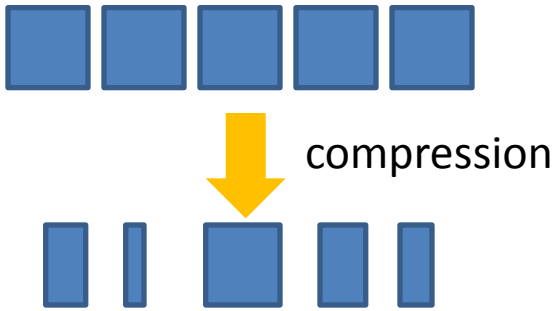
concerns of compression

disks (mostly) read/write
4KB units

so do file-systems and
operating systems

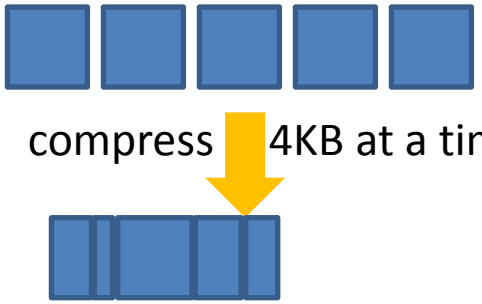
CPU works harder

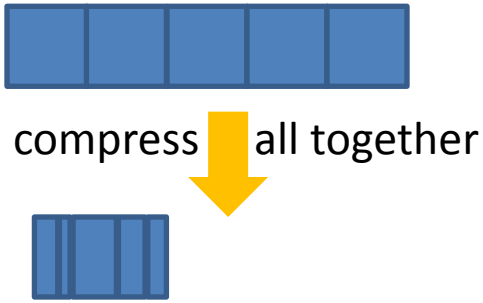
compression turns
4KB-aligned data to variable-
sized chunks



**may need more complex
data structures**

compression improves as chunk size increases





file systems and applications often try to compress larger chunks

potentially at the cost of **increased accesses to disk (increase SSD wear)**

redundant reads and read-modify-write of entire chunk

outline

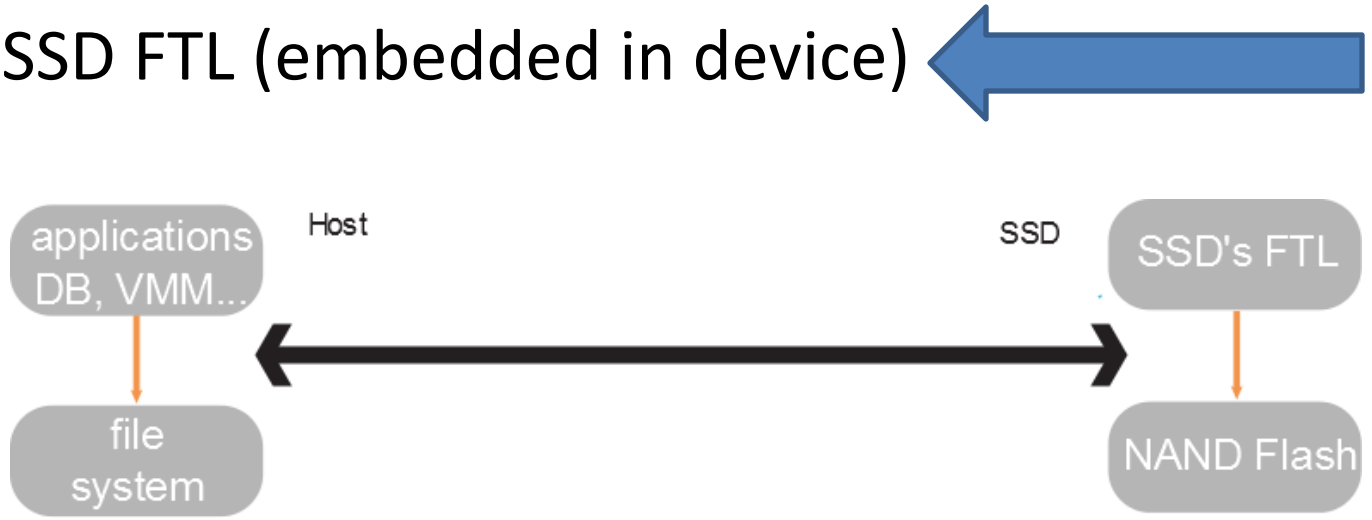
- Introduction
- Compression and SSDs
  - Where and how
  - Typical use case
- Evaluation
- Conclusion

where and how

- Where should we place compression?
  - Host or device?
- How should we do it?
  - What granularity?
  - Which layout?

where to compress

- Compression can be enabled in different levels:
  - application (database)
  - file system
  - SSD FTL (embedded in device)



SSD use cases

- SSDs vs. magnetic disks (HDDs)
  - faster random accesses
  - but more expensive
- Mostly used for
  - OLTP = OnLine Transaction Processing (MySQL, Oracle)
  - Caching solutions

Known to generate highly-compressible data (2-5x compressible)

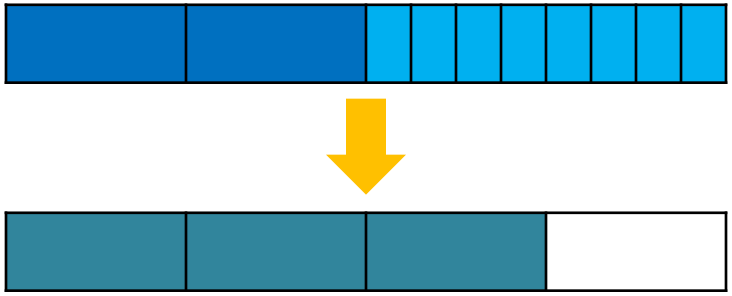
mysql* database compression

data stored in fixed-size B-tree pages (e.g. 16KB)

compress to smaller fixed-size compressed page (e.g. 4K, 8K)

in-page modifications log

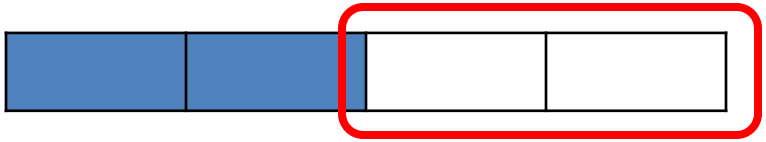
when log is full, re-compress entire page



* most popular freely-available open-source database system.

fixed-size compression pages → compression failures

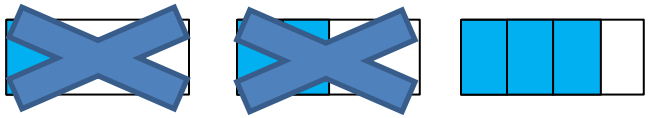
padding and unutilized space → reduces compression gains



may cause read-modify-write behavior

logical view

physical view



file system compression

- Btrfs – B-tree file system. In Linux kernel, still considered experimental
- ZFS – focused on protecting against data corruption (more conservative and robust)
- ext4 – most popular Linux file system

- ZFS/Btrfs are copy-on-write file systems
  - Support compression (easier with COW)
  - Aligned storage units

- COW good for compression, but causes fragmentation
  - Splitting + recompressing storage units
- Read-modify-copy still possible
  - Records/extents not always fragmented
- Unutilized space

brief summary

- Read-modify-write behavior
- Unutilized space
- Fragmentation

"All problems in computer science can be solved by another level of indirection…except of course for the problem of too many indirections"

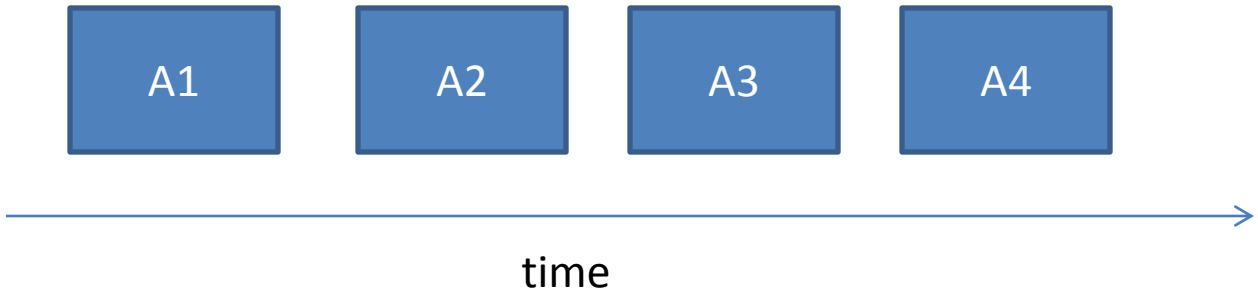
David Wheeler

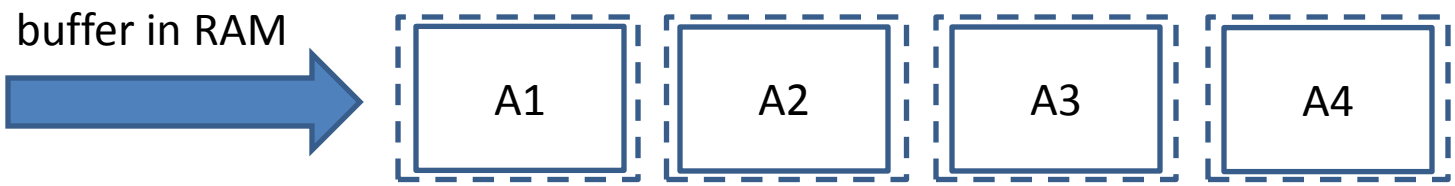
intra-SSD compression

- ✓ FTLs utilize indirection anyway (fixed-size)
- ✓ Re-use mapping table for compression
  - Buffer write requests
  - Compress
  - Dump compressed data to flash
  - Update mapping

How? four possible packing schemes

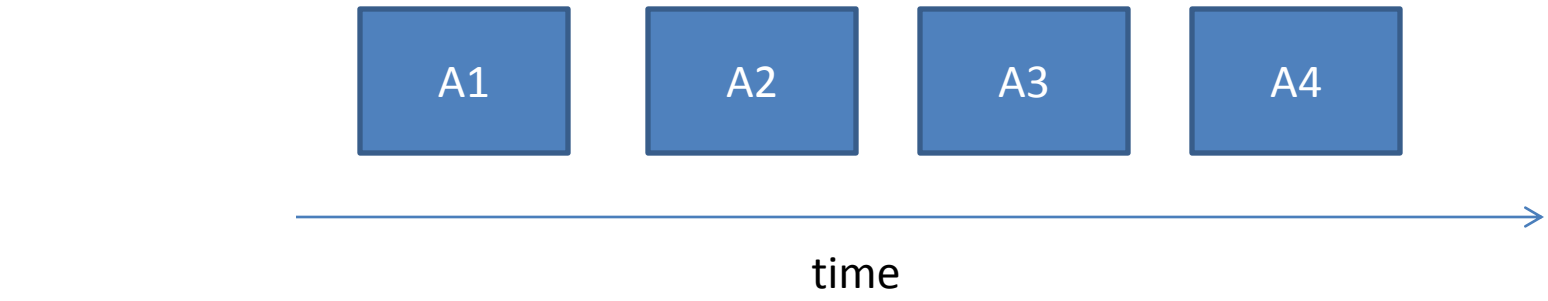
chunk-based



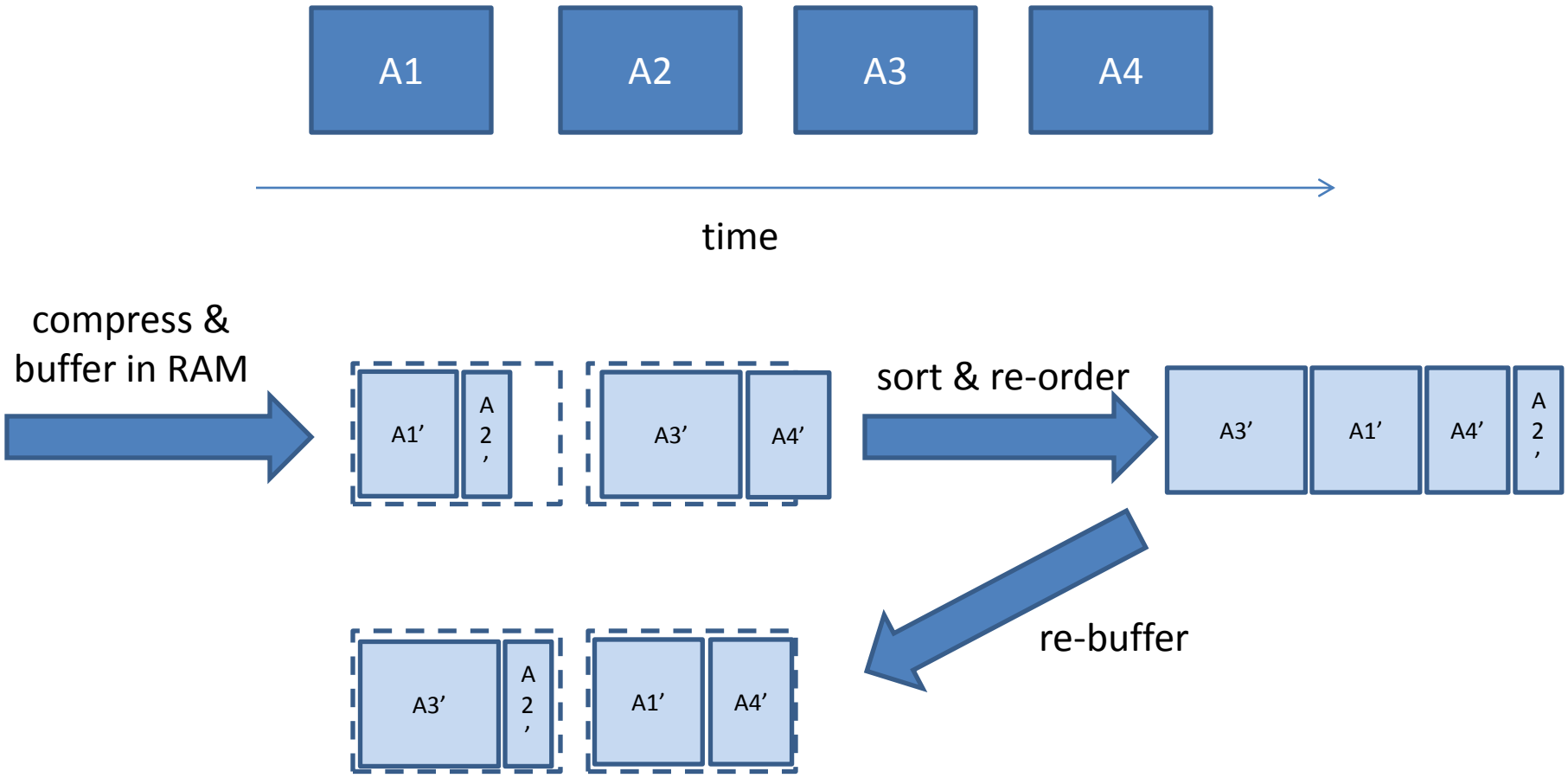


compress as single large chunk…

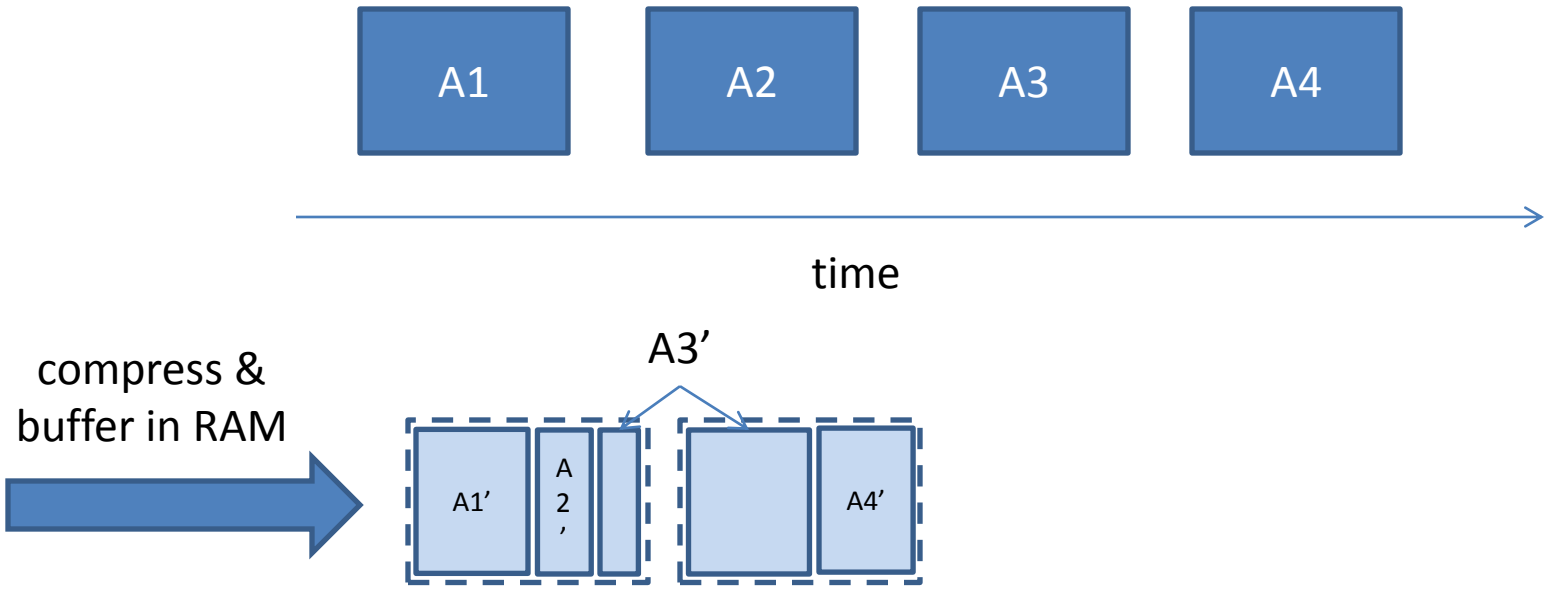
binpacking


A1
A2
A3
A4
time
compress &
buffer in RAM
A4'`
A
2
'
A3'
A4'
dump first buffer to make room
dump to flash
A1
A
2
'

re-ordering


A1
A2
A3
A4
time
compress &
buffer in RAM
A1'
A2'
A3'
A4'
sort & re-order
A3'
A1'
A4'
A2'
re-buffer
A3'
A2'
A1'
A4'

compaction

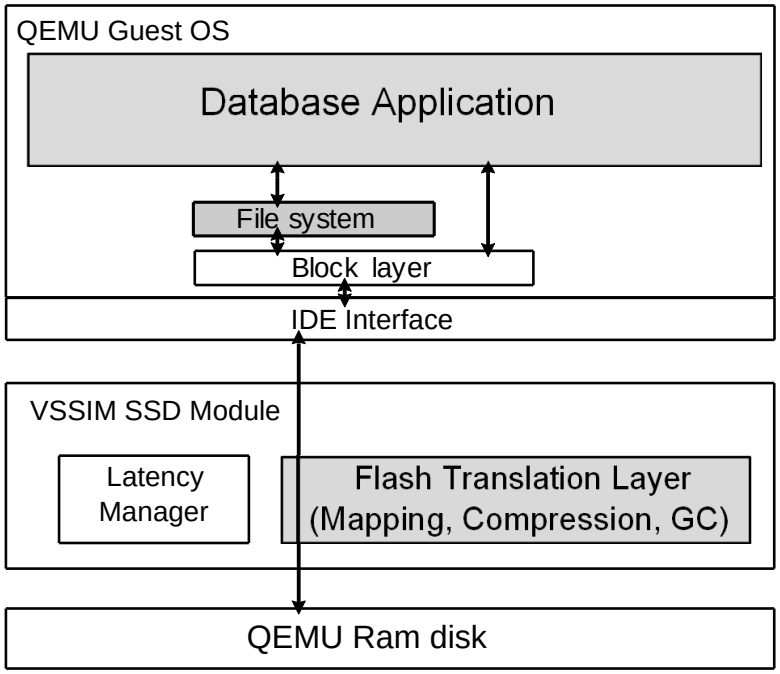


outline

- Introduction
- Compression and SSDs
  - Where and how
  - Typical use case
- Evaluation
- Conclusion

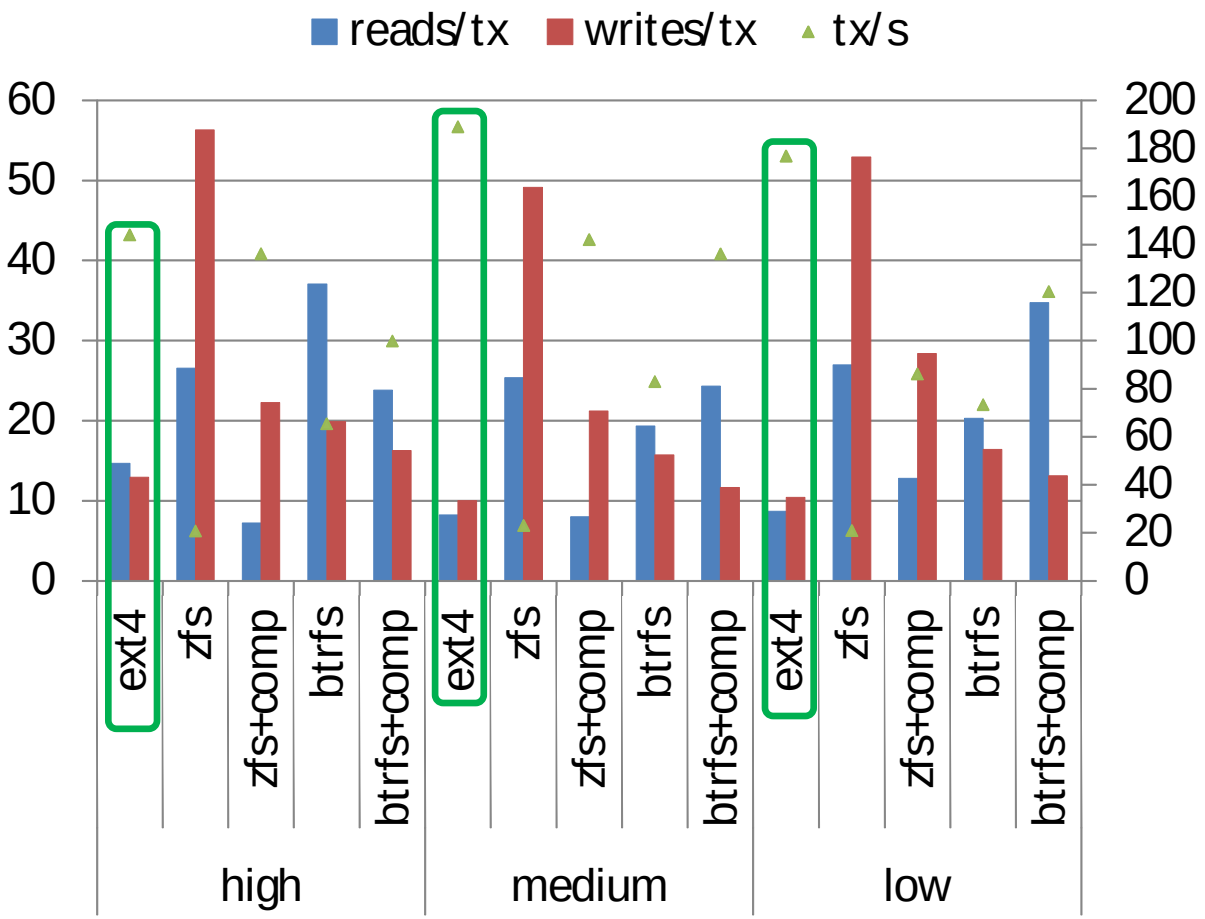
methodology

- 4GB SSD
  - VSSIM SSD emulator
  - 4KB pages, 64 pages/block
  - 50/250us read/write latency
- TPC-C workload (OLTP)
  - Modified to provide three levels of compressibility (high/medium/low)



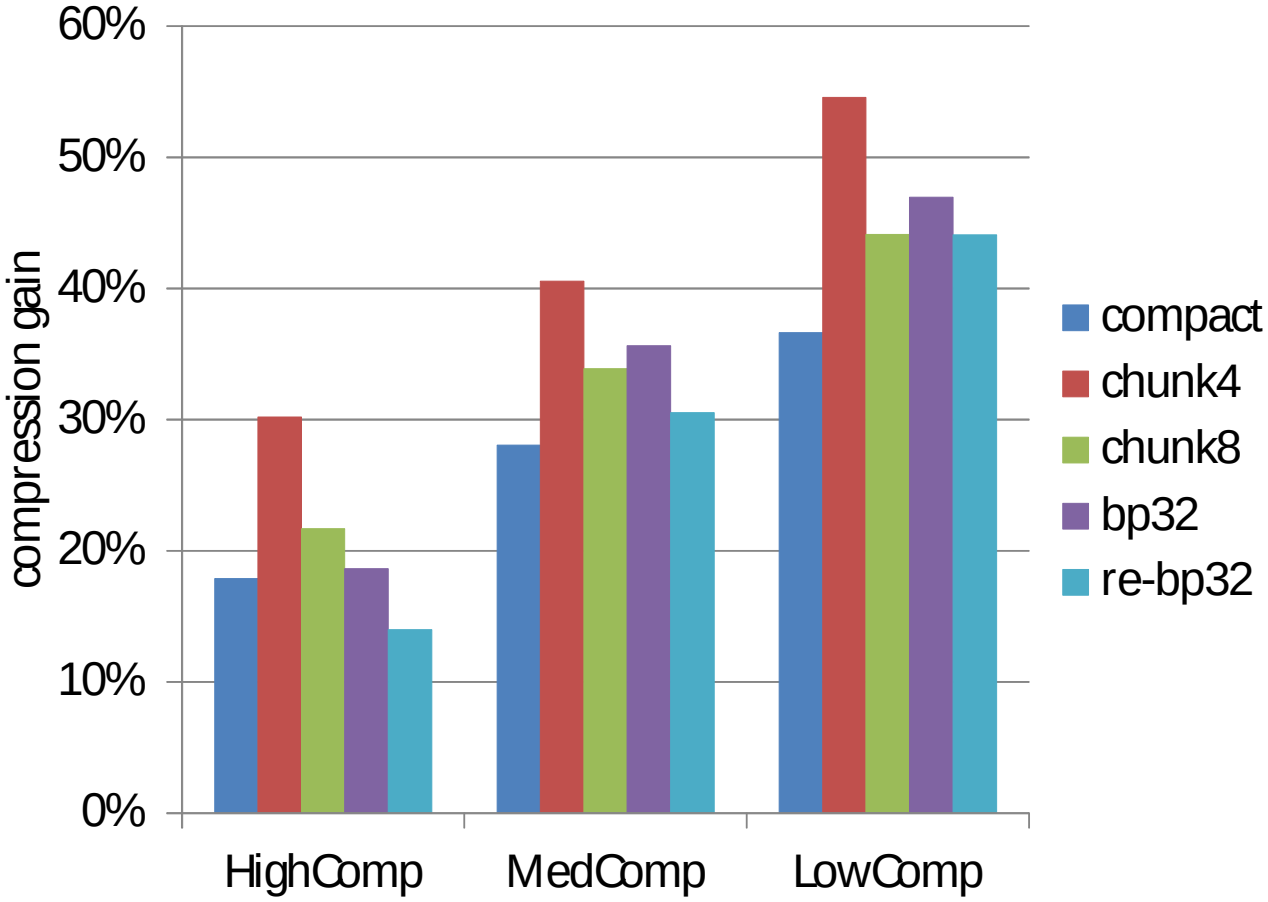
repeated with compression enabled in every level

file-system compression



- **ext4 w/o compression yields best results**
- MySQL+compression much worse in all configurations

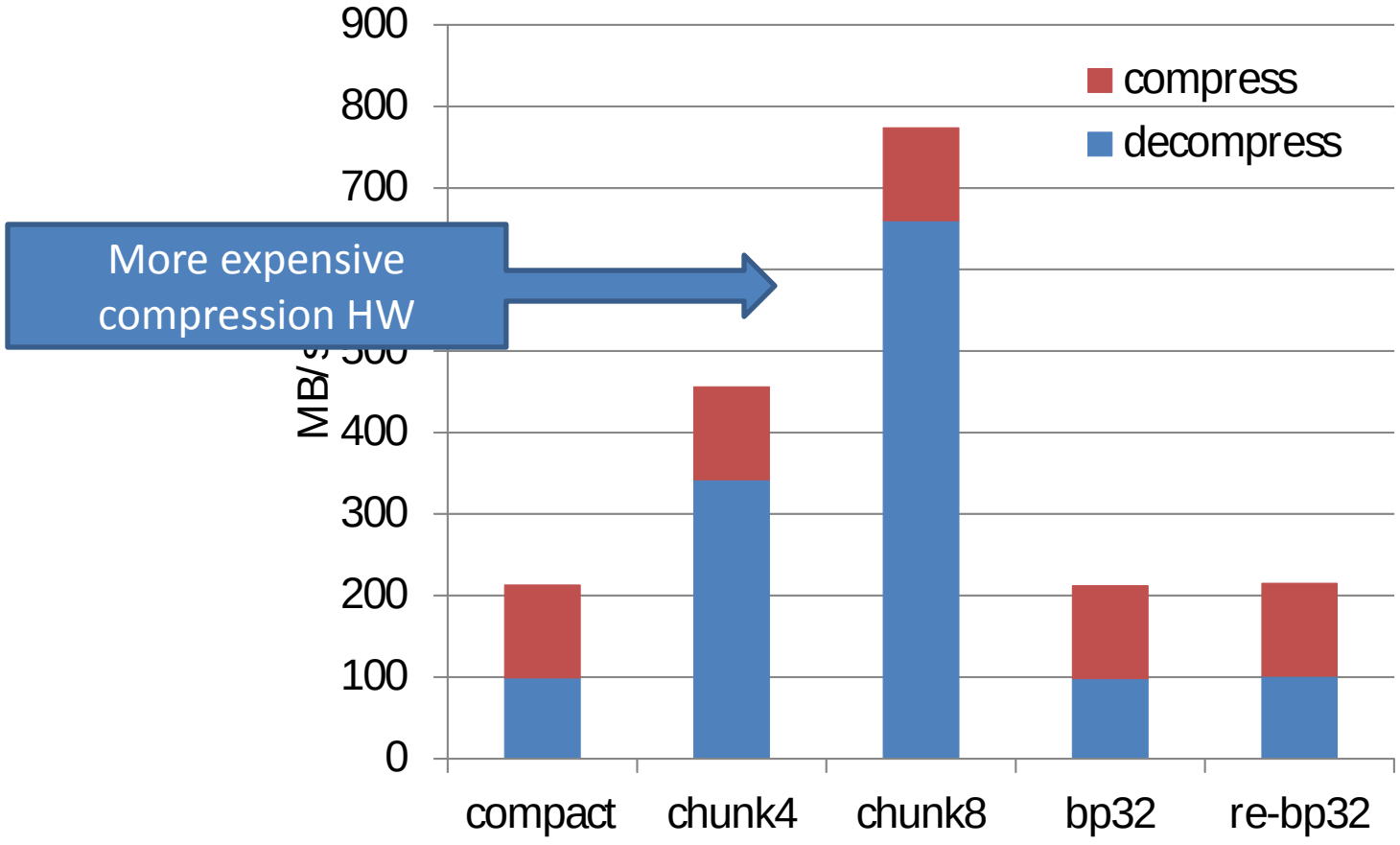
embedded compression gains



performance (tx/s) improved by 10-15% vs. no compression

- Total physical writes vs. FTL w/o compression
- Compact and re-ordering schemes deliver similar performance
  - re-ordering requires 30% less RAM for mapping data structures

compression hardware



outline

- Introduction
- Compression and SSDs
  - Where and how
  - Typical use case
- Evaluation
- Conclusion

conclusion

- **Intra-SSD compression superior** to using it in the existing application and file systems
  - improvement vs. popular database application and file systems in OLTP workload
- Compressing OLTP workloads in **larger chunks not always better**
- **Enhanced re-ordering scheme delivers optimal improvement** using 30% less RAM requirements than new compact scheme

questions?

Aviad Zuck, Sivan Toledo, Dmitry Sotnikov, Danny Harnik

(http://www.tau.ac.il/~aviadzuc)