New Computer Trends And How This Affect Us

Francesc Alted
Freelance Consultant
http://www.blosc.org/professional-services.html

April 10rd, 2016

“No sensible decision can be made any longer without taking into account not only the world as it is, but the world as it will be.”

— *Isaac Asimov*

“No sensible decision can be made any longer without taking into account not only the computer as it is, but the computer as it will be.”

— My own rephrasing

About Me

- Physicist by training
- Computer scientist by passion
- Open Source enthusiast by philosophy
  - PyTables (2002 - 2011)
  - **Blosc** (2009 - now)
  - bcolz (2010 - now)

Why Open Source Projects?

"*The art is in the execution of an idea. Not in the idea. There is not much left just from an idea.*"

–Manuel Oltra, music composer

"*Real artists ship*"

–Seth Godin, writer

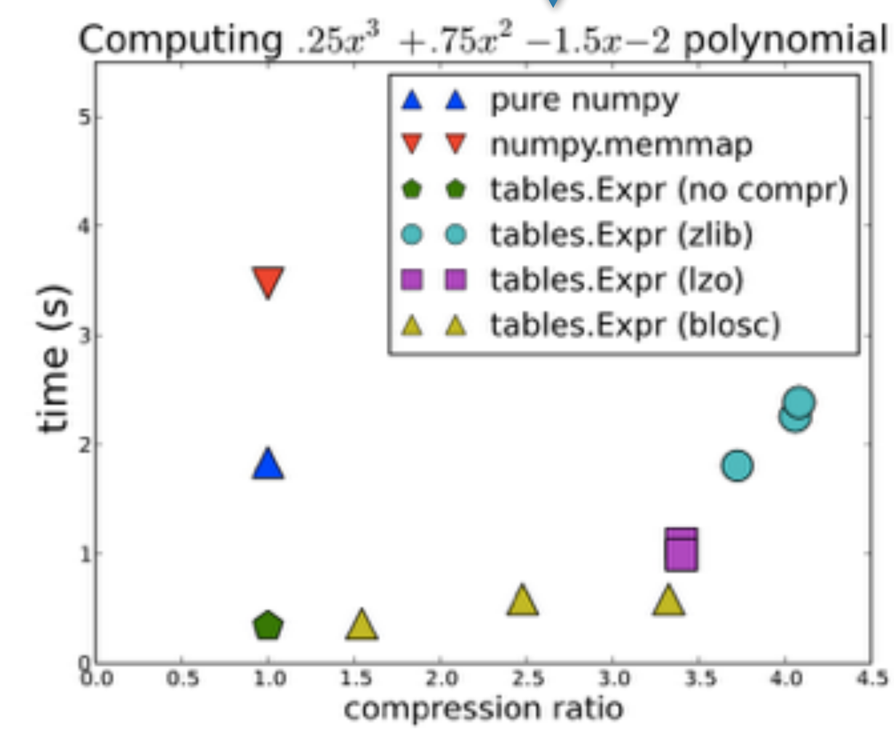
- Nice way to realize yourself while helping others

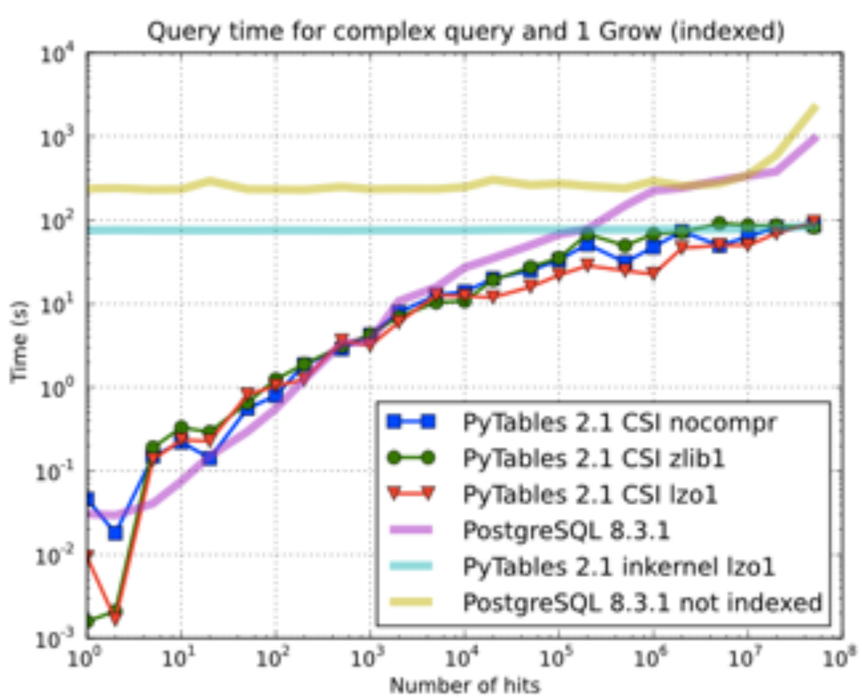
PyTables

Out-of-core
Expressions

The technology platform to make a difference in your relationship with large and complex data

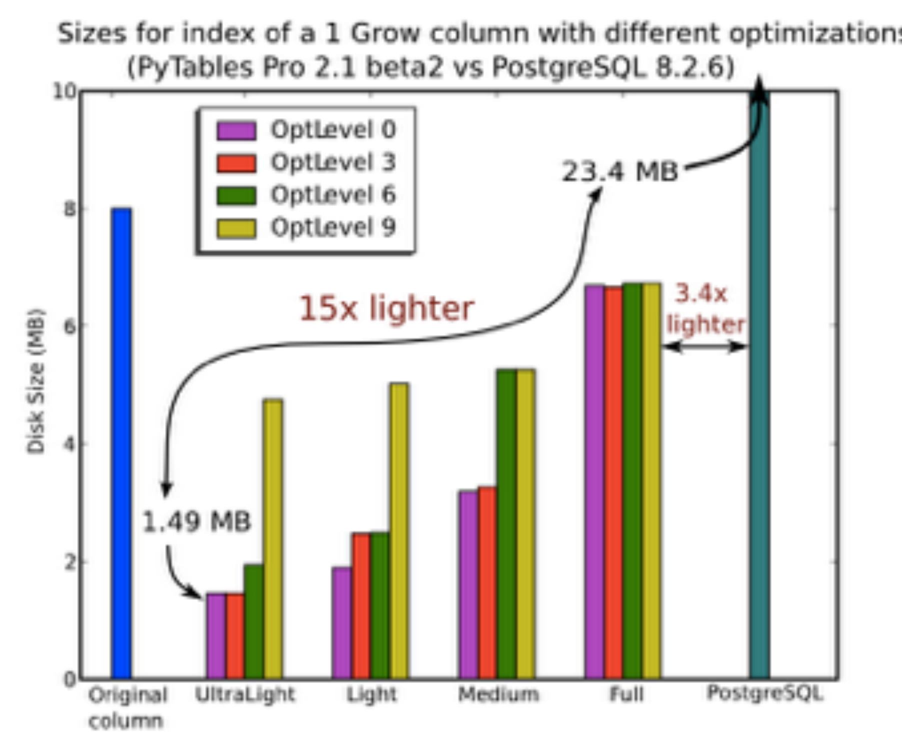
HDF5 + a Twist





OPSI

Indexed
Queries



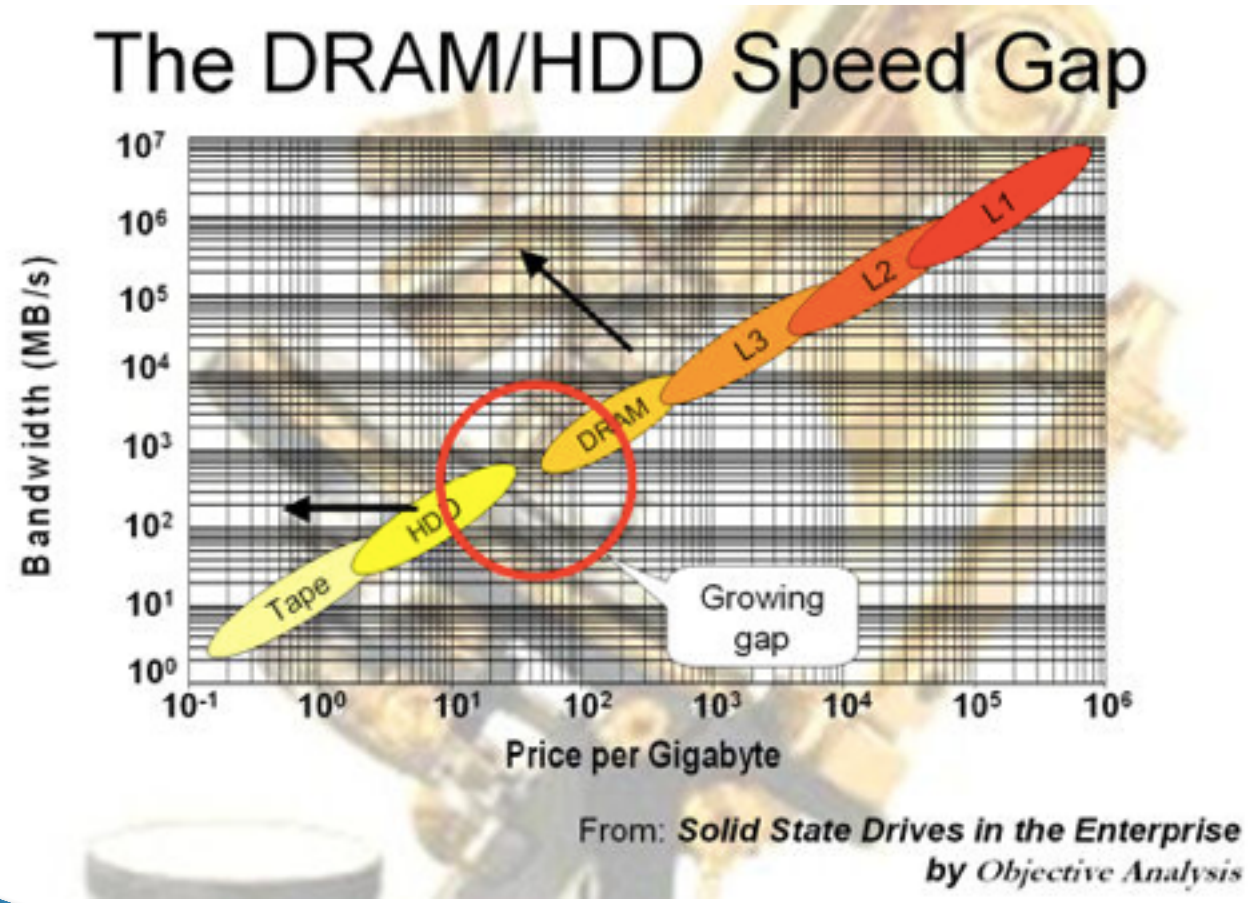
Overview

- Recent trends in computer architecture
- The need for speed: storing and processing as much data as possible with your existing resources
- **Blosc & bcolz** as examples of compressor and data containers for large datasets that follow the principles of the newer computer architectures

Trends in Computer Storage

Forthcoming Trends

The growing gap between DRAM and HDD is facilitating the introduction of new SDD devices



PCIe SSD

M.2 SSD

BGA SSD

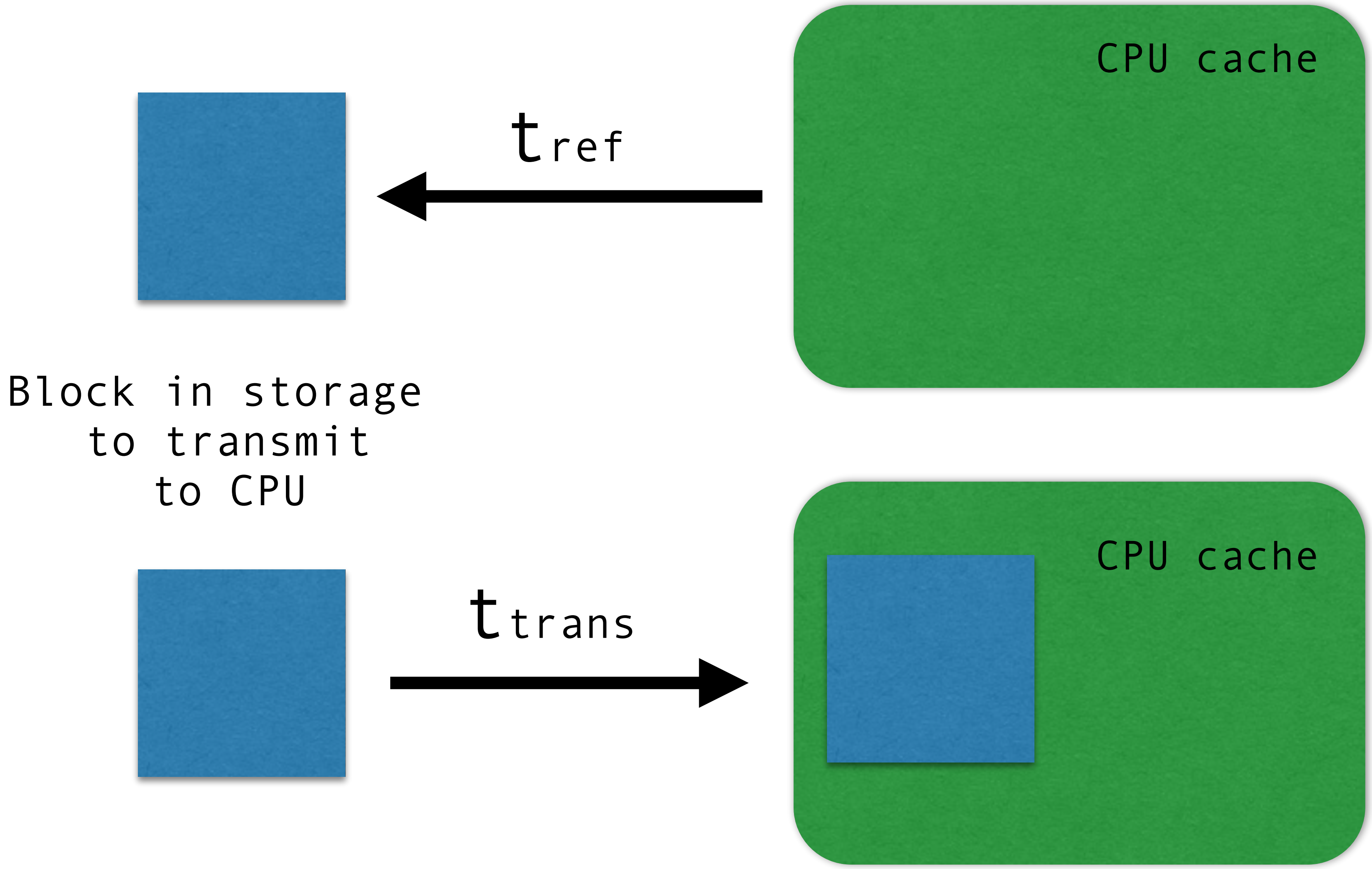
Latency Numbers Every Programmer Should Know

```
Latency Comparison Numbers
--------------------------
L1 cache reference                           0.5 ns
Branch mispredict                            5   ns
L2 cache reference                           7   ns                      14x L1 cache
Mutex lock/unlock                           25   ns
Main memory reference                      100   ns                      20x L2 cache, 200x L1 cache
Read 4K randomly from memory             1,000   ns     0.001 ms
Compress 1K bytes with Zippy             3,000   ns
Send 1K bytes over 1 Gbps network       10,000   ns     0.01 ms
Read 4K randomly from SSD*             150,000   ns     0.15 ms
Read 1 MB sequentially from memory     250,000   ns     0.25 ms
Round trip within same datacenter      500,000   ns     0.5  ms
Read 1 MB sequentially from SSD*     1,000,000   ns     1    ms  4X memory
Disk seek                           10,000,000   ns    10    ms  20x datacenter roundtrip
Read 1 MB sequentially from disk    20,000,000   ns    20    ms  80x memory, 20X SSD
Send packet CA->Netherlands->CA    150,000,000   ns   150    ms
```

Source: Jeff Dean and Peter Norvig (Google), with some additions

http://www.eecs.berkeley.edu/~rcs/research/interactive_latency.html

Reference Time vs Transmission Time



$t_{ref} \sim= t_{trans}$ => optimizes storage access

Not All Storage Layers Are Created Equal

Memory: t_{ref}: 100 ns / t_{trans} (**1 KB**): ~100 ns

Solid State Disk: t_{ref}: 10 us / t_{trans} (**4 KB**): ~10 us

Mechanical Disk: t_{ref}: 10 ms / t_{trans} (**1 MB**): ~10 ms

The slower the media, the larger the block that is worth to transmit

But essentially, a blocked data access is mandatory for speed!

We Need More Data Blocking In Our Infrastructure!

- Not many data containers focused on blocking access

- No silver bullet: we won't be able to find a single container that makes everybody happy; it's all about tradeoffs

- With blocked access we can use persistent media (disk) as it is ephemeral (memory) and the other way around -> independency of media!

Can We Get Better Bandwidth Than Hardware Allows?

Compression for Random & Sequential Access in SSDs

Performance Specification	Incompressible Data	Compressible Data
Sequential Write Bandwidth (Mbp/s)	235	520
Sequential Read Bandwidth (Mbp/s)	550	550
Random Write (IOPS)	16,500 (65MB/s)	60,000 (240MB/s)
Random Read (IOPS)	46,000 (180MB/s)	50,000 (200MB/s)

3. Source: *Intel® Solid-State Drive 520 Series Product Specification*; Random reads based on 4KB Queue Depth 32

- Compression does help performance!

Compression for Random & Sequential Access in SSDs

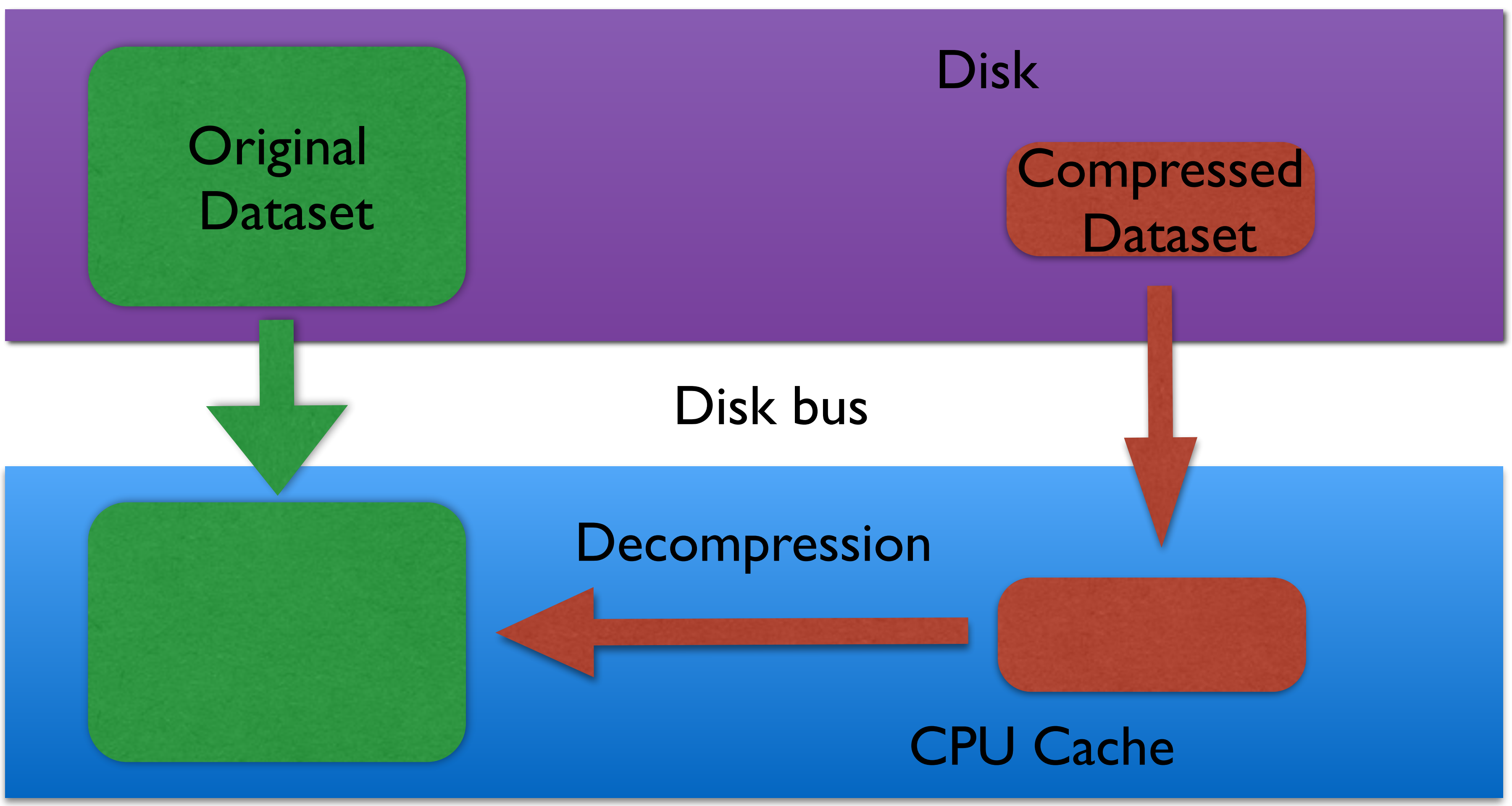
Performance Specification	Incompressible Data	Compressible Data
Sequential Write Bandwidth (Mbp/s)	235	520
Sequential Read Bandwidth (Mbp/s)	550	550
Random Write (IOPS)	16,500 (65MB/s)	60,000 (240MB/s)
Random Read (IOPS)	46,000 (180MB/s)	50,000 (200MB/s)

3. Source: *Intel® Solid-State Drive 520 Series Product Specification*; Random reads based on 4KB Queue Depth 32

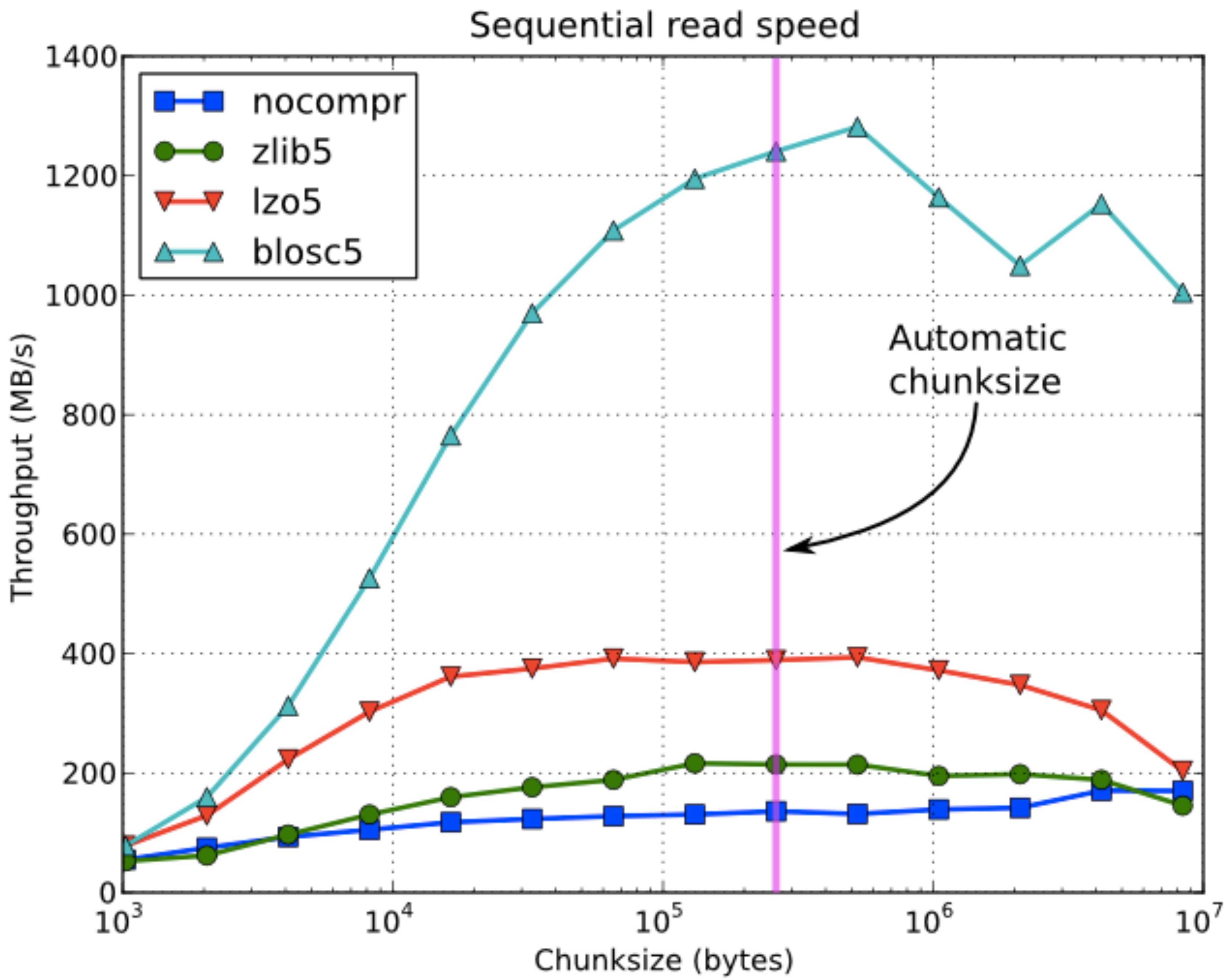
- Compression does help performance!
- However, limited by SATA bandwidth

Leveraging Compression Straight To CPU

Less data needs to be transmitted to the CPU



Transmission + decompression faster than direct transfer?

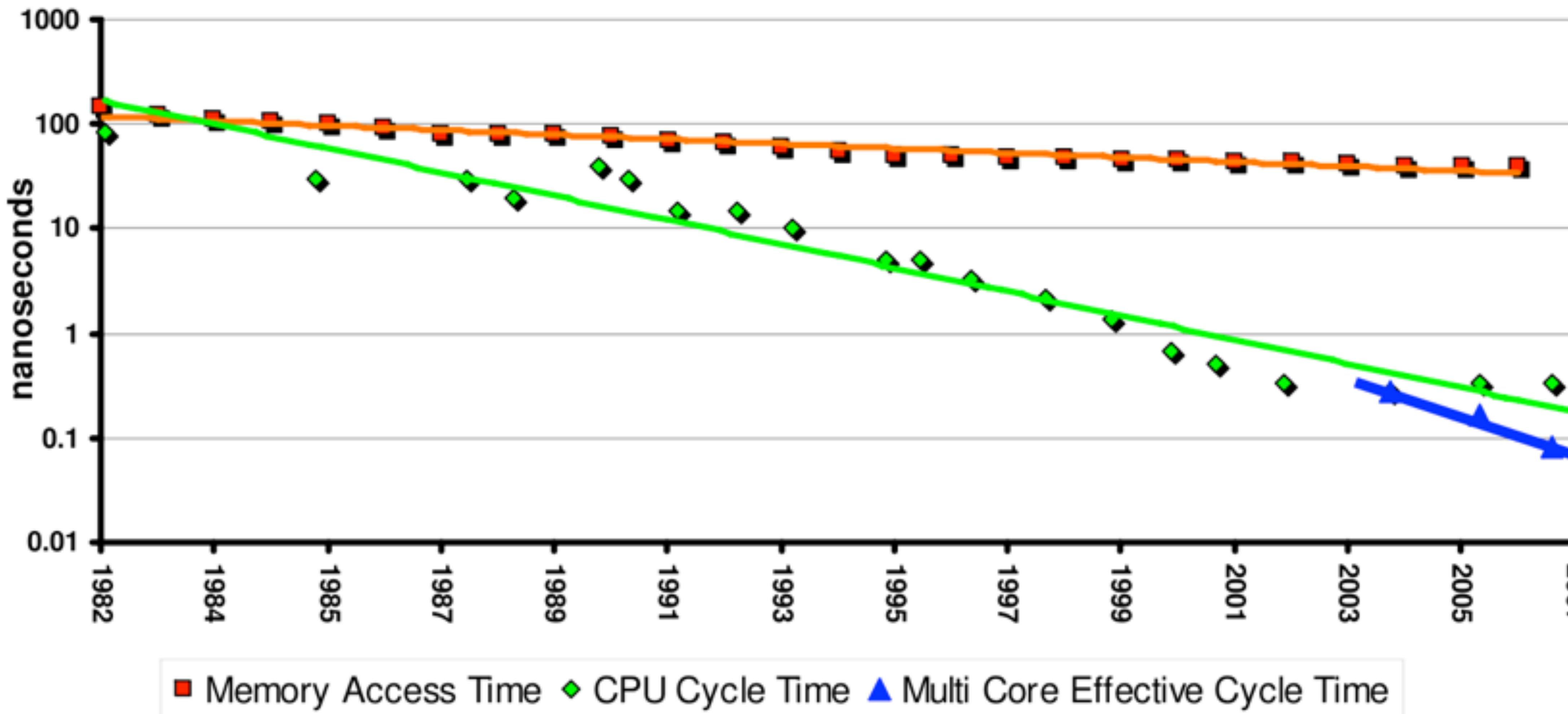


When we have a fast enough compressor
we can get rid of the limitations of the bus bandwidth.

How to get maximum compression performance?

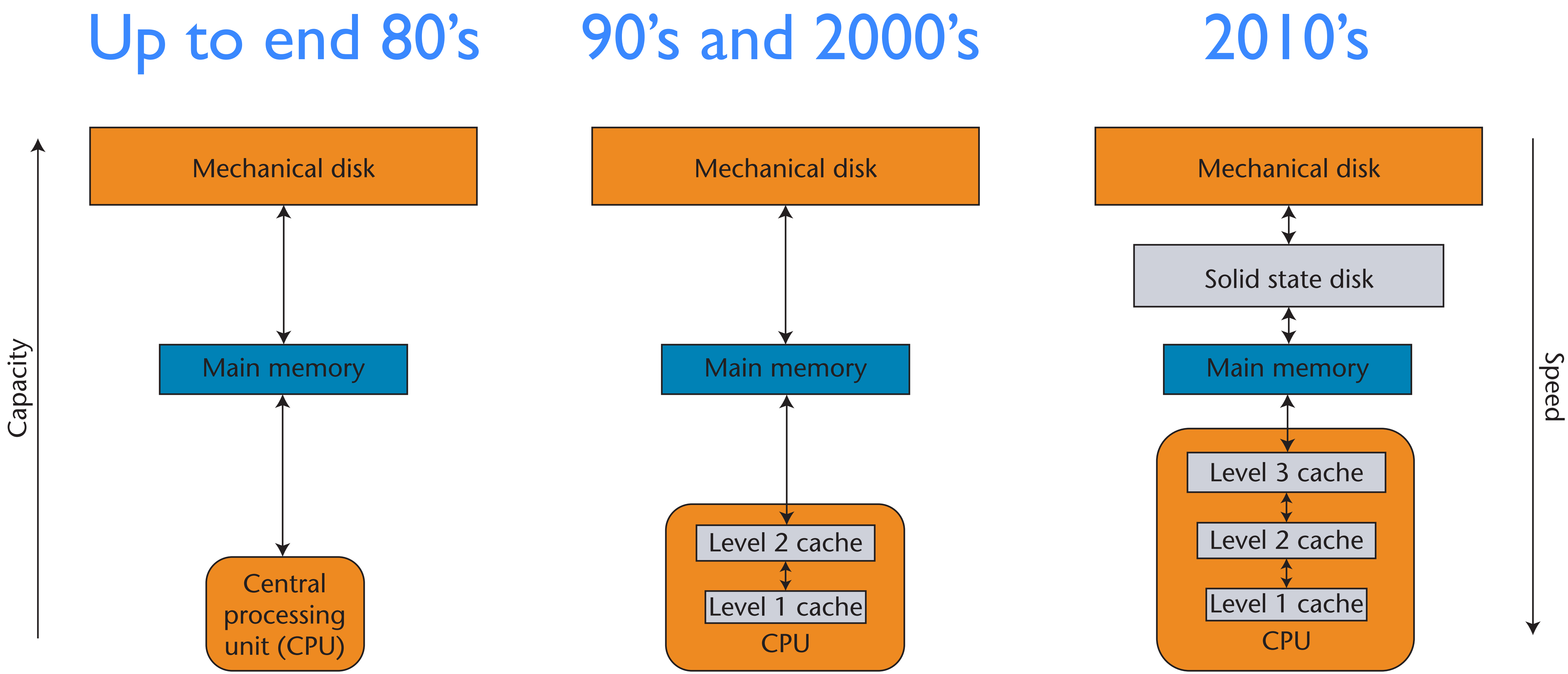
Recent Trends In Computer CPUs

Memory Access Time vs CPU Cycle Time



The gap is wide and still opening!

Computer Architecture Evolution



Hierarchy of Memory
By 2018 (Educated Guess)

HDD (persistent)

SSD SATA (persistent)

SSD PCIe (persistent)

XPoint (persistent)

RAM (addressable)

L4

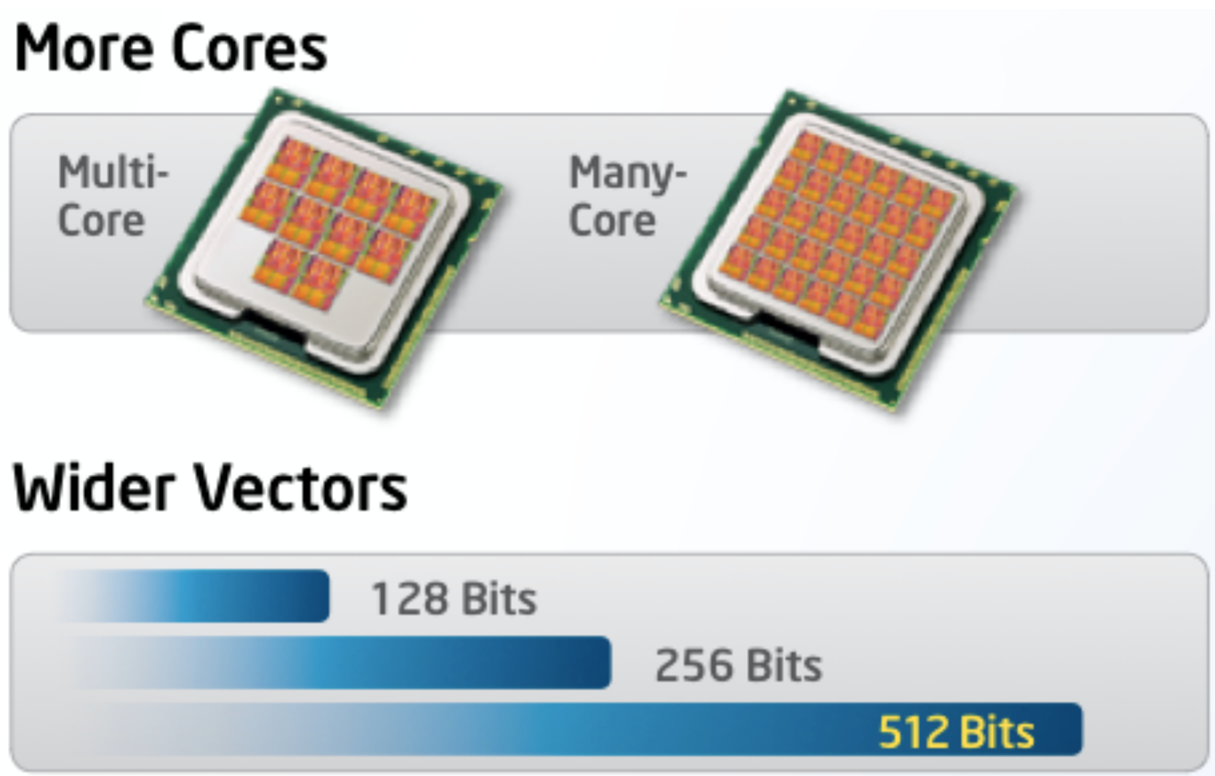
L3

L2

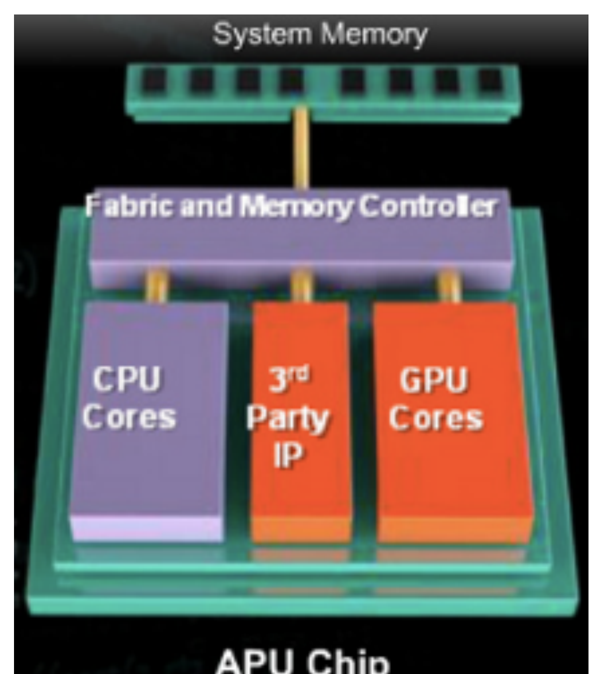
L1

9 levels will be common!

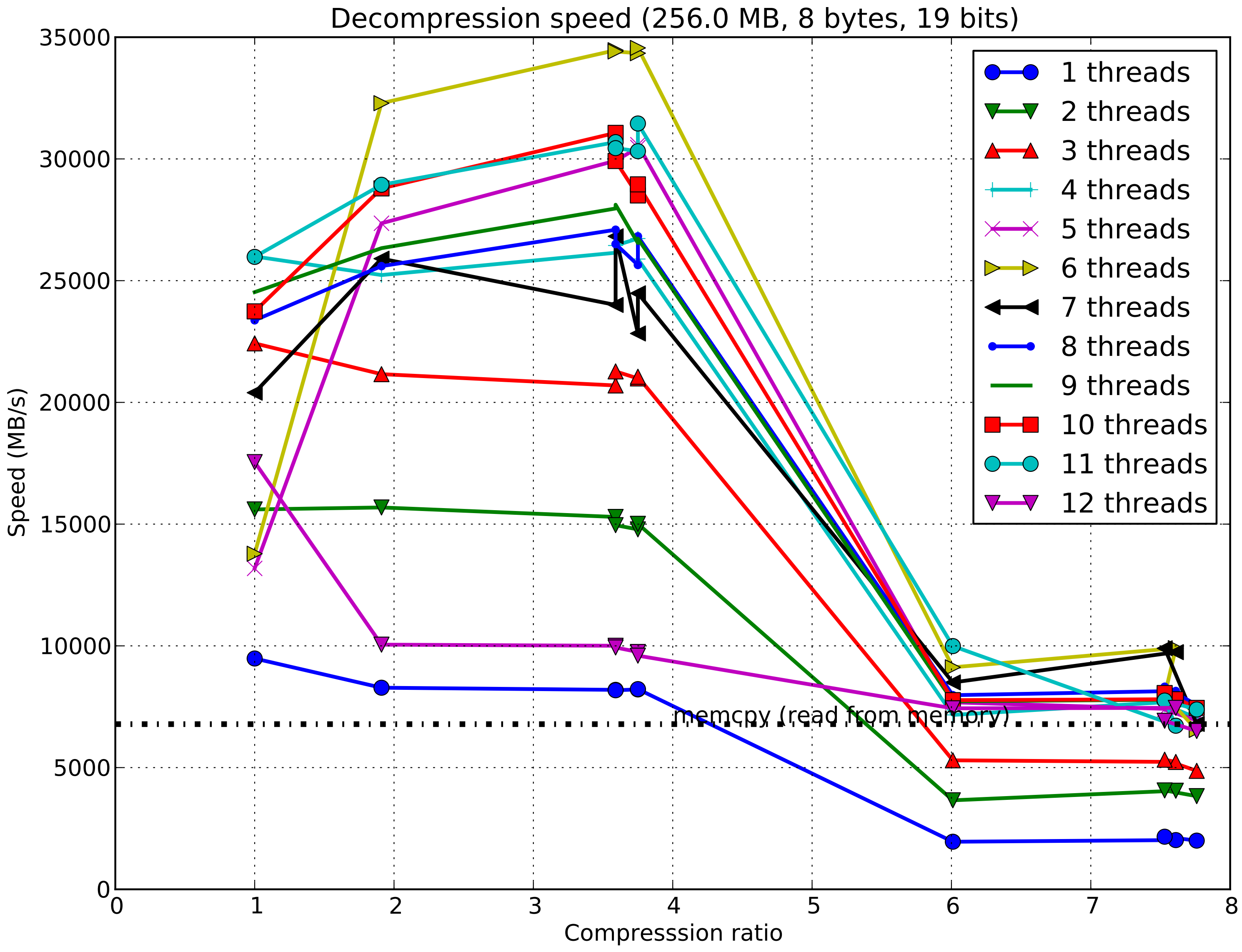
Forthcoming Trends



CPU+GPU Integration

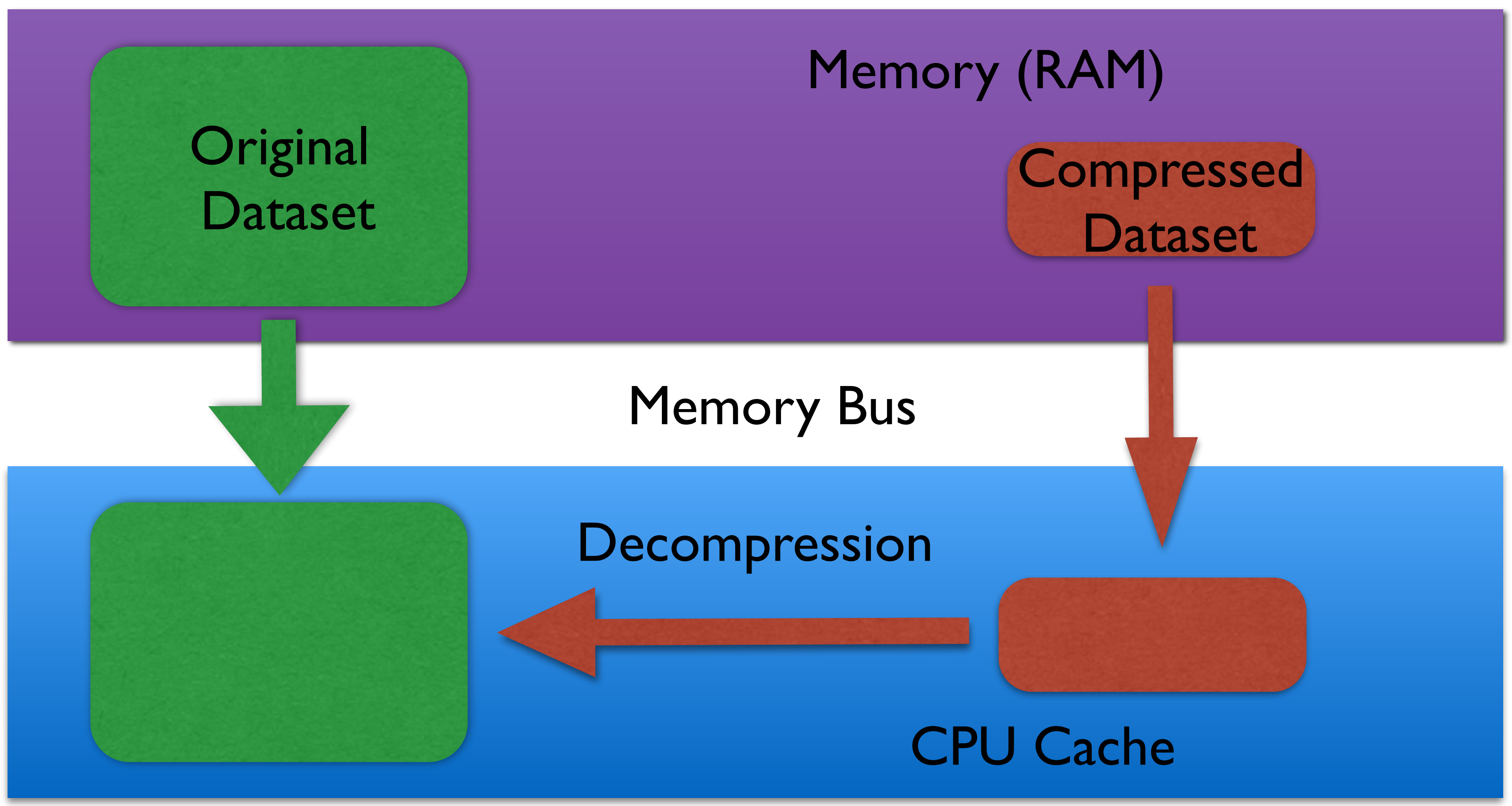


Blosc: Compressing Faster Than *memcpy()*



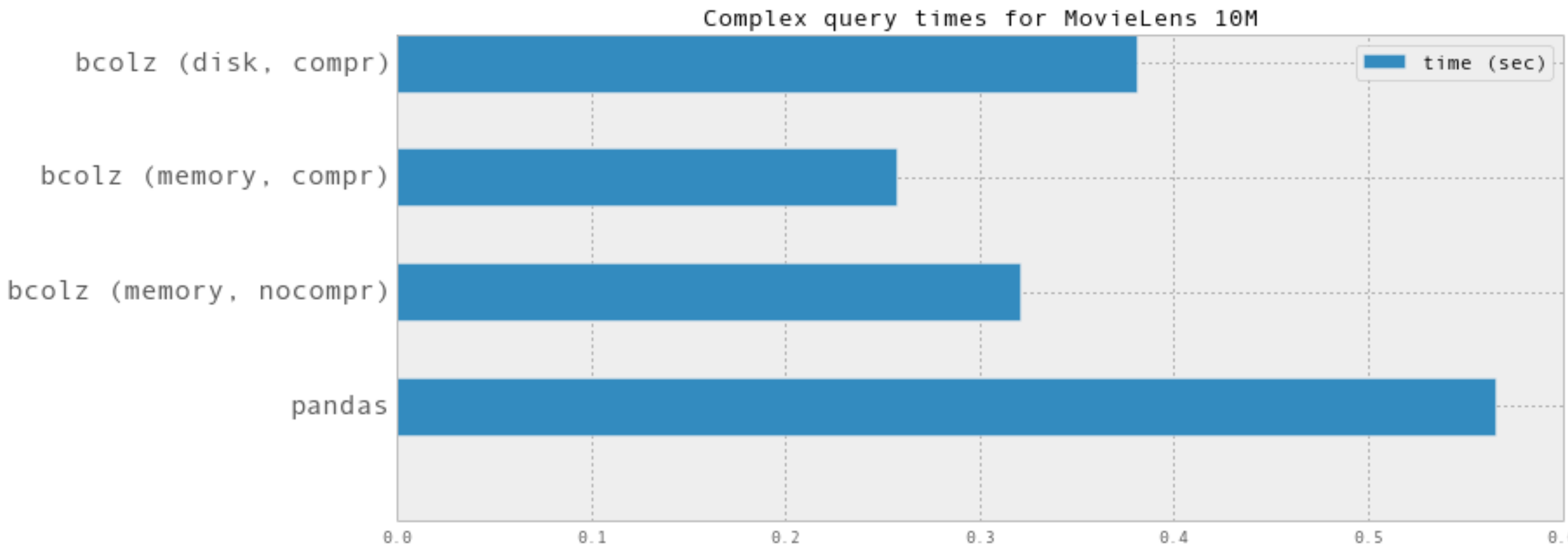
Improving RAM Speed?

Less data needs to be transmitted to the CPU



Transmission + decompression faster than direct transfer?

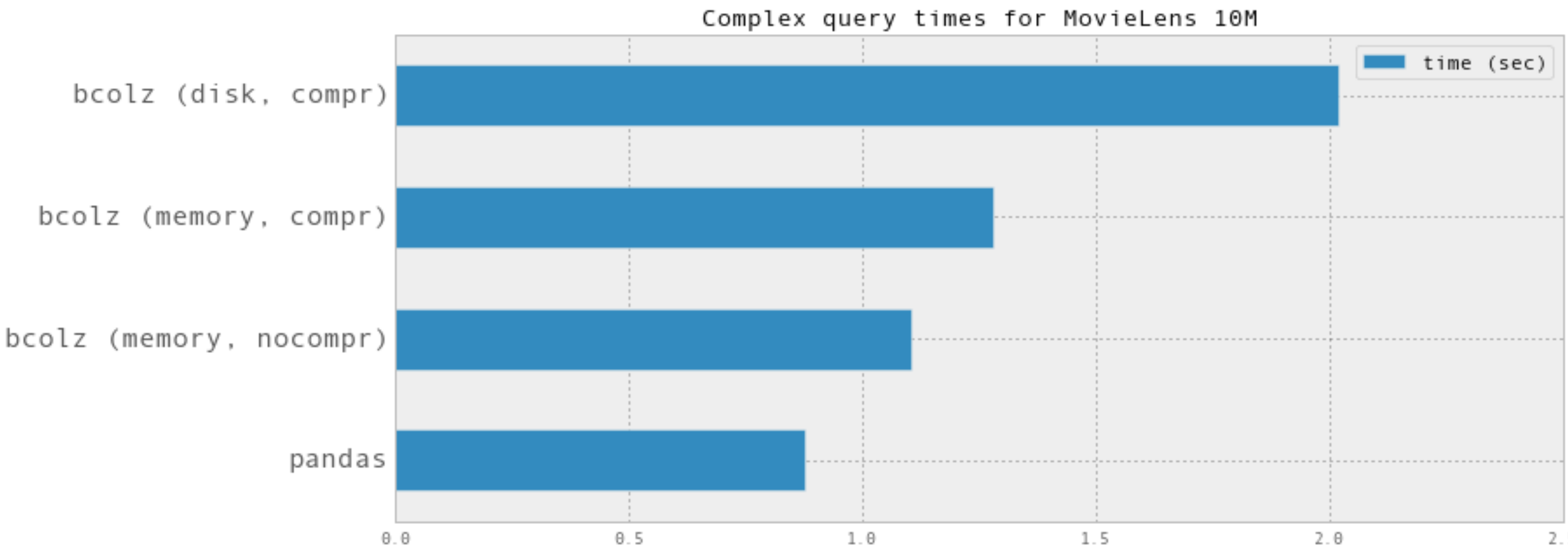
Source: https://github.com/Blosc/movielens-bench



Query Times

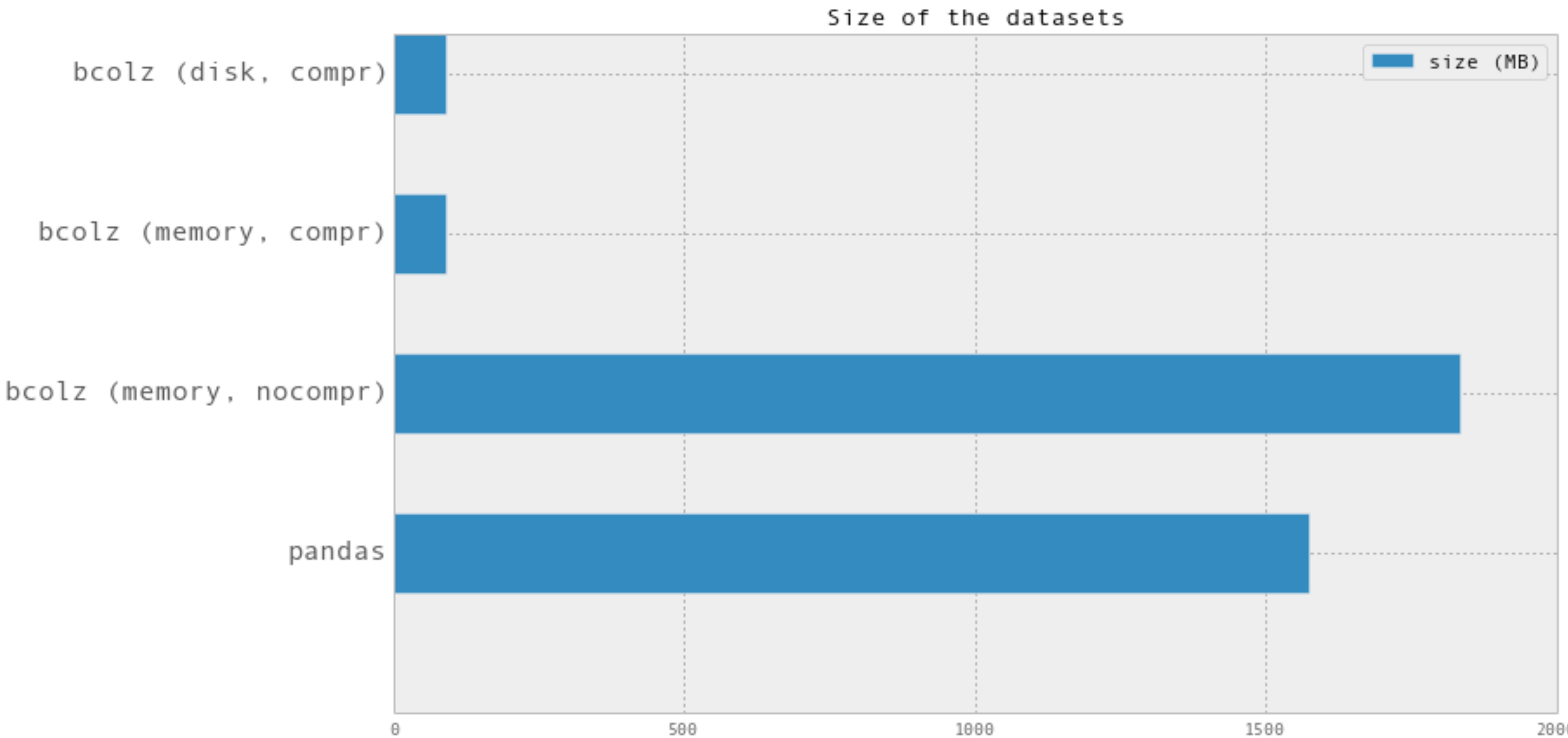
2012 old laptop (Intel Ivy-Bridge, 2 cores)
Compression **speeds** things up

Source: https://github.com/Blosc/movielens-bench



Query Times

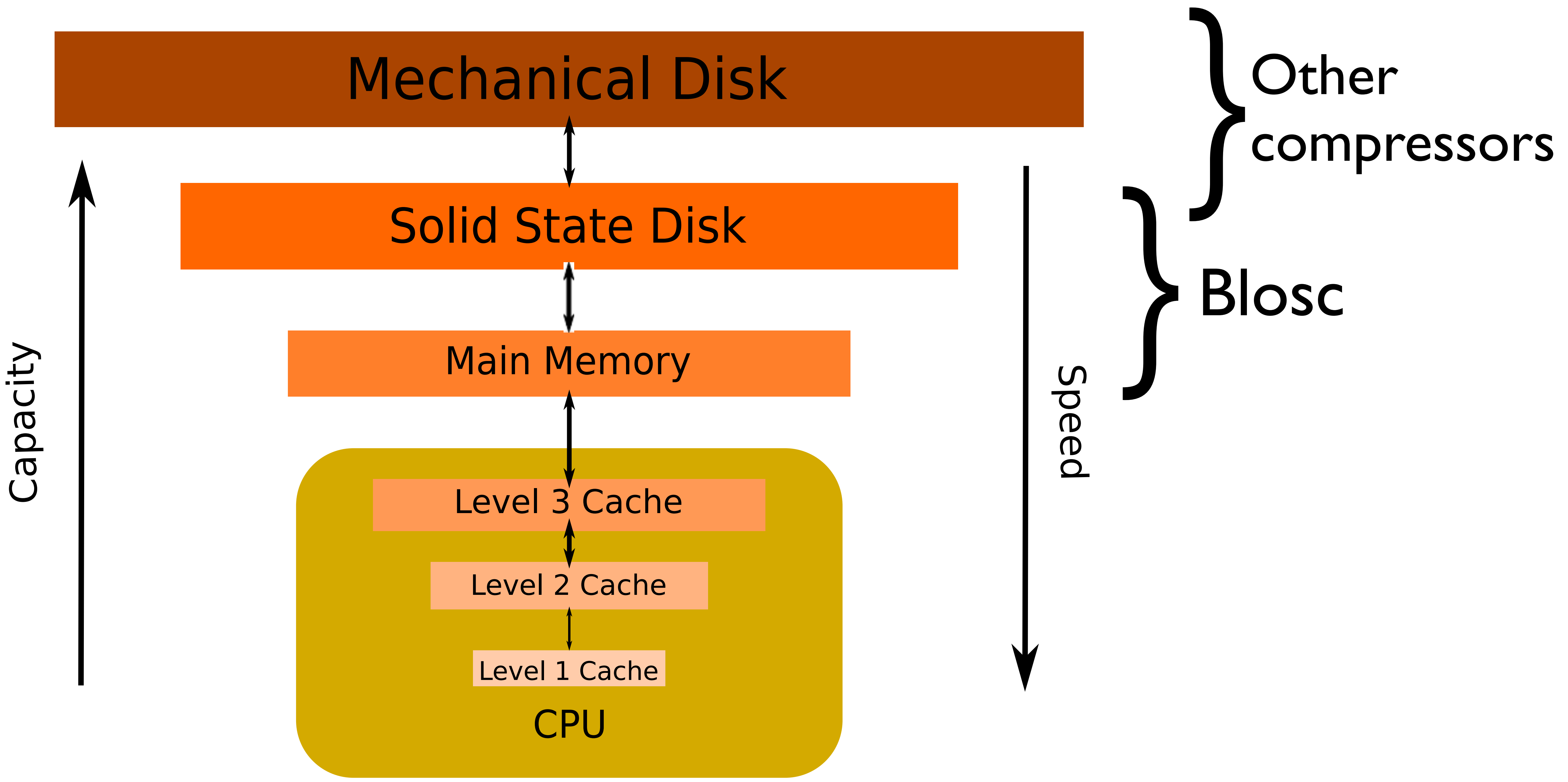
2010 laptop (Intel Core2, 2 cores)
Compression still slow things down



bcolz vs pandas (size)

bcolz can store 20x more data than pandas by using compression

Accelerating I/O With Blosc



“Blosc compressors are the fastest ones out there at this point; there is no better publicly available option that I'm aware of. That's not just ‘yet another compressor library’ case.”

— Ivan Smirnov
(advocating for Blosc inclusion in h5py)

Compression matters!

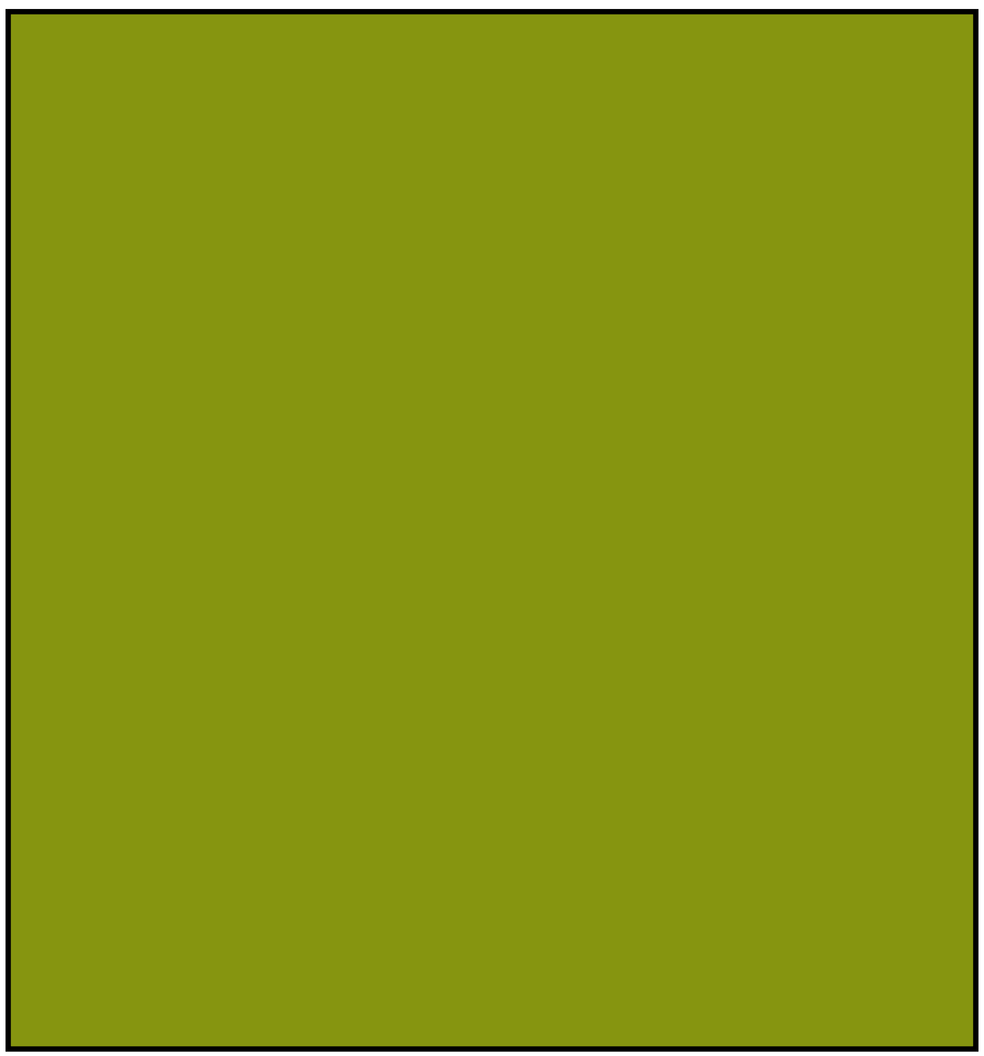
Bcolz: An Example Of Data Containers Applying The Principles Of New Hardware

What is bcolz?

- bcolz provides data containers that can be used in a similar way than the ones in NumPy, Pandas
- The main difference is that data storage is **chunked**, not **contiguous**
- Two flavors:
    - **carray**: homogenous, n-dim data types
    - **ctable**: heterogeneous types, columnar

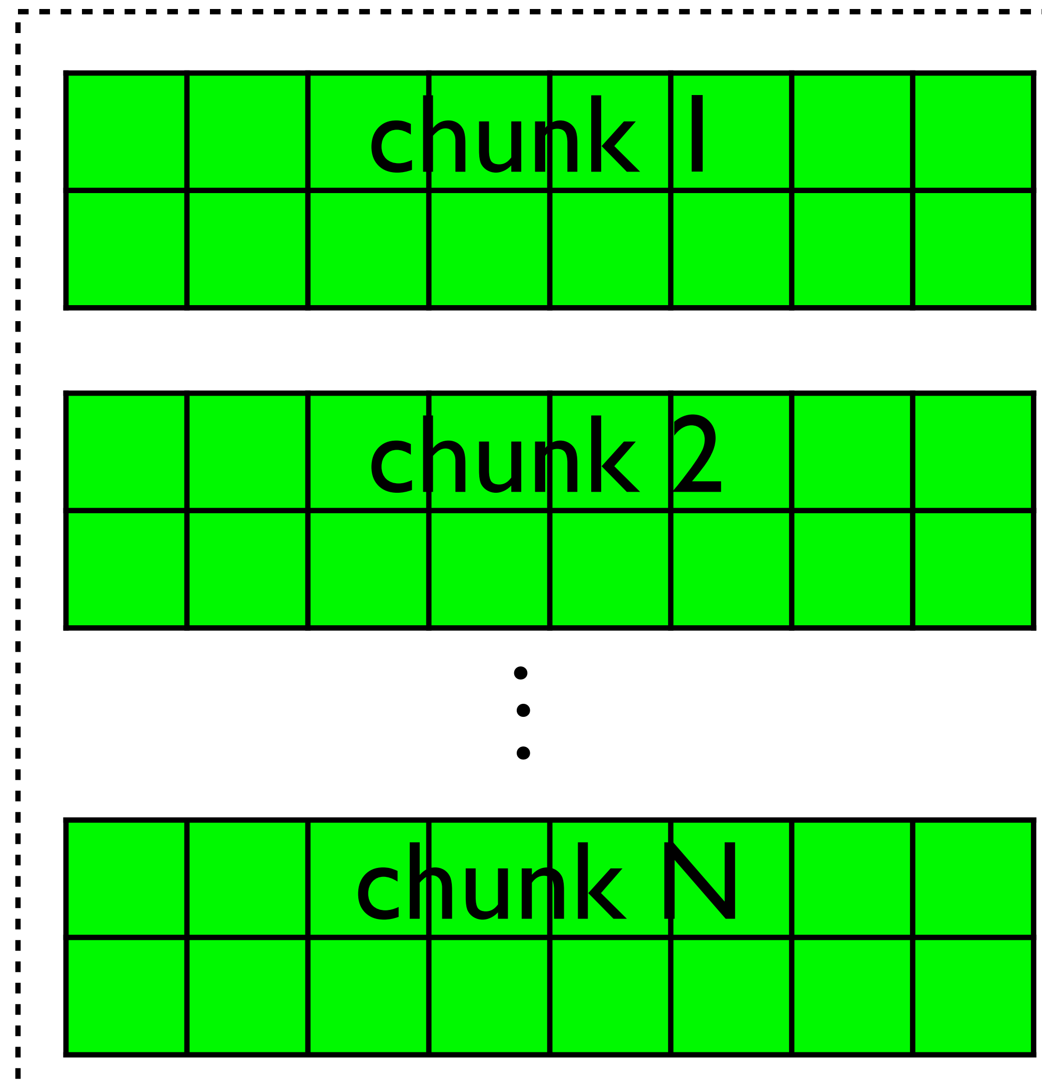
Contiguous vs Chunked

NumPy container



Contiguous memory

carray container

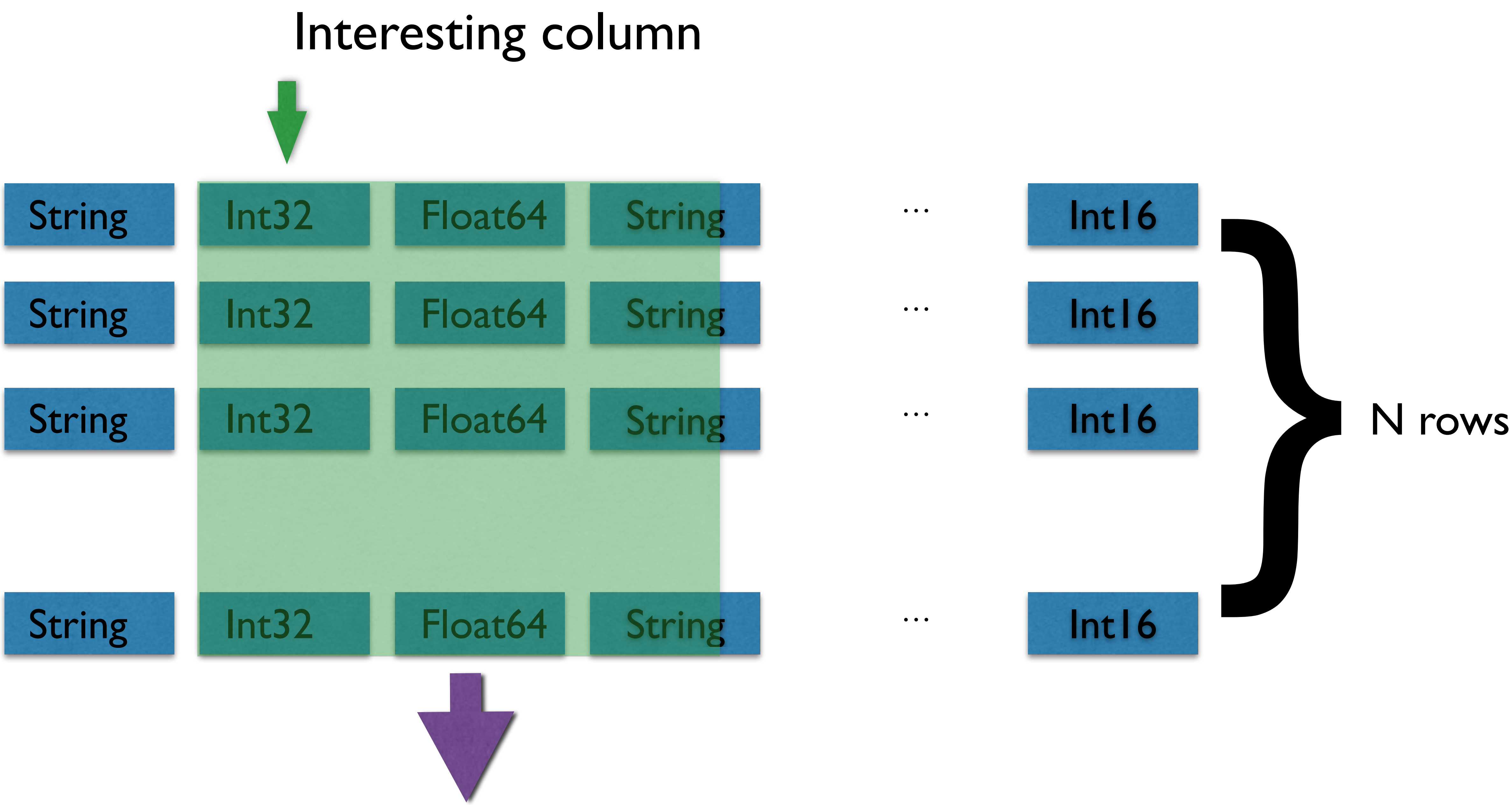


Discontiguous memory

Why Columnar?

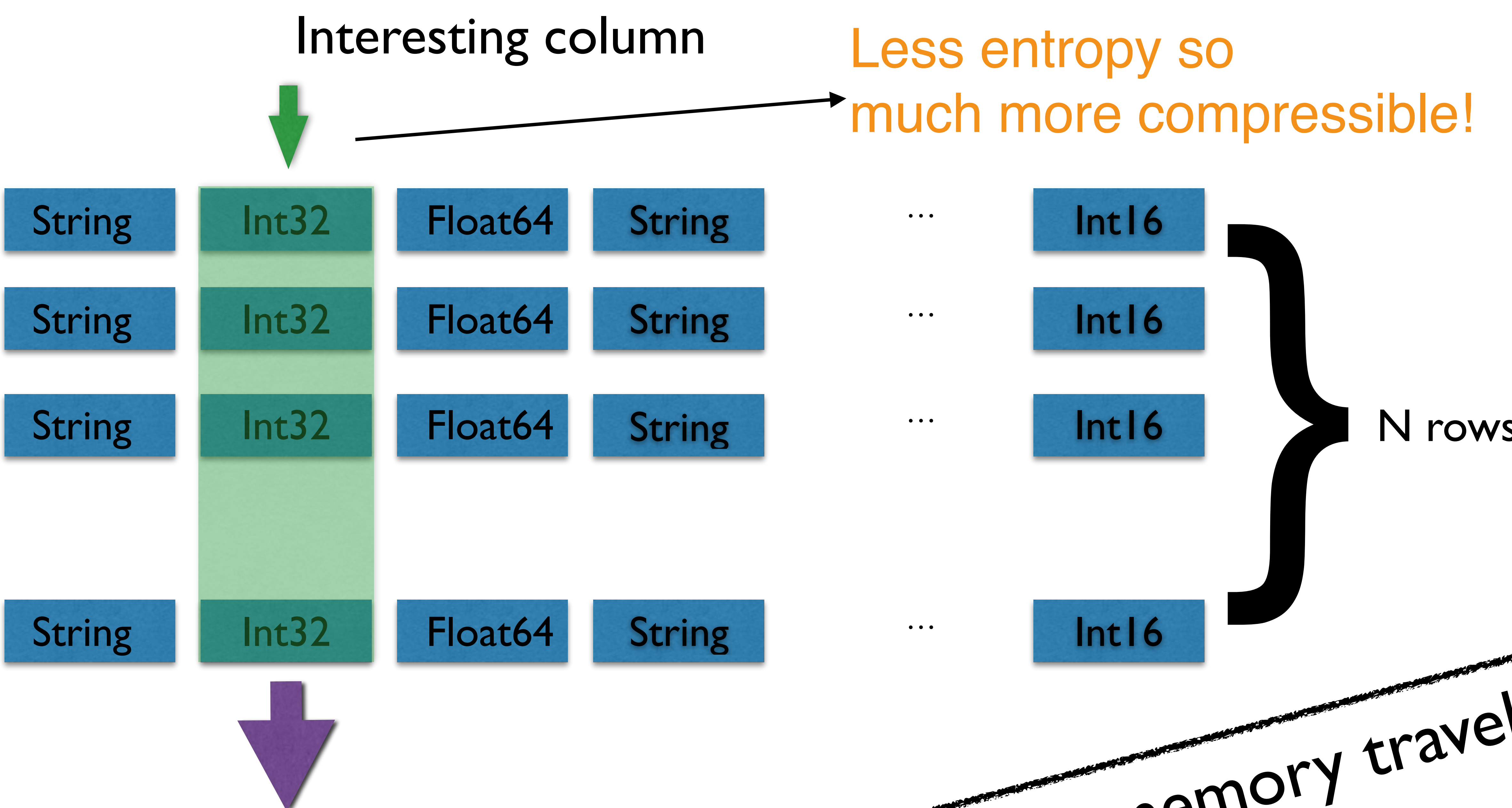
- Because it adapts better to newer computer architectures

In-Memory Row-Wise Table (Structured NumPy array)



Interesting Data: N * 4 bytes (Int32)
Actual Data Read: N * 64 bytes (cache line)

In-Memory Column-Wise Table (bcolz *ctable*)



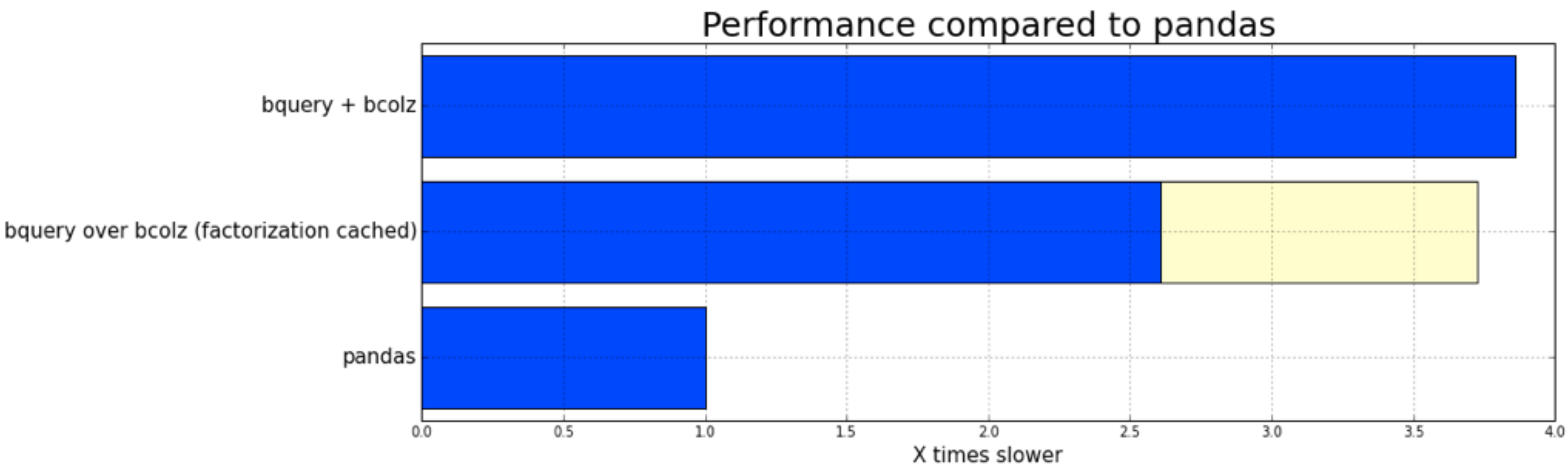
Interesting Data: N * 4 bytes (Int32)
Actual Data Read: N * 4 bytes (Int32)

Less memory travels to CPU!

Some Projects Using bcolz

- Visualfabriq’s bquery (out-of-core groupby’s): https://github.com/visualfabriq/bquery
- Scikit-allel: http://scikit-allel.readthedocs.org/
- Quantopian: http://quantopian.github.io/talks/NeedForSpeed/slides#/

bquery - On-Disk GroupBy

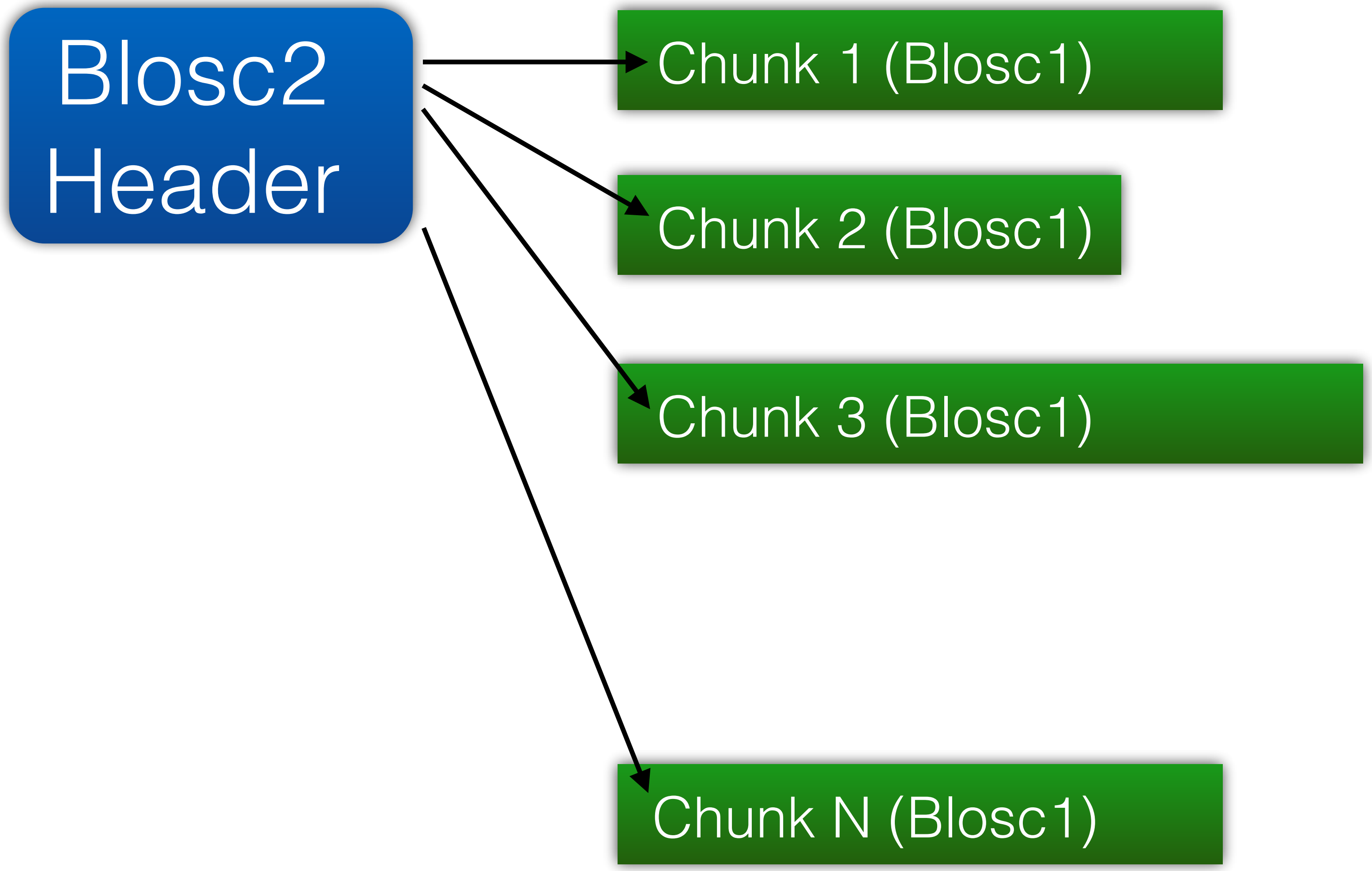


In-memory (pandas) vs on-disk (bquery+bcolz) groupby

“Switching to bcolz enabled us to have a much better scalable architecture yet with near in-memory performance”
— Carst Vaartjes, co-founder visualfabriq

“The future for me clearly involves lots of block-wise processing of multidimensional **bcolz** carrays"”

–Alistair Miles
Head of Epidemiological Informatics for the Kwiatkowski group. Author of ***scikit-allel****.*



Introducing Blosc2

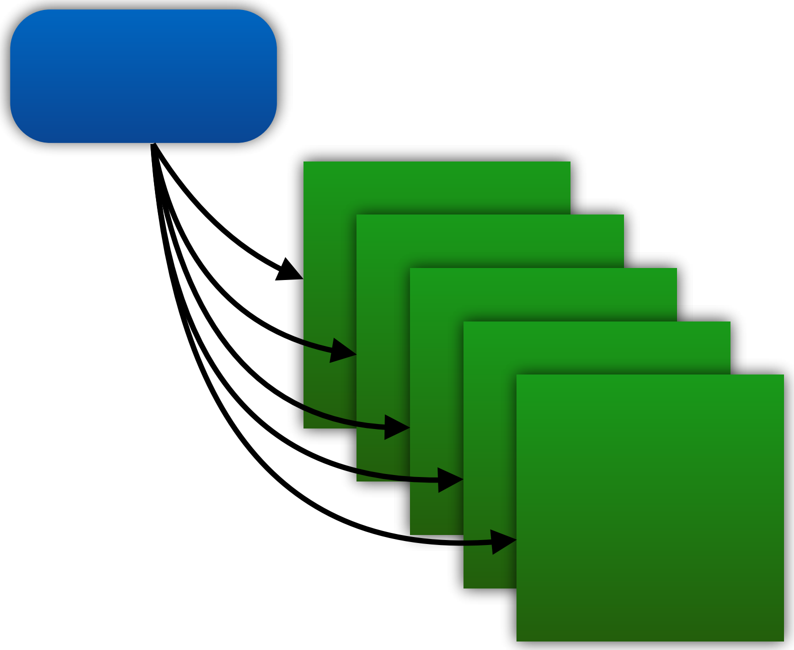
Next generation of Blosc

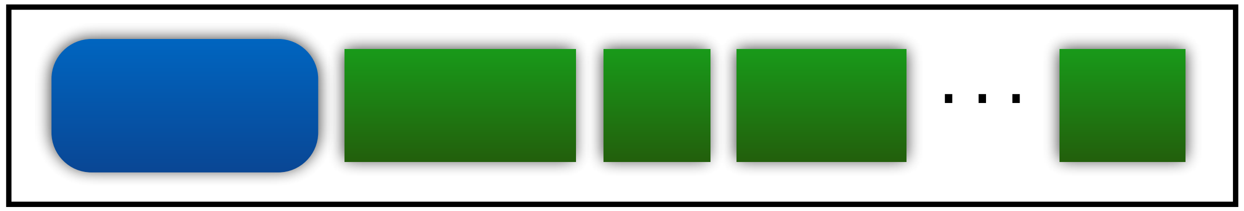
Planned features for Blosc2

- Looking into inter-chunk redundancies (delta filter)

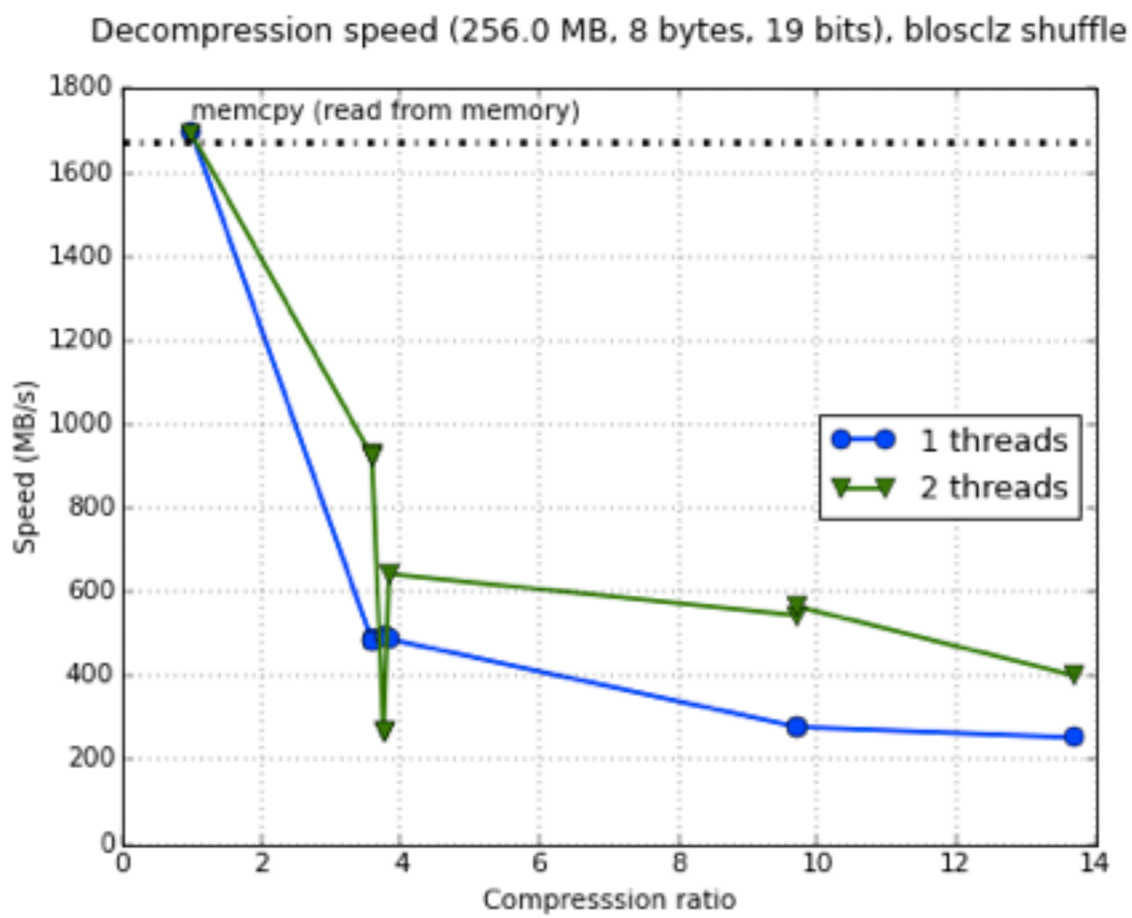
- Support for more codecs (Zstd is there already!)

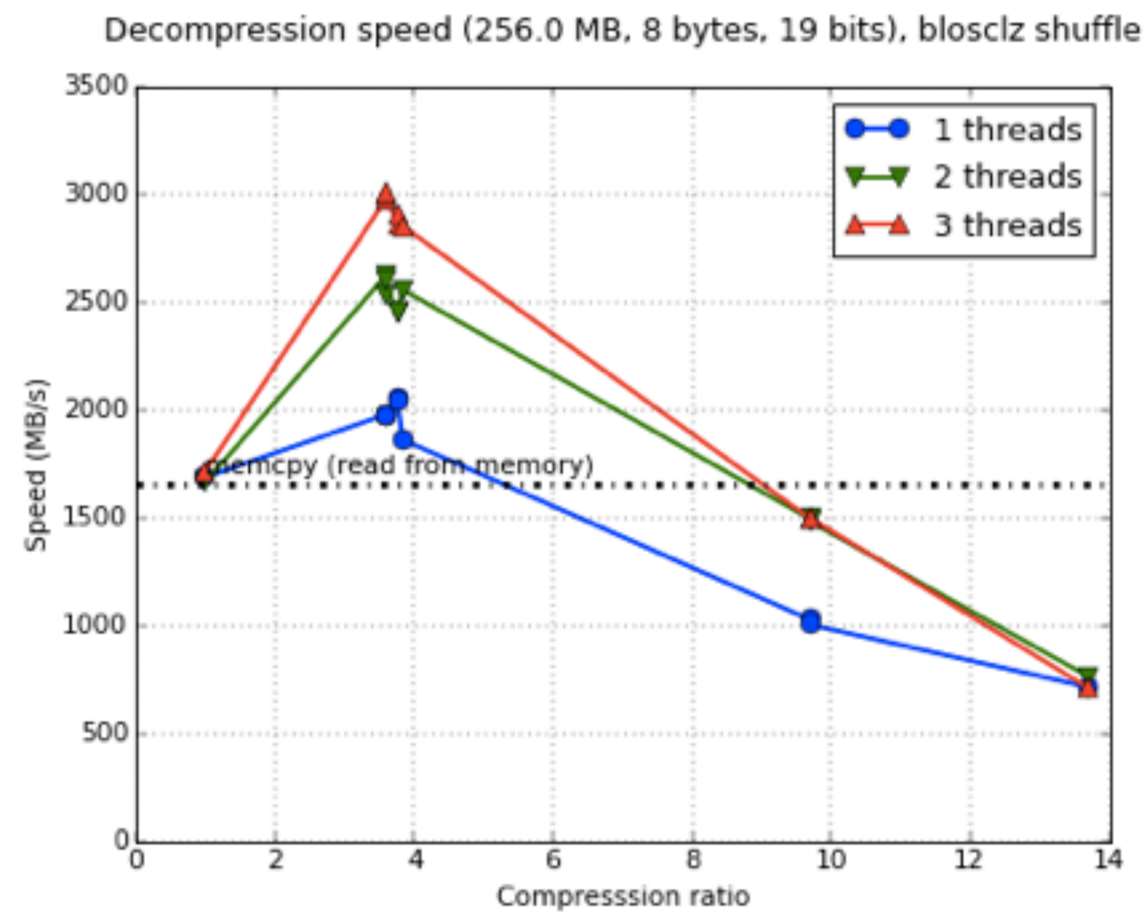
- Serialized version of the super-chunk (disk, network)





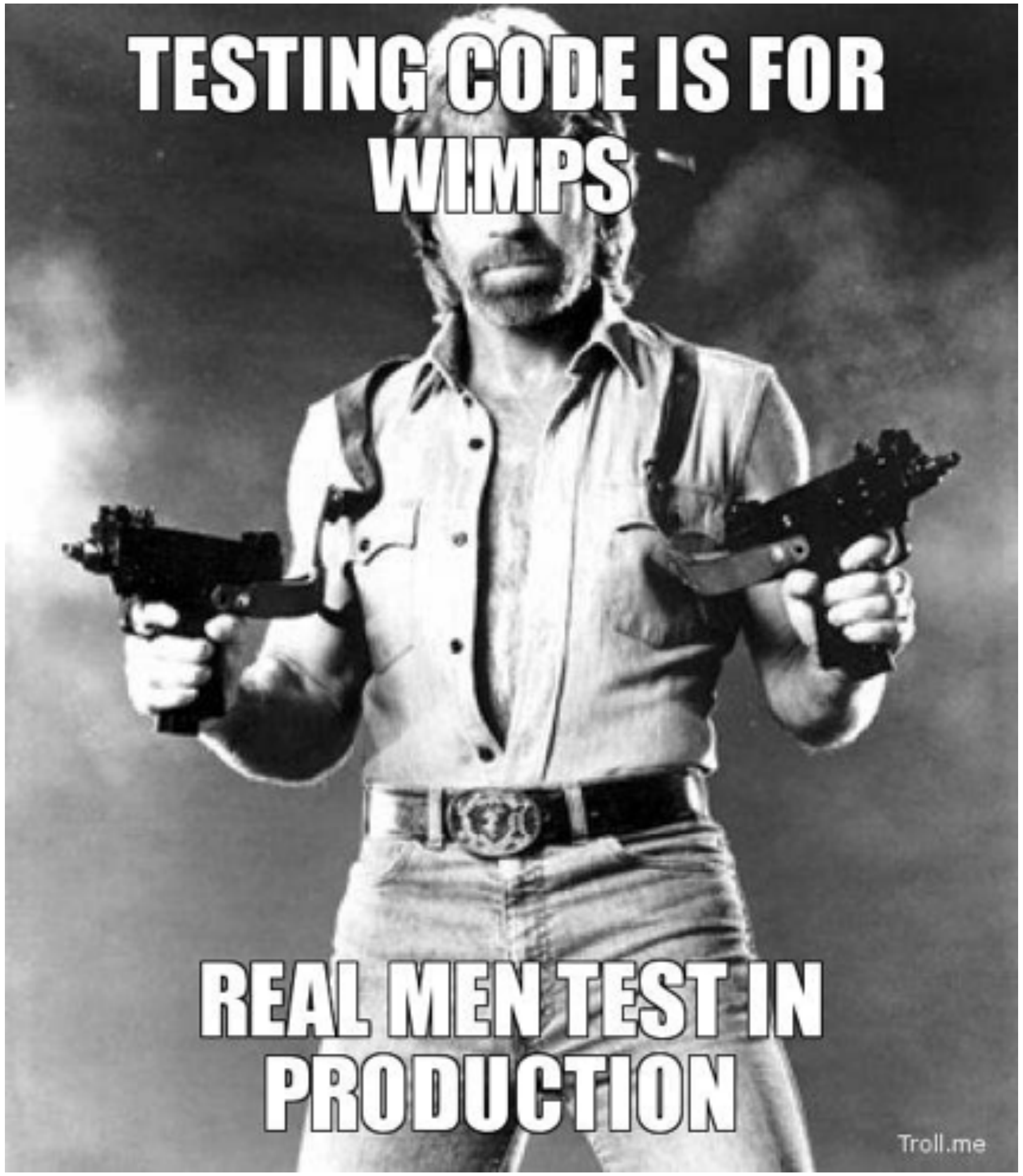
Not using NEON	Using NEON





- At 3 GB/s, Blosc2 on ARM achieves one of the best bandwidth/Watt ratios in the market

- Profound implications for the density of data storage devices (e.g. arrays of disks driven by ARM)



Blosc2 has its own repo

https://github.com/Blosc/c-blosc2

Meant to be usable only when heavily tested!

`(bcolz2 will follow after Blosc2)`

Closing Notes

- Due to the evolution in computer architecture, the compression can be effective for two reasons:
  - We can work with more data using the same resources.
  - We can reduce the overhead of compression to near zero, and even beyond than that!

“In science, one can learn the most by studying what seems the least.”

–Marvin Minsky

¡Gracias!