Investigations on hardware compression of IBM Power9 processors

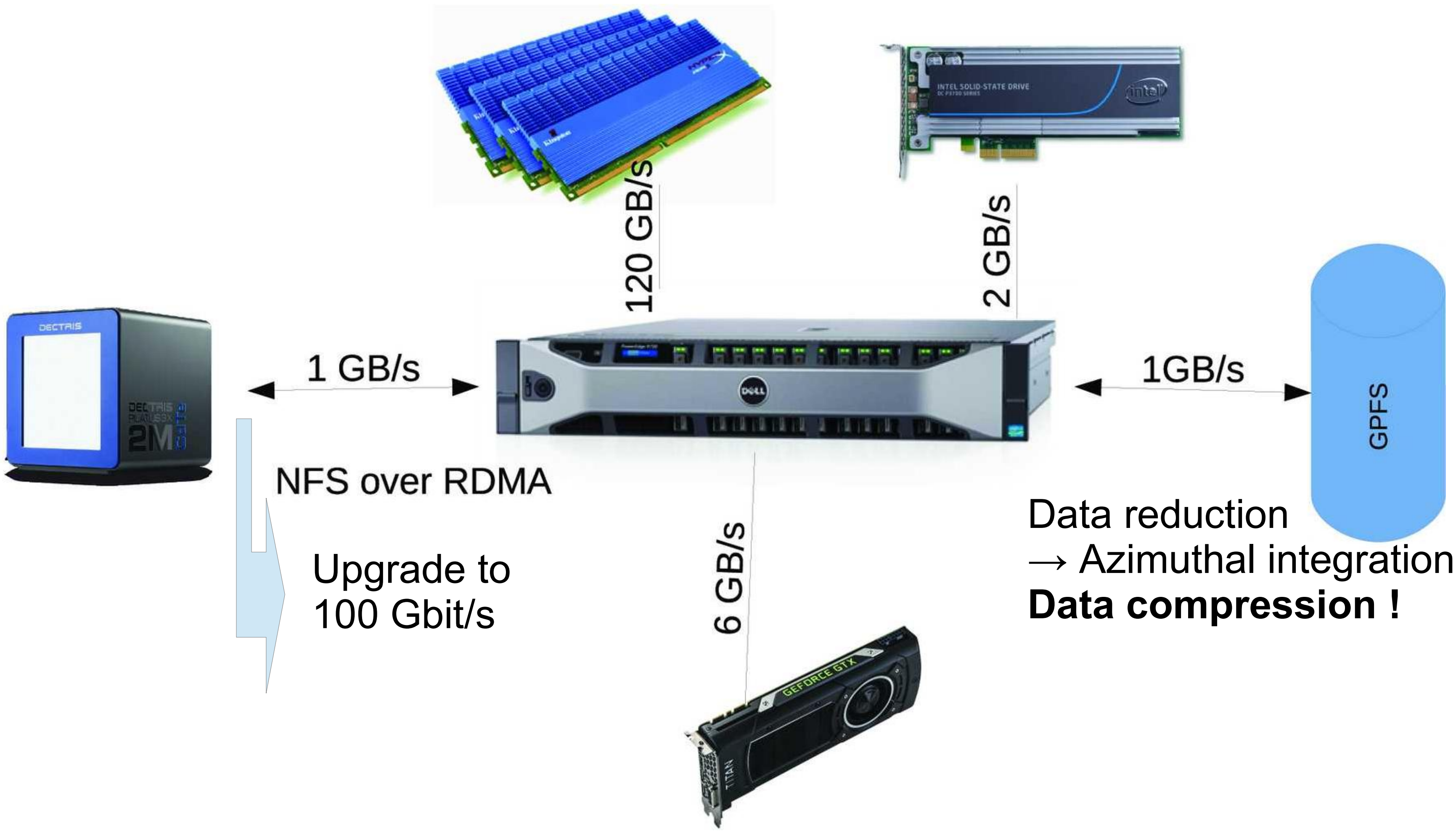
Jérome Kieffer, Pierre Paleo, Antoine Roux, Benoît Rousselle

Outline

- **The bandwidth issue at synchrotrons sources**
- **Presentation of the evaluated systems:**
  - Intel Xeon vs IBM Power9
  - Benchmarks on bandwidth
- **The need for compression of scientific data**
  - Compression as part of HDF5
  - The hardware compression engine NX-gzip within Power9
  - Gzip performance benchmark
  - Bitshuffle-LZ4 benchmark
  - Filter optimizations
  - Benchmark of parallel filtered gzip
- **Conclusions**
  - on the hardware
  - on the compression pipeline in HDF5

Bandwidth issue at synchrotrons sources

Data analysis computer with the main interconnections and their associated bandwidth.

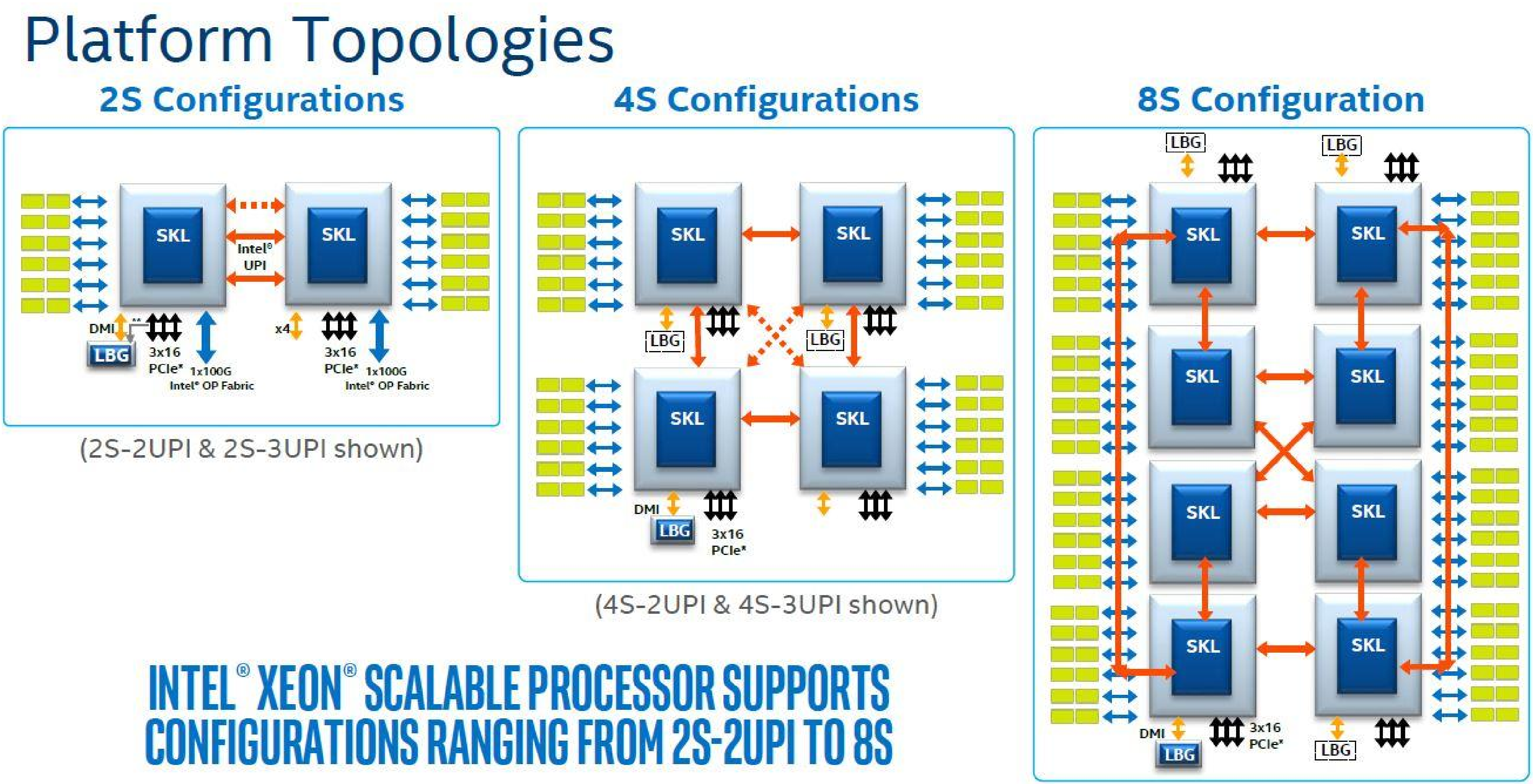


Figures from former generation of servers

JOURNAL OF SYNCHROTRON RADIATION

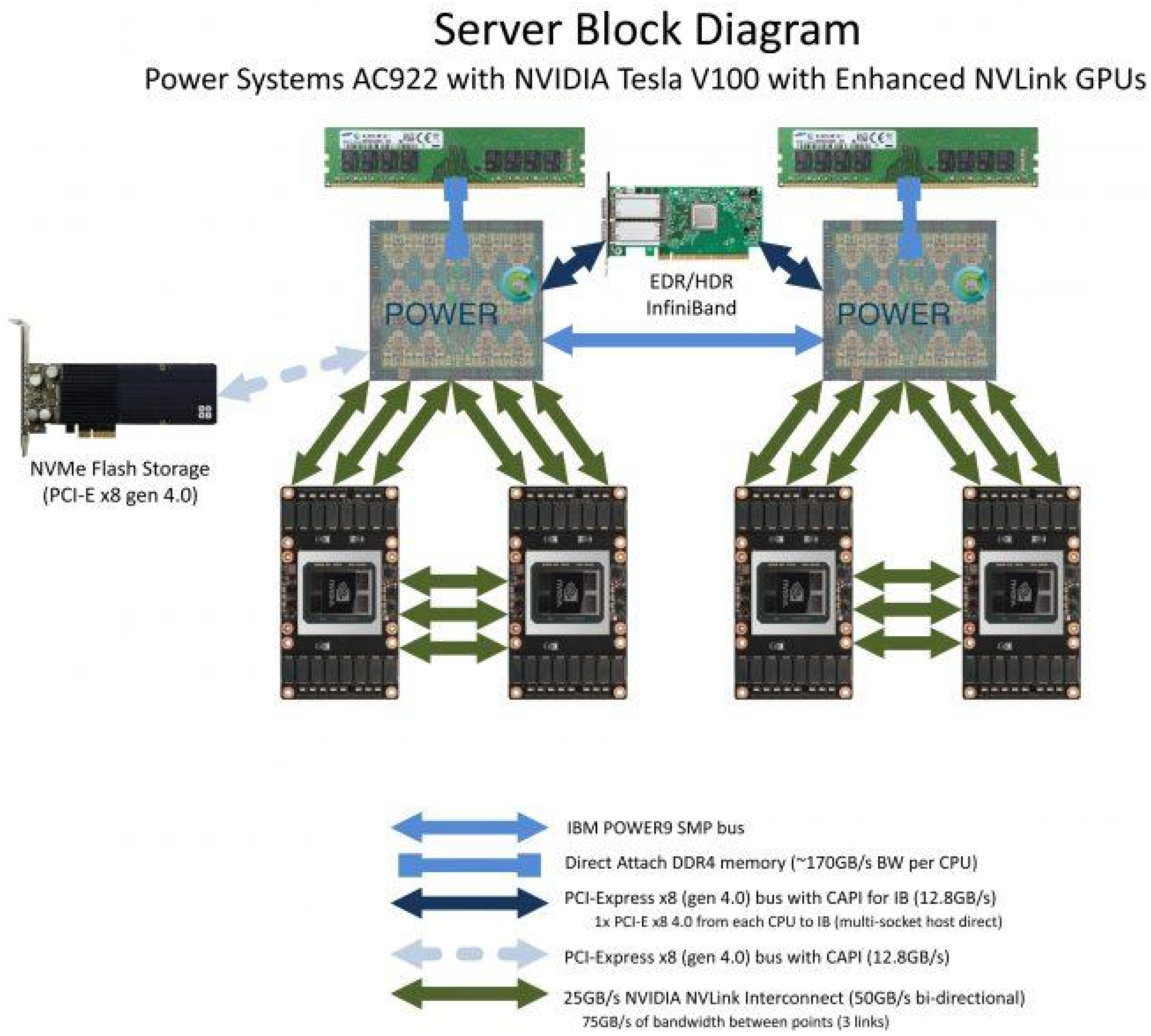
Kieffer *et al.*
Volume 25 | **Part 2** | March 2018 | Pages 612–617 | 10.1107/S1600577518000607

Topologies of Intel Xeon servers in 2019



Source: intel.com

Architecture of the AC922 server from IBM featuring Power9



Bandwidth measurement: Xeon vs Power9

Computer	Dell R840	IBM AC922
Processor	4 Intel Xeon (12 cores) 2.6 GHz	2 IBM Power9 (16 cores) 2.7 GHz
Cache (L3)	19 MB	8x 10 MB
Memory channels	4x 6 DDR4	2x 8 DDR4
Memory capacity	→ 3TB	→ 2TB
Memory speed theory	512 GB/s	340 GB/s
Measured memory speed	160 GB/s	270 GB/s
Interconnects	PCIe v3	PCIe v4 NVlink2 & CAPI2
GP-GPU co-processor	2Tesla V100 PCIe v3	2Tesla V100 NVlink2
Interconnect speed CPU ↔ GPU	12 GB/s	48 GB/s

Strength and weaknesses of the OpenPower architecture

While amd64 **is today's *de facto* standard in HPC, it has a few competitors:** arm64, ppc64le **and to a less extend** riscv **and** mips64.

- **Strength of IBM Power9 vs Intel Xeon:**
  - Huge bandwidth everywhere: memory, Nvlink2, PCiev4, OpenCAPI
  - Easy to recompile since the Power9 is little-endian
  - Open source everywhere, down to the architecture (ISA)
  - Runs the two fastest computer in the world: *Summit* & *Sierra*
  - Competitive in price

- **Weaknesses of IBM Power9 vs Intel Xeon:**
  - Much smaller user base
  - *Virtually No* commercial software available
  - Limited size vector instruction set. *ALTIVEC* ≈ *SSE2* 128bits SIMD
  - Less optimized code

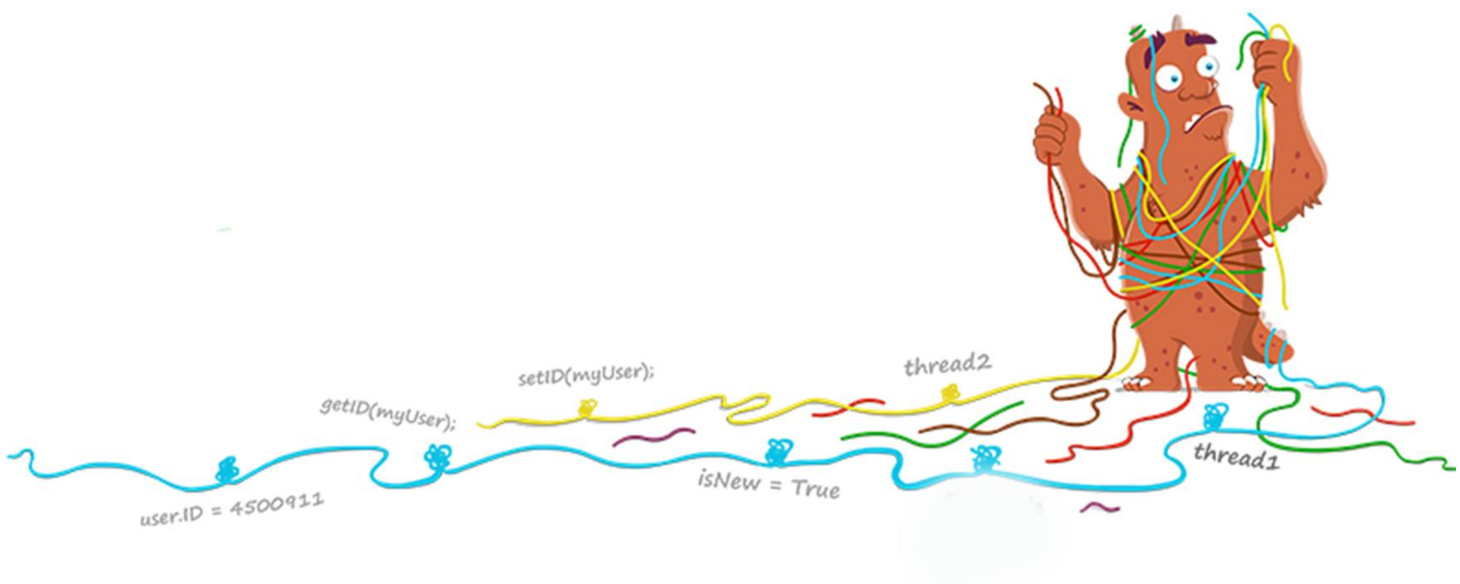
Compression of scientific data

Use-case: images acquired by fast 2D detectors

The need for compression of scientific data

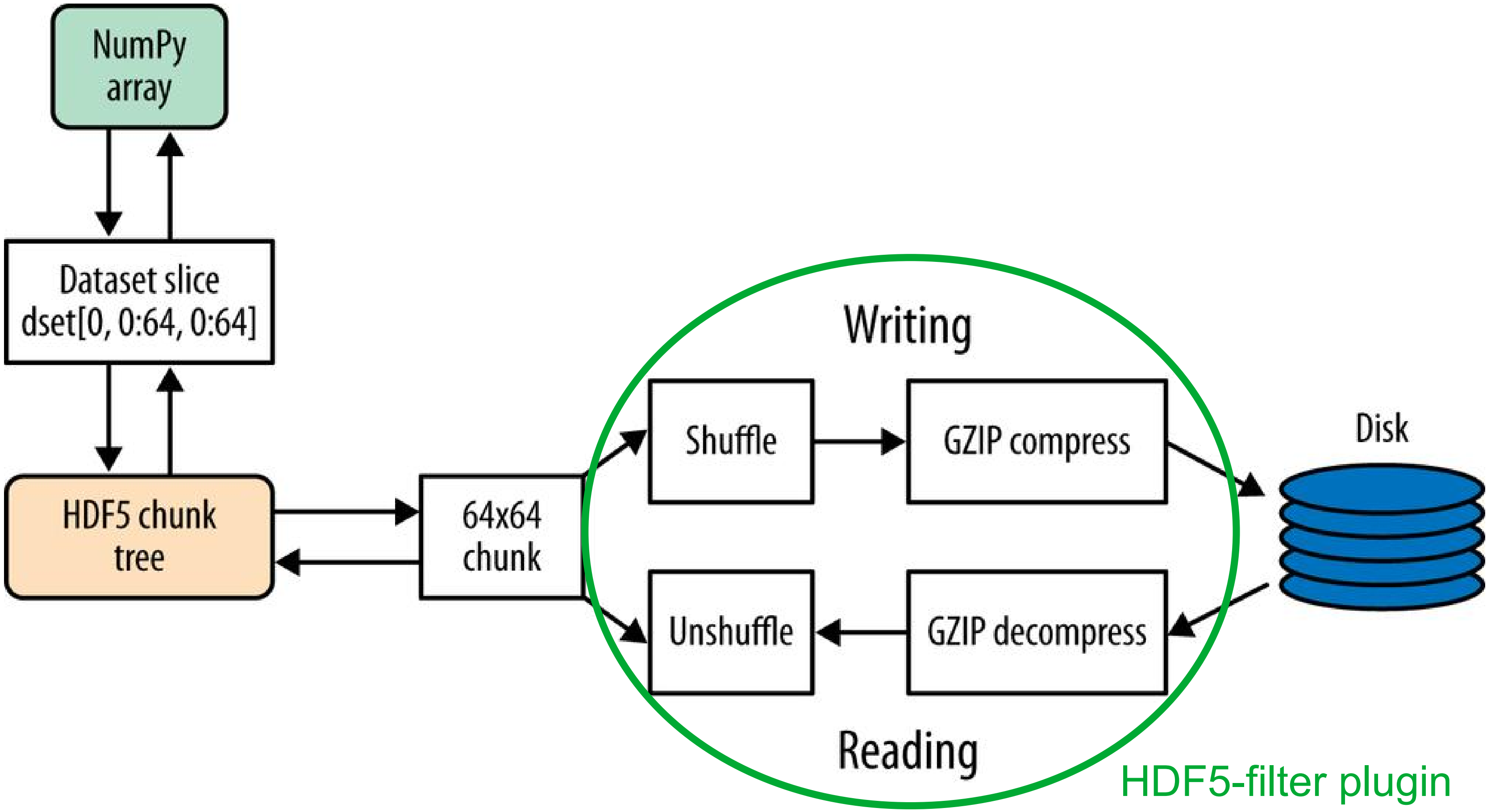
Especially true for large raw data coming directly from detector:

- **Unbiased:** lossy compression must not bias the data
- **Nearly lossless:** must not decrease the sensitivity of the data.
- **Fast decompression** decompress must be faster than I/O.
- **Threaded:** multi-threaded (de-)compression for performance
- **Thread-safe:** one day, HDF5 may become multi-threaded (we all hope)



K.Masui et al. / Astronomy and Computing 12 (2015)181–190

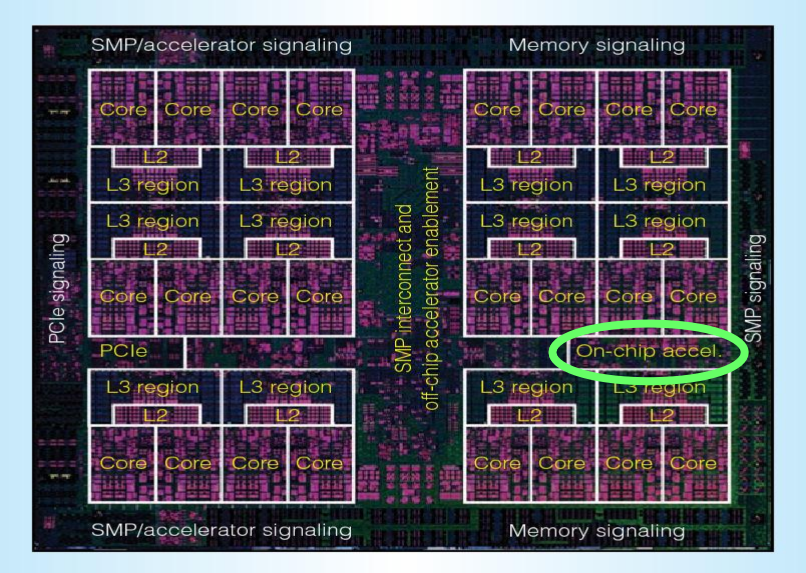
Compression within the HDF5 library: the gzip case



“The SHUFFLE filter is […] very, very fast (negligible compared to the compression time)”

Python and HDF5 by Andrew Collette

The NX hardware accelerator of Power9

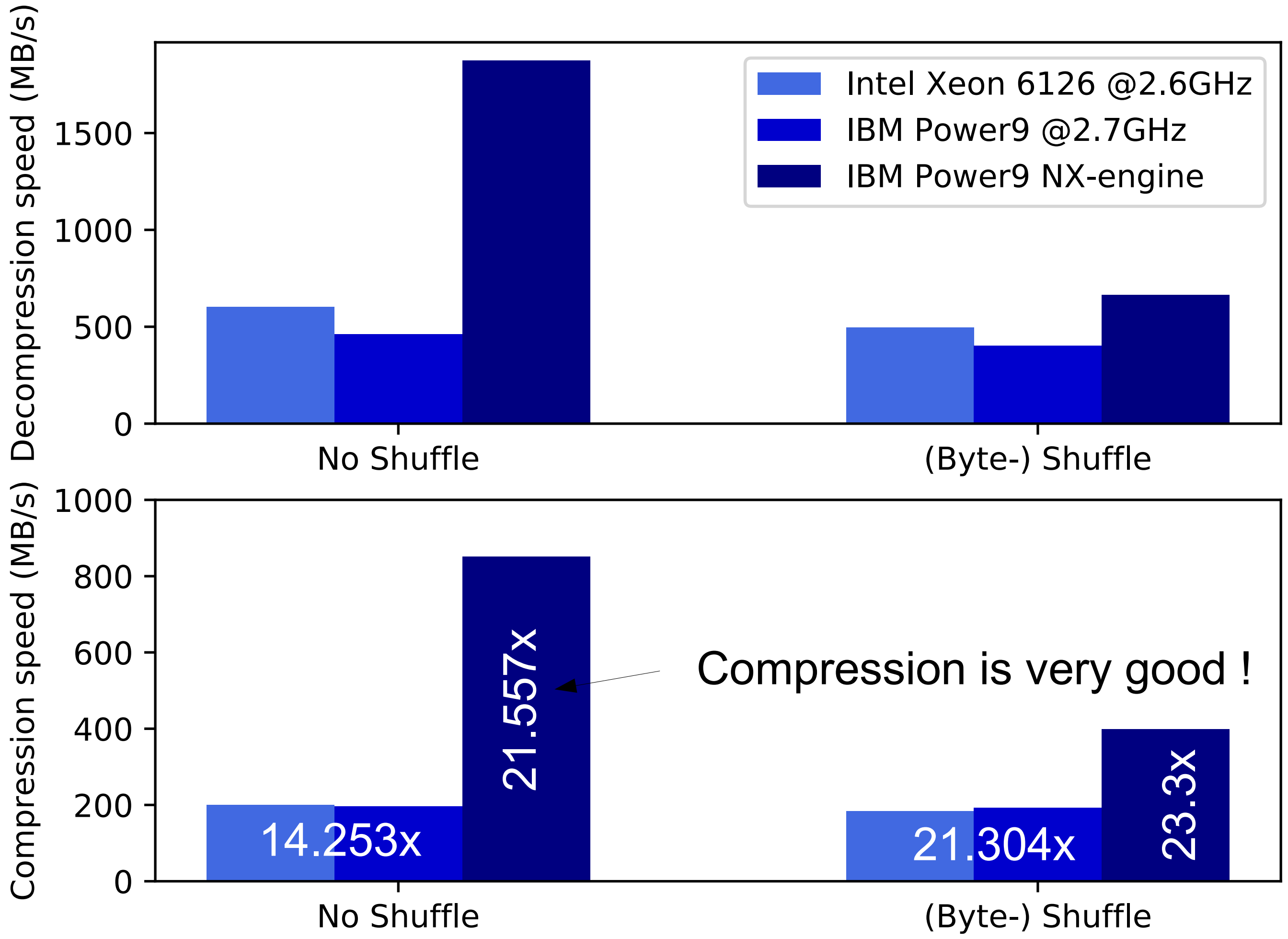


- **One NX-engine per Power9 processor**
  - Industry standard gzip (deflate)
  - Up to 16 GB/s of gzip or gunzip
  - Source code available: https://github.com/abalib/power-gzip.git
  - Just LD_PRELOAD=libnxz.so
  - Works out of the box with HDF5

- **Example used for the benchmark:**
  - Lysozyme dataset provided by Dectris for their Eiger4M
  - 1800 frames of 2167x2070 uint32 (4 bytes/pixel)
    30 GByte raw data
  - Initially compressed with first generation LZ4 HDF5 plugin
    5GByte compressed data (6.23x compression ratio)

Performance of the NX-compressor used with HDF5

- **Software: libhdf5 1.10.5 with *gzip* compression level 1:**
  - Only 1 core is used: HDF5 is single threaded
  - The shuffle filter kills the performances of the NX-engine

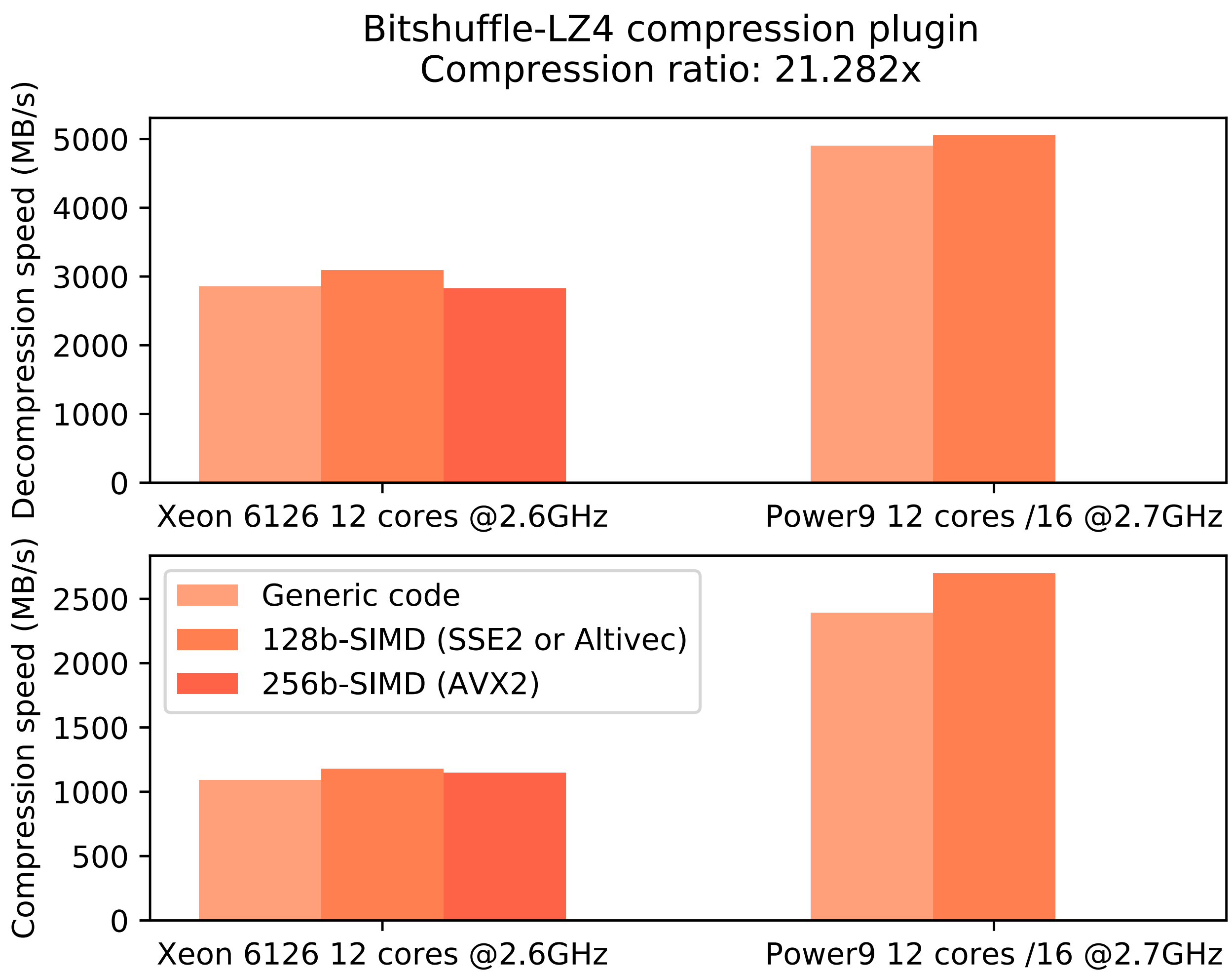


HDF5 filter plugins

- **Importance of the compressor stage:**
  - Bzip2, gzip, lz4, … new compressors are under development
- **Importance of the pre-filter stage:**
  - Shuffle ! Bitshuffle ? Delta ?
  - Issue for building the HDF5 plugin (acces to datatype size)
- **The Blosc library (→ talk from Francesc Alted)**
  - HDF5 plugin already exists
  - “Raw” filters and compressors are available
    - **Currently C-blosc2 beta4**
    - **SIMD implementation are available for better performances**
    - **GCC-8 offers** “SSE2 → ALTIVEC” **code translation**
- **Few question are remaining ...**
  - How fast are actually those filters ?
  - Does the implementation matter ?

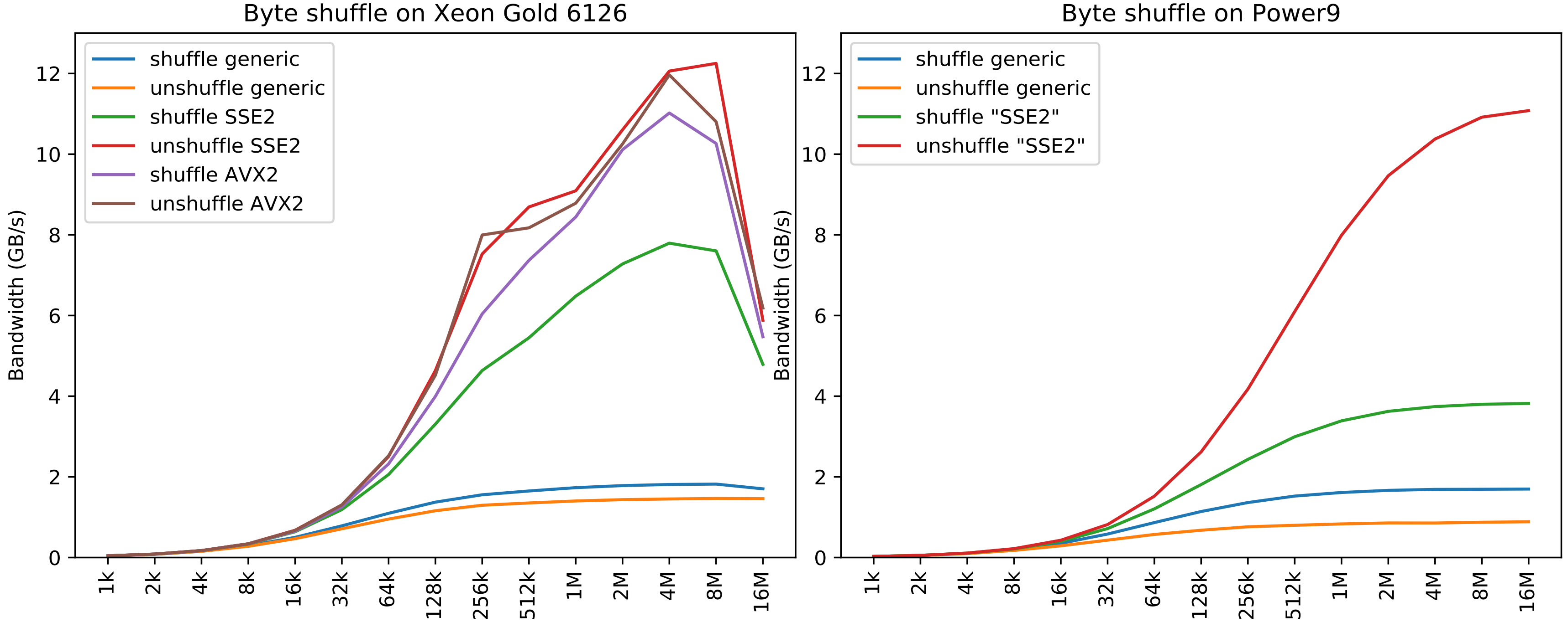
Bitshuffle-LZ4 plugin (used in newer Eiger firmwares)

- **Without pre-filtering, the compression ratio were not that great.**
  - Bit-shuffling increases even further the compression ratio (6x → 21x)
  - Coupled with the fast *lz4* compressor & multi-threaded



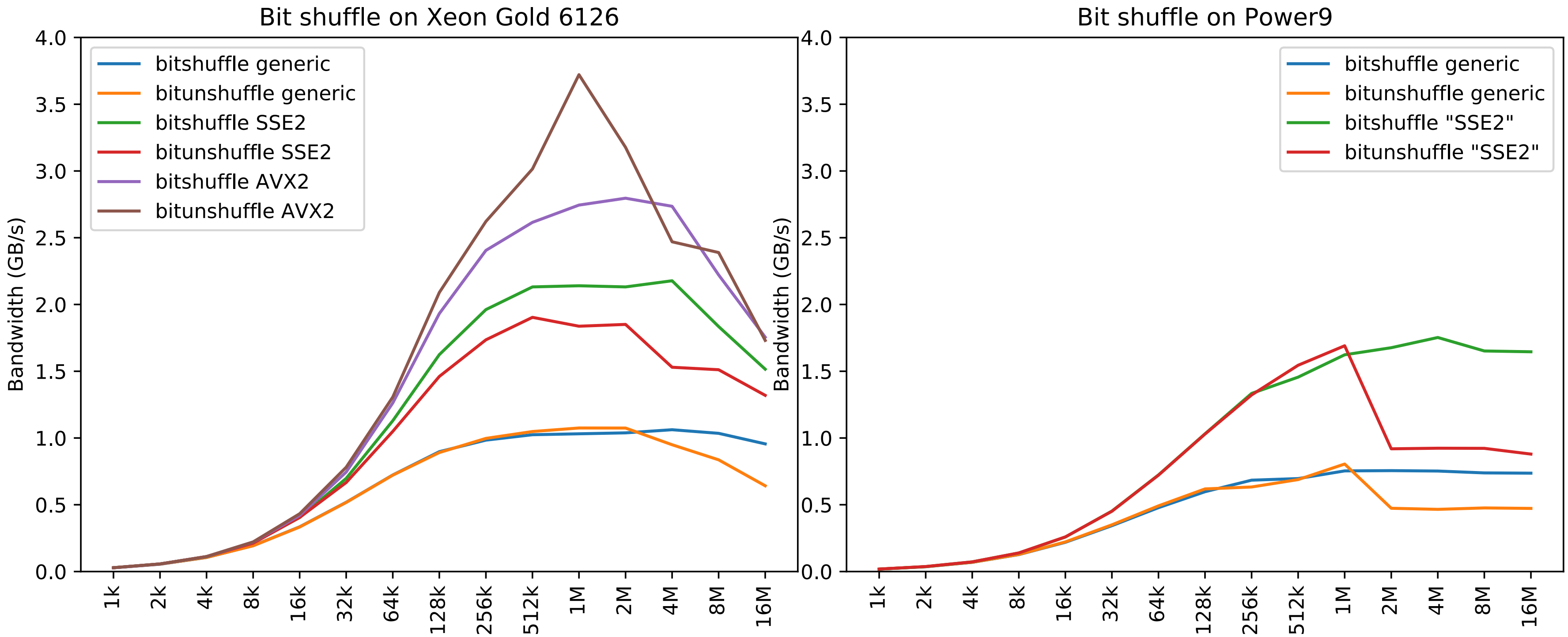
- The Power9 lacks wider SIMD like AVX
- Automatic translation of SSE2 since gcc-8
- Only 12 out of 16 cores used to be fair

Blosc-2: Bandwidth of the shuffle filter



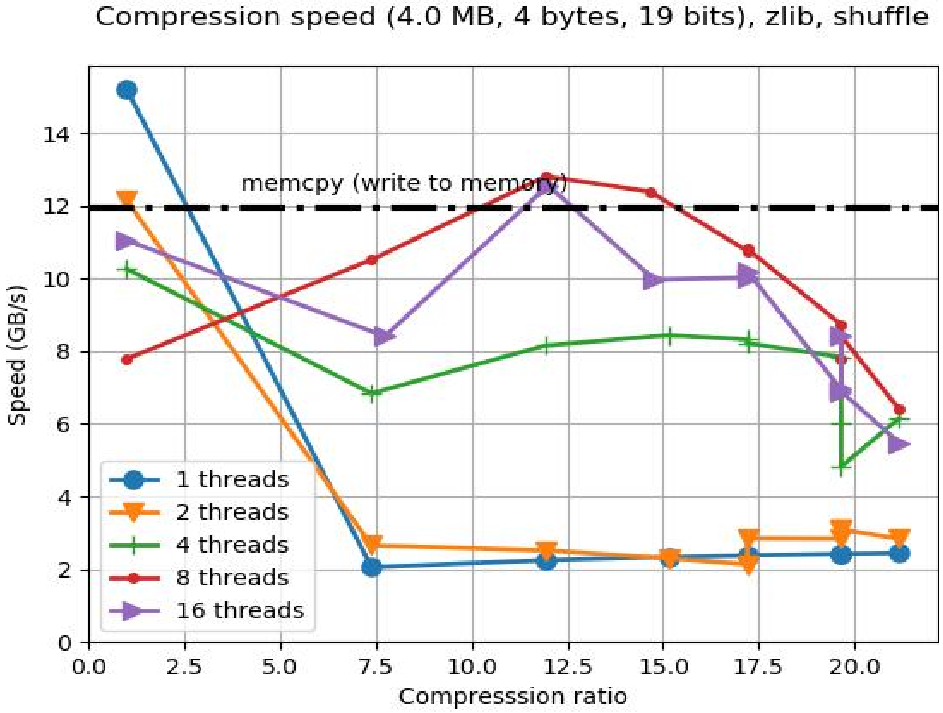
- **Benchmark conditions: C-blosc2 (2.0beta4)**
  - One thread, various sizes to probe the cache
  - 4 bytes per data (int32)
  - Shuffle requires 2 buffers *in* and *out*
  - Python / *timeit* (best of 5) + *ctypes* bindings

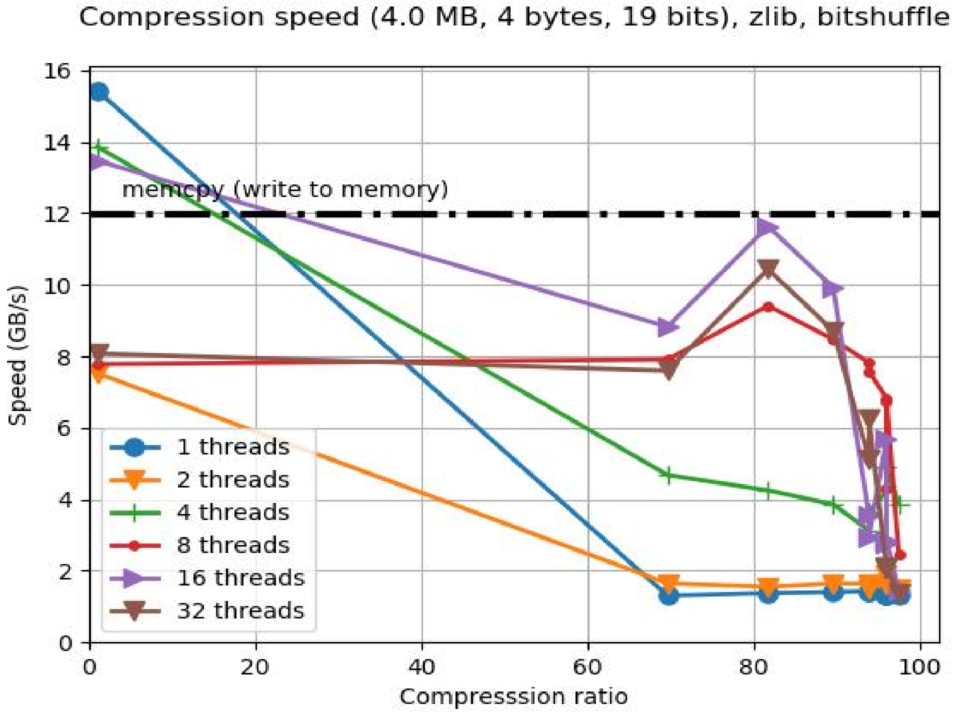
Blosc-2: Bandwidth of the bitshuffle filters



- **Benchmark conditions: C-blosc2 (2.0beta4)**
  - One thread, various sizes to probe the cache
  - 4 bytes per data (int32)
  - Bitshuffle requires 3 buffers
  - Python / *timeit* (best of 5) + *ctypes* bindings

Multi-threaded filters + hardware gzip





- **Best performances obtained: 12 GB/s speed**
  - Operating on one socket out of 2
  - All cores of the socket used but without SMT
  - Moderate compression level=2
  - Much better compression for bitshuffle than shuffle

Conclusion on the computers

- **About the Power9 architecture**
  - All tested code run after simple recompilation
  - Automatic SSE2 → ALTIVEC code translation since gcc-8
    - **Probably not as good as native ALTIVEC code**
  - The Power9 wins where bandwidth maters
  - The hardware compression engine does the complex job for free
  - Python and other interpreted languages are slower

- **About the Dell R840**
  - 3TB of RAM !
  - Memory bandwidth is only a third of theoretical value
  - Limited by PCIe v3 bandwidth
  - The R840 runs much warmer than the R740 (2 processors)
  - The size of the cache L3 of the processor matters !

Conclusion on HDF5's compression framework

- **No support for multi-core computers ?**
  - Multi-threading, OpenMP
- **Gzip/shuffle implemented in 2002**
  - left untouched since then ?
  - SIMD implementation are missing for shuffle
- **Many compression features are missing like:**
  - Bitshuffle, proven supperior to shuffle
  - Many compressors are missing while free to redistribute
    - **Why are they not provided by the HDFgroup ?**
  - Plugins are not actual plugins … but libraries !
    as they are linked to only ONE version of HDF5 !
- **Tested from python/h5py**
  - Whould the picture be different if tested from C or C++?

Acknowledgments

- **IBM:**
  - Thibaud Besson
  - Bruno Mesnet
  - Alexandre Castellane
  - Fabrice Moyen
  - Jean-Pierre Rascalou
  - Pascal Vezolle
  - Frederic Barrat
  - Laurent Vanel
- **Scasicomp:**
  - Marc Ruhlmann
  - Martin Poupon
- **Blosc:**
  - Francesc Alted
- **ESRF:**
  - Andy Götz
  - V. Armando Solé