SDC 20

BY Developers FOR Developers

Storage Developer Conference
September 22-23, 2020

# Deep Compression at Inline Speed for All-Flash Array

**Chris Mao** (mao@purestorage.com)

**Principal Engineer**

**Pure Storage**

# Agenda

- Need for speed: deep compression in All-flash array
- Programmable FPGA accelerator card
- Make SW API simple
- Innovate on hardware architecture design
- Continuous algorithm development
- Benchmarks

# Pure-Flash Array for Enterprise Storage

FlashArray//C: QLC Flash
Capacity Optimized

FlashArray//X: 100% NVMe
Performance Optimized

FlashBlade: Scale-Out
Unified Fast File & Object

**7,800+**
Global Customers

**21%**
Year-over-Year Growth

**82**
NPS Score = Top 1% of B2B Companies

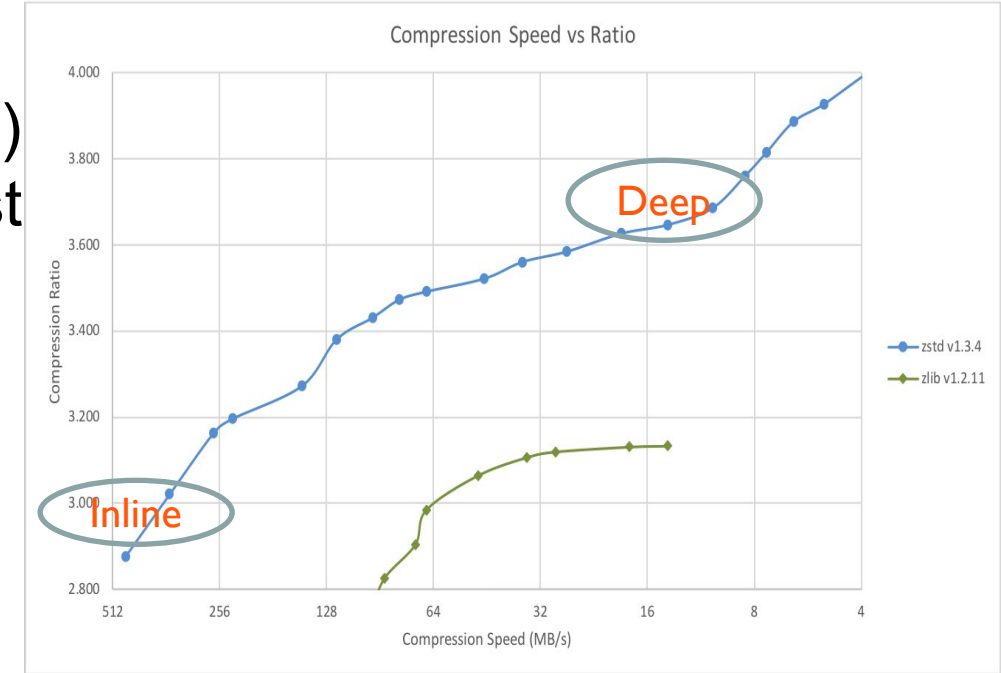
**$1.64B**
Annual Revenue



PURESTORAGE®

# Not all compressions are created equal

- Inline compression
  Fast(~0.5GB/s/core)
  LZO,LZ4, ZSTD-fast
  etc
- Deep Compression
  Slow (10X slower)
  Compress ~30%
  Better
  ZSTD,Mermaid…



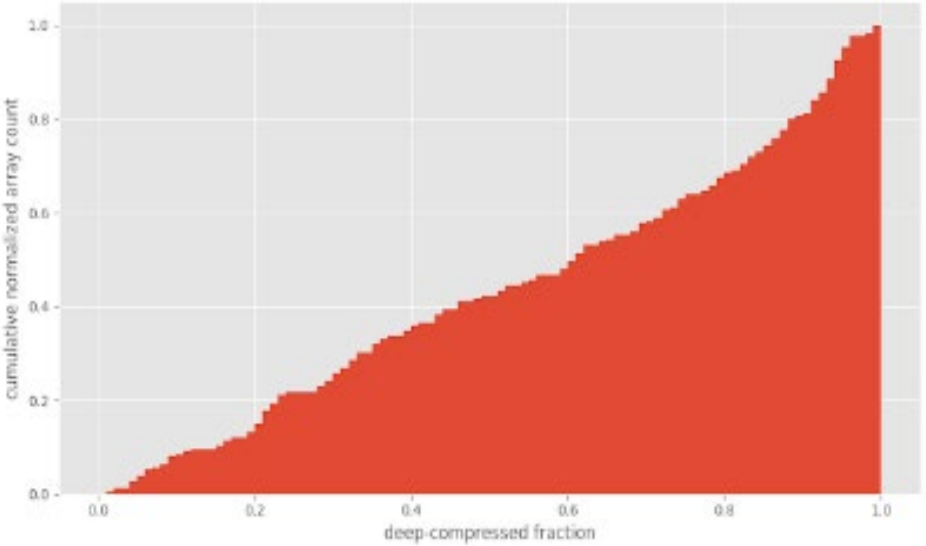

Source: https://facebook.github.io/zstd/



PURESTORAGE®

# Not all data gets an equal chance

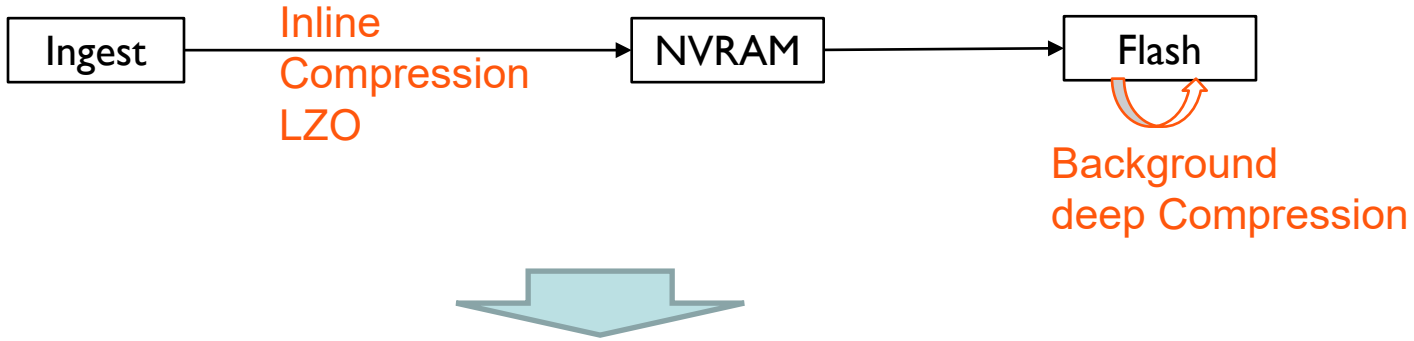
- 100% data gets inline compressed
- Deep compression is opportunistic
- Sustained load + large capacity => lost opportunity for deep compression

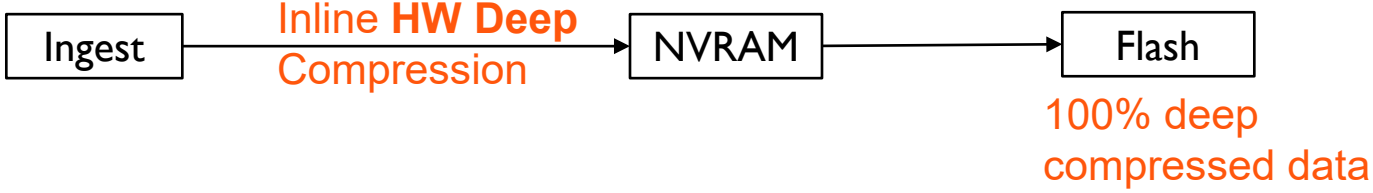





PURESTORAGE®

# Lower $/Gb by up to 30%, anyone?






PURESTORAGE®

# Programmable FPGA accelerator card

- Deep compression offload at inline speed: target 10X faster than x86 CPU
- Power efficient
- Low NRE for HW development
- Programmable accelerator platform

# Why not ASIC?

- Off-the-shelf ASICs: fixed gzip/zlib that is suboptimal in both speed and compression ratio
- ZSTD on the roadmap: better, fast in decode, but not the fastest. Years away for chip to be available
- Custom in-house ASIC … not for the faint-hearted…
- Always lagging years behind in algorithm development: every 1% compression ratio is $$$

# PAC (Pure Accelerator Card) 1.0

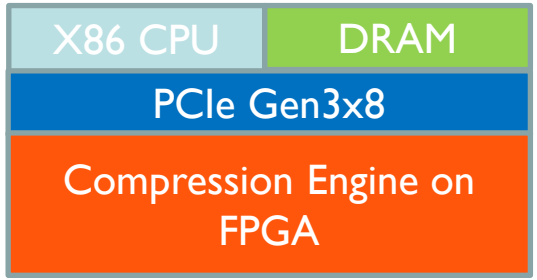
- Standard HHHL PCIe adaptor card
- PCIe Gen3x8
- 6GB/s
- <25Watt
- Mid-range FPGA device
- User space VFIO driver




PURESTORAGE®

# KISS HW API: Keep It Simple & Stupid

- int compress(src_buf,dst_buf,src_len) {
  pcie_write(SQ,src_buf,dst_buf,src_len)
  poll_wait_for_completion()
  }
- Compression requests in SQ complete in strict FIFO order



# Compression algorithms: which one?

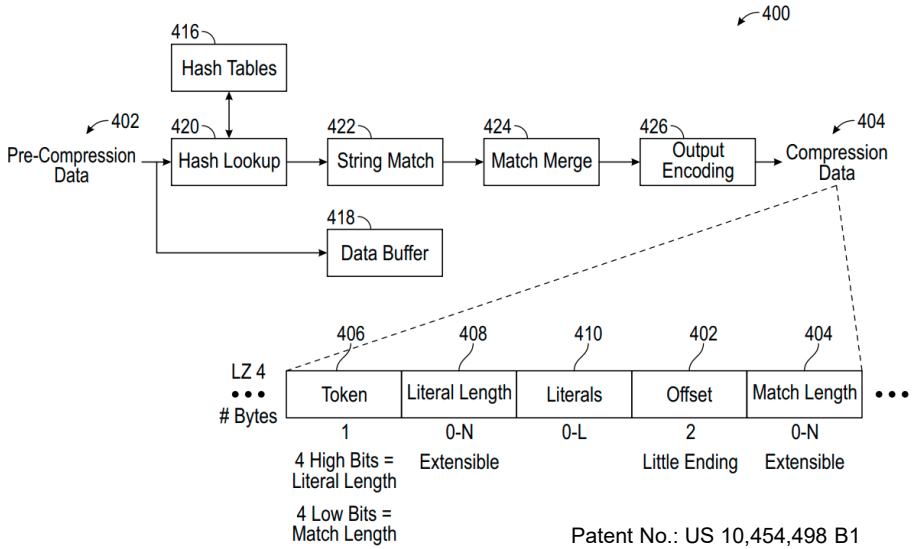
- Generally all based on LZ + Entropy encoding
- Differentiate on the quality of LZ search, and encoding
- Open source: gzip/zlib, lz4, zstd…
- Closed source/commercial: LZO, Mermaid…

# Algorithm: roll your own?

- Block compression: block size <64KB
- Super fast decoding in SW (>2-3GB/s/core)
- Best possible compression ratio for real workload
- => in-house developed algorithm that is trained by workload in our fleet of 10,000s of flash arrays

# LZ compression in HW: pipeline design

- Freedom in architecture & logic design
- Parallelism
- Fast on-chip SRAM 1000s of 1R1W block RAMs
- General purpose CPU limited by von-Nuemann arch



# LZ search in HW: hash tables

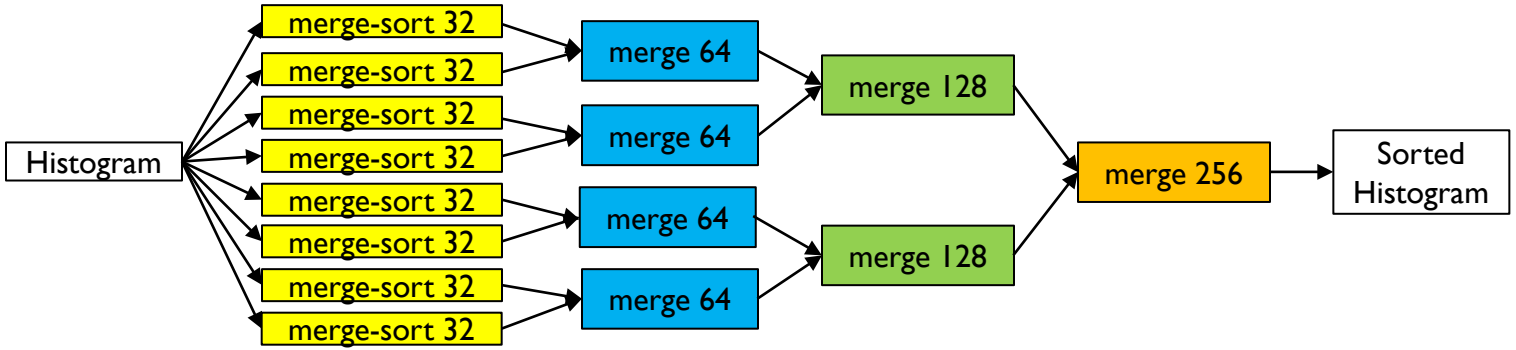
- Process 4 bytes positions every clock cycle : 4 hash updates (writes) and 4 hash lookups (reads) from/to memory => **19.2GB/s** BW @300Mhz, 128KB in size
- 2 hash tables/engine => **38.2GB/s/engine** memory BW
- 4 compression engines => **153.6GB/s** hash table memory BW
- HW implementation: 64x 1R1W 2KB memory blocks

# Huffman encoding in HW

- Literals: dynamic Huffman encoding
- Metadata sections (Match offset, Match size, Literal size etc) encoded with 64 static tables.
- Static tables trained by real workload across fleet
- Muti-stream Huffman encoding for faster decoding

# Huffman encoding: sorting histograms

- Histogram sorting: 256 random numbers
- Merge-sort: NlogN=2000 clock cycles
- HW merge-sort latency: 160[32log32] + 64 + 128 +256 = 608 cycles
- HW merge-sort pipeline throughput: sustain one sort every 256 cycles.






PURESTORAGE®

# Microbencharmark #1 (32KB block)

Time spent in ns/byte (blockbench, 32768)

encode  decode ratio  total bytes

e:  1.860  d:  0.378  r:  2.574  t:  5951982  anacharsis0

e:  1.662  d:  0.591  r:  2.373  t:  6454818  lzopro13 (inline)

e:  1.523  d:  0.364  r:  2.729  t:  5611961  arion0 (HW inline)

e:  1.824  d:  0.561  r:  3.277  t:  4673778  **arion3 (HW deep)**

e: 11.574  d:  0.592  r:  3.474  t:  4408983  mermaid4

e: 12.458  d:  0.747  r:  3.725  t:  4112044  anacharsis4

# Microbenchmark #2 (4KB block)

Time spent in ns/byte (blockbench, 4096)

encode decode ratio total bytes

e: 2.161 d: 0.468 r: 2.112 t: 7251089 anacharsis0

e: 1.698 d: 0.610 r: 2.061 t: 7431928 lzopro13 (inline)

e: 3.181 d: 0.426 r: 2.356 t: 6501966 arion0 (HW inline)

e: 3.703 d: 0.769 r: 2.795 t: 5481220 arion3 (HW deep)

e: 16.782 d: 0.691 r: 2.558 t: 5987246 mermaid4

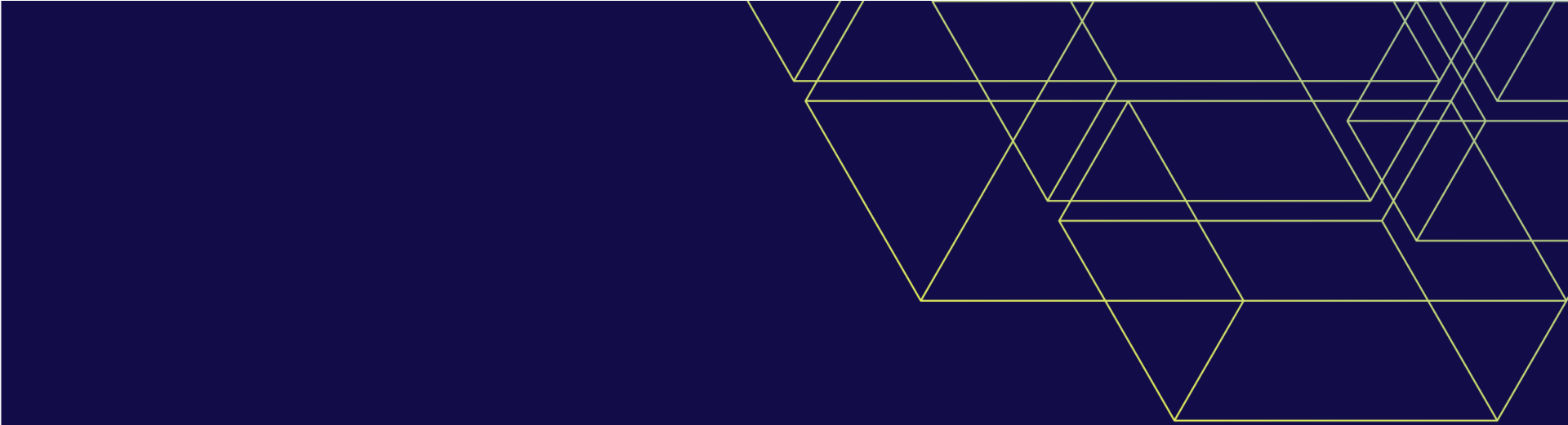
e: 21.891 d: 0.986 r: 3.070 t: 4989952 anacharsis4

# Continuous algorithm development

- LZ+: better LZ search algorithm (NP-hard)
- Filters: auto-detection of input data stream types, and apply possible transformation for better compression ratio
- the unknown unknowns…

# Summary

- Deep compression at inline speed: better $/Gb up to 30%
- FPGA accelerator card can hit the performance, cost & power target
- Domain specific architecture/logic design key to success
- Programmable platform enables continuous improvements and non-disruptive field upgrade



**Please take a moment**
**to rate this session.**

**Your feedback matters to us.**