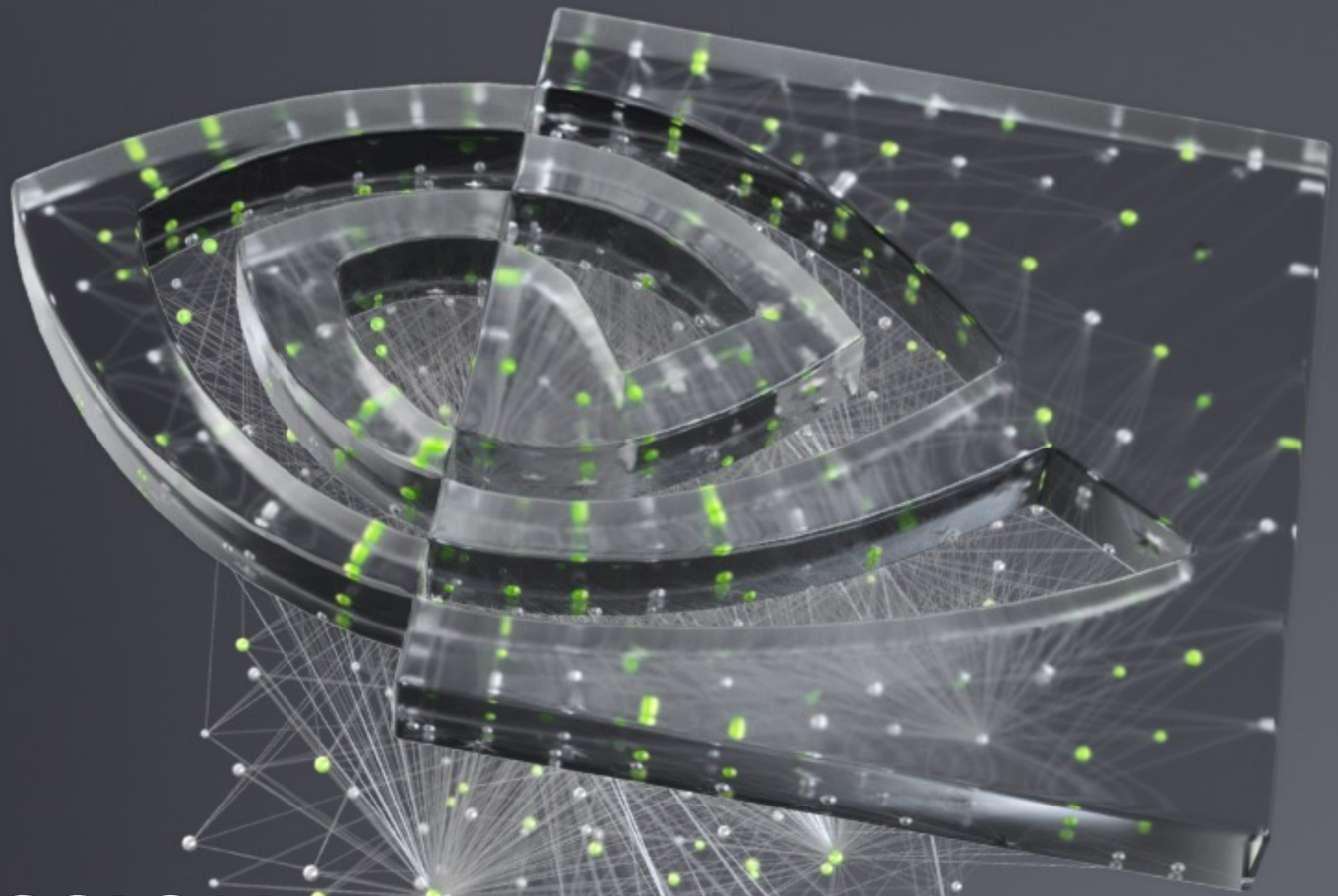


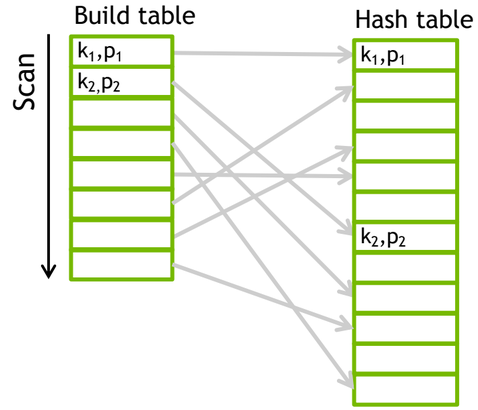
FAST DATA COMPRESSION ON GPU AND DPU

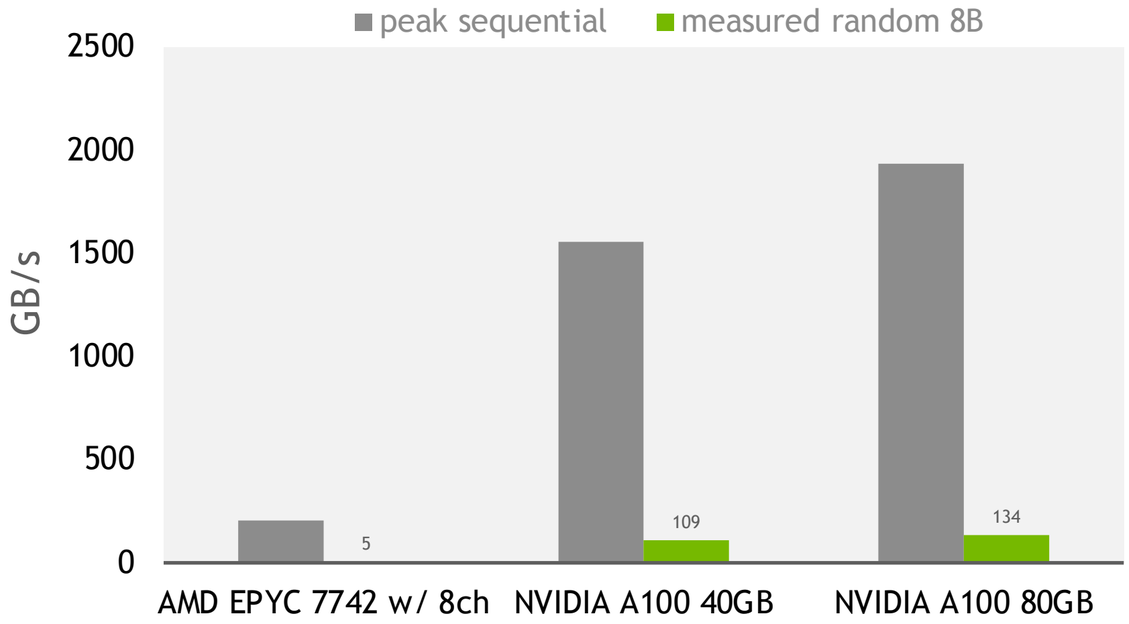
Nikolay Sakharnykh, 8/16/2021

ACCELERATING DATABASES ON GPU

Databases are very **memory intensive**, and not so much compute intensive (like GEMMs)

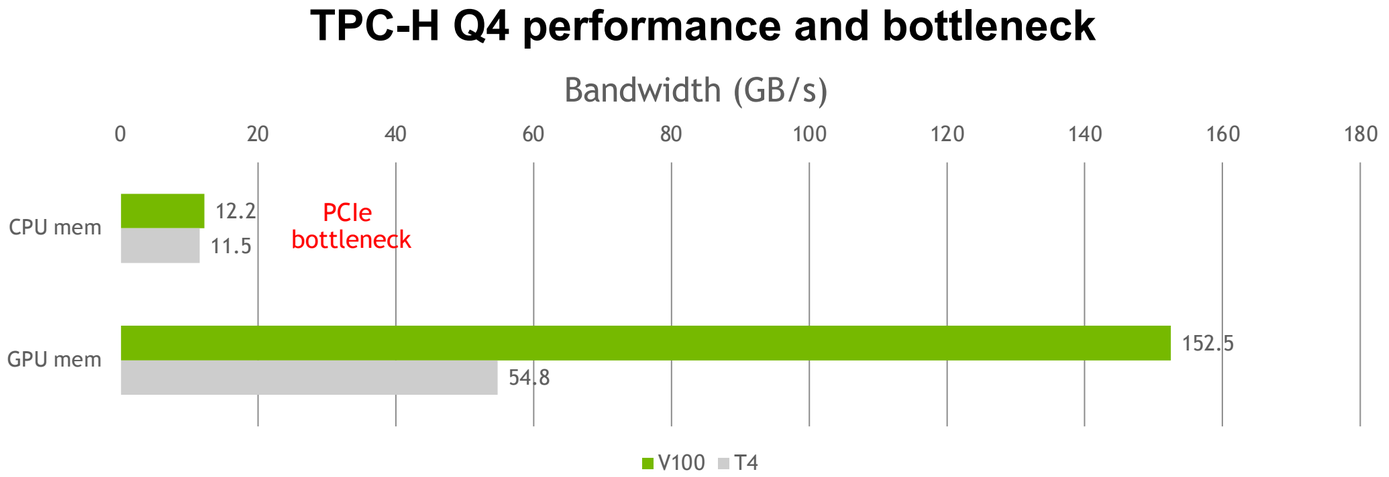
Often limited by random access performance, e.g. hash tables in join and group-by

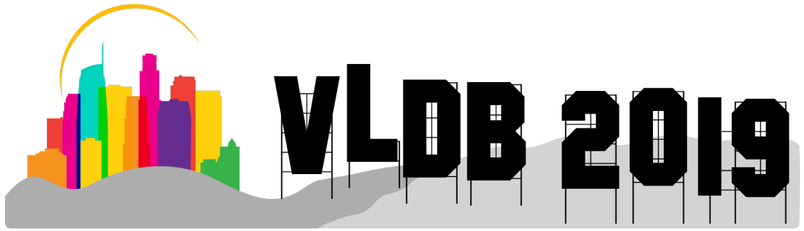




When the data fits in memory, GPUs crush CPUs!

WHAT ABOUT **VERY LARGE** DATABASES?





ADMS'19 keynote: "2019: GPU Odyssey"

Fast-forward to now (2021):

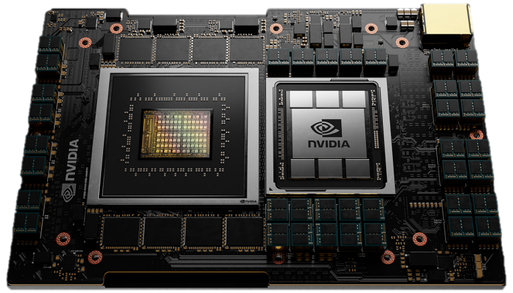
PCIe Gen3 -> PCIe Gen4, 2x improvement

V100 -> A100 random access, >2x improvement

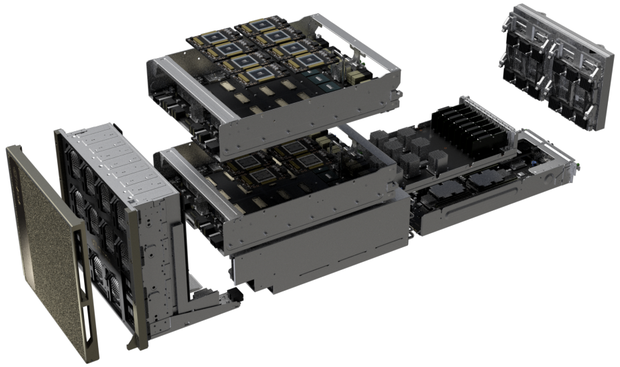
Result: still PCIe bound!

NVIDIA.

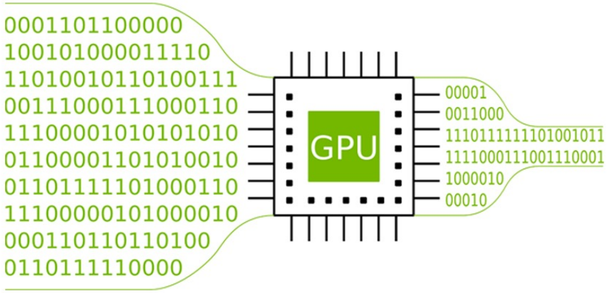
HOW DO WE OVERCOME PCI-E BOTTLENECK



Faster Interconnect

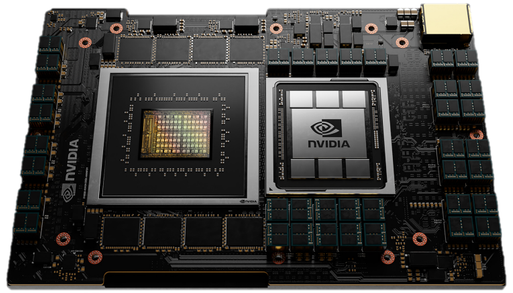


More GPUs



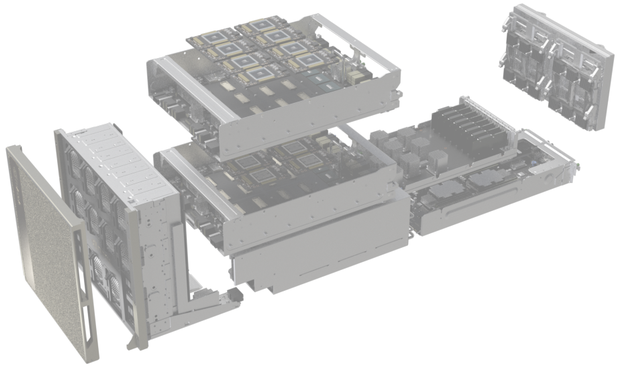
Compression

HOW DO WE OVERCOME PCI-E BOTTLENECK

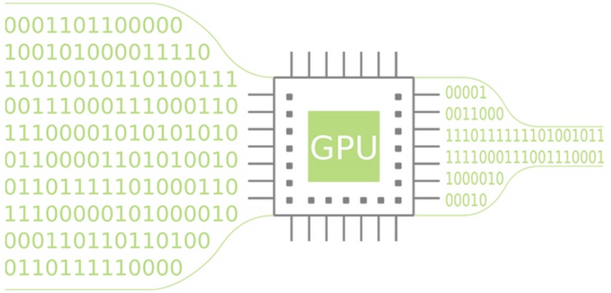


Faster Interconnect

V100 + P9 features NVLINK 2.0
150GB/s bidir BW



More GPUs

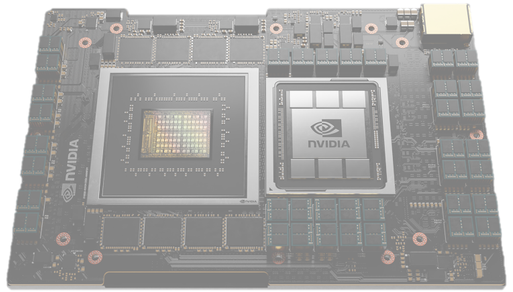


Compression

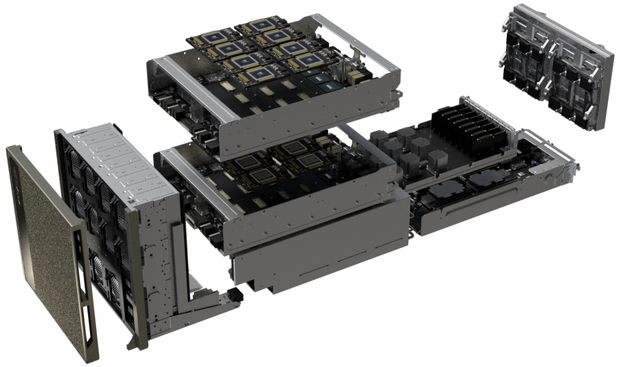
SIGMOD'20 best paper:
Pump Up the Volume: Processing Large Data on GPUs with Fast Interconnects

At **GTC'2021** NVIDIA announced Grace CPU tightly coupled with GPUs

HOW DO WE OVERCOME PCI-E BOTTLENECK

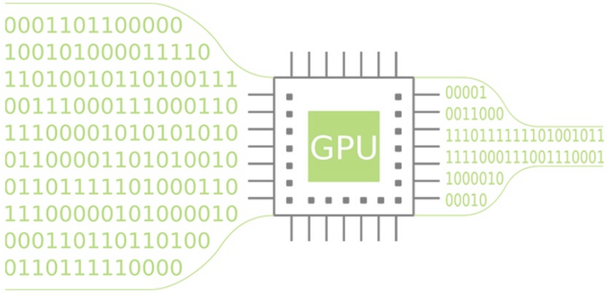


Faster Interconnect



More GPUs

DGX-A100: 640GB of GPU memory



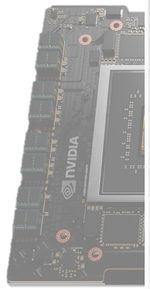
Compression

GTC'19:
Effective, Scalable, Multi-GPU Join

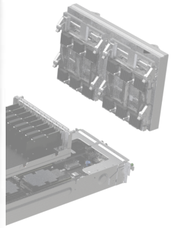
VLDB'21:
Efficient Join Algorithms For Large Database Tables
in a Multi-GPU Environment

ADMS'21:
Scaling Join to a Thousand GPUs

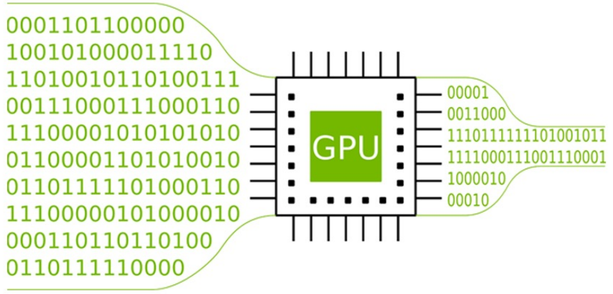
HOW DO WE OVERCOME PCI-E BOTTLENECK



Fas
V100 +
1



emory



Compression

BUT FIRST, HOW DO WE GET DATA TO THE GPU EFFICIENTLY?

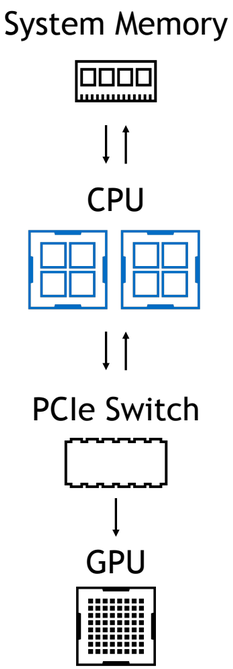
CPU-GPU DATA MOVEMENT

cudaMemcpy – DMA/copy engines

- Efficient for sending large data blocks, but suboptimal for small transfer sizes
- For best efficiently memory needs to be pinned & registered
- Pageable memory copies are implemented with a staging buffer internally

Zero-copy access – direct access (cudaHostRegister)

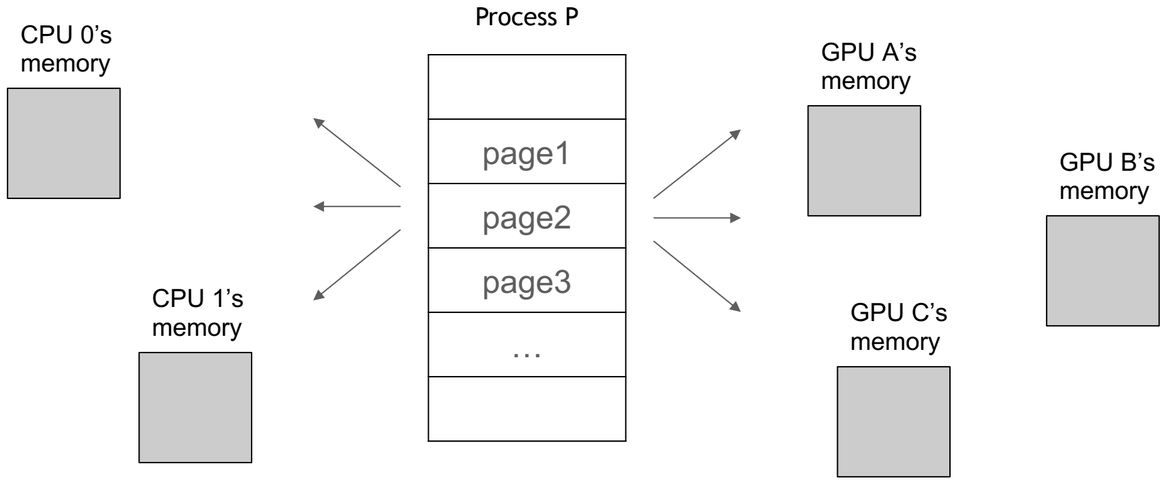
- Great for small fine-grained accesses
- GPU cores engaged in generating memory requests



NVIDIA.

CPU-GPU DATA MOVEMENT

Automatic page migration (cudaMallocManaged)



Migrate data to the accessing processor on first touch

Minimum migration granularity is 4KB but can be as large as 2MB

Things to look out for: frequent page faults or thrashing (can be mitigated with hints)

https://developer.nvidia.com/blog/maximizing-unified-memory-performance-cuda/

Look out for a new tech blog on improving memory oversubscription perf!

To be published in the next few weeks

NVIDIA.

GETTING DATA TO GPUS: IO

GPUDirectStorage

Skips CPU bounce buffer via DMA

Works for both local and remote storage

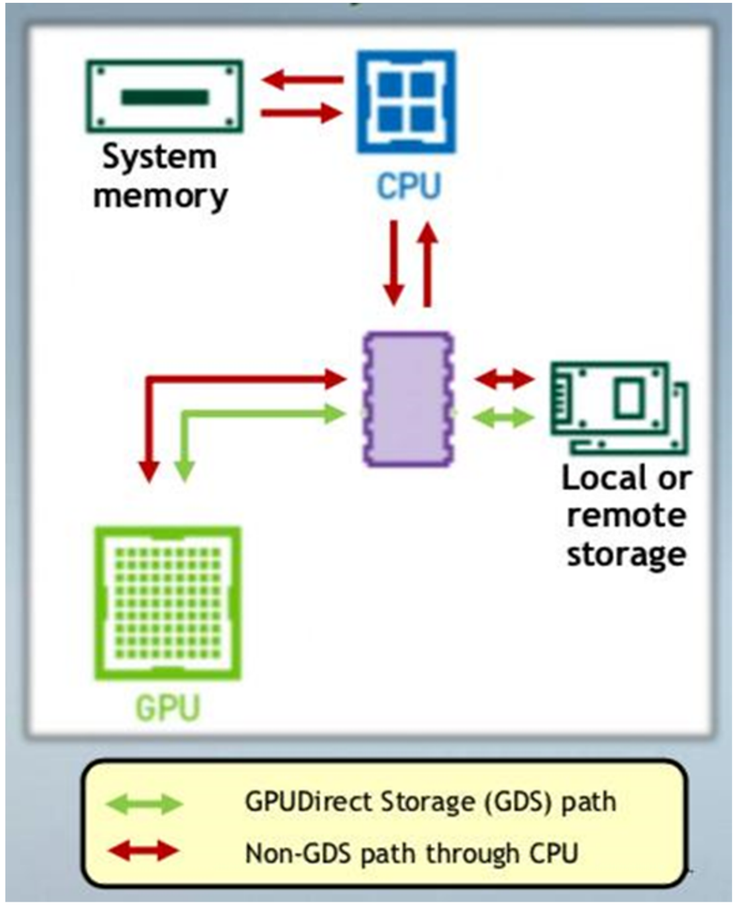
Accessed via new CUDA cuFile APIs

No special HW needed

Advantages of GDS:

Higher peak bandwidth

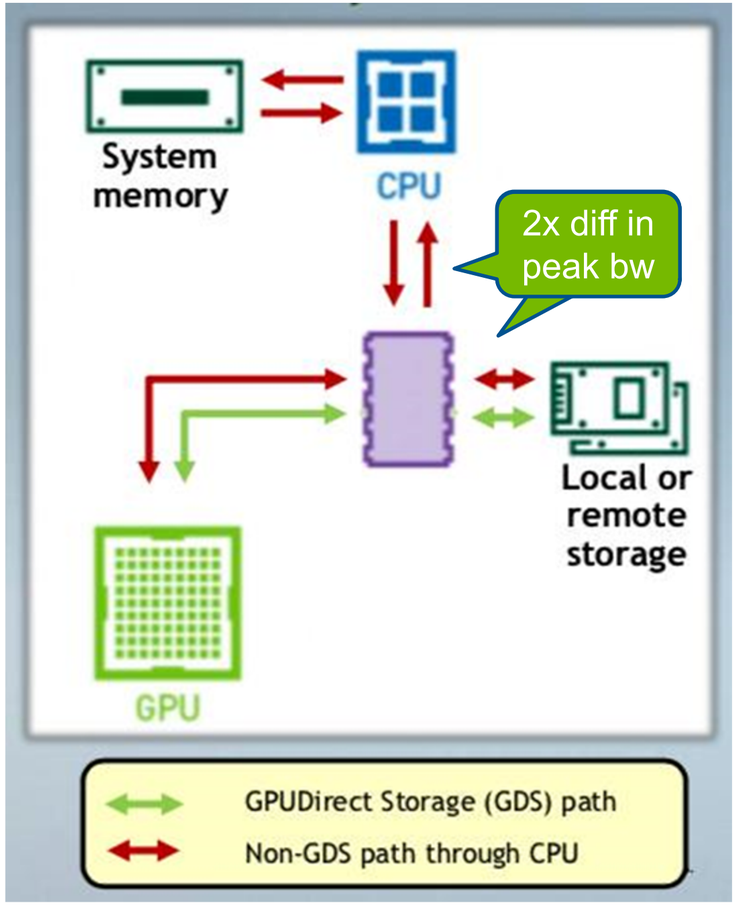
Lower latency by avoiding extra copies and dynamic routing



DIRECT DATA PATH ADVANTAGE ON DGX

Endpoint		DGX-2 (Gen3), GB/s	DGX-A100 (Gen4), GB/s
CPU		50, peak	100, peak
Switch		100, peak	200, peak
Endpoint	**Capacity**	**Measured**	
CPU sysmem	O(1TB)	48-50 @ 4 PCIe	96-100 @ 4 PCIe
Local storage	O(100TB)	53+ @ 16 drives	53+ @ 8 drives
NICs	O(1PB)	88 @ 8 NICs	180+ @ 8 NICs

2x diff in peak bw

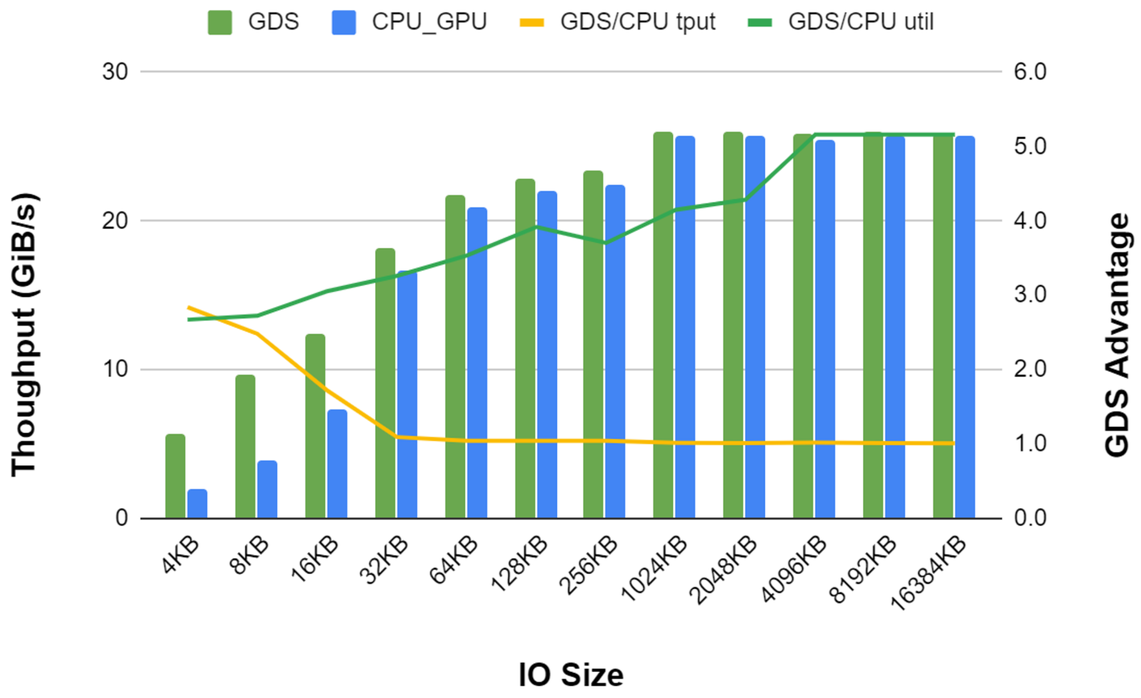


GDS: DGX-A100 LOCAL STORAGE PERFORMANCE

Setup: 4 GPUs, 4 NVMes, each GPU reads from ext4 mount on NVMe, aggregated NVMe BW of 26 GiB/s

Small IO sizes: up to 2.8x performance advantage over no-GDS case

Large IO sizes: perf advantage shrinks to 1x (capped by limited throughput of local drives), but we see up to 5x CPU utilization advantage

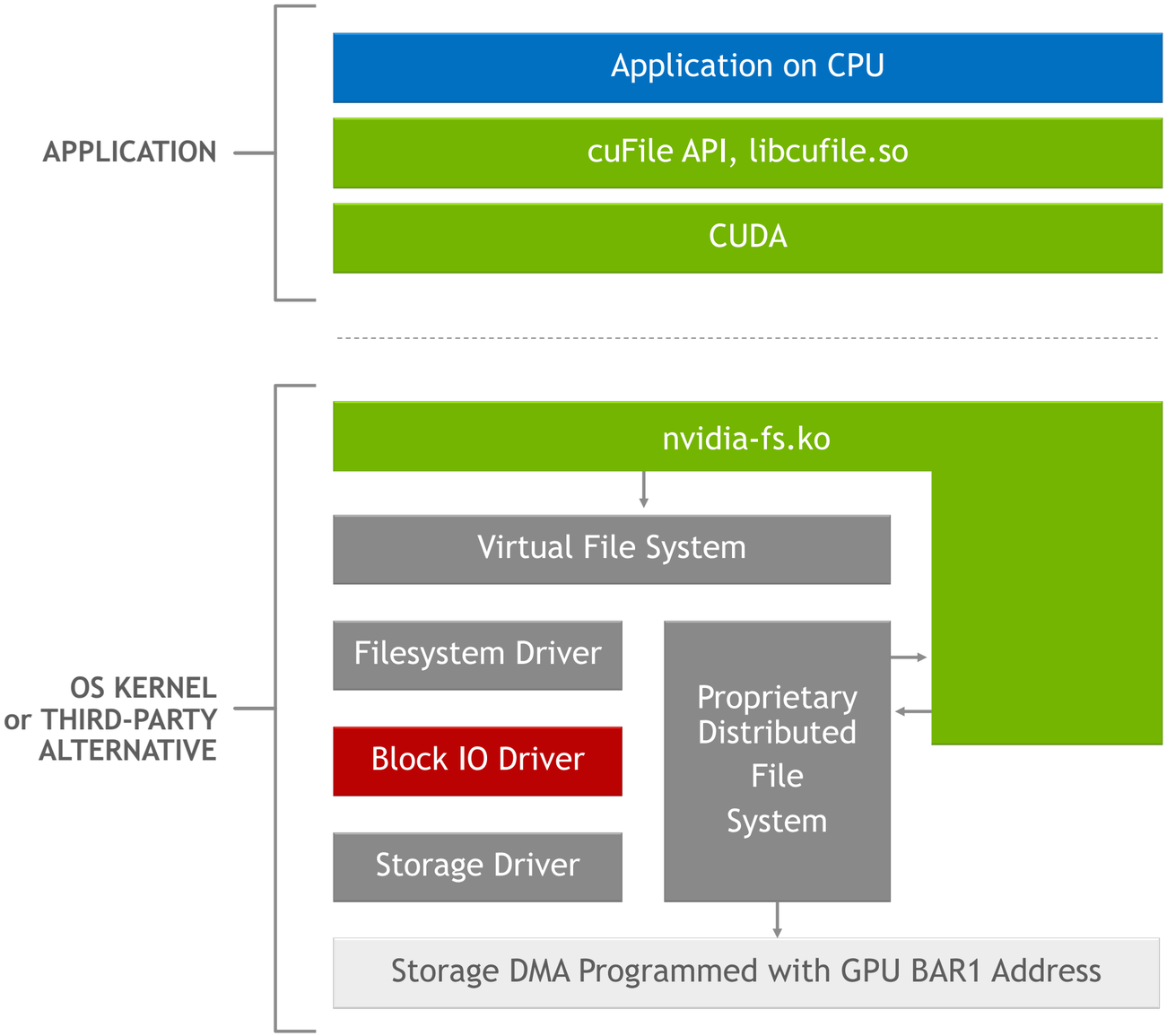


GPUDIRECT STORAGE SW ARCHITECTURE

cuFile user API
Enduring API for applications and frameworks

nvidia-fs driver API
For file system and block IO drivers
Vendor-proprietary solutions: no patching avoids lack of Linux enabling

NVIDIA is actively working with the community on upstream first to enable Linux to handle GPU VAs for DMA



cuFILE LIBRARY

- **Features**
  - Shared user-space file IO library for GDS-ready DMA and RDMA file systems
  - Supports aligned and unaligned file IO offsets and sizes
  - Avoids cudaMemcpy, intermediate copies to page cache, and user-allocated sys memory
  - User-space RDMA connection and buffer registration management
  - Dynamically routes IO using optimal path based on HW topology and GPU resources
  - Supports compatibility mode that works with user-level installation only for development
- **Requirements**
  - Needs files to be opened in O_DIRECT mode to bypass page cache and buffering in system memory for POSIX-based file systems

NVIDIA.

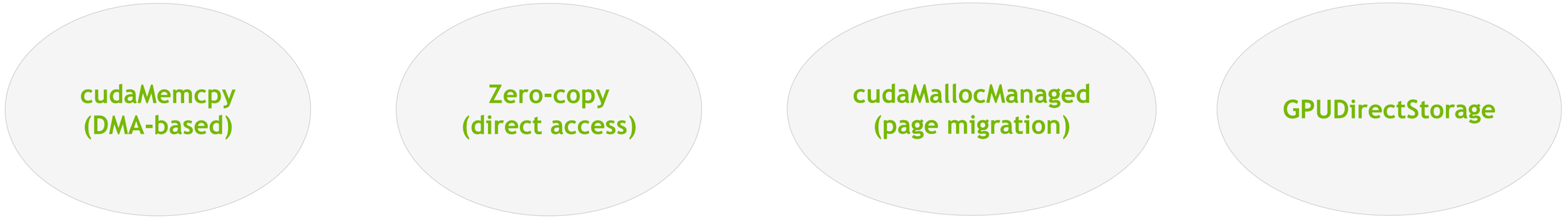
cuFILE USAGE EXAMPLE

Write from GPU memory to Local or Remote Storage

```
int main(void) {
        …                                                       // set GPU device, etc.

        CUfileError_t status;
        CUFileDescr_t cf_descr;                                 // general enough to support Windows
        CUFileHandle_t cf_handle;

        status = cuFileDriverOpen();                            // initialize

        fd = open(TESTFILE, O_WRONLY|O_DIRECT, 0644);           // interop with normal file IO
        cf_descr.handle.fd = fd;
        status = cuFileHandleRegister(&cf_handle, &cf_descr);   // check support for file at this mount

        cuda_result = cuMemAlloc(&devPtr, size);                // user allocates memory

        status = cuFileBufRegister(devPtr, size);               // performance optimization
        cudaMemset((void *) devPtr, 0xab, size);
        ret = cuFileWrite(cf_handle, devPtr, size, 0, 0);       // ~pwrite: file handle, GPU base address, size, offset, GPU offset

        status = cuFileBufDeregister(devPtr);                   // cleanup
        cuFree(devPtr);
        cuFileHandleDeregiser(cf_handle);
        close(fd);
        cuFileDriverClose();
        return 0;
}
```


NVIDIA.

RECAP DATA MOVEMENT



Bottom line - if you're not reaching CPU-GPU peak rates you're not doing it right

ARE GPUS A GOOD FIT FOR (DE)COMPRESSION ALGORITHMS?

COMPRESSION TECHNIQUES

Fundamental compression ops

- **Data deduplication** – RLE, LZ, Dictionary
- **Difference encoding** – Delta, Frame-of-reference
- **Entropy encoding** – bit-packing, Huffman, ANS

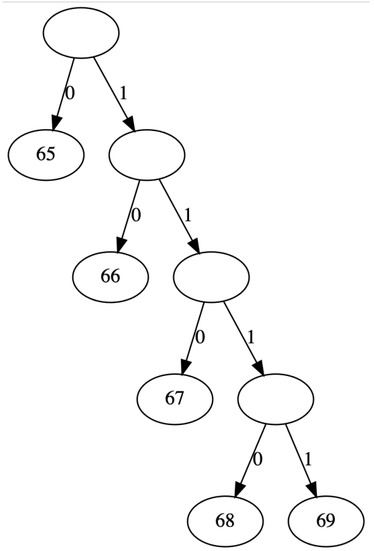
Many well-knowns schemes combine these together

- Deflate = LZ + Huffman
- zstd = LZ + Huffman + FSE/ANS

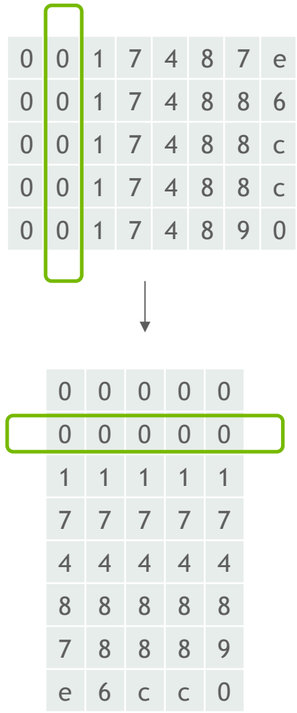
Lots of pre-processing techniques (usually domain-specific)

- Bit-plane transpose, sort, BWT

Huffman tree



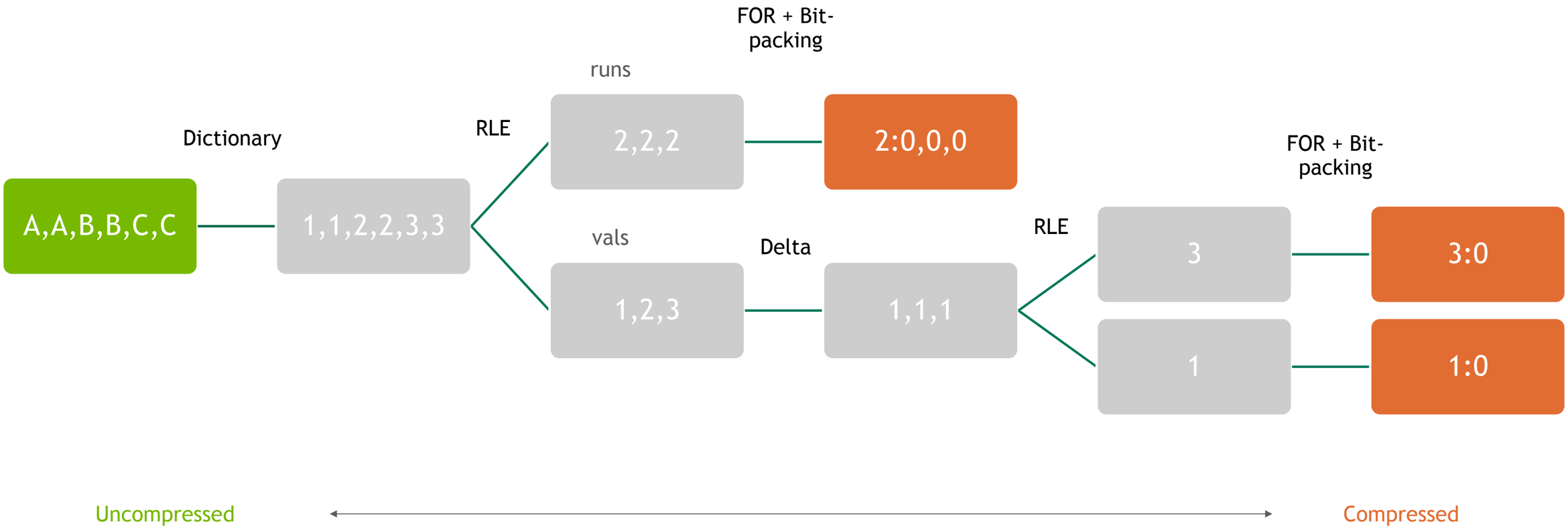
Bit-shuffle/transpose



NVIDIA.

CASCADED COMPRESSOR

Combining the blocks together: RLE + Delta + FOR + bit-packing

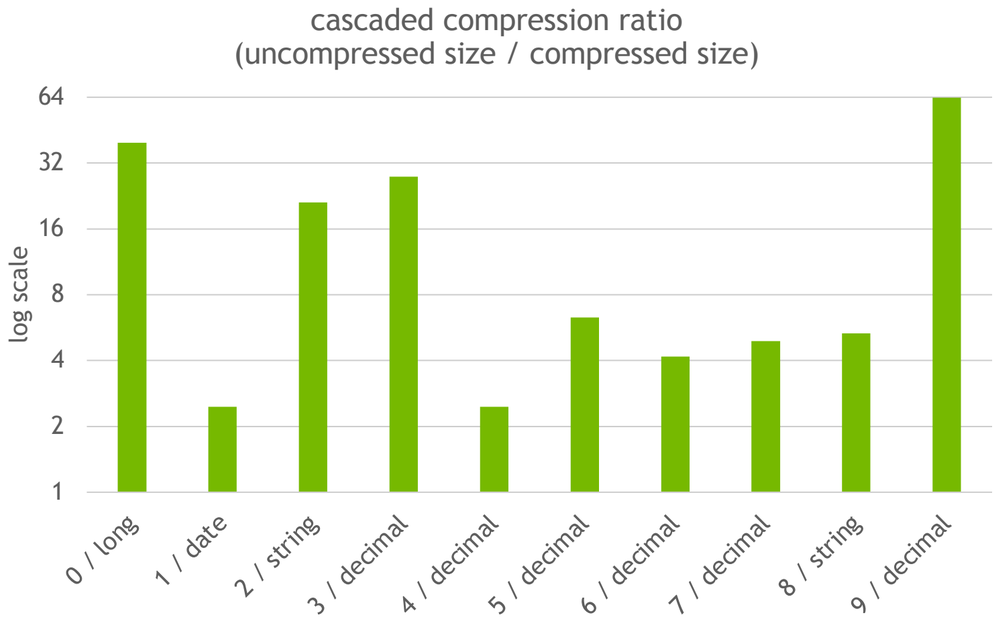


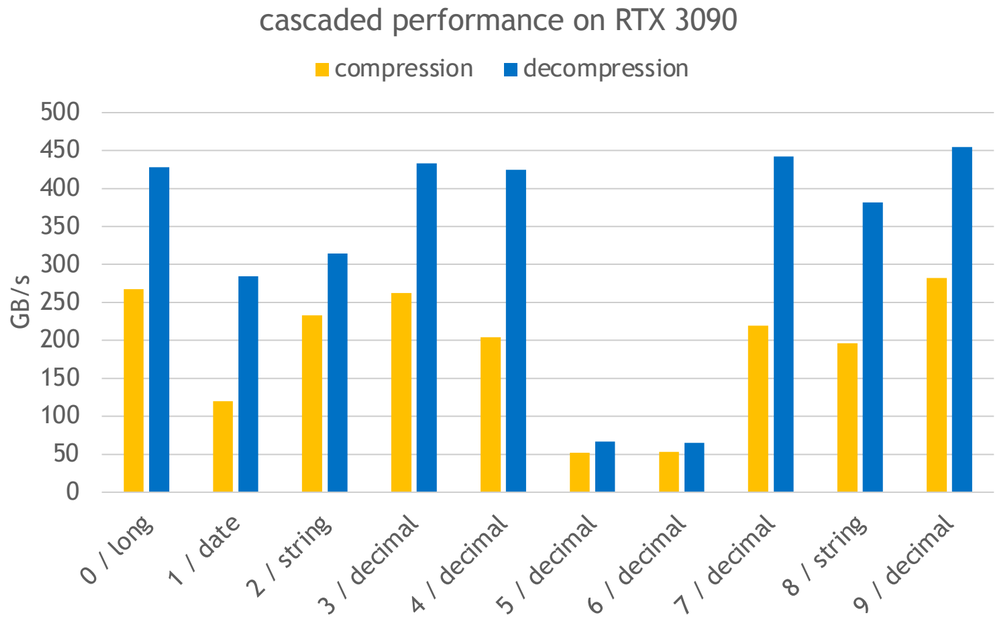
GTC 2020: Software-Based Compression for Analytical Workloads

NVIDIA.

CASCADED RATIOS AND PERF

Great fit for structured data from analytics datasets





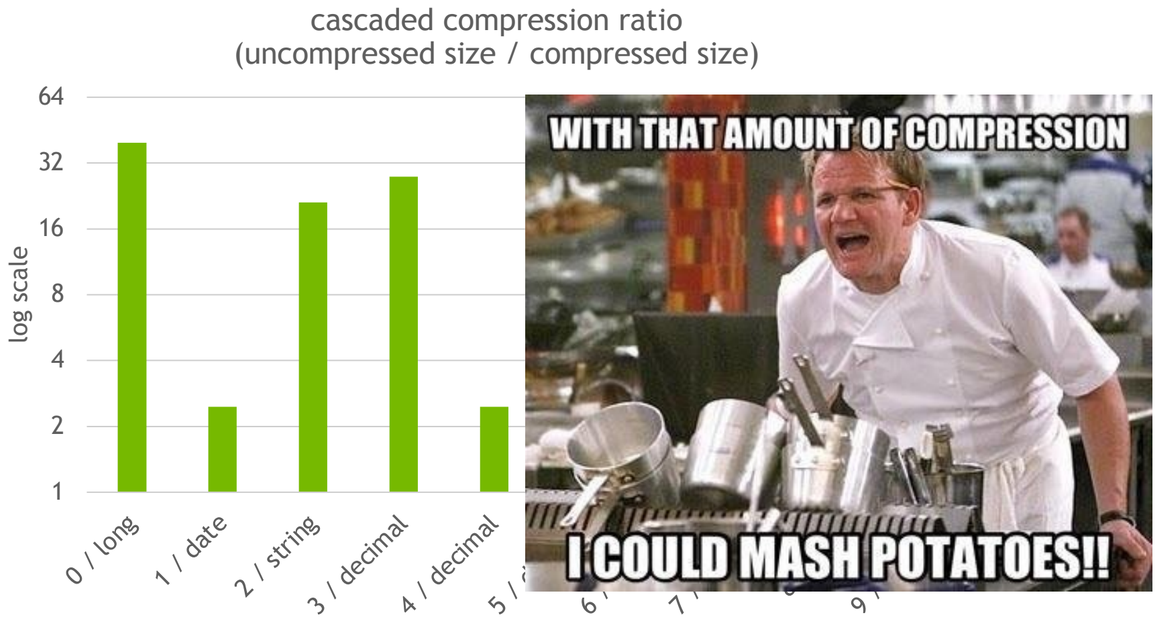
decimal interpreted as 8B integer, string and date as 4B integers

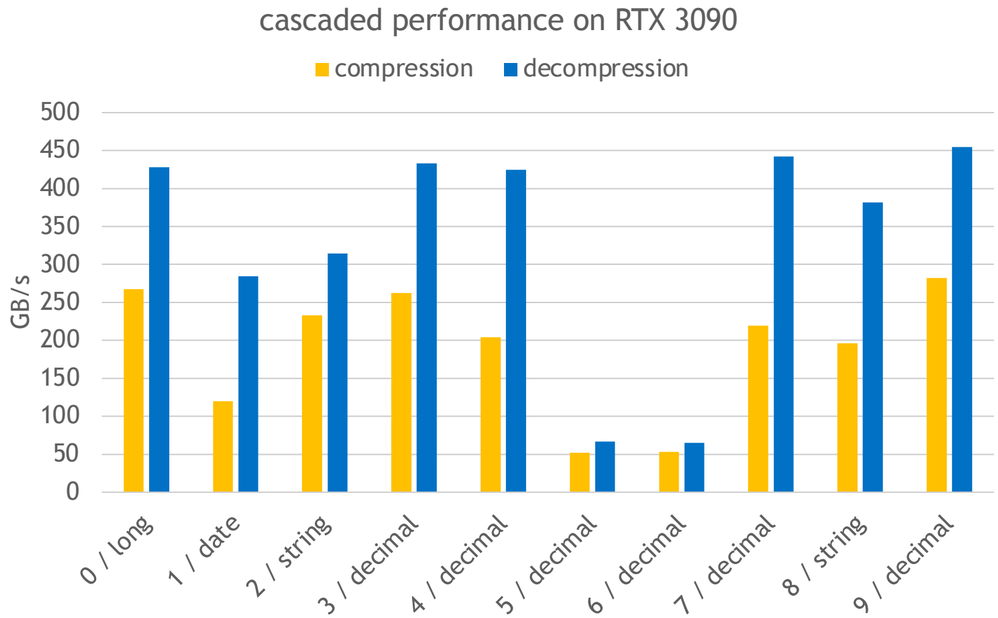
Dataset is derived from Fannie Mae's Single-Family Loan Performance Data and can be obtained here: https://rapidsai.github.io/demos/datasets/mortgage-data
Each column is 100-200MB of uncompressed data
Sample row from the dataset: `100005072756|12/01/2001|GMAC MORTGAGE, LLC|8.0|124352.34|12.0|348.0|0.0|12/2030|27100.0|...`

NVIDIA.

CASCADED RATIOS AND PERF

Great fit for structured data from analytics datasets



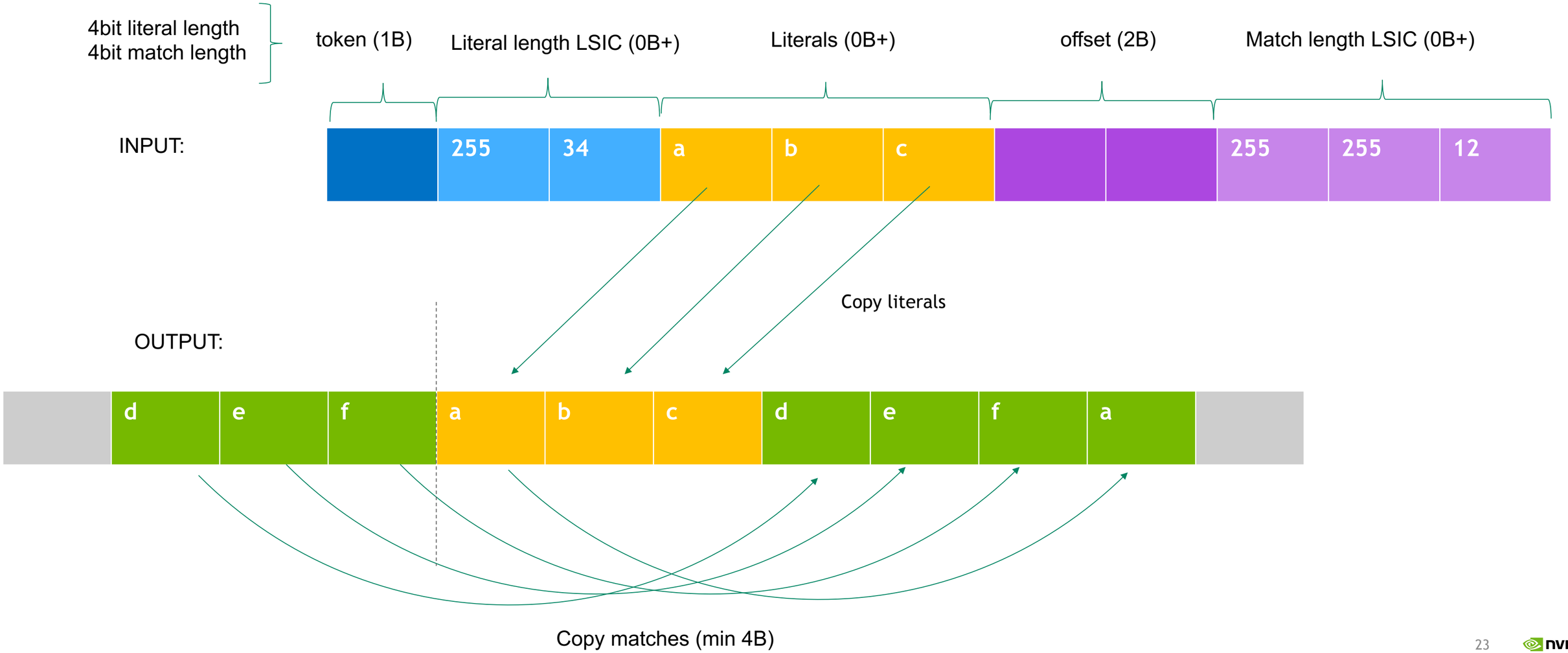


decimal interpreted as 8B integer, string and date as 4B integers

Dataset is derived from Fannie Mae's Single-Family Loan Performance Data and can be obtained here: https://rapidsai.github.io/demos/datasets/mortgage-data
Each column is 100-200MB of uncompressed data
Sample row from the dataset: `100005072756|12/01/2001|GMAC MORTGAGE, LLC|8.0|124352.34|12.0|348.0|0.0|12/2030|27100.0|...`

NVIDIA.

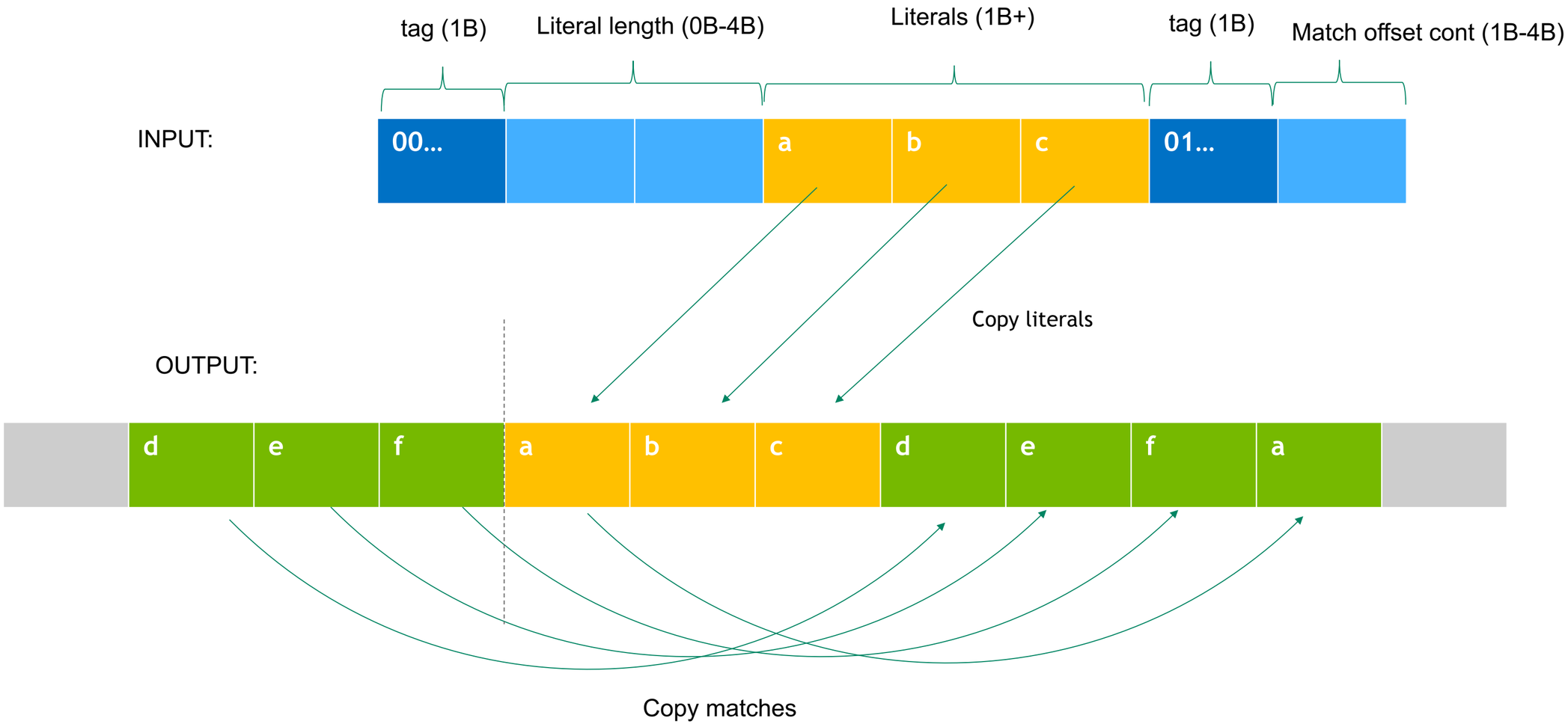
LZ4


4bit literal length
4bit match length
token (1B)
Literal length LSIC (0B+)
Literals (0B+)
offset (2B)
Match length LSIC (0B+)
INPUT:
255
34
a
b
c
255
255
12
Copy literals
OUTPUT:
d
e
f
a
b
c
d
e
f
a
Copy matches (min 4B)

NVIDIA.

SNAPPY

First 2 bits of the tag byte:
00 – literal, 6bit len, 0B-4B len
01 – copy, 3bit len, 3bit offset, 1B offset
10 – copy, 6bit len, 2B offset
11 – copy, 6bit len, 4B offset

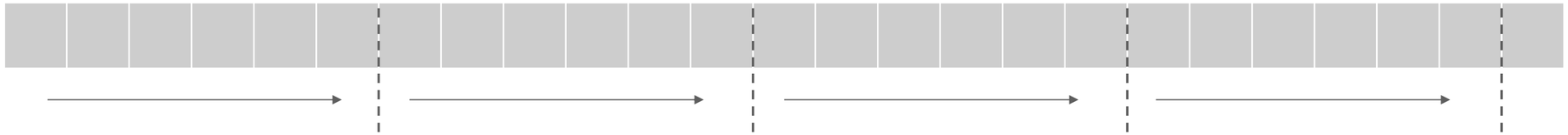


LEVELS OF PARALLELISM

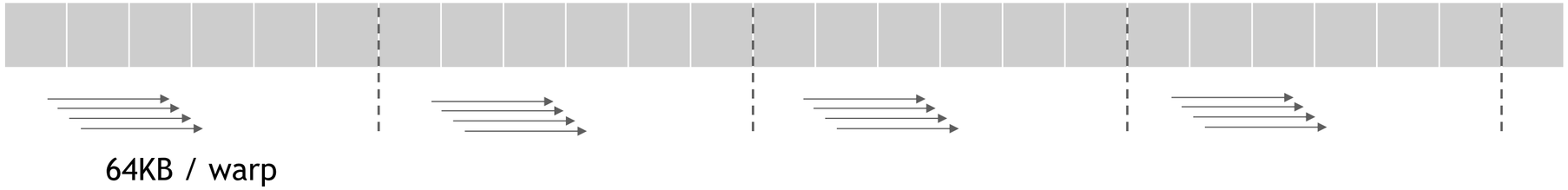
Chunking is critical

one LZ stream – difficult to map to the GPU, creates dependencies between copies, future copies depend on previous data

many smaller independent LZ streams – better mapping to the GPU, more concurrent tasks; need to store offsets/sizes

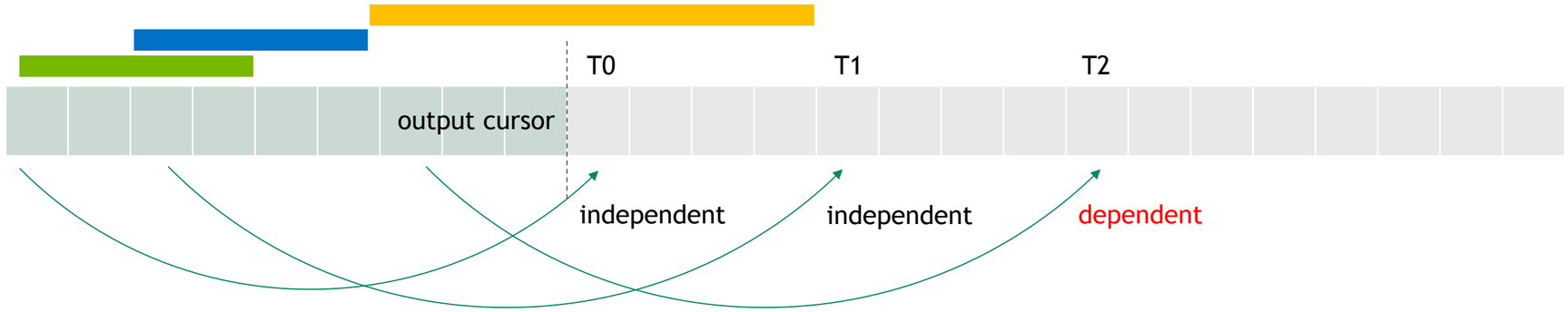


warp or threadblock-level parallelism within a LZ stream – threads cooperating on writing the output buffer



NVIDIA.

EFFICIENT MATCH COPIES



Each thread in a warp calculates if the end of the match is before the cursor -> independent copy

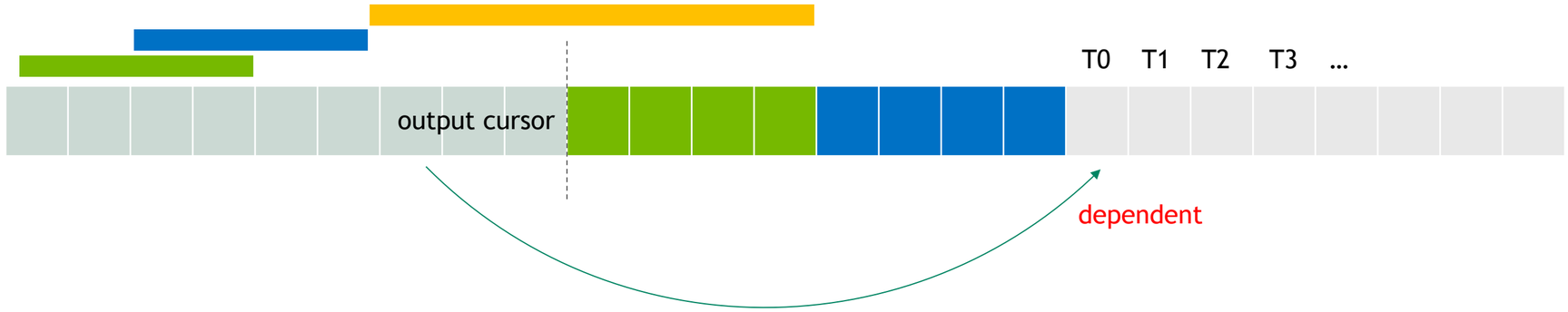
First, we process **independent copies**:

- eligible threads load up to 16B into registers (up to 5x4B loads with funnel shifts in registers)

- store from registers to global memory (up to 5x4B atomicOrs)

NVIDIA.

EFFICIENT MATCH COPIES



Dependent copies and tails of independent copies are processed one-by-one cooperatively by the whole warp

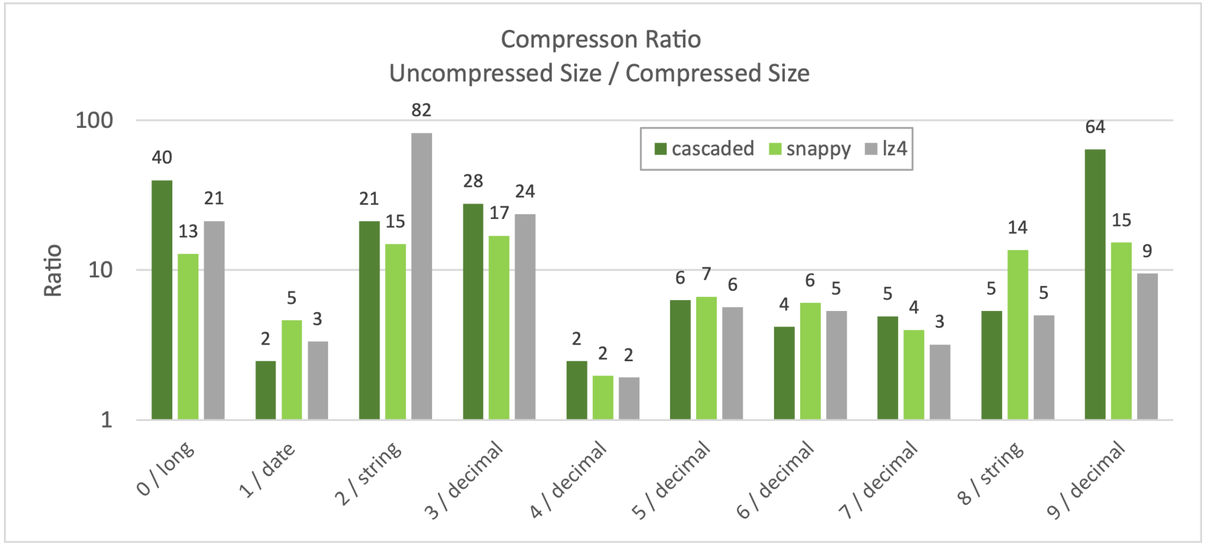
- short copies: each thread copies 1B

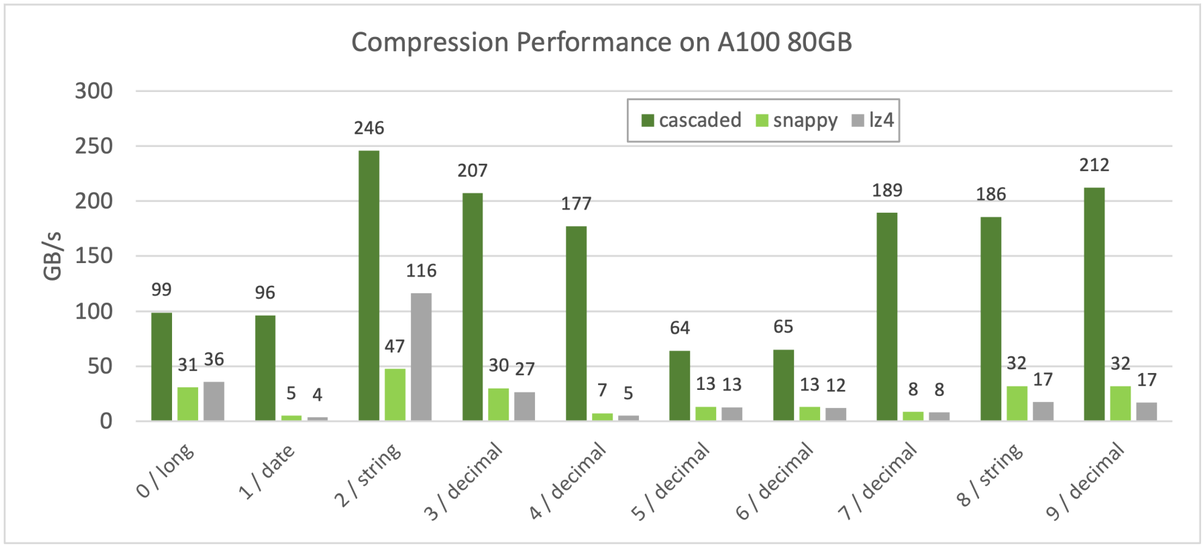
```
currDst[i] = currSrc[i % currOffset];  // take care of the overlapping copies with modulo
```

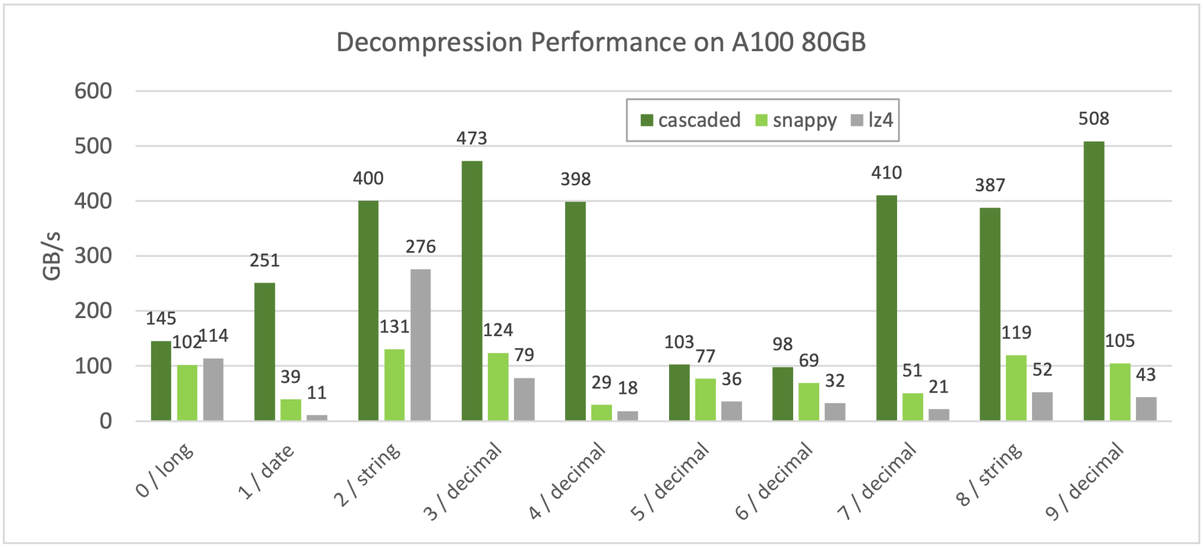
- long copies: unroll the copy loop for better ILP, each thread copies multiple bytes

NVIDIA.

PERFORMANCE



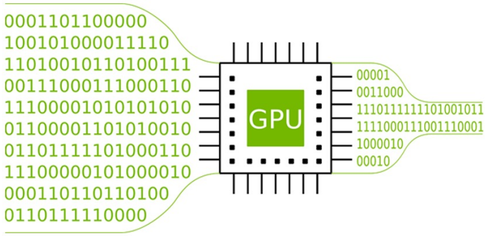




NVIDIA

NVCOMP

CUDA library for GPU compression/decompression



https://github.com/NVIDIA/nvcomp, https://developer.nvidia.com/nvcomp

Goal: generic API across all compressors, so developers can switch between compressors easily in their apps

1.0 release – July 2020: Cascaded and LZ4 with a single interface

2.0 release – April 2021: Cascaded, LZ4, Snappy, GDeflate, Bitcomp

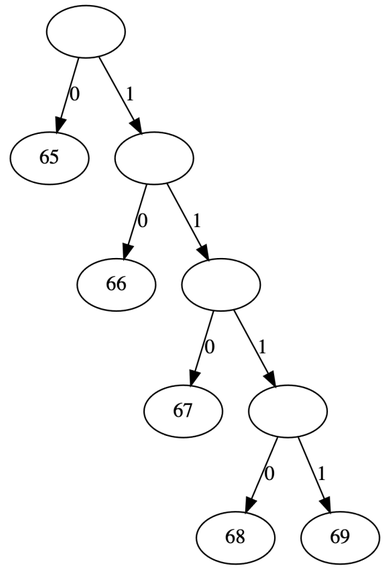
- **Low-level** is targeting advanced users — metadata and chunking must be handled outside of nvCOMP, low-level nvCOMP APIs perform batch compression/decompression of multiple streams, they are light-weight and fully asynchronous.
- **High-level** is provided for ease of use — metadata and chunking is handled internally by nvCOMP, this enables the easiest way to ramp up and use nvCOMP in applications, some of the high-level APIs are synchronous and for best performance/flexibility it's recommended to use the low-level APIs.
- Examples: interop with existing CPU libraries (liblz4), use GPUDirectStorage

NVIDIA.

THAT'S ALL GREAT, BUT WHAT ABOUT DEFLATE?

compressed block
dynamic huffman

compressed data:
- 1B literal write
- 3-258B match copies



111010110111011 0...

literal match match

Don't know where the N-th symbol starts in the stream until we decode all previous symbols

Significantly limits parallelism on GPUs!

Can rearrange stream for parallel decoding, but then it's no longer standard deflate

NVIDIA.

DPU

WHAT IS DPU?

BlueField-2 DPU

200 Gbps Ethernet & InfiniBand, NRZ & PAM4 modulation

Powered by ConnectX-6 Dx

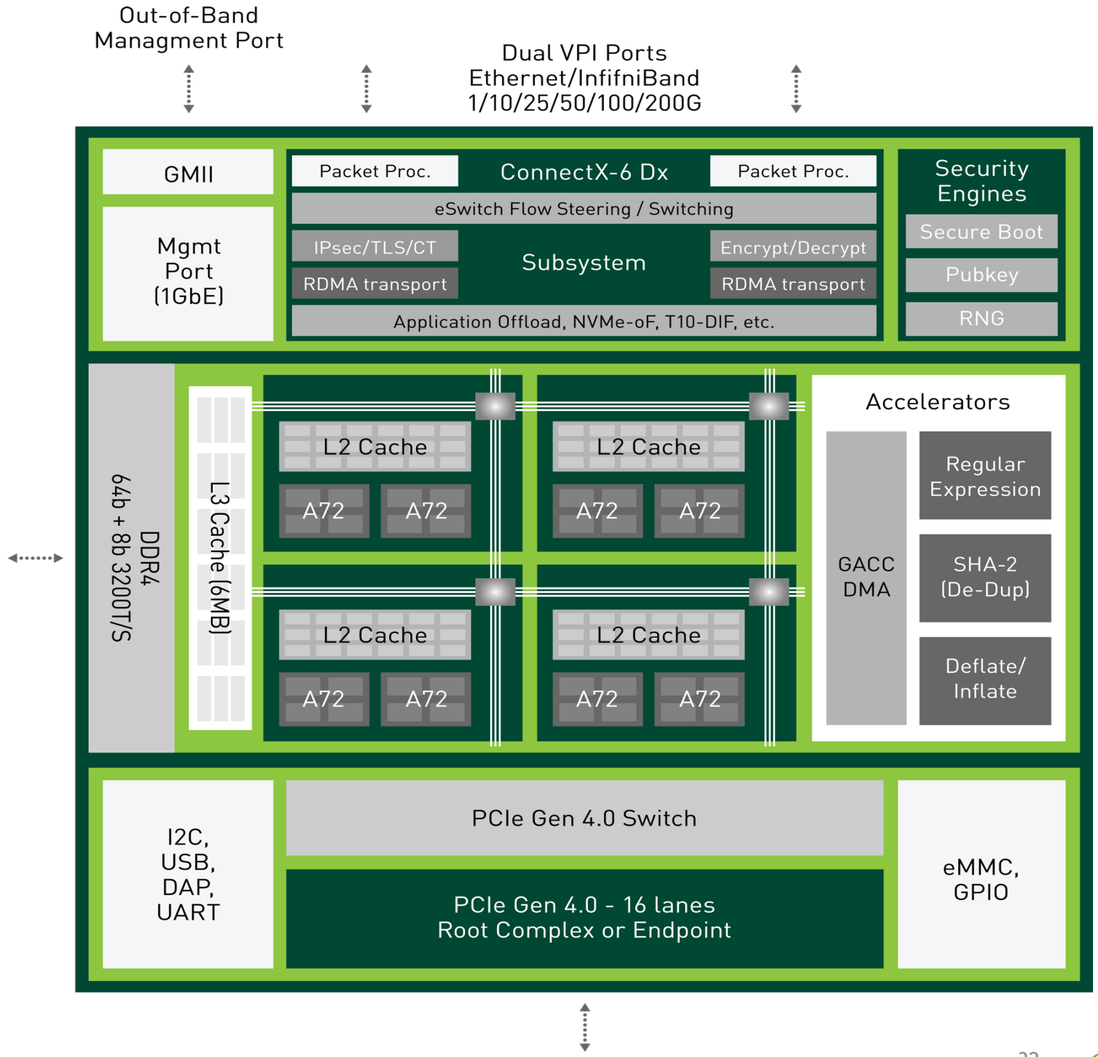
8 ARM A72 CPUs subsystem in a Tile architecture

- Accelerators for RegEx, compression, SHA hashing

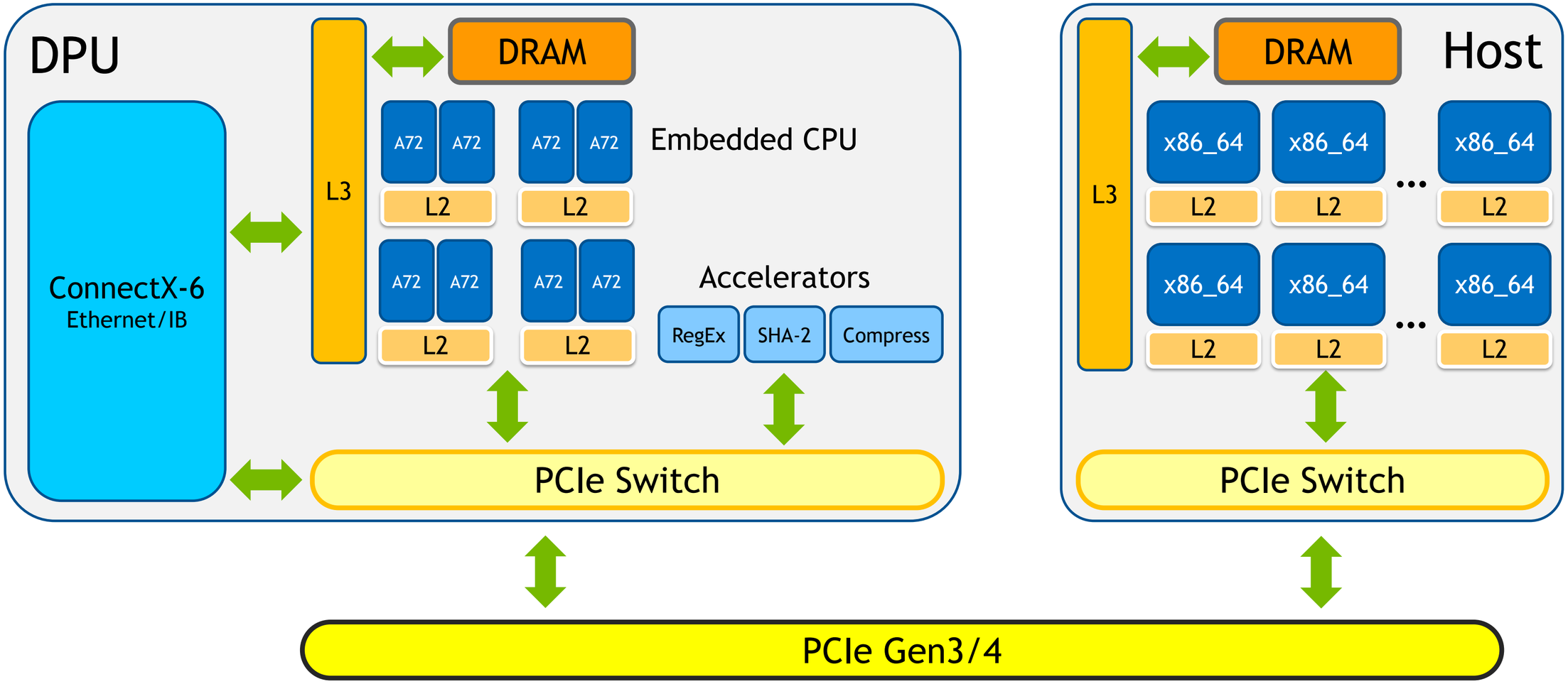
Runs its own OS (Ubuntu based)

Fully integrated PCIe switch, 16 bi-furcated Gen4.0

- Root Complex or End Point modes

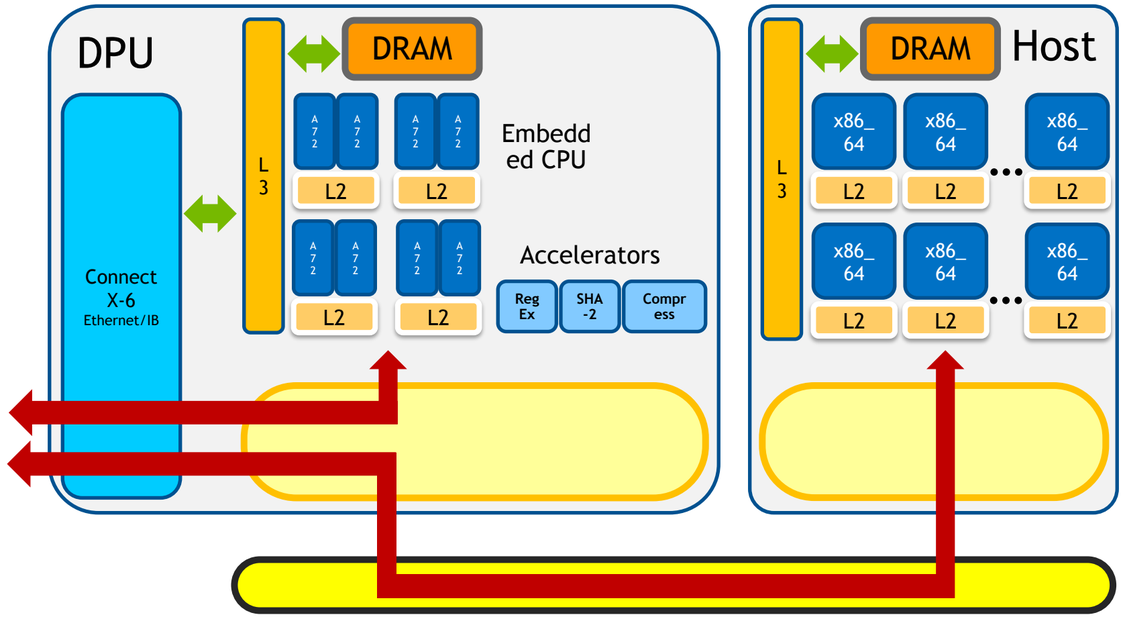


BLUEFIELD-2 SYSTEM (HOST + DPU) ARCHITECTURE



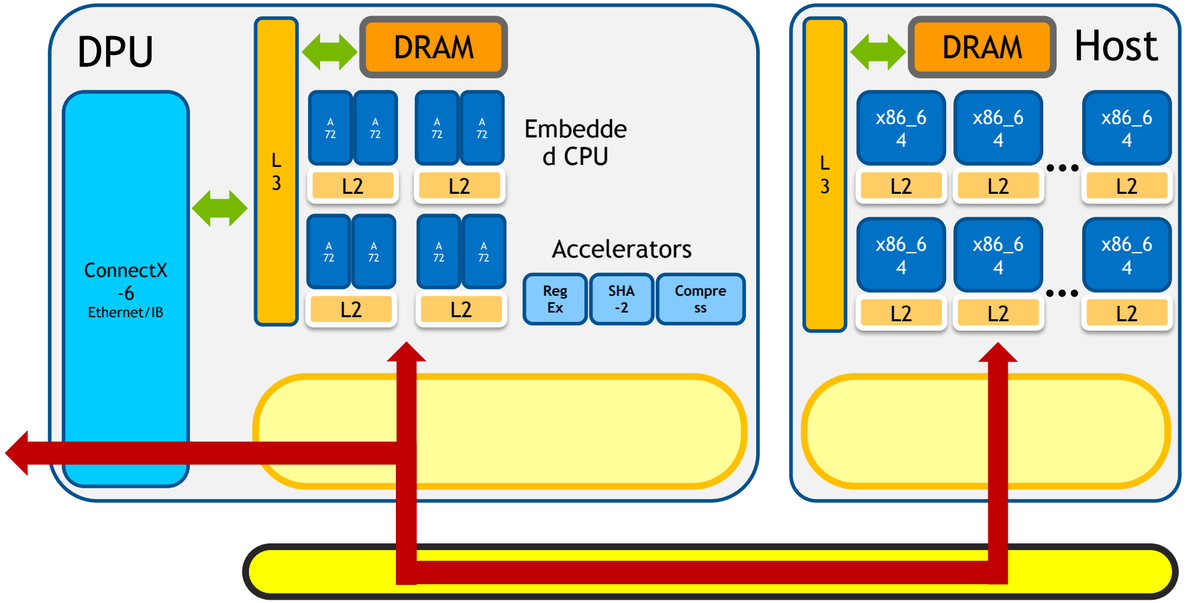
DPU OPERATION MODES

Separated Host



- *Host* and *Embedded CPU* are symmetric, and can operate independently, separately, and simultaneously

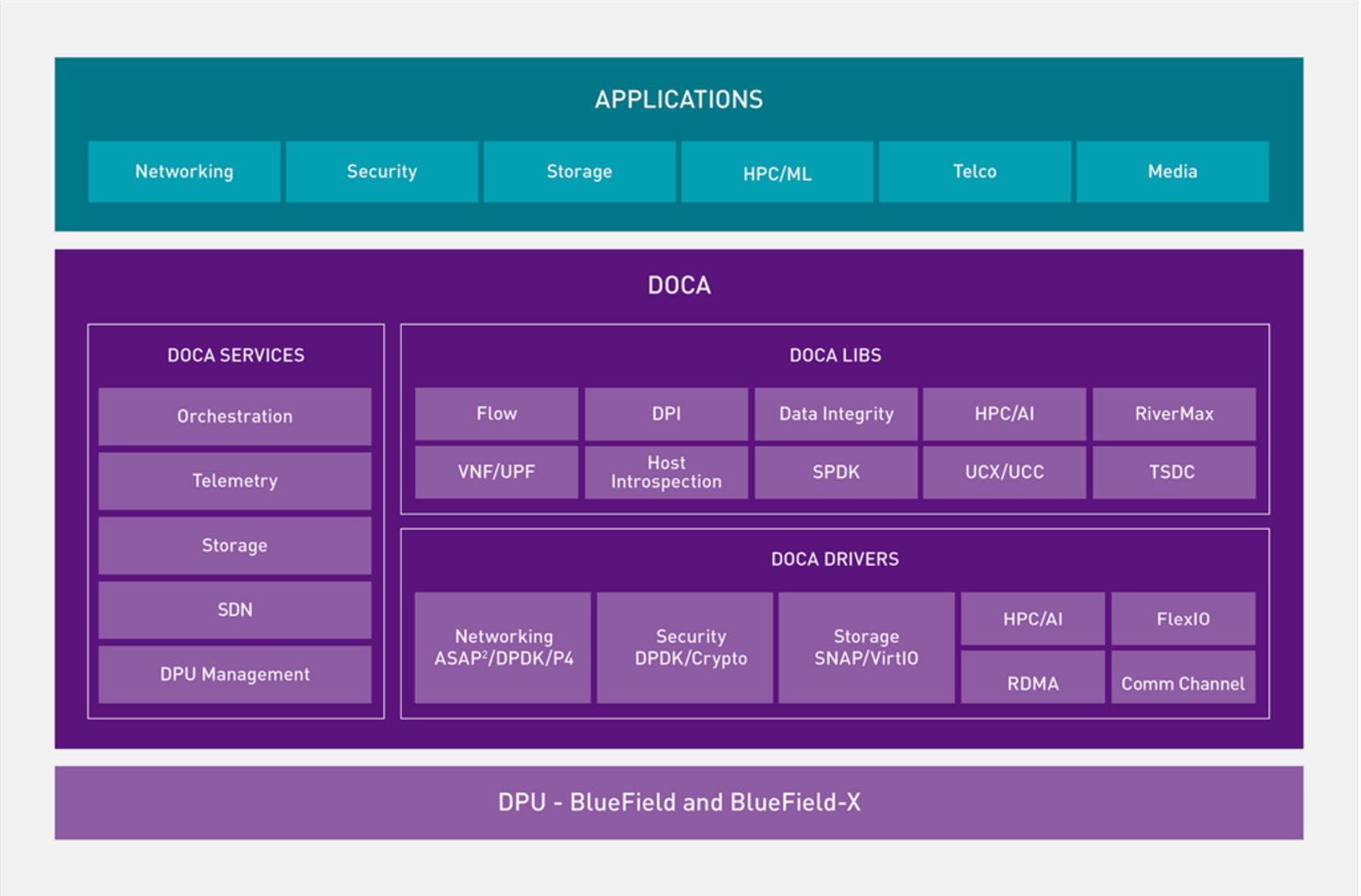
Embedded CPU Function



- NIC resources and functionality are owned and controlled by the *Embedded CPU*

DATA CENTER ON-A-CHIP ARCHITECTURE (DOCA) SDK

Initial release 1.0 in April 2021, current release 1.1



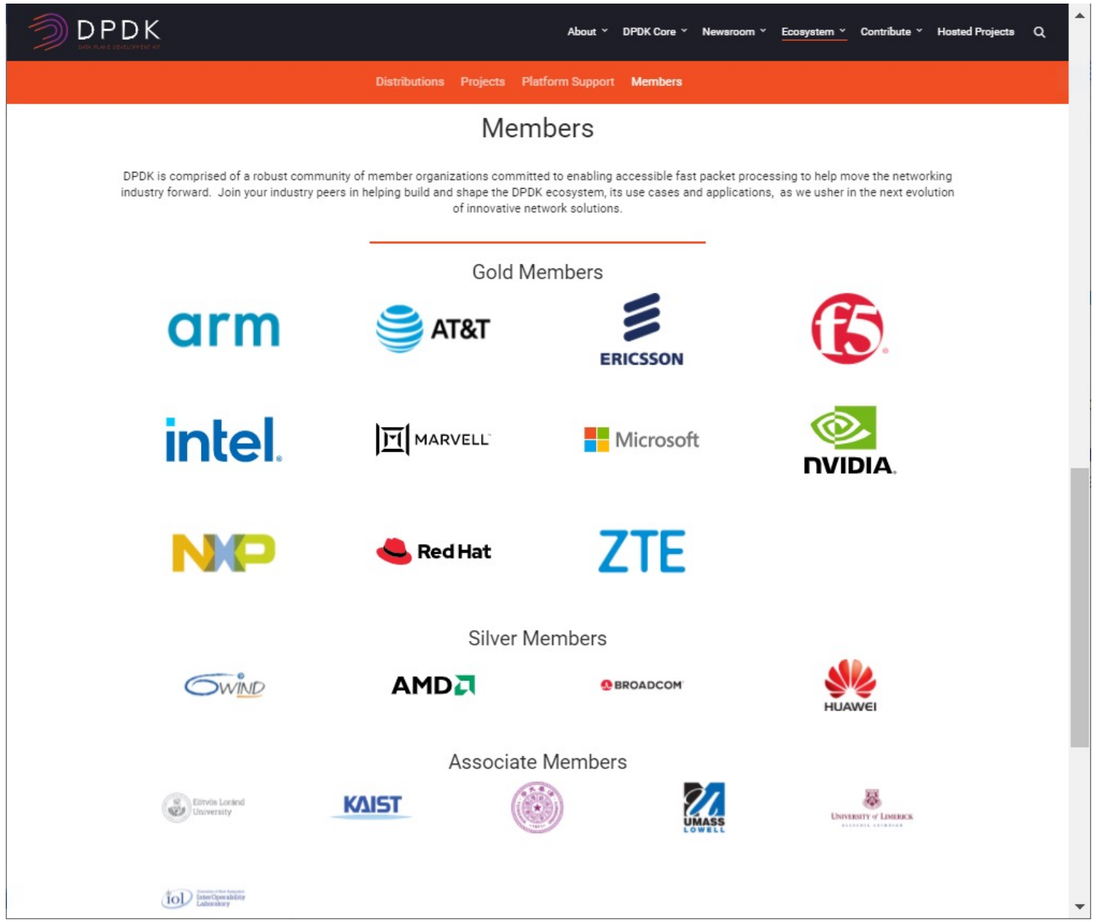
NVIDIA.

DATA PLANE DEVELOPMENT KIT (DPDK)

DPDK was created in 2010 by Intel and made available under a permissive open-source license

De facto standard for high performance user space networking in the telecommunications industry

Provides framework for Bluefield accelerators in the NVIDIA DOCA SDK



DEFLATE TEST

Silesia corpus dataset, using `dpdk-test-compress-perf` from DPDK 21.05

Preliminary BlueField-2 results:

Comp throughput **39.06 Gbps = 4.88GB/s**

Decomp throughput **52.90 Gbps = 6.61GB/s**

Comp ratio 42.45%

https://github.com/inikep/lzbench

1 core of Intel Core i7-8700K, Ubuntu 18.04.3 64-bit

app	Compression	Decompression	Compressed size	Compression ratio
lz4 1.9.2	737 MB/s	4448 MB/s	100880800	47.60
lz4fast 1.9.2 -3	838 MB/s	4423 MB/s	107066190	50.52
lz4fast 1.9.2 -17	1201 MB/s	4632 MB/s	131732802	62.15
lz4hc 1.9.2 -1	131 MB/s	4071 MB/s	83803769	39.54
lz4hc 1.9.2 -4	81 MB/s	4210 MB/s	79807909	37.65
lz4hc 1.9.2 -9	33 MB/s	4378 MB/s	77884448	36.75
lz4hc 1.9.2 -12	11 MB/s	4427 MB/s	77262620	36.45
libdeflate 1.3 -1	201 MB/s	865 MB/s	73318371	34.59
libdeflate 1.3 -3	161 MB/s	912 MB/s	70668968	33.34
libdeflate 1.3 -6	99 MB/s	924 MB/s	67928189	32.05
libdeflate 1.3 -9	16 MB/s	898 MB/s	65701539	31.00
libdeflate 1.3 -12	7.39 MB/s	900 MB/s	64801629	30.57

NVIDIA.

DEFLATE TEST

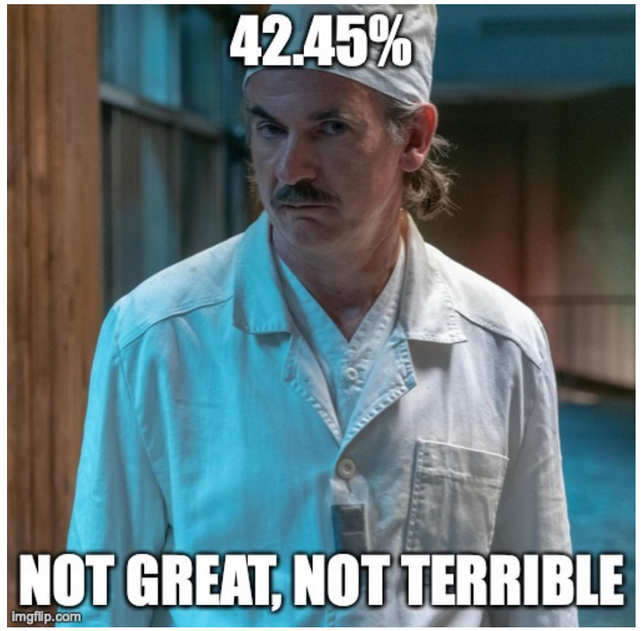
Silesia corpus dataset, using `dpdk-test-compress-perf` from DPDK 21.05

Preliminary BlueField-2 results:

Comp throughput **39.06 Gbps = 4.88GB/s**

Decomp throughput **52.90 Gbps = 6.61GB/s**

Comp ratio 42.45%



https://github.com/inikep/lzbench

1 core of Intel Core i7-8700K, Ubuntu 18.04.3 64-bit

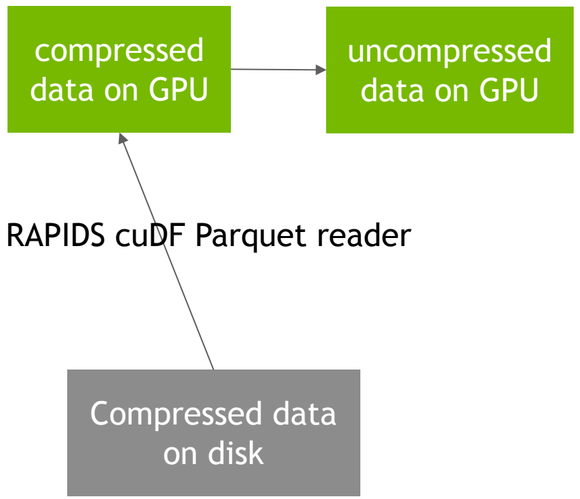
app	Compression	Decompression	Compressed size	Compression ratio
lz4 1.9.2	737 MB/s	4448 MB/s	100880800	47.60
lz4fast 1.9.2 -3	838 MB/s	4423 MB/s	107066190	50.52
lz4fast 1.9.2 -17	1201 MB/s	4632 MB/s	131732802	62.15
lz4hc 1.9.2 -1	131 MB/s	4071 MB/s	83803769	39.54
lz4hc 1.9.2 -4	81 MB/s	4210 MB/s	79807909	37.65
lz4hc 1.9.2 -9	33 MB/s	4378 MB/s	77884448	36.75
lz4hc 1.9.2 -12	11 MB/s	4427 MB/s	77262620	36.45
libdeflate 1.3 -1	201 MB/s	865 MB/s	73318371	34.59
libdeflate 1.3 -3	161 MB/s	912 MB/s	70668968	33.34
libdeflate 1.3 -6	99 MB/s	924 MB/s	67928189	32.05
libdeflate 1.3 -9	16 MB/s	898 MB/s	65701539	31.00
libdeflate 1.3 -12	7.39 MB/s	900 MB/s	64801629	30.57

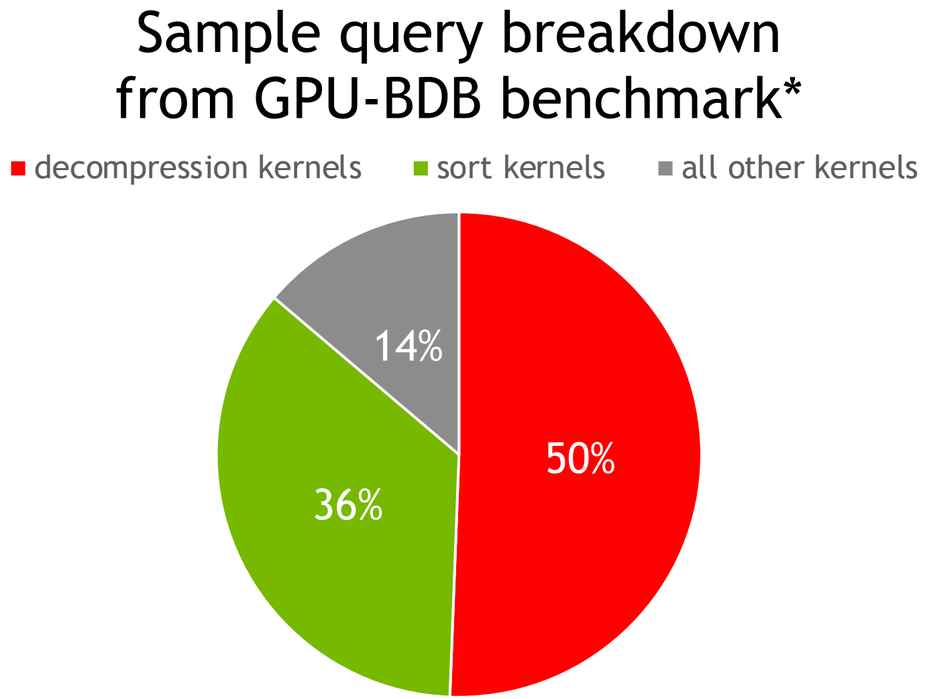
NVIDIA.

USE CASES

WHY COMPRESSION?

Accelerating IO in Data Center





WHAT MATTERS

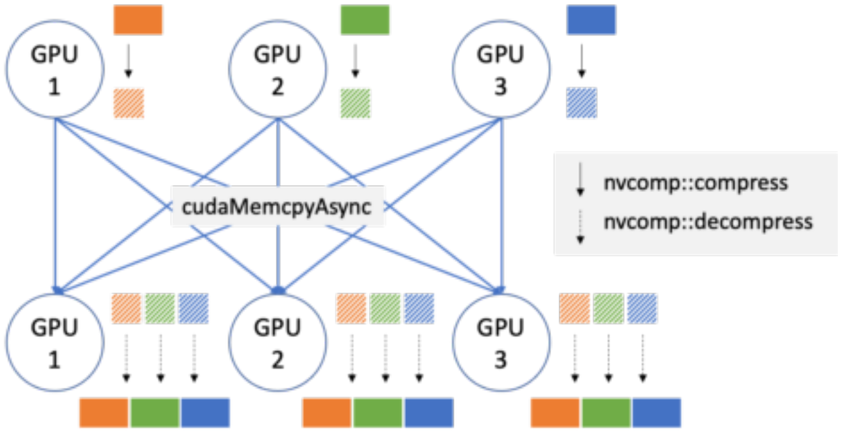
Decompression speed
Support for standard formats

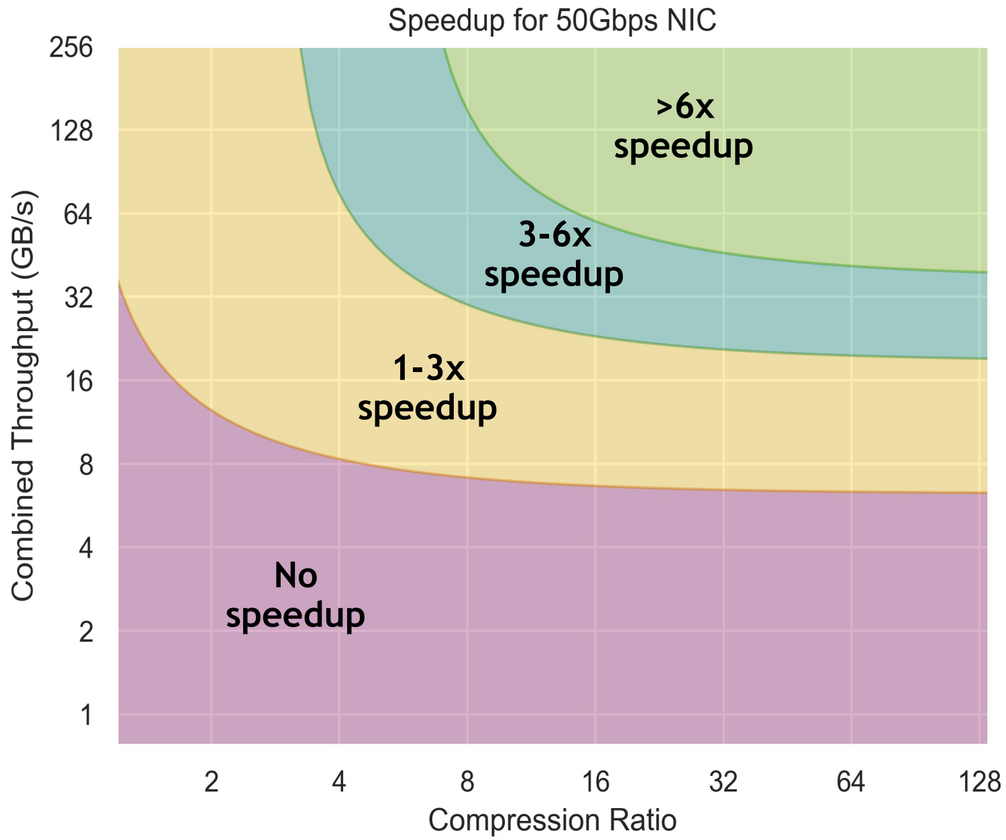
* https://github.com/rapidsai/gpu-bdb

NVIDIA.

WHY COMPRESSION?

Accelerating Communications





WHAT MATTERS

Compression/decompression speed
Compression ratio

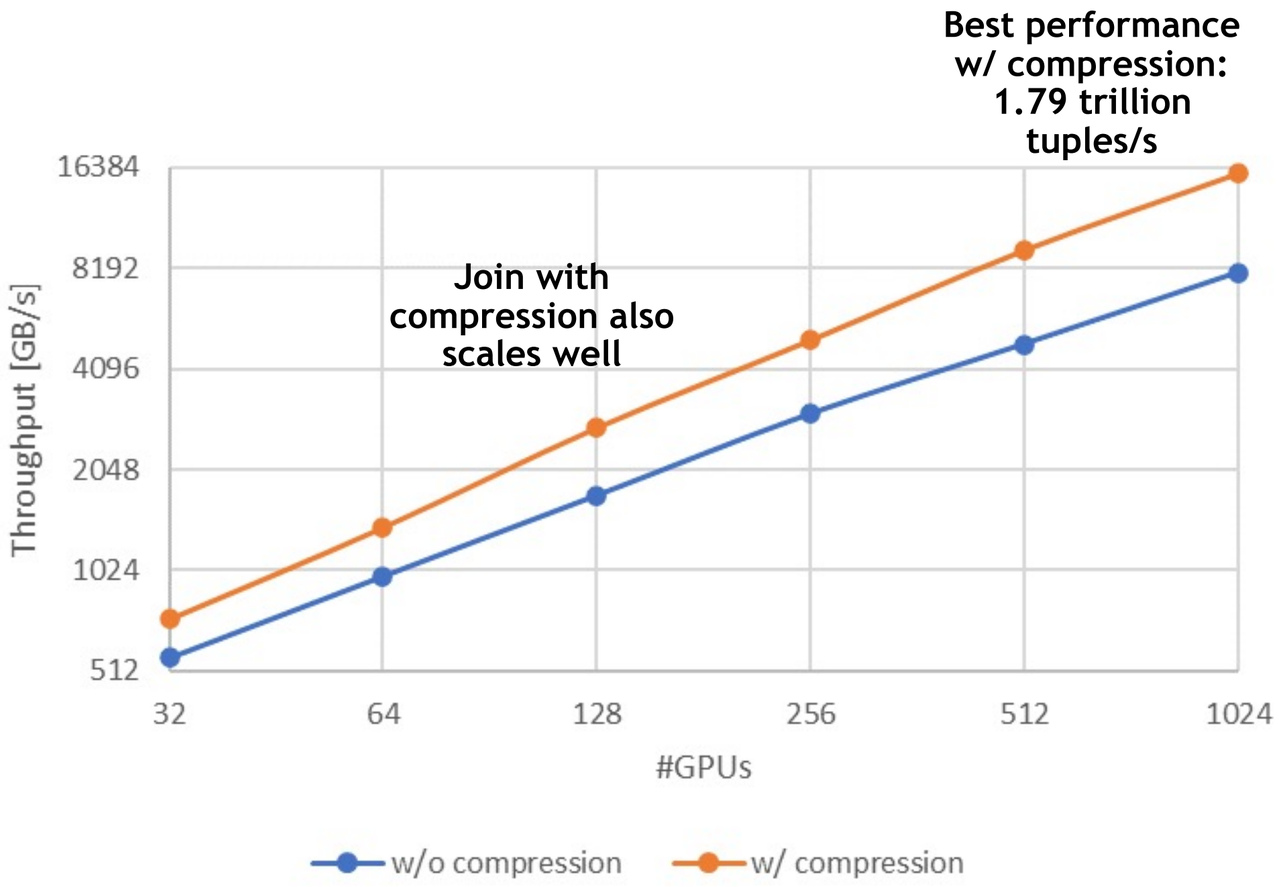
https://developer.nvidia.com/blog/optimizing-data-transfer-using-lossless-compression-with-nvcomp/

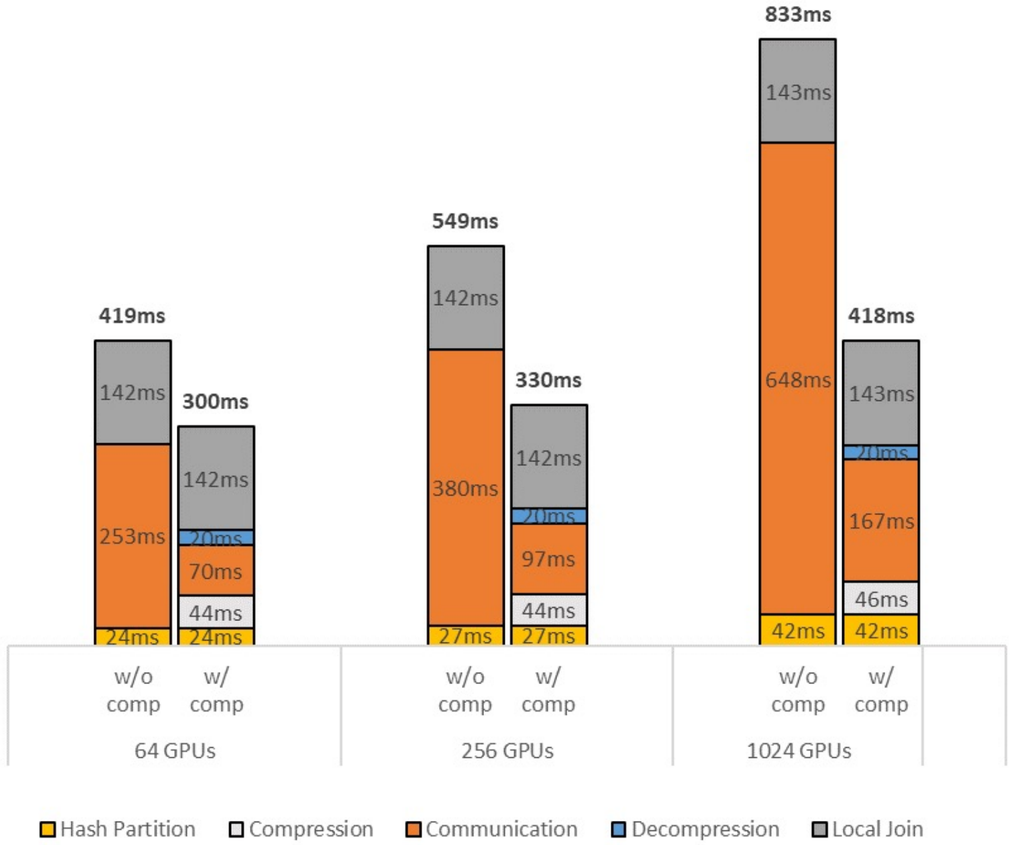
GTC 2021 S31645: Optimizing Data Movement with Compression for Database Applications

NVIDIA.

SCALING JOIN TO A THOUSAND GPUS

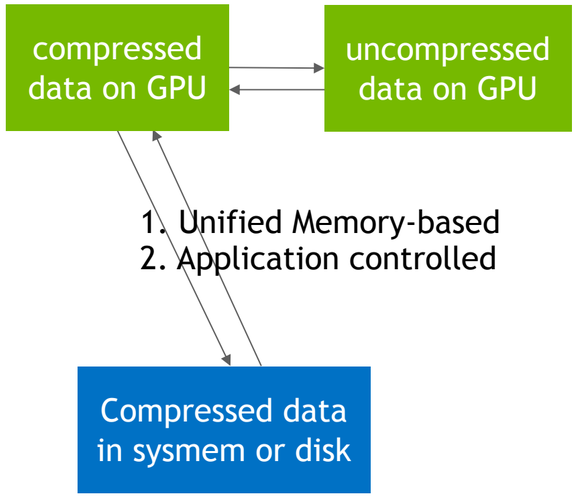
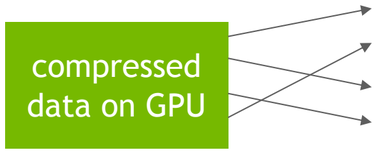
Session 4 @ 8pm CET, *Hao Gao*





WHY COMPRESSION?

Emerging use cases: extending GPU capacity

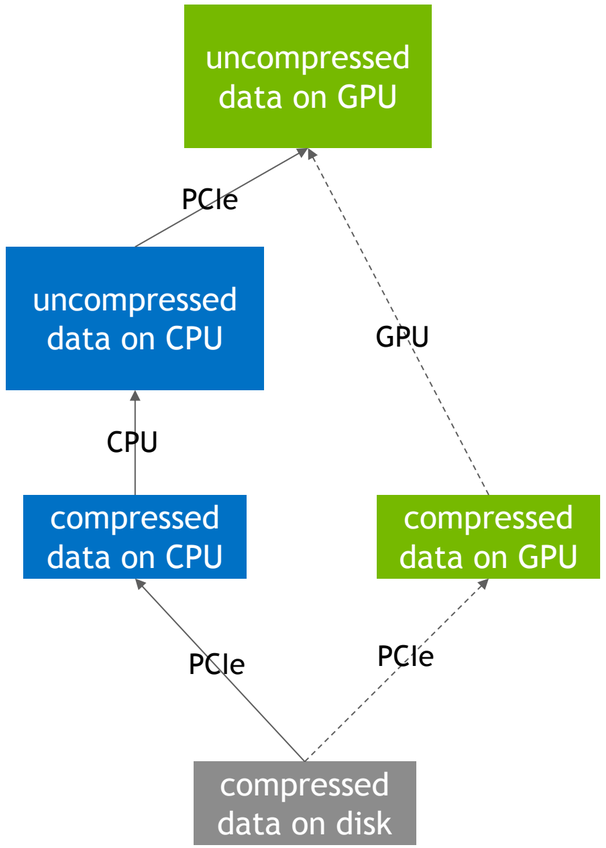
	Oversubscription	In-place decompression
		
EXAMPLES	1. Unified Memory-based 2. Application controlled	Doesn't exist today in any frameworks
WHAT MATTERS	Compression ***AND*** decompression speed Relatively coarse granularity (**page or many pages**)	Decompression speed Relatively fine granularity (**element-wise access**)

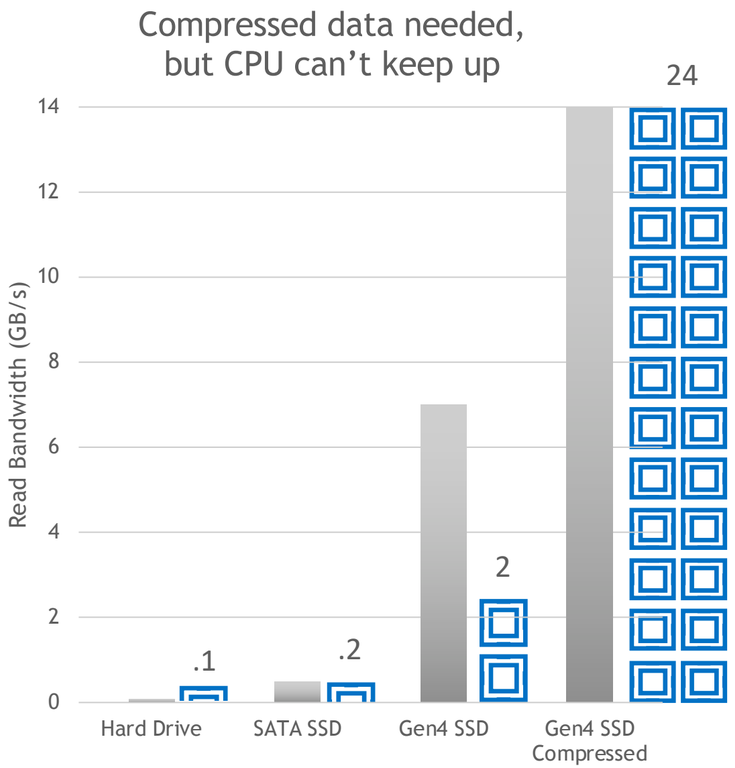
WHY COMPRESSION?

Accelerating IO in Graphics

1) bulk loading of levels and large scenes

2) streaming large geometry/textures



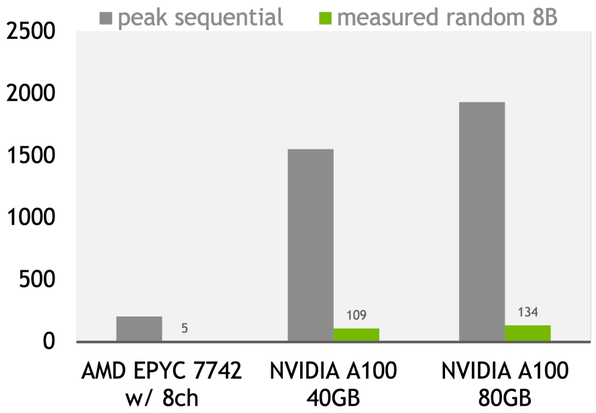


https://www.nvidia.com/en-us/geforce/news/rtx-io-gpu-accelerated-storage-technology/

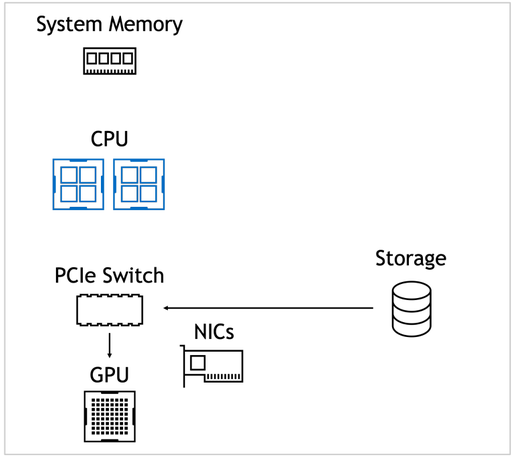
NVIDIA

RECAP

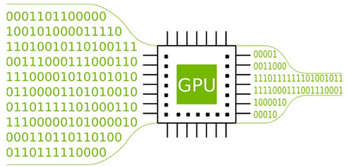
Fast Memory

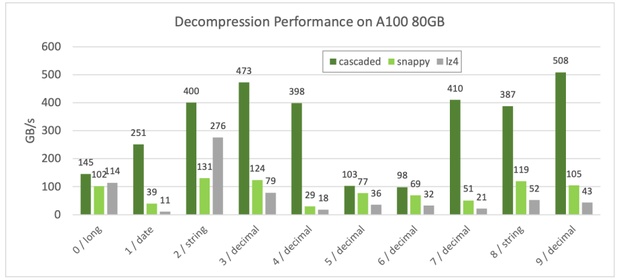


GPUDirectStorage

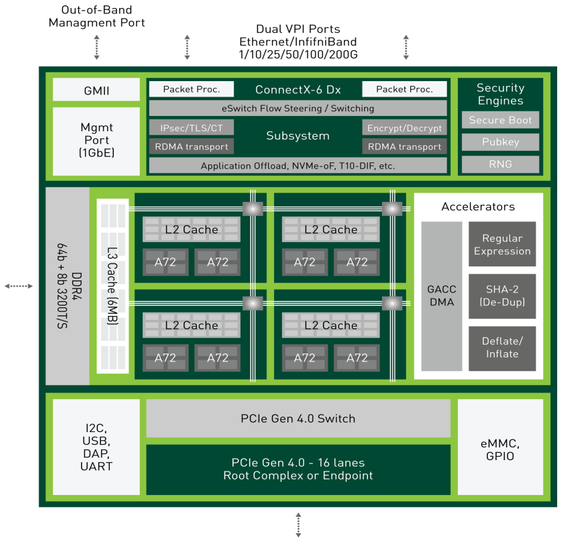


nvCOMP

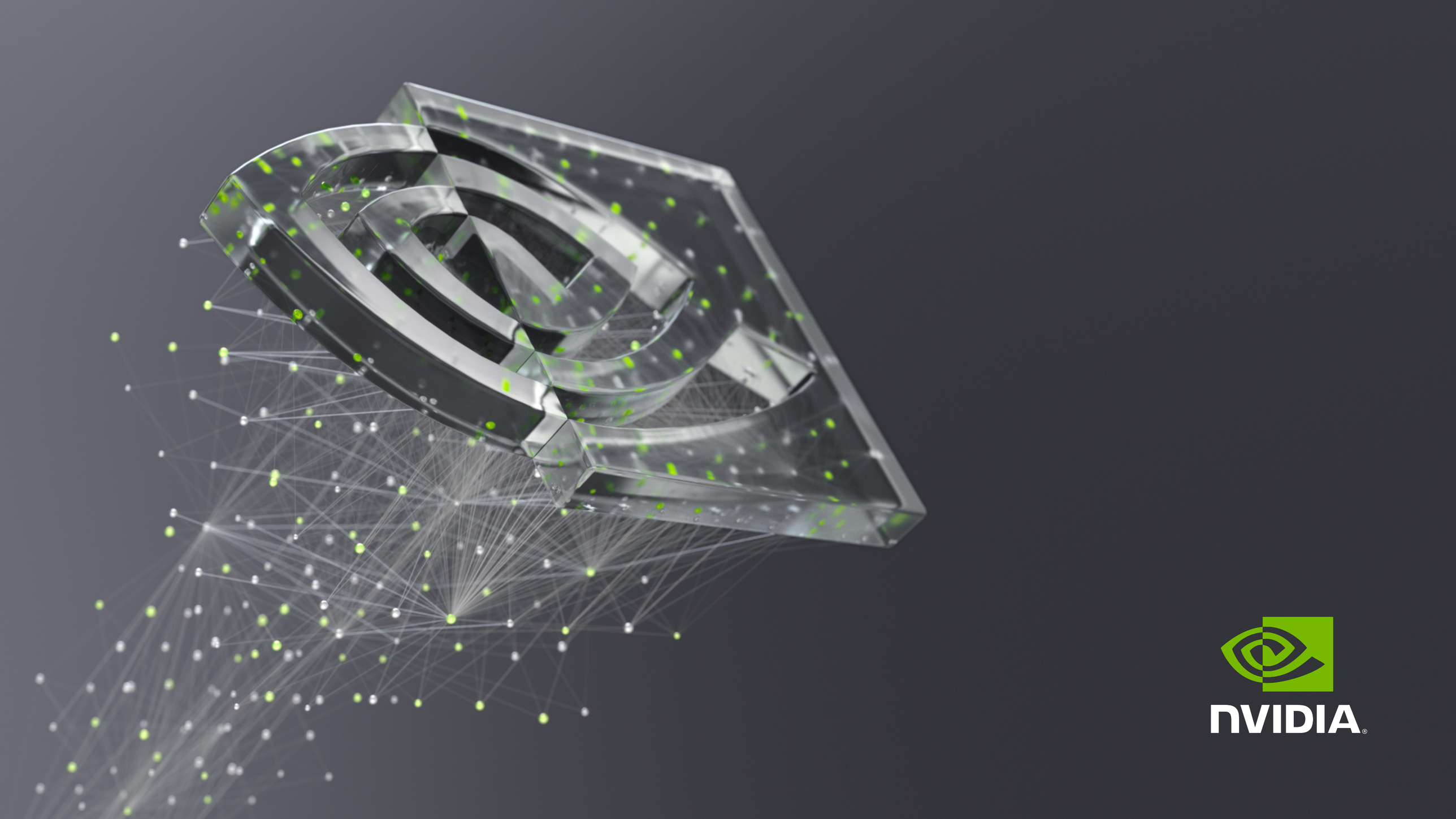




DPU



1. Think about ways to leverage compression in your applications
2. Try nvCOMP - participate in shaping out the future of compression/decompression on GPU
3. Lots of open ground for researching new comp techniques for massively parallel architectures


NVIDIA