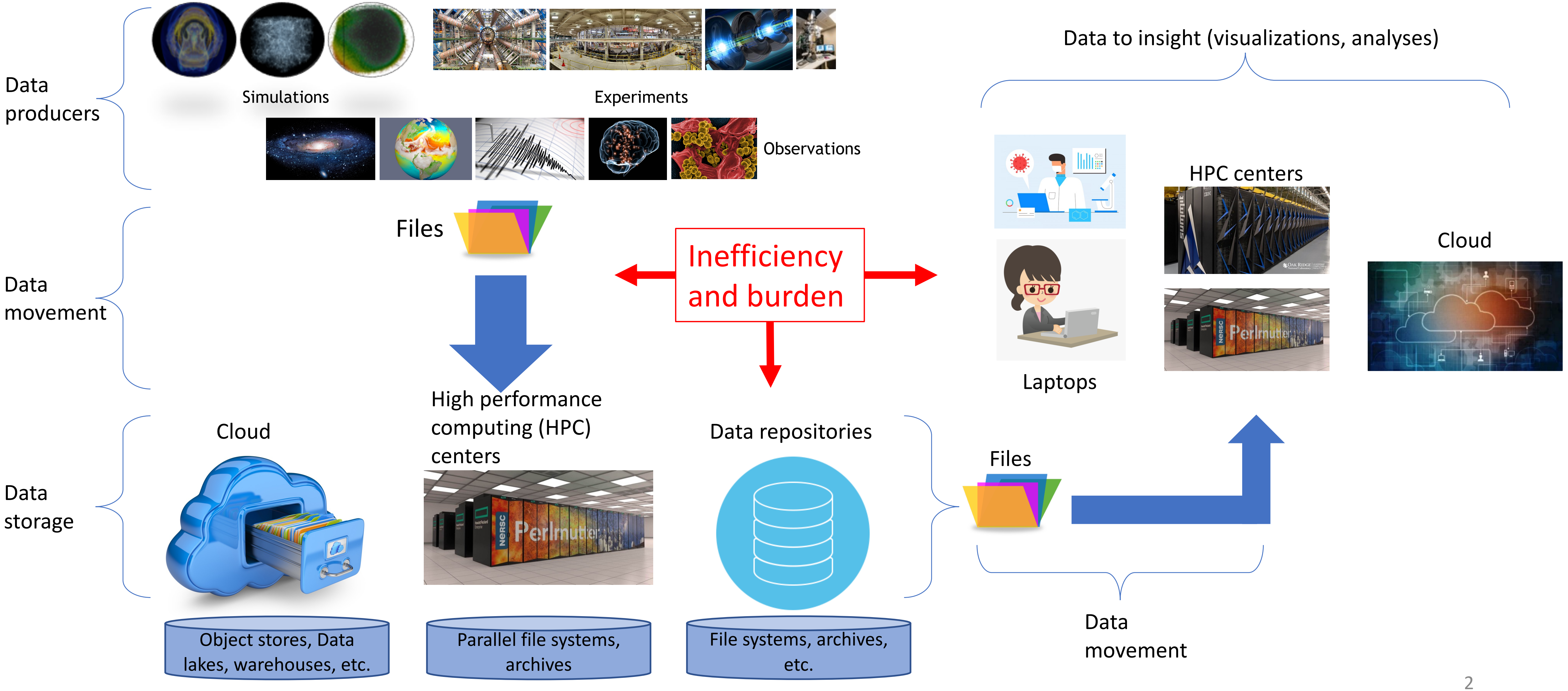
Runway: In-transit Data Compression on Heterogeneous HPC Systems

John Ravi, Suren Byna, and Michela Becchi

North Carolina State University, The Ohio State University, and Lawrence Berkeley National Laboratory

Scientific data storage and access – Many sources of inefficiency



Trends in computing devices

- Heterogeneous processing devices
  - CPUs
  - GPUs
  - FPGAs
  - Special purpose accelerators
    - Data processing units (DPUs)
    - Smart SSDs
    - Smart NICs
- Massive concurrency
- Locations of data generation and consumption
  - Traditional: In compute nodes
  - Trends: In network, in storage, and at the edge

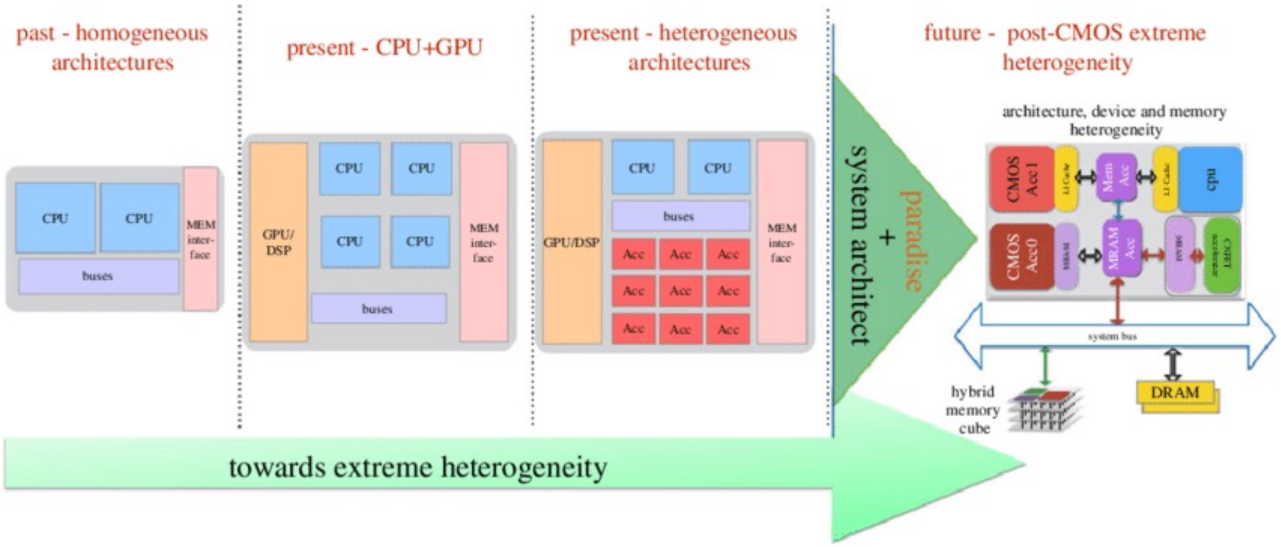
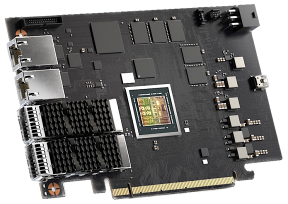


Image from D. Vasudeven, via J. Shalf, Extreme Heterogeneity workshop report

DPUs



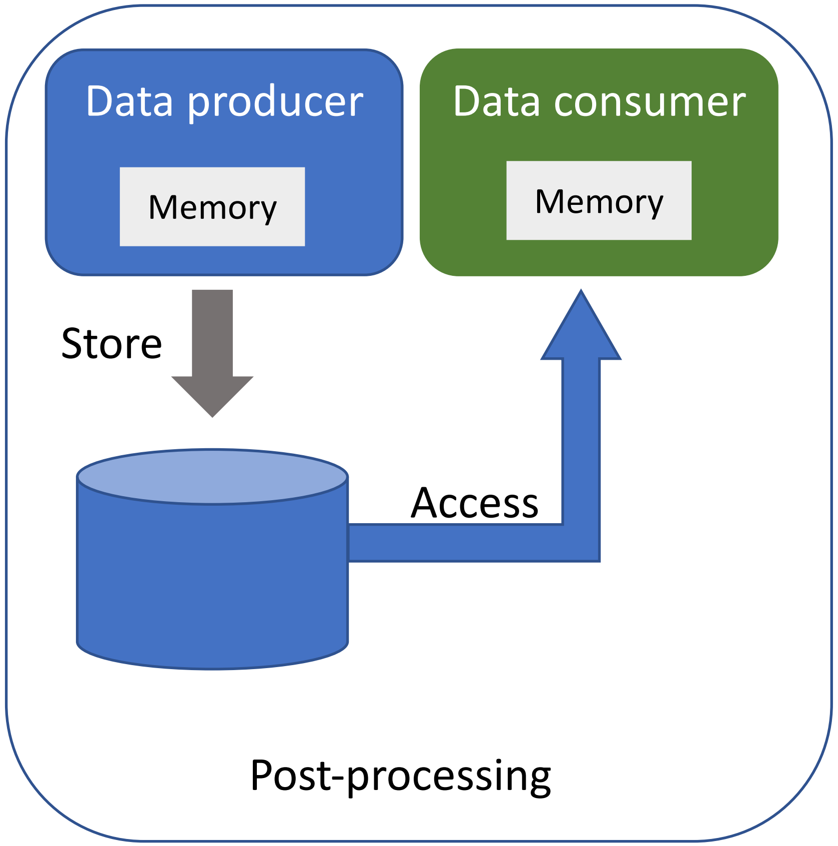
Research questions

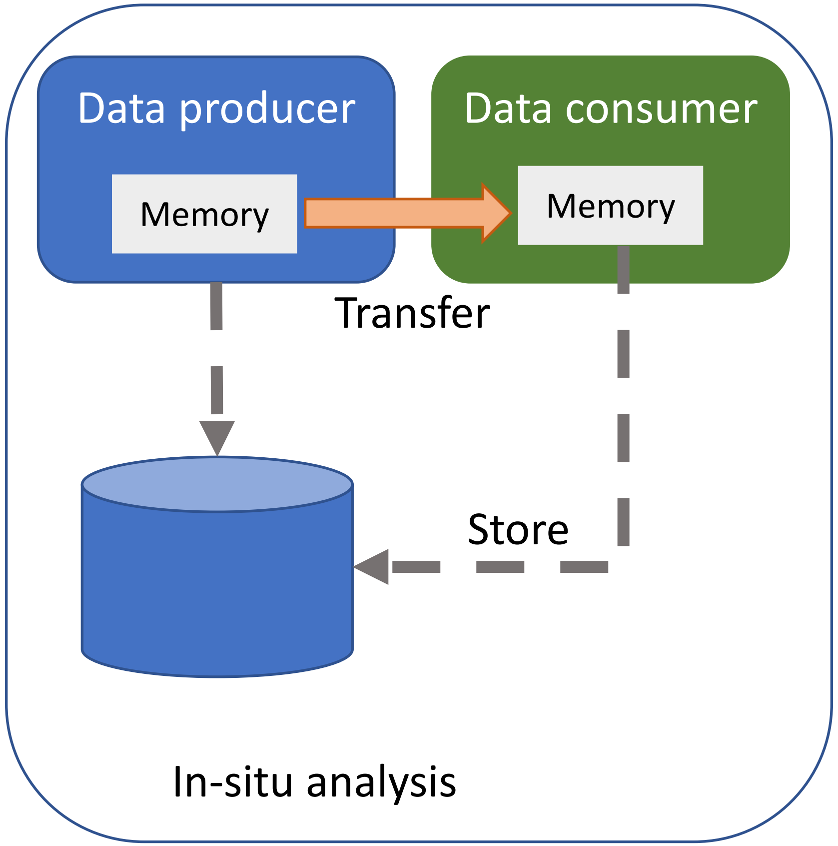
- How would accelerators benefit data analysis or transforms?
  - While data is moving between memory and storage

- Can we predict data transform (compression) cost on CPUs and GPUs to design a scheduler?

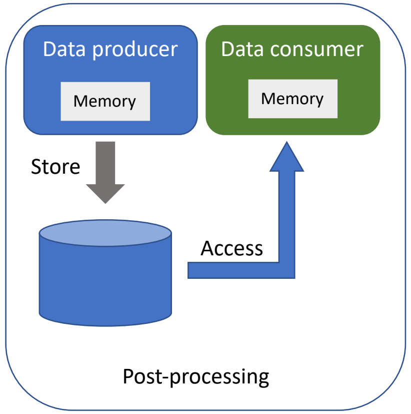
- Is non-uniform compression on different regions of the data beneficial?

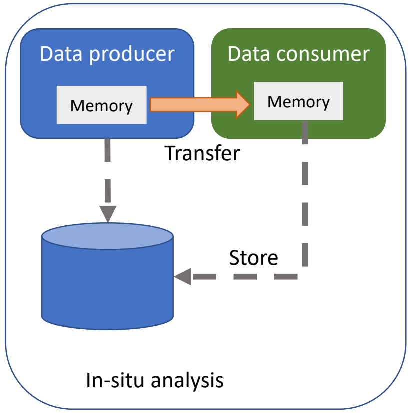
Analysis paradigms

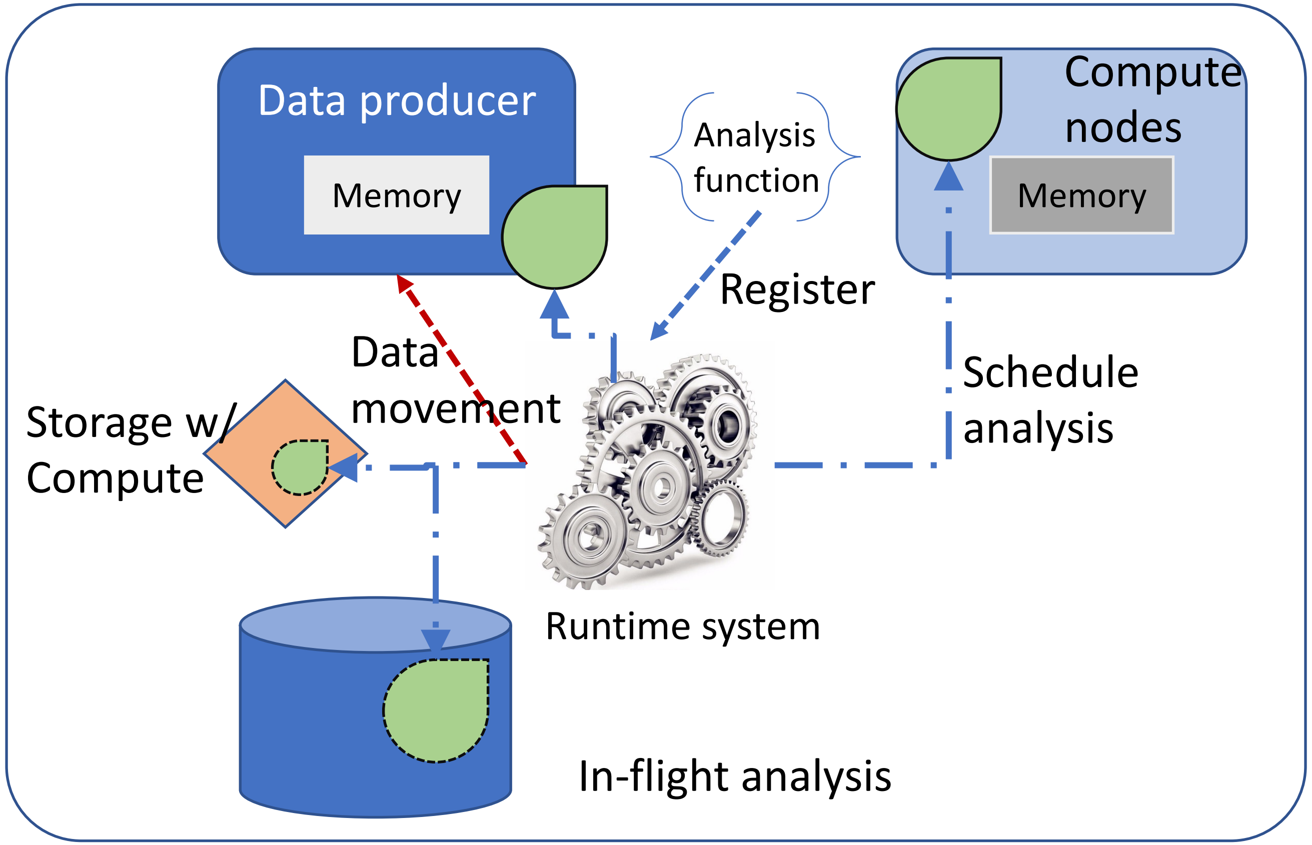




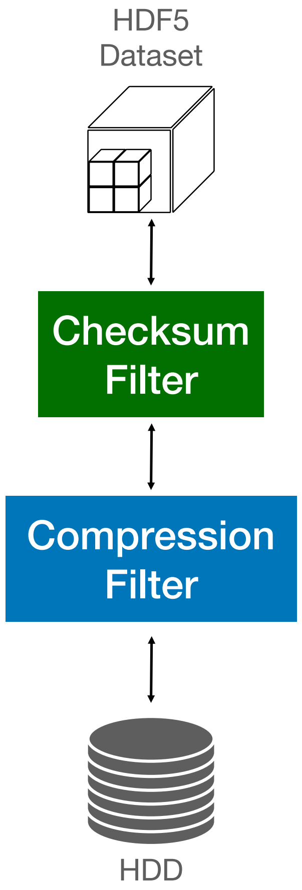
In-transit / in-flight analysis







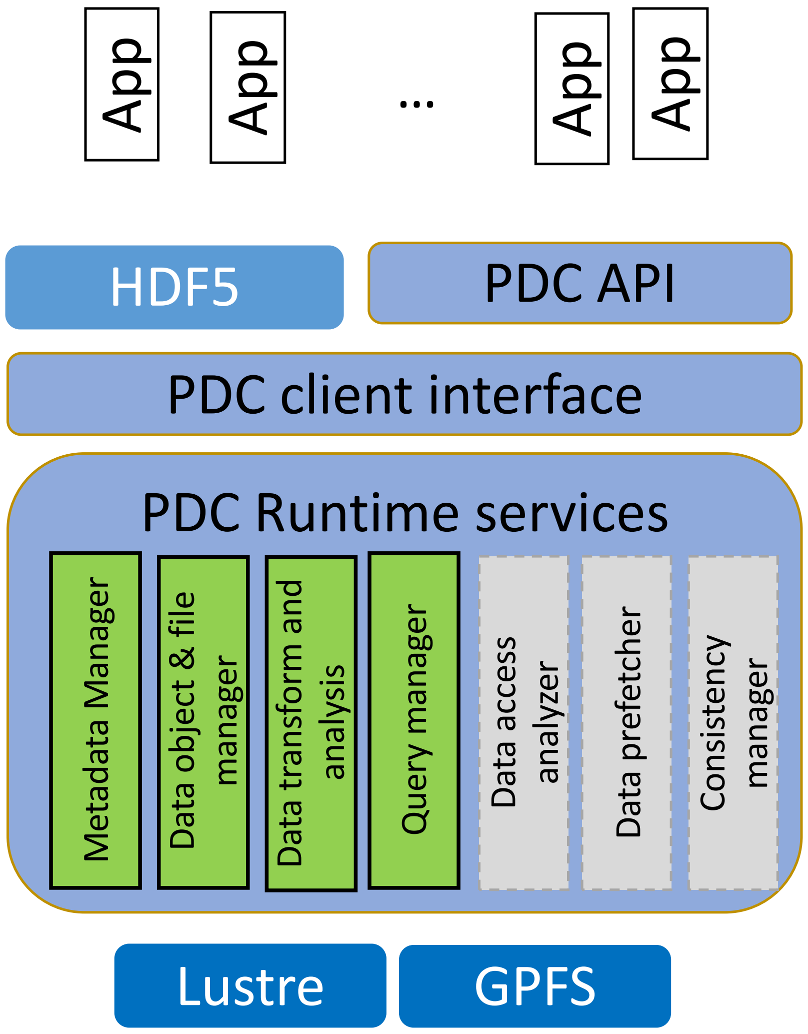
In-transit data compression



- Focus on compression
  - Decrease storage and bandwidth requirements for applications
- Current I/O middleware
  - HDF5 Filters allow in-transit compression
  - HDF5 has no policy to map compute to a particular device
  - Shared environments—need a daemon to monitor device utilization, cost to transfer data

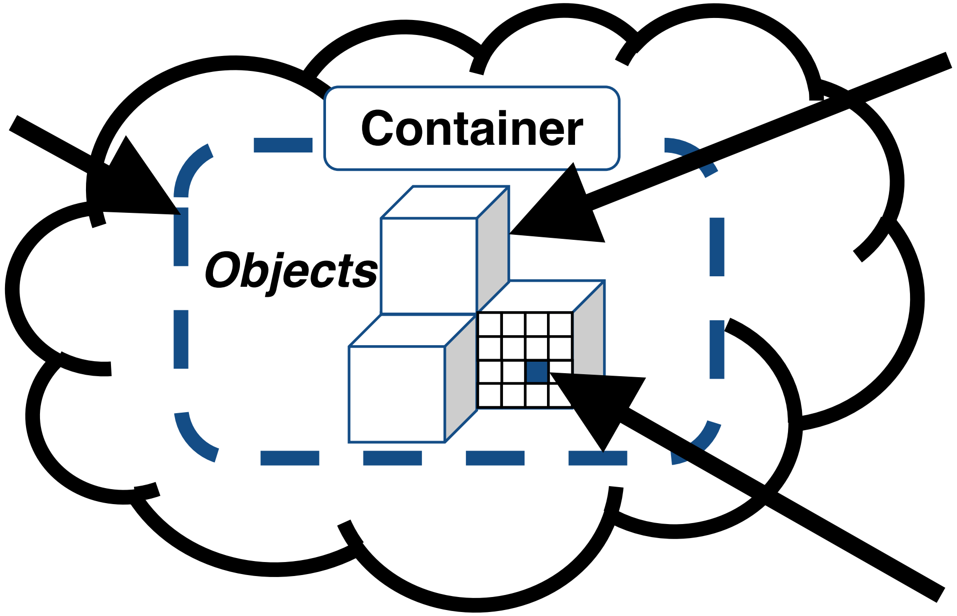
Proactive Data Containers (PDC)

- Object-centric abstraction with a runtime system for data movement orchestration
- Autonomous data management
  - Proactive use of memory hierarchy
- Support for extracting information from data
  - Information management
  - Simulation time analytics
  - Interaction among multiple datasets



Proactive Data Containers - object abstraction

PDC organizes data as a set of *objects* within a *Container*

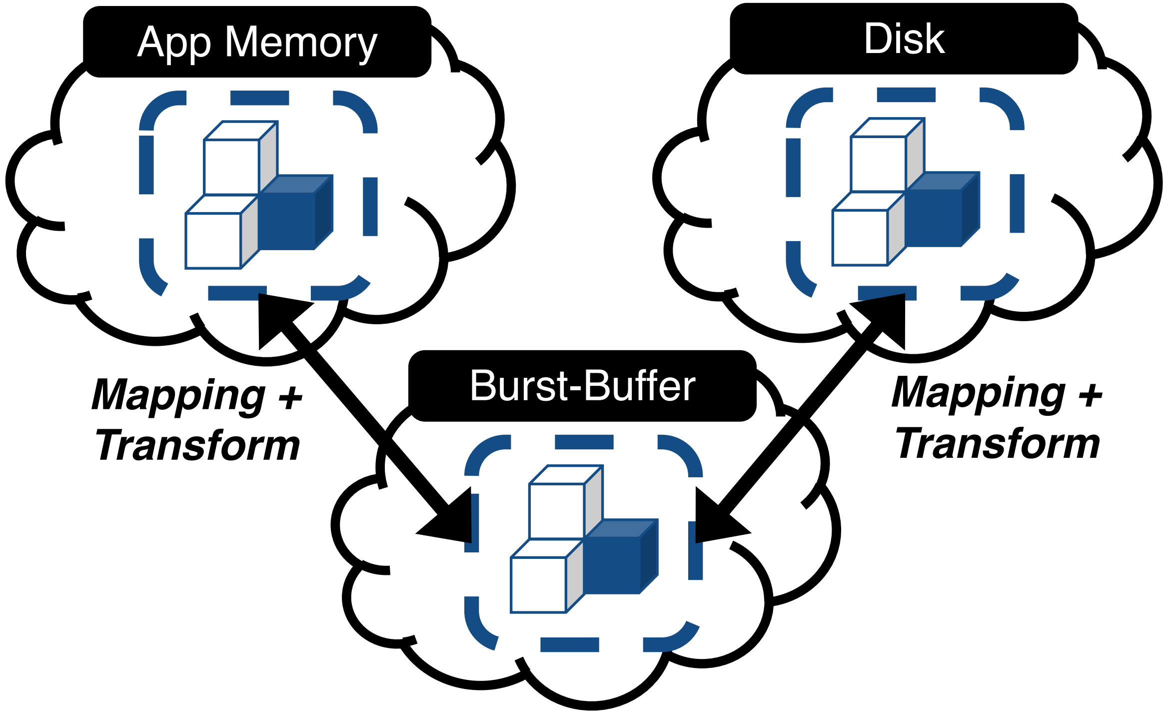


Object is a generic term to describe byte streams in an abstract manner

Region is the basic and fine-grain unit for data movement operations in PDC

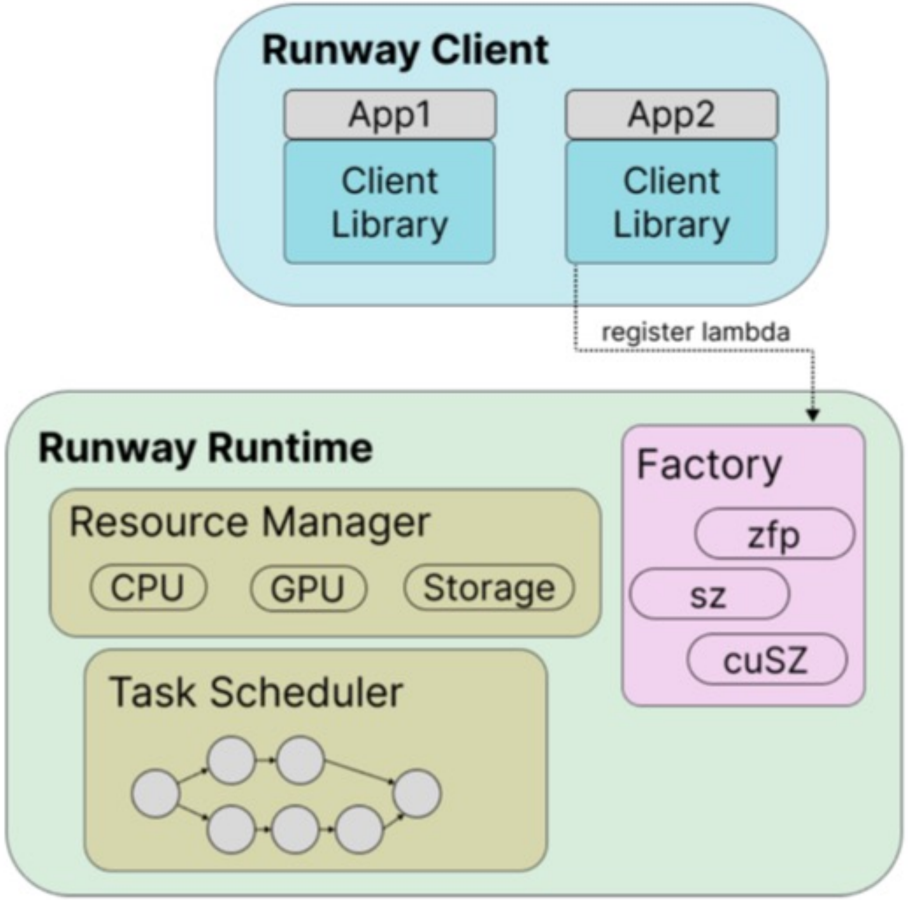
Proactive Data Containers - object operations

- No explicit data movement
- Object mapping
  - Data movement operations *implicit*
  - Similar to `mmap()`
  - Transform
- Concurrent access
  - Explicit lock operation per region
  - Unlocked region = data movement can occur from/to that region
- Primitives: `map`/`unmap` & `transfer` (`wait` for completing a `transfer`)



https://github.com/hpc-io/pdc

Runway for in-transit transforms (compression)



- Simplified interface: register *compression* variants
- Active monitoring: dynamic resource mapping on available devices (CPU, GPU, DPU)
- Region-based compression: more compressibility, higher throughput

Runway analysis framework in PDC

- Runway analysis registration relies on the data movement infrastructure consisting of Clients + Servers + Mercury RPC + ***function registration APIs*** provided by Proactive Data Containers (PDC)

Analysis Registration:

```
PDCobj_analysis_register("user-defined-analysis-function", input1_iter,
result1_iter);
```

Transform Registration:

```
PDCregion_transform_register("pdc_transform_compress", &x[0], region_x,
obj_xx, region_xx, 0, INCR_STATE, DATA_OUT);
```

Experimental setup – Platforms and workloads

System Configuration

	Testbed Two-node system	NERSC Perlmutter Large scale cluster
CPU	2x Intel Xeon ES-2530 v4, 10-Core	AMD EPYC 7763 64-Core
RAM	126GB DDR4	256GB DDR4
GPU	NVIDIA A30 PCIe 24GB	NVIDIA A100 SXM4 40GB
DPU	NVIDIA Bluefield-2	-
OS	Ubuntu 18.04.5	SUSE Linux 15
Drivers	NVIDIA Driver 515.48.07, CUDA 11.7	

Datasets from SDRBench

Dataset	Data Objects	Entropy	Dimensions	Mem. Req.
Nyx	Temperature	23.99	512x512x512 Single Precision	512 MB
Hurricane ISABEL	QCLOUD	1.30	100x500x500 Single Precision	100 MB
	QRAIN	21.45		
	QVAPOR	24.19		
QMCPACK	QMCPack (einspline)	26.08	115x69x69x28 Single Precision	612 MB
S3D	Pressure	26.77	500x500x500 Double Precision	1 GB
Miranda	Density	22.5	96 regions of 3072x3072x3072 Single Precision	106 GB

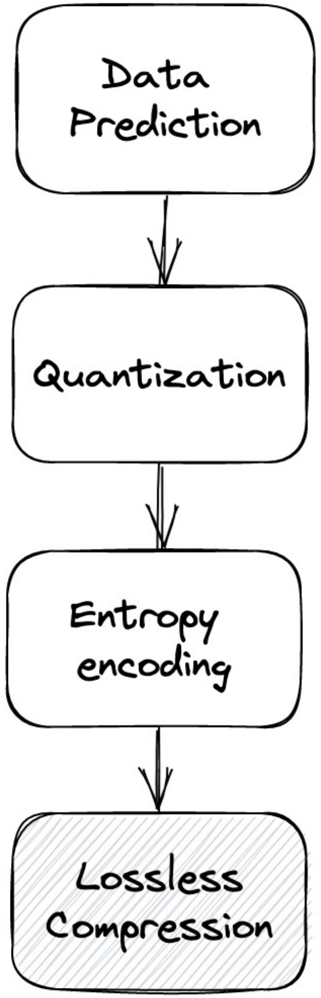
- SDRBench — scientific data reduction benchmark from authors of SZ
  - Includes data for visualization and application checkpoints
  - We developed a proxy write benchmark for each dataset

Data compression stages

- Lossless compressors
  - Zlib (LZ + Huffman)
  - Zstd (LZ + FSE)

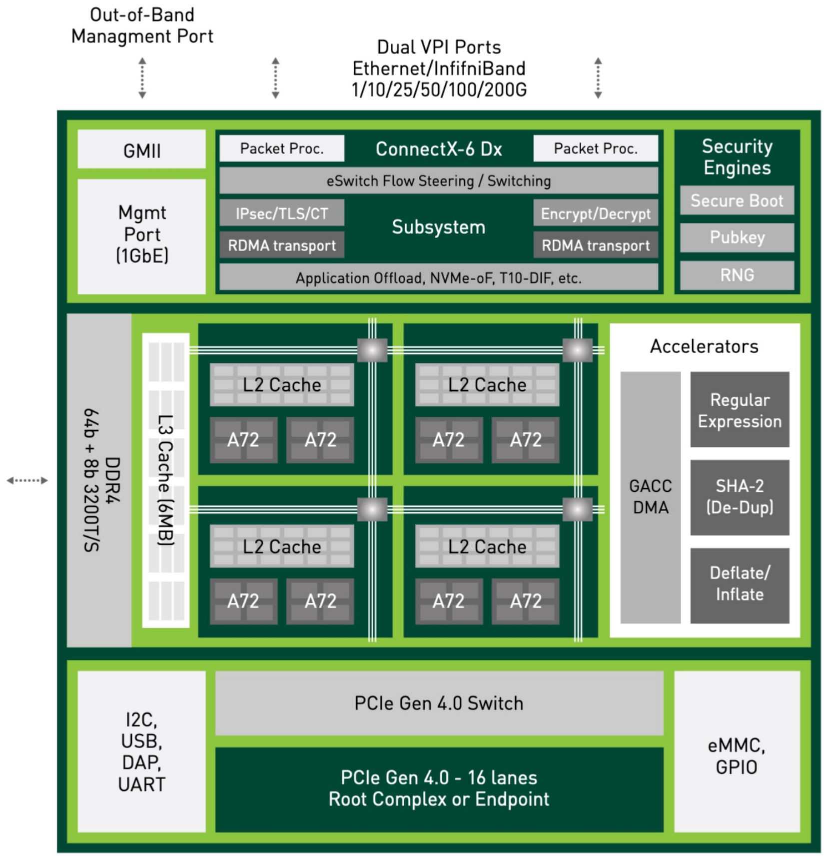
- Lossy compression methods
  - ZFP—fixed-rate, fixed-precision
  - MGARD—MultiGrid Adaptive Reduction of Data
  - SZ—Modular Error-bounded Lossy Compression Framework

SZ algorithm



Experimental Setup - DPU

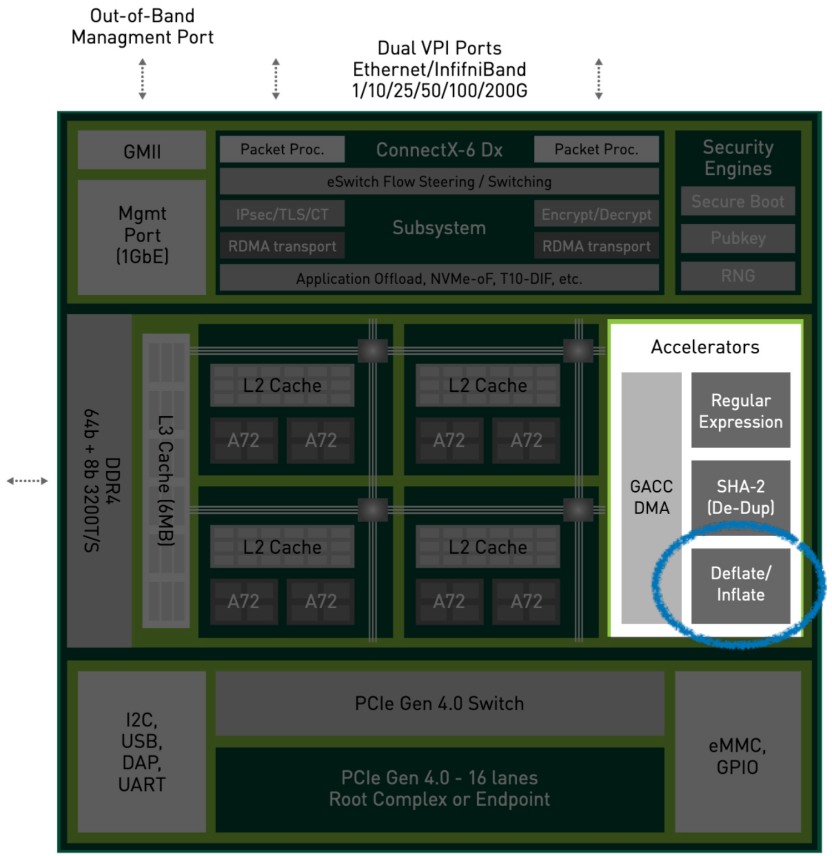
- NVIDIA Bluefield-2
  - NIC accelerator— 2x25Gbps
  - Arm Cortex A72, 16GB DDR4
  - OS—Ubuntu 20.04
  - API—NVIDIA DOCA v1.5.1
  - Legacy API—DPDK (Intel Data Plane Development Kit)



https://network.nvidia.com/files/doc-2020/pb-bluefield-2-dpu.pdf

Experimental Setup - DPU

- NVIDIA Bluefield-2
  - NIC accelerator— 2x25Gbps
  - Arm Cortex A72, 16GB DDR4
  - OS—Ubuntu 20.04
  - API—NVIDIA DOCA v1.5.1
  - Legacy API—DPDK (Intel Data Plane Development Kit)
    - Uses DEFLATE accelerator on DPU (LZ77 + Huffman coding)

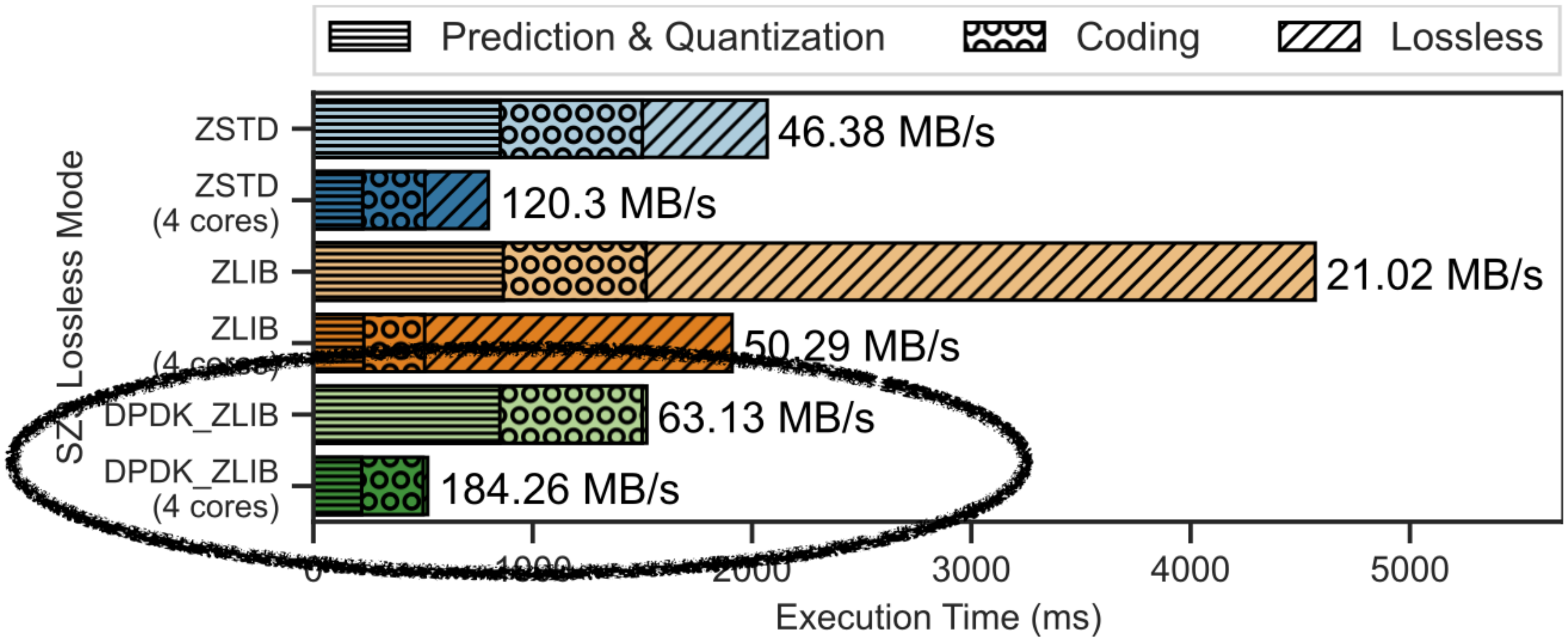


https://network.nvidia.com/files/doc-2020/pb-bluefield-2-dpu.pdf

Research questions

- How would accelerators benefit data analysis or transforms?
  - While data is moving between memory and storage
- Can we predict data transform (compression) cost on CPUs and GPUs to design a scheduler?
- Is non-uniform compression on different regions of the data beneficial?

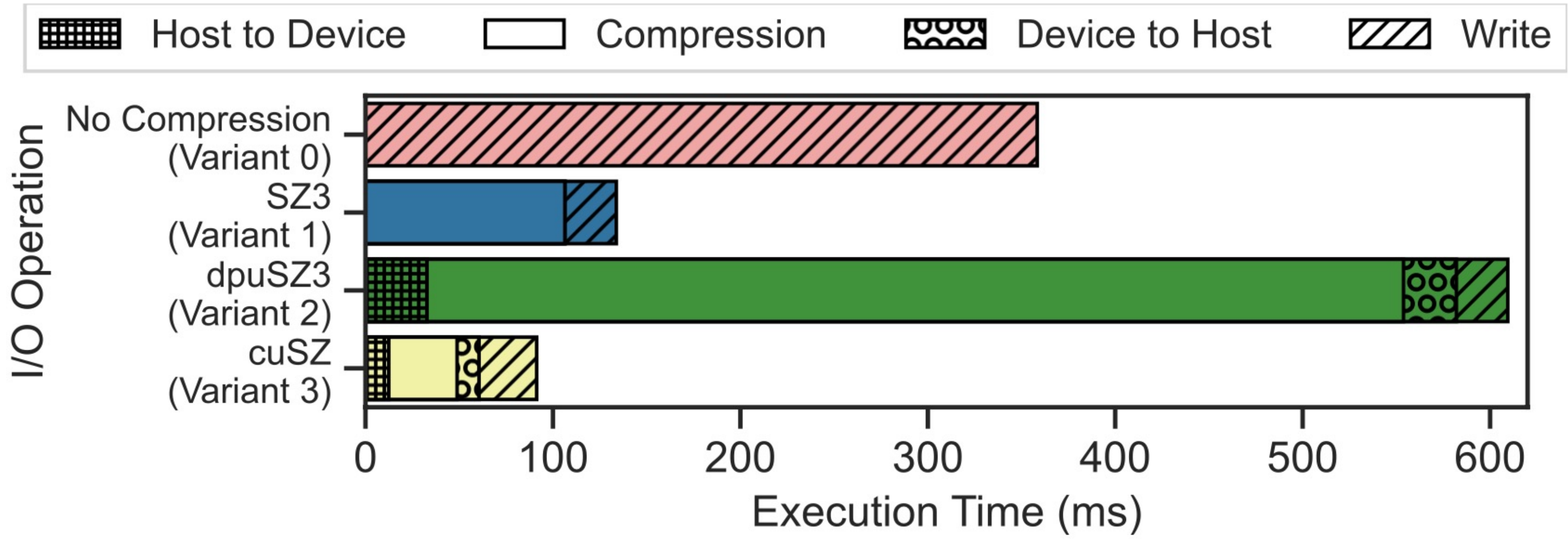
Compression performance with DPU



Hurricane ISABEL, QVAPOR data object

- DPDK_ZLIB utilizing the DPU DEFLATE Accelerator is 27X faster than ZLIB

Data Compression on different devices – CPU, GPU, and DPU



Hurricane ISABEL, QVAPOR data object

- Comparison of different variants of SZ—GPU is fastest

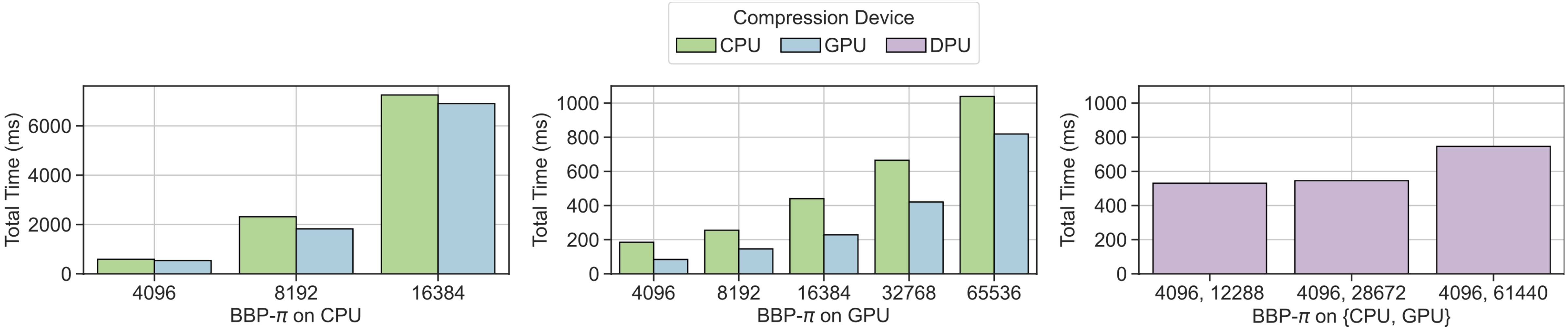
Emulated computation to keep CPU and GPU busy – BBP-π

$$\{16^n\pi\} = \{4\{16^n S_1\} - 2\{16^n S_4\} - \{16^n S_5\} - \{16^n S_6\}\}$$

$$\{16^n S_j\} = \{\{\sum_{k=0}^{n} \frac{16^{n-k} mod 8k + j}{8k + j}\} + \sum_{k=n+1}^{n+100} \frac{16^{n-k}}{8k + j}\}$$

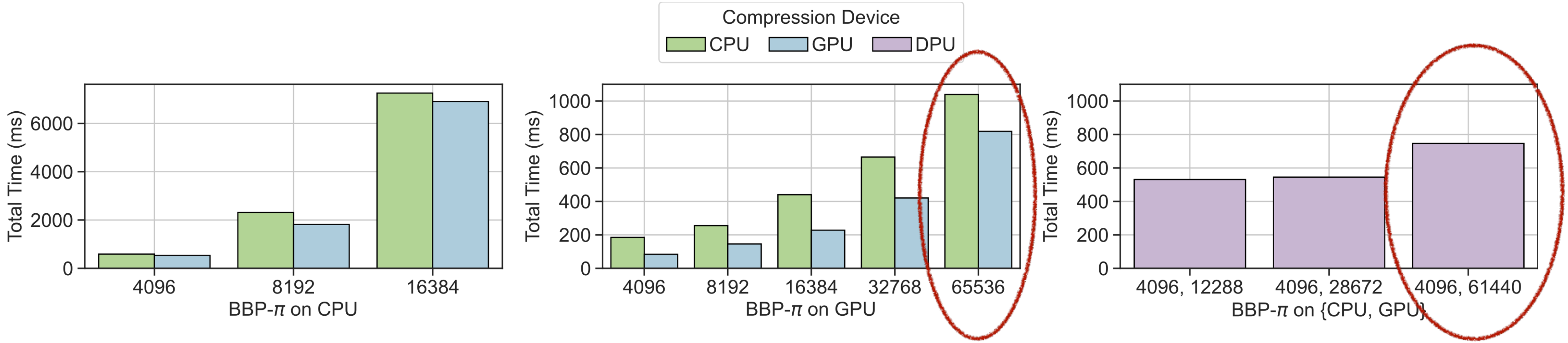
- Bailey-Borwein-Plouffe algorithm for calculating π
- Calculate the n-th hexadecimal digit of π without calculating the first n – 1 digits
- Scales linearithmically, O(n log n)
- Parallel implementation (OpenMP, CUDA)—each thread computes a digit

In-transit Data Compression - Static resource mapping



- Co-running *BBP-π* and *QVAPOR-IO* kernel
  - Green: *QVAPOR-IO* runs on CPU
  - Blue: *QVAPOR-IO* runs on GPU
  - Purple: *QVAPOR-IO* runs on DPU

In-transit data compression – Static resource mapping



- Co-running *BBP-π* and *QVAPOR-IO* kernel
  - Green: *QVAPOR-IO* runs on CPU
  - Blue: *QVAPOR-IO* runs on GPU
  - Purple: *QVAPOR-IO* runs on DPU

Hurricane ISABEL, QVAPOR data object

Research questions

- How would accelerators benefit data analysis or transforms?
  - While data is moving between memory and storage
- Can we predict data transform (compression) cost on CPUs and GPUs to design a scheduler?
- Is non-uniform compression on different regions of the data beneficial?

In-transit analysis cost prediction

- Goal—based on previous runs, predict analysis (compression) time

$$t_{latency} = t_{h2d\_time} + t_{compute} + t_{d2h\_time}$$

$$f_{est\_compute} \implies y_i = \beta_0 * x_{i,0}^3 + \beta_1 * x_{i,1}^2 + \beta_2 * x_{i,1} + \beta_3$$

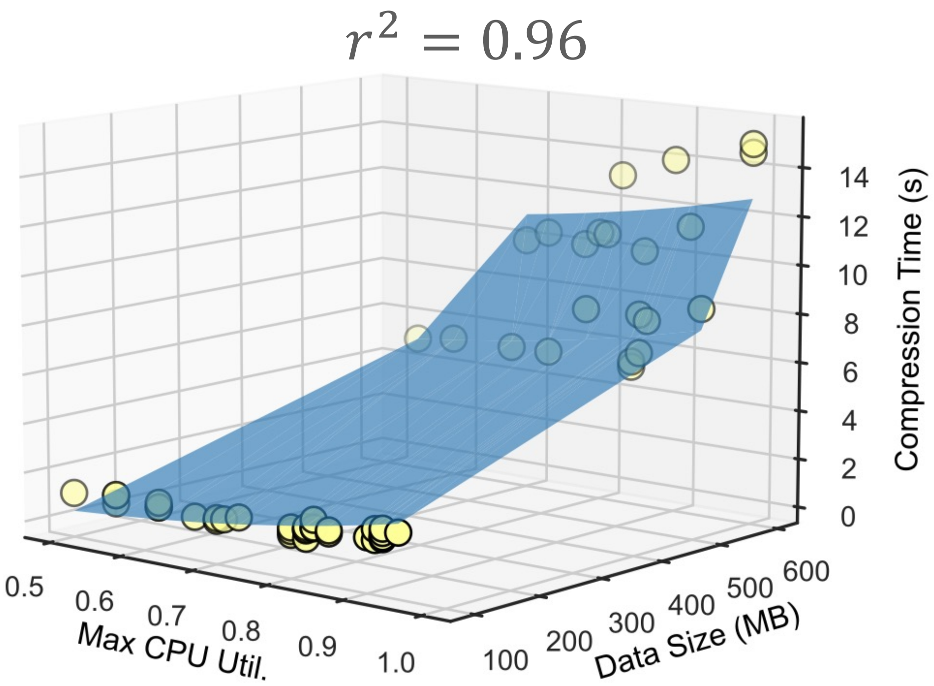
Measured compute time

Device Utilization

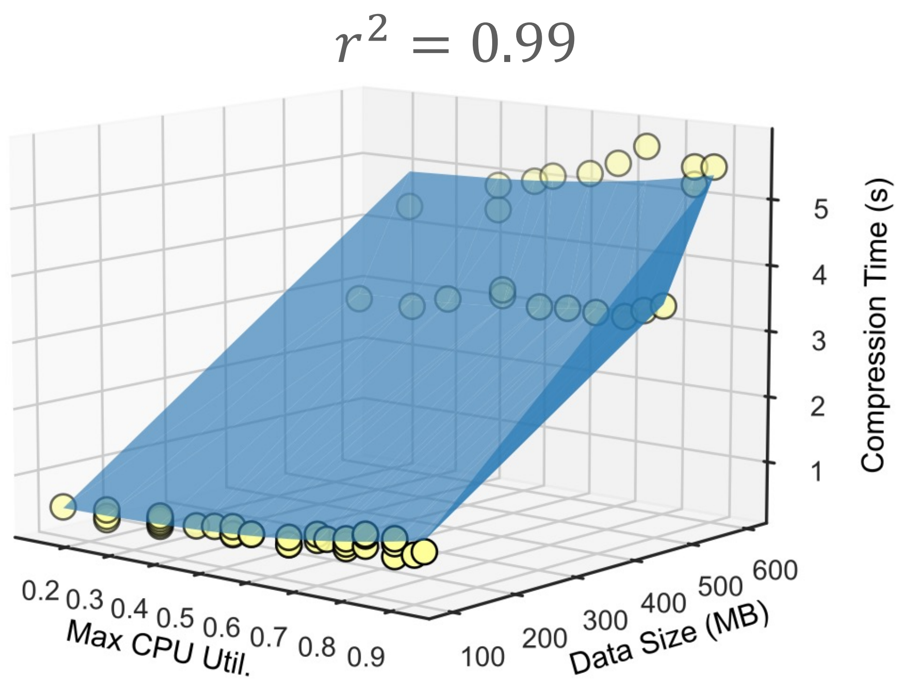
Data Size

- Polynomial regression solved using non-linear least squares

Modeling compression time on CPU

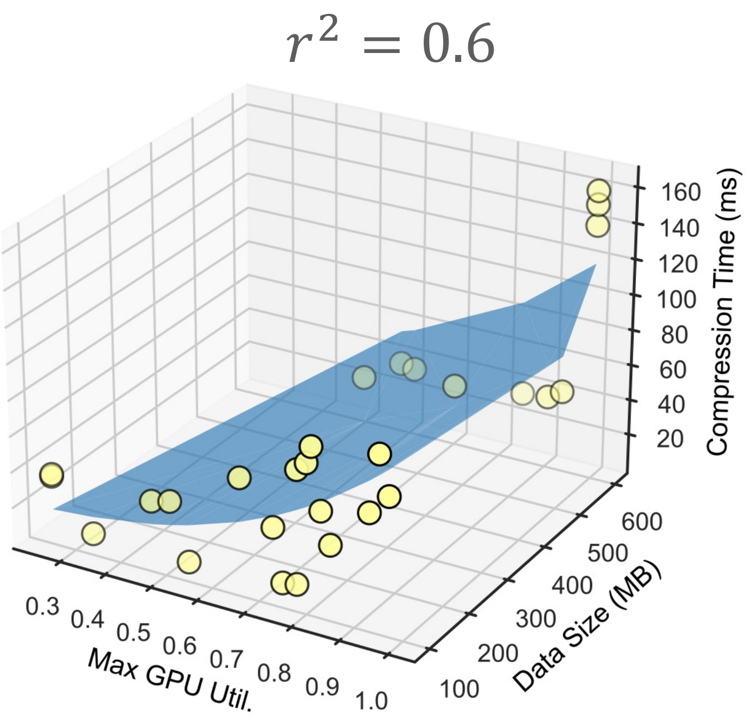


CPU (Intel Xeon) on testbed

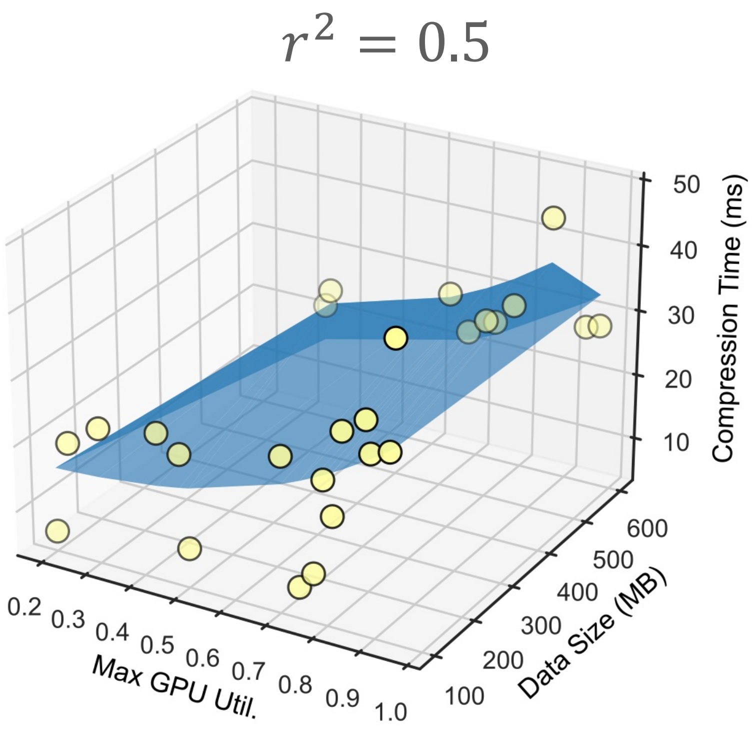


CPU (AMD Epyc) on Perlmutter

Estimating compression cost on GPU

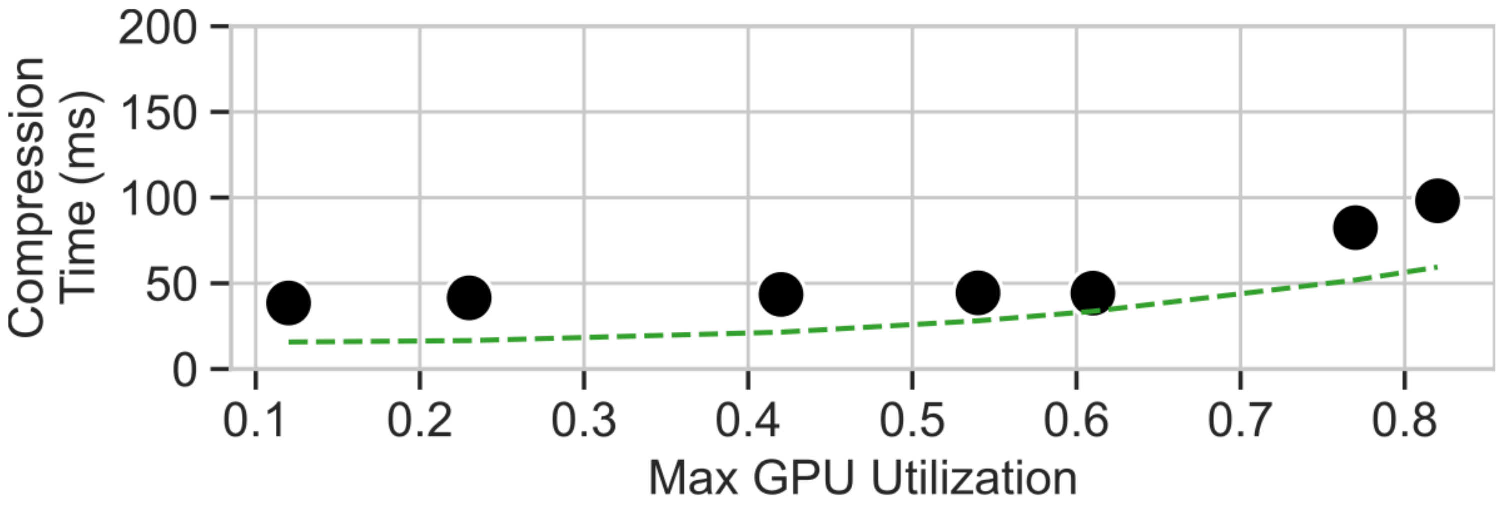


GPU (NVIDIA A30) on testbed



GPU (NVIDIA A100) on Perlmutter

Compression cost on GPU with varying utilization



- Co-run *BBP-π* and *QVAPOR-IO* kernel both sharing the same resource (NVIDIA A30 GPU).
- We vary the BBP-π duration to vary the GPU utilization (X-axis).
- Compression cost increases with increasing GPU utilization (Y-axis)
- The estimated time using prediction is shown as a dotted line.

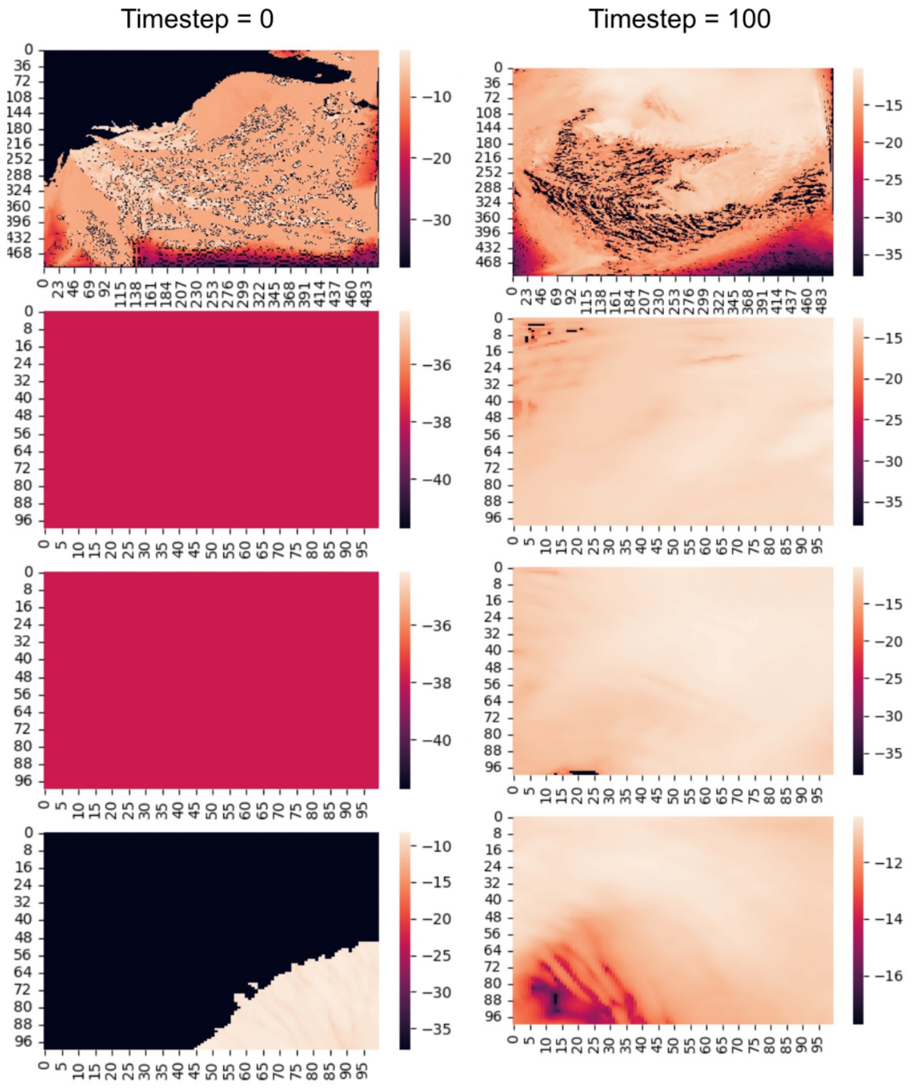
Research questions

- How would accelerators benefit data analysis or transforms?
  - While data is moving between memory and storage
- Can we predict data transform (compression) cost on CPUs and GPUs to design a scheduler?
- Is non-uniform compression on different regions of the data beneficial?

Region-based data compression

- Perform compression at region level versus object level

- Motivation: Non-uniform compression parameters to improve performance on low entropy regions

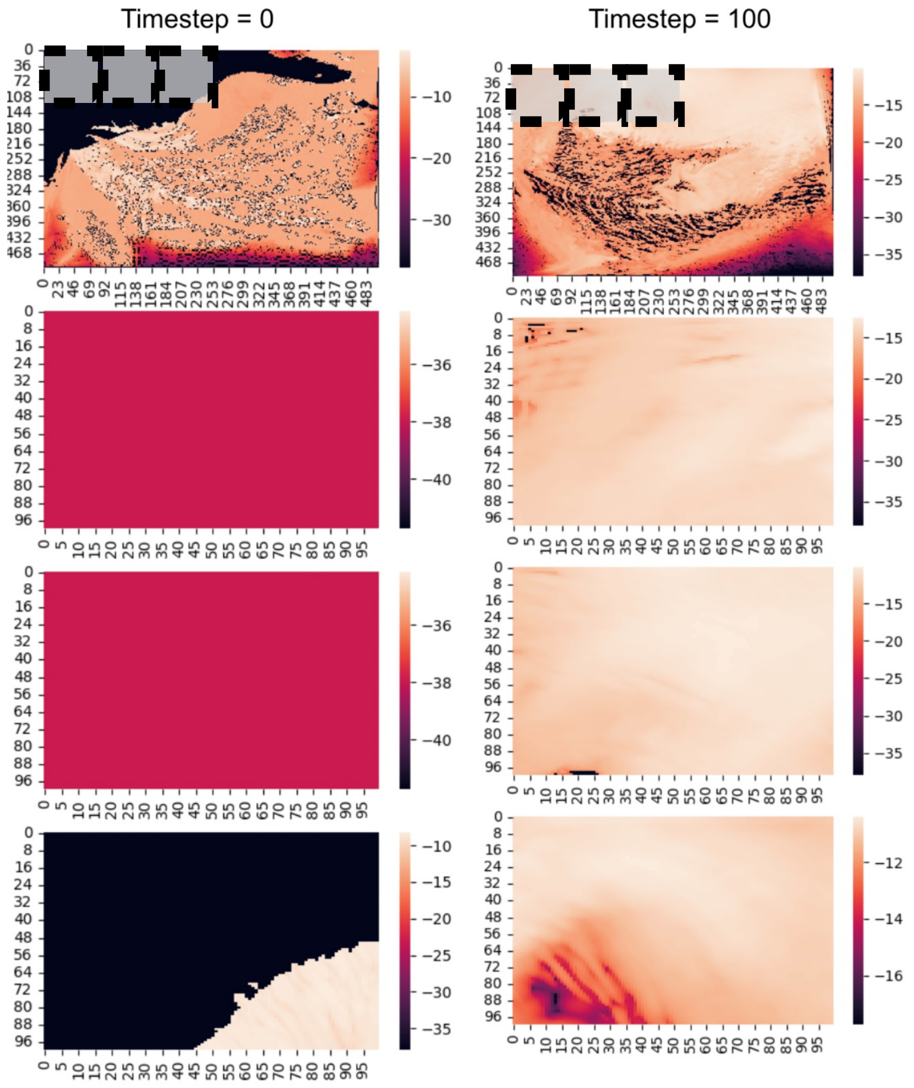


QRAIN from Hurricane ISABEL Dataset

Region-based data compression

- Perform compression at region level versus object level

- Motivation: Non-uniform compression parameters to improve performance on low entropy regions

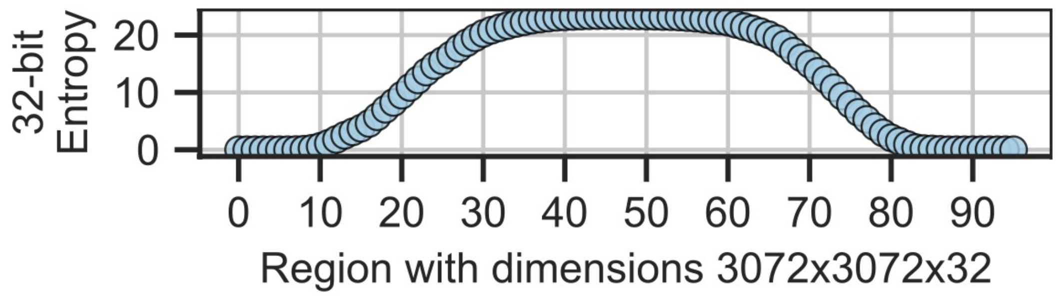


QRAIN from Hurricane ISABEL Dataset

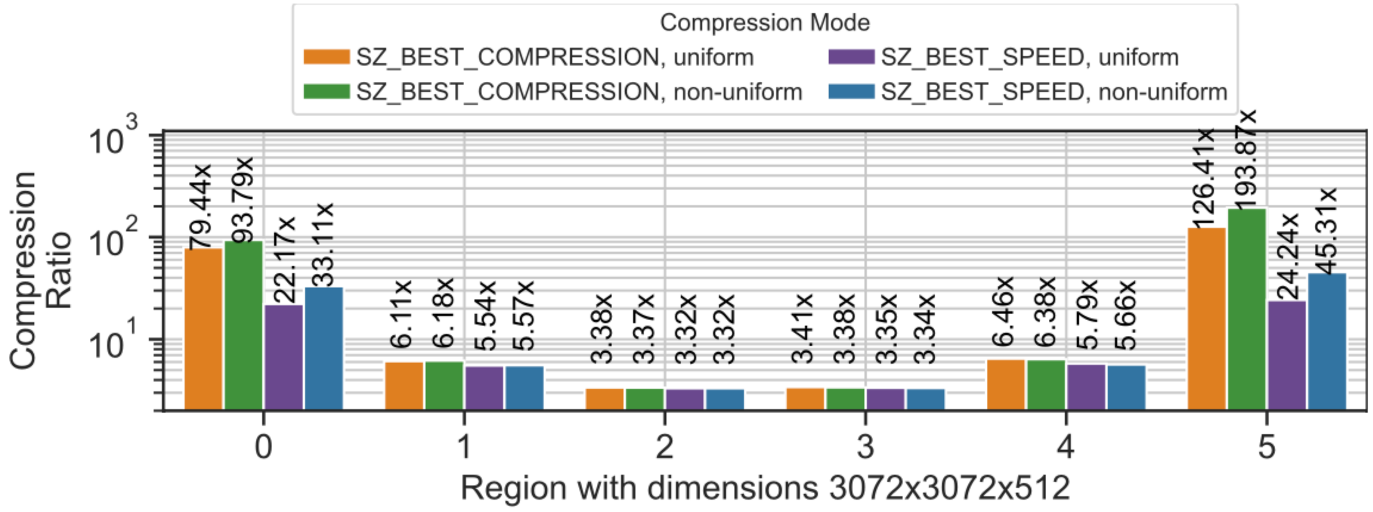
Region-based data compression – speed vs. compression ratio

- Each 3072x3072x32 region is ~1GB
- Some regions have 0 entropy

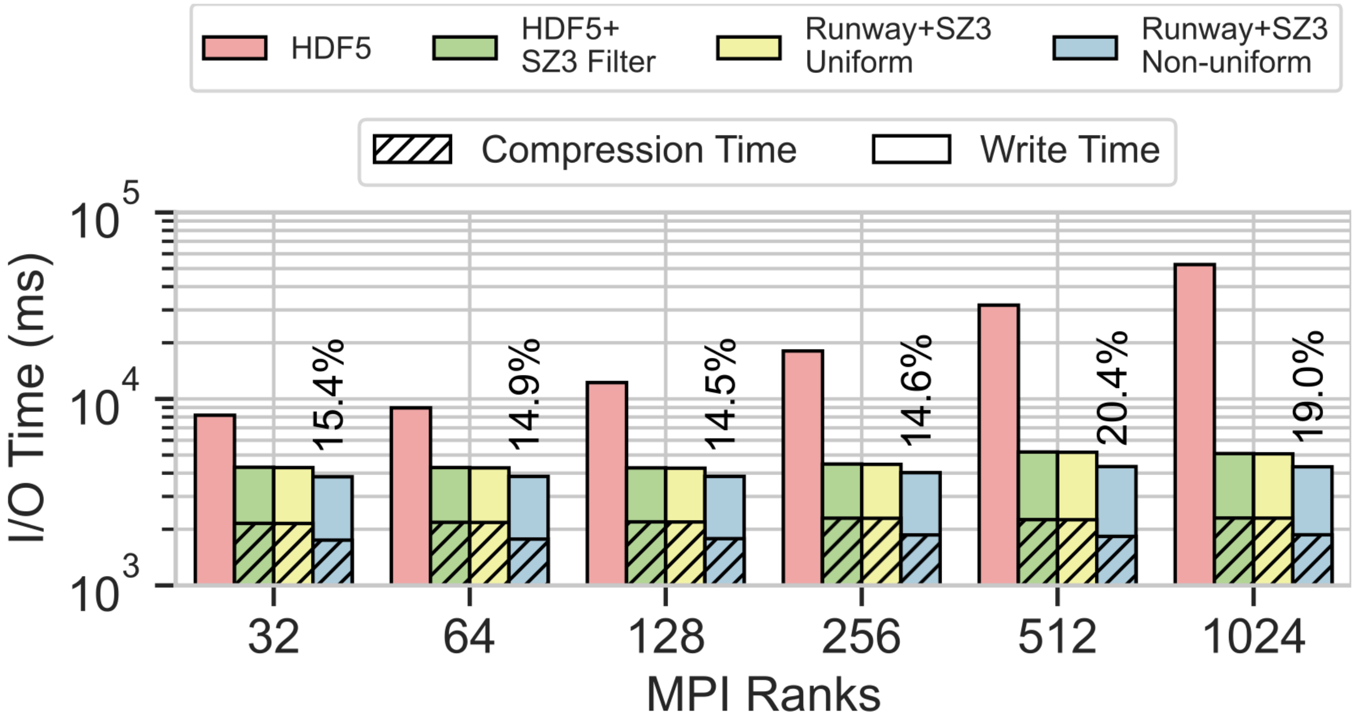
Density, Data Object with dimensions 3072x3072x3072
Miranda Dataset



- SZ_BEST_COMPRESSION- Lossless enabled
- SZ_BEST_SPEED— Lossless disabled



Region-based data compression at large scale



Pressure, Data Object from S3D dataset

Weak-scaling problem: Data size increases with the number of ranks

- Per-region compression is advantageous – between 15% and 20% for an S3D dataset

Conclusions

Thanks to:

- How would accelerators benefit data analysis or transforms?
  - For data compression, GPUs often provide good performance
  - When GPUs are busier than 75%, DPUs can help
- Can we predict data transform (compression) cost on CPUs and GPUs to design a scheduler?
  - Predicting compression cost on CPUs is accurate.
  - On GPUs, prediction model works well for large datasets when the compression cost is high
- Is non-uniform compression on different regions of the data beneficial?
  - Region-based compression accuracy is beneficial
- Future work
  - Offload overhead on future DPUs may be less

https://github.com/hpc-io/pdc

Contact: Suren Byna – https://sbyna.github.io