# CSE 5449: Intermediate Studies in Scientific Data Management

## Lecture 23: Proactive Data Containers – Data Management

Dr. Suren Byna

The Ohio State University

E-mail: byna.1@osu.edu

https://sbyna.github.io

04/18/2023

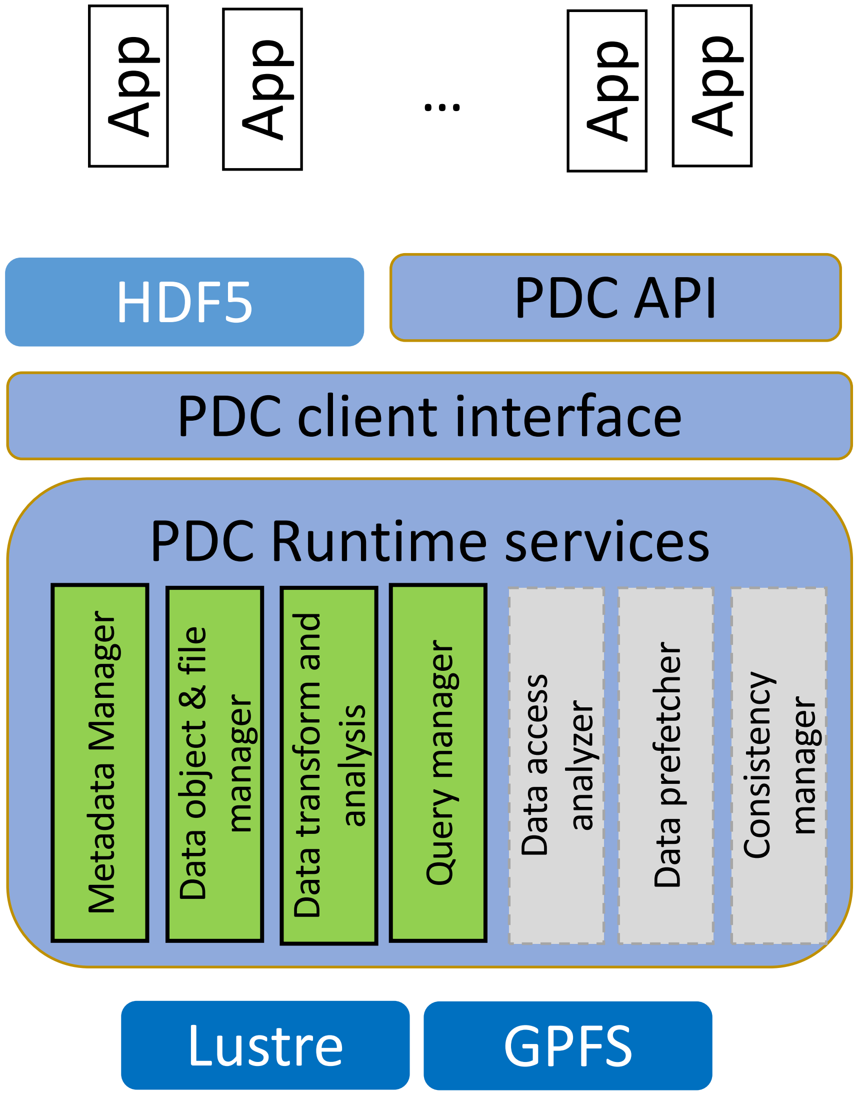
# Today's class

- Any questions?

- Class presentation topic

- Today's class –
  - Proactive Data Containers – Data management

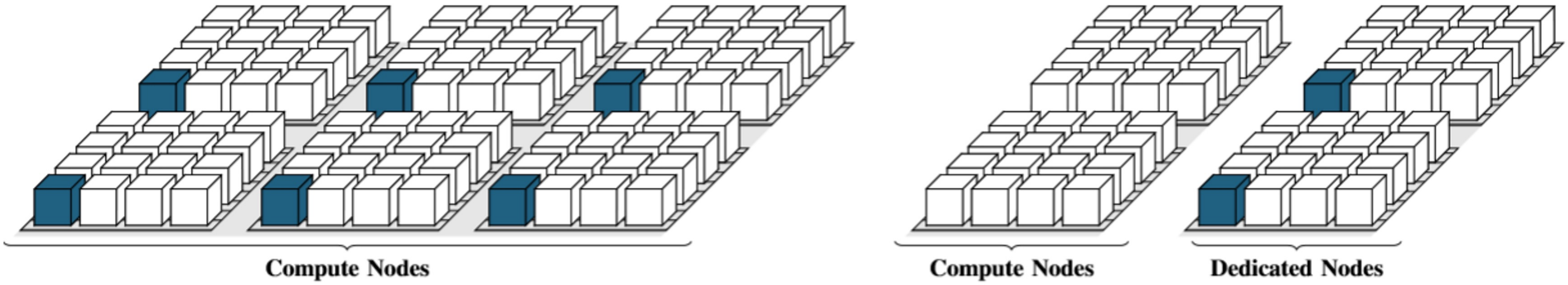
# Proactive Data Containers (PDC): An autonomous object-centric data management services framework




- Advantages of PDC
  - Application-level object abstractions - Freedom from file management
  - Transparent utilization of storage hierarchy and data movement
  - Superior and scalable performance
  - Live system for data management services
    - Metadata management, analysis, indexing and querying services, *consistency, data placement,* etc.

# Data management in PDC

- PDC servers run in background, manages data and metadata.
- Data objects can be stored on different layers of memory hierarchy.
- Large data objects are decomposed into smaller regions.
- Metadata is cached in server’s memory and persisted to storage.
- Application send requests through linked PDC client library.

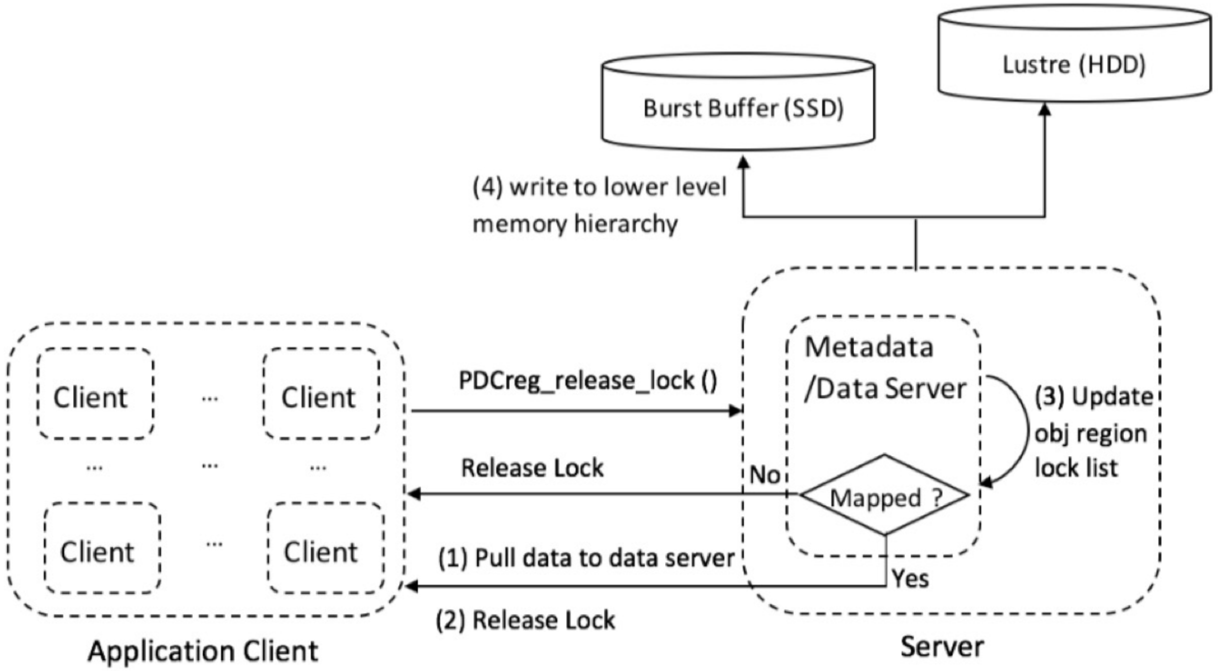



(a) Shared server modes: servers and clients are located on the same node.

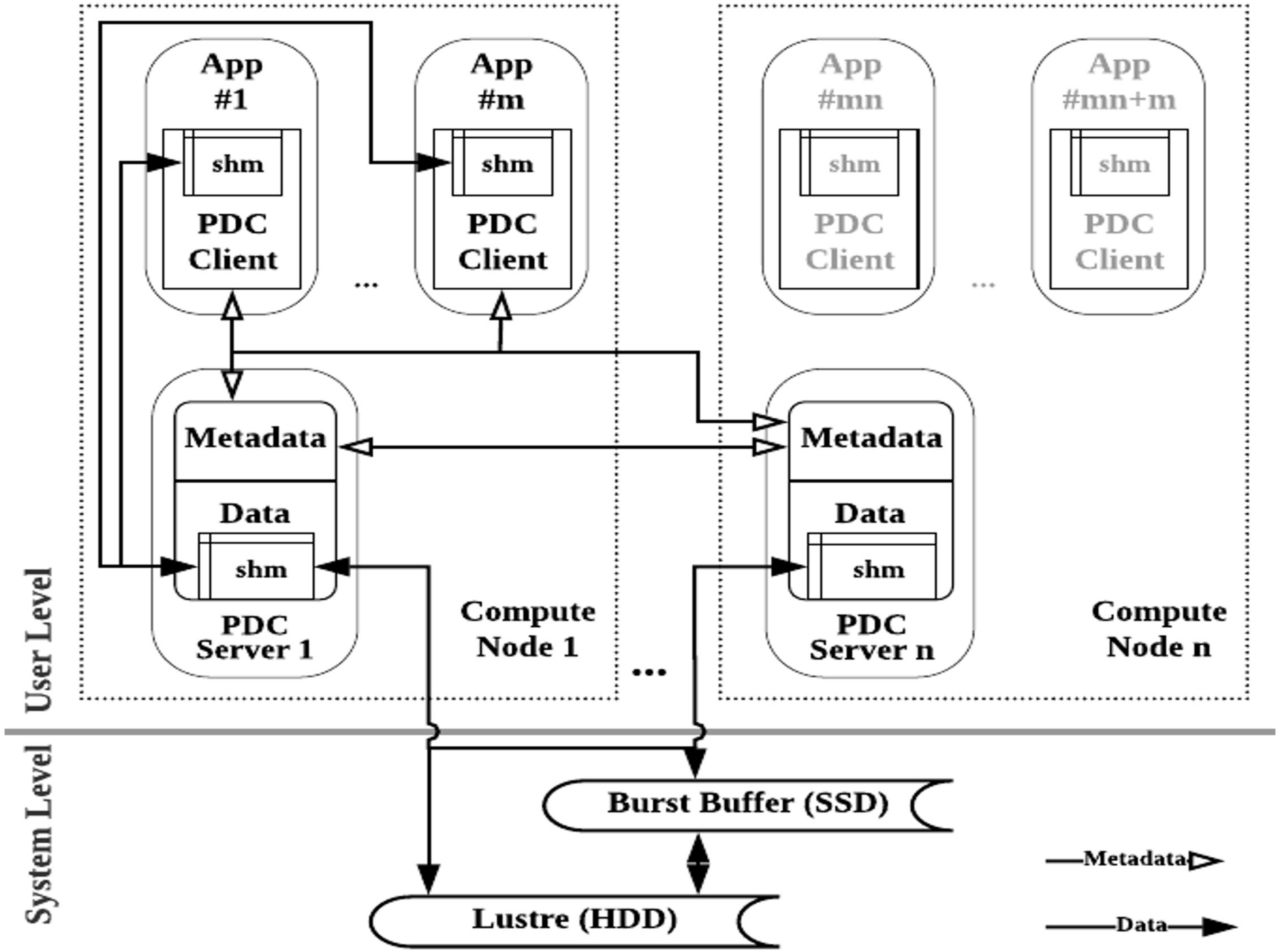
(b) Dedicated server modes: servers are on separate nodes.

# Data management and operations

- Data movement and I/O realized *asynchronously*
  - Transfers to deeper levels of the storage hierarchy are handled by PDC server
  - Overlap computation with I/O operations
- Application's buffers, when mapped, can only be used and modified once a lock is acquired

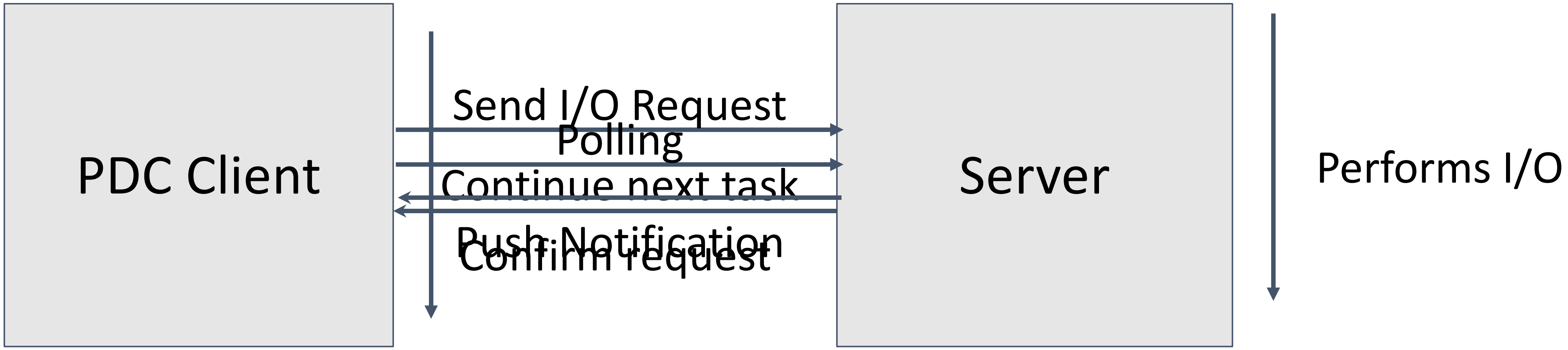


# PDC System



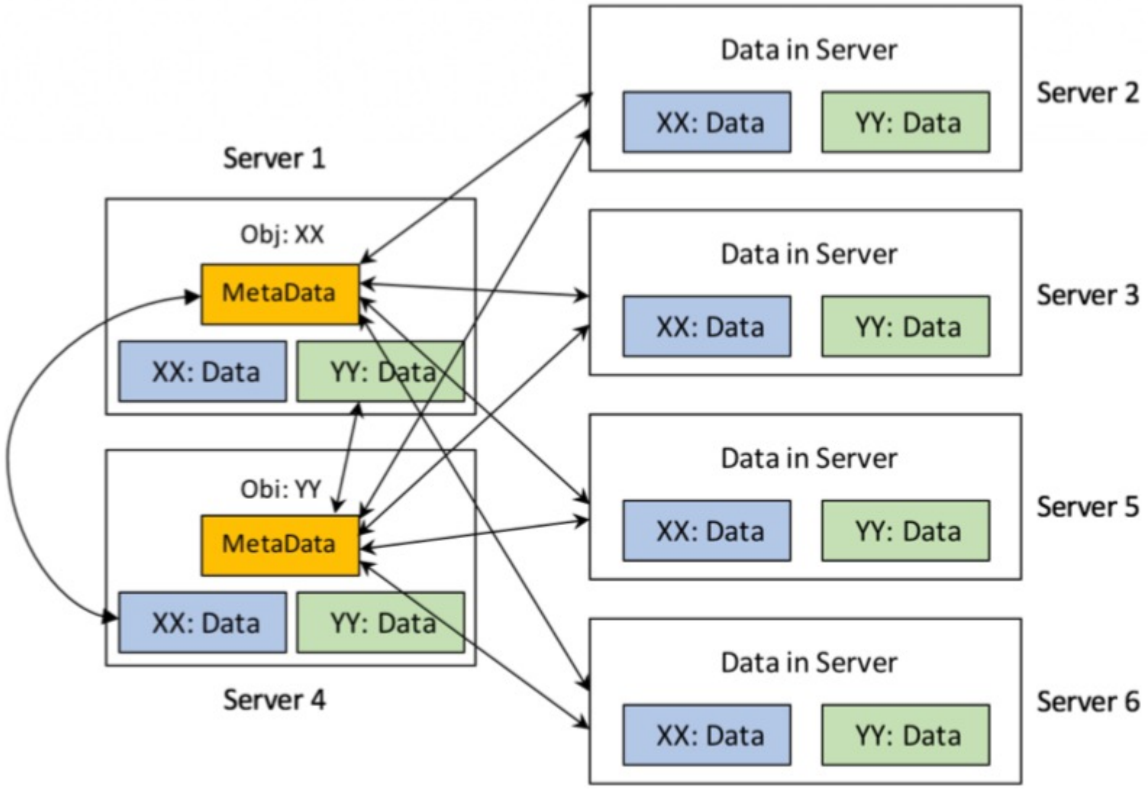
App #1
shm
PDC Client
...
App #m
shm
PDC Client
App #mn
shm
PDC Client
...
App #mn+m
shm
PDC Client
Metadata
Data
shm
PDC Server 1
Compute Node 1
...
Metadata
Data
shm
PDC Server n
Compute Node n
User Level
System Level
Burst Buffer (SSD)
Lustre (HDD)
Metadata
Data

# Asynchronous I/O



# User-space Client-Server Model

- PDC servers are responsible for executing both metadata and data management operations
- Metadata and data distribution in PDC Data Server
- PDC servers are multi-threaded
- Control and data transfers using RPC mechanism
  - Mercury (https://mercury-hpc.github.io )



# Storage Hierarchy-Aware Data Management

- **Memory**
  - Fastest.
  - Temporary and limited storage space.
- **Burst Buffer**
  - Fast.
  - Temporary and limited storage space.
- **Lustre**
  - Slower and requires expertise in performance tuning
  - Long term storage with enough storage space.

# Data Management Optimizations

- **Node-local data aggregation**
  - Each server aggregates I/O requests from node local clients.
  - Effective use of shared memory to transfer data.
  - Log-structured write
  - At the storage layer,
    - uses one file per region by default
    - can arrange multiple regions into a file
- **Automatic Lustre Tuning**
  - Automatically sets stripe count, size, OST index

# Metadata Optimizations

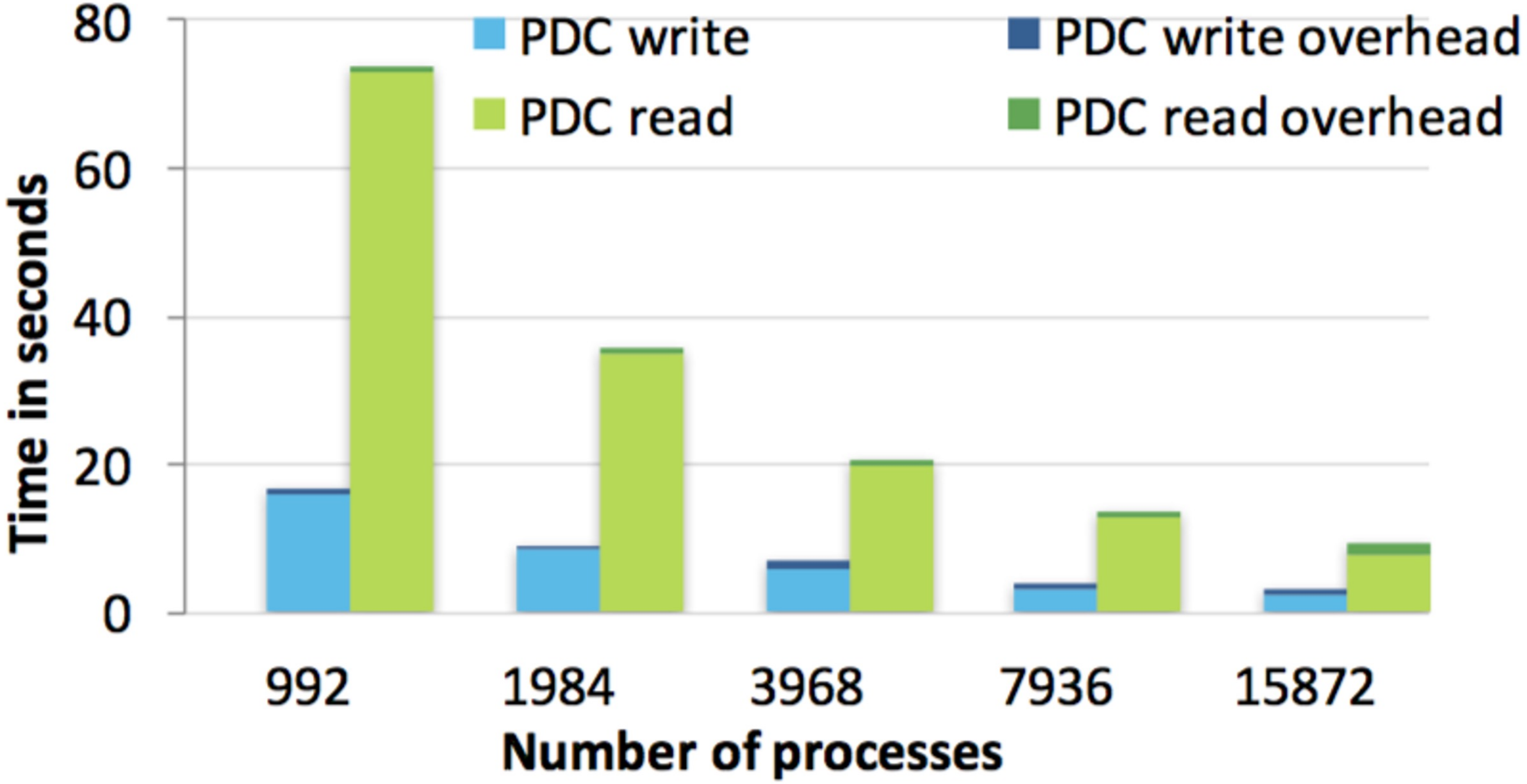
- **Collective Metadata querying.**
  - Aggregate the requests and retrieve corresponding metadata.
  - Reduce communication cost.

- **Relaxed metadata consistency.**
  - Delay some metadata updates and bundle with others.
  - Reduce communication cost.

# Experimental Setup

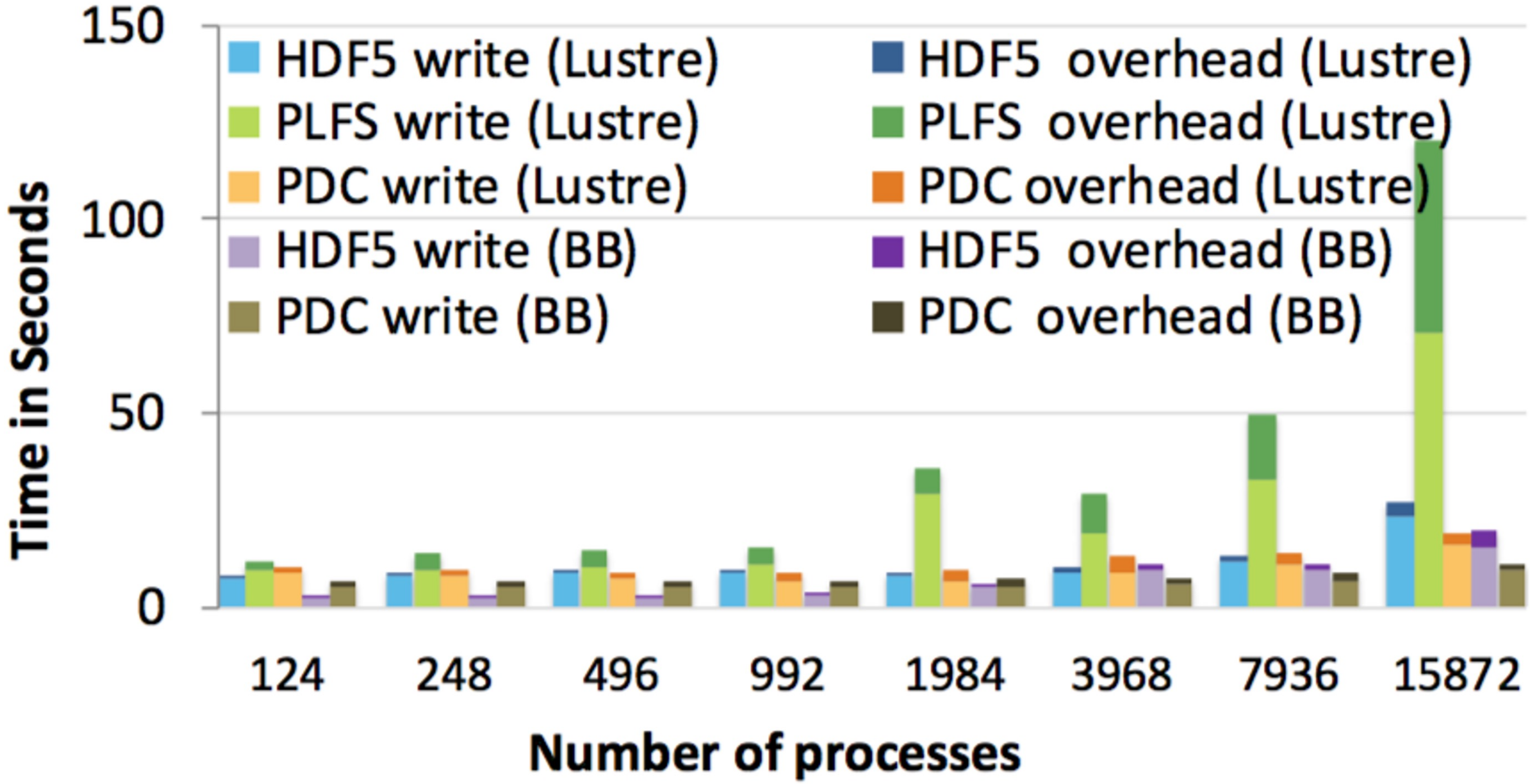
| | |
|---|---|
| HPC Systems | Cori (NERSC), Cooley (Argonne) |
| Comparison | PDC, HDF5, and PLFS |
| Workloads | Benchmarks<br>IO Kernels (VPIC-IO, BDCATS-IO) |
| Operations | Write, read with single and multiple time steps.<br>Strong and weak scaling |
| Storage | Main Memory<br>SSD-based Burst Buffer<br>Hard disk drive (Lustre and GPFS) |

# I/O Strong Scaling



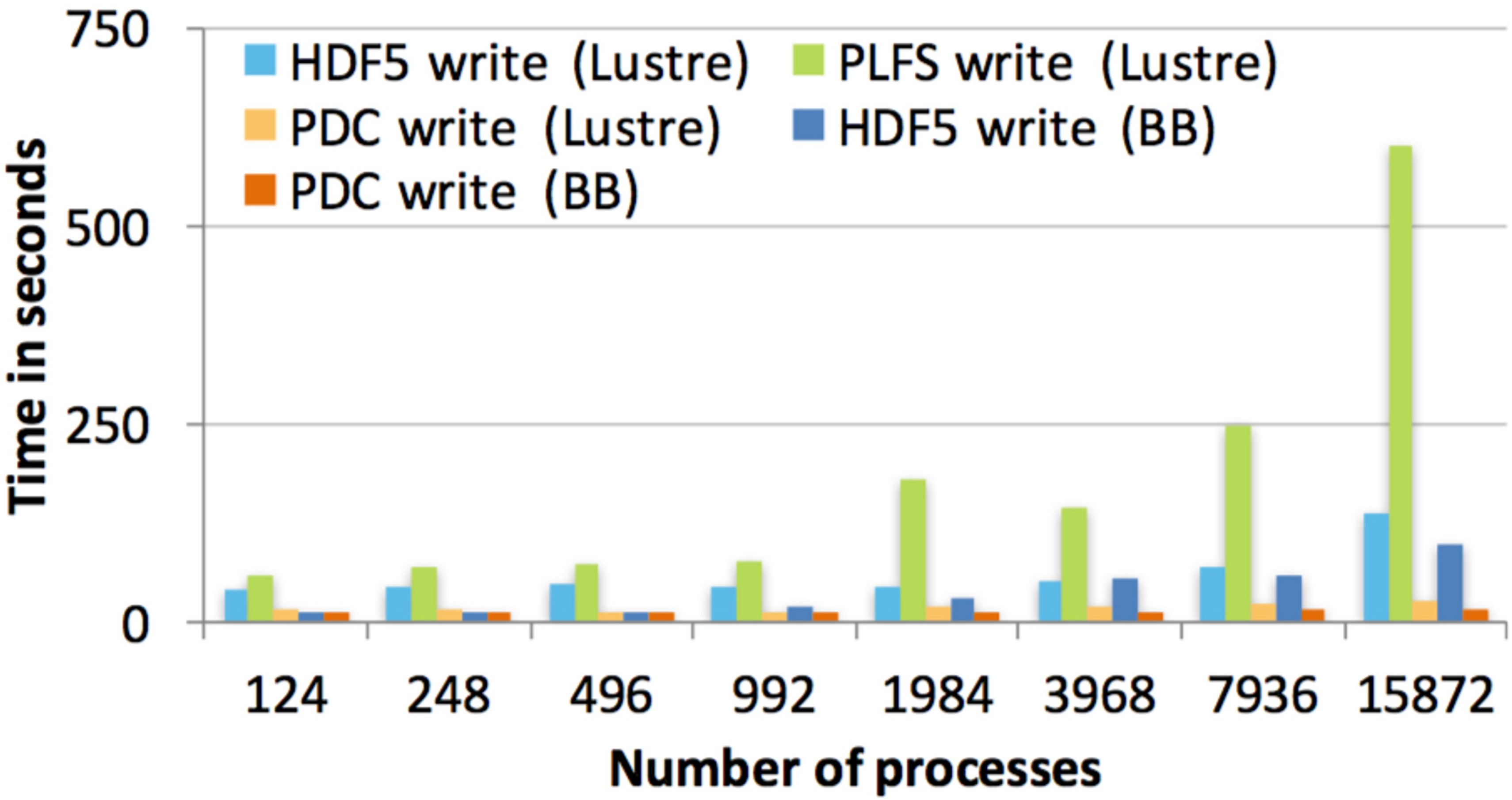

PDC strong scaling performance for writing and reading 512GB data on Lustre.

# VPIC-IO (Weak Scaling) Single-timestep Write



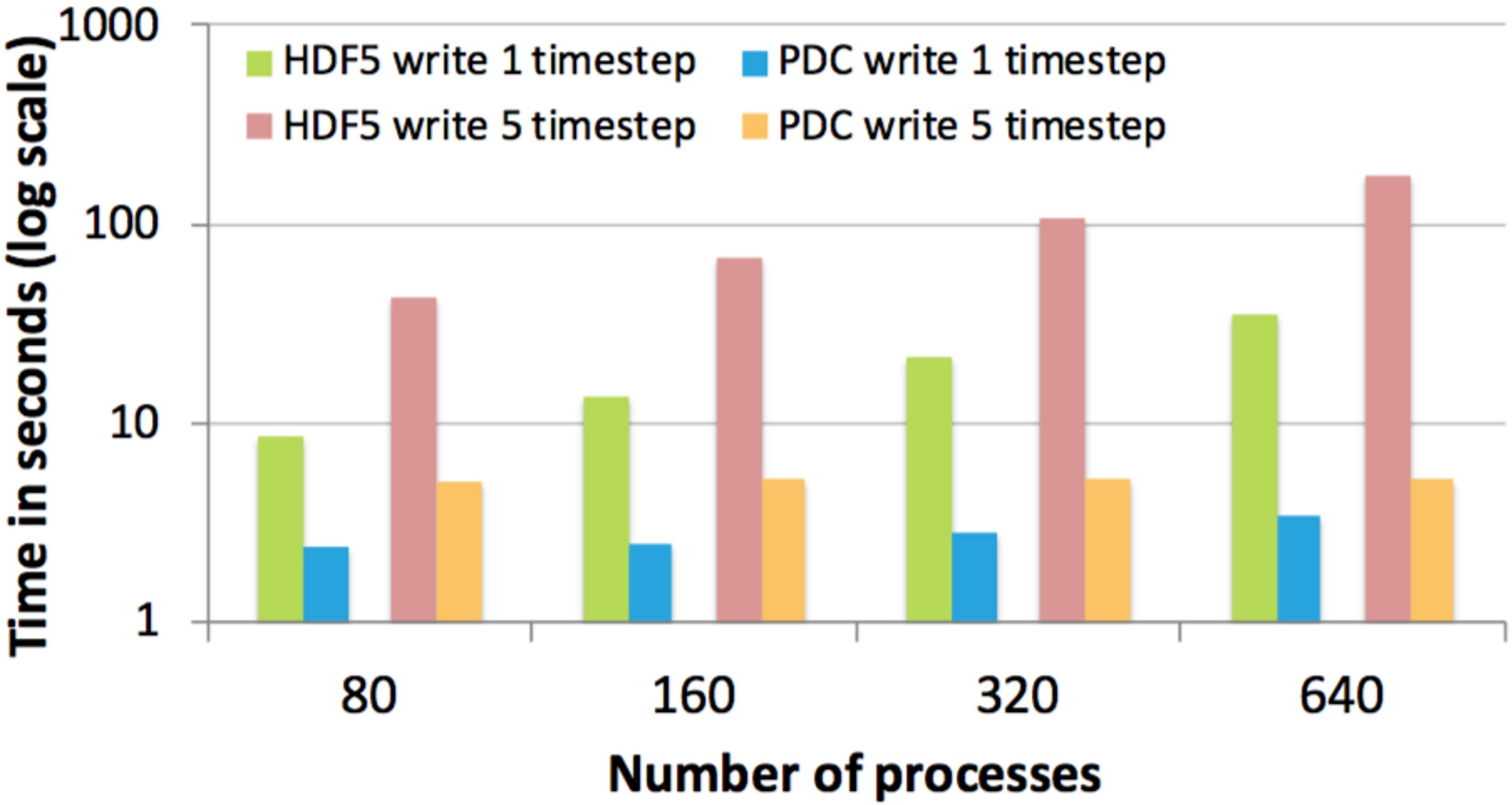

Total time for writing 1 timestep to Lustre and Burst Buffer using HDF5, PLFS, and PDC on Cori. PDC is up to **1.7x** faster than HDF5 and **9.2x** over PLFS

# VPIC-IO (Weak Scaling) Multi-timestep Write



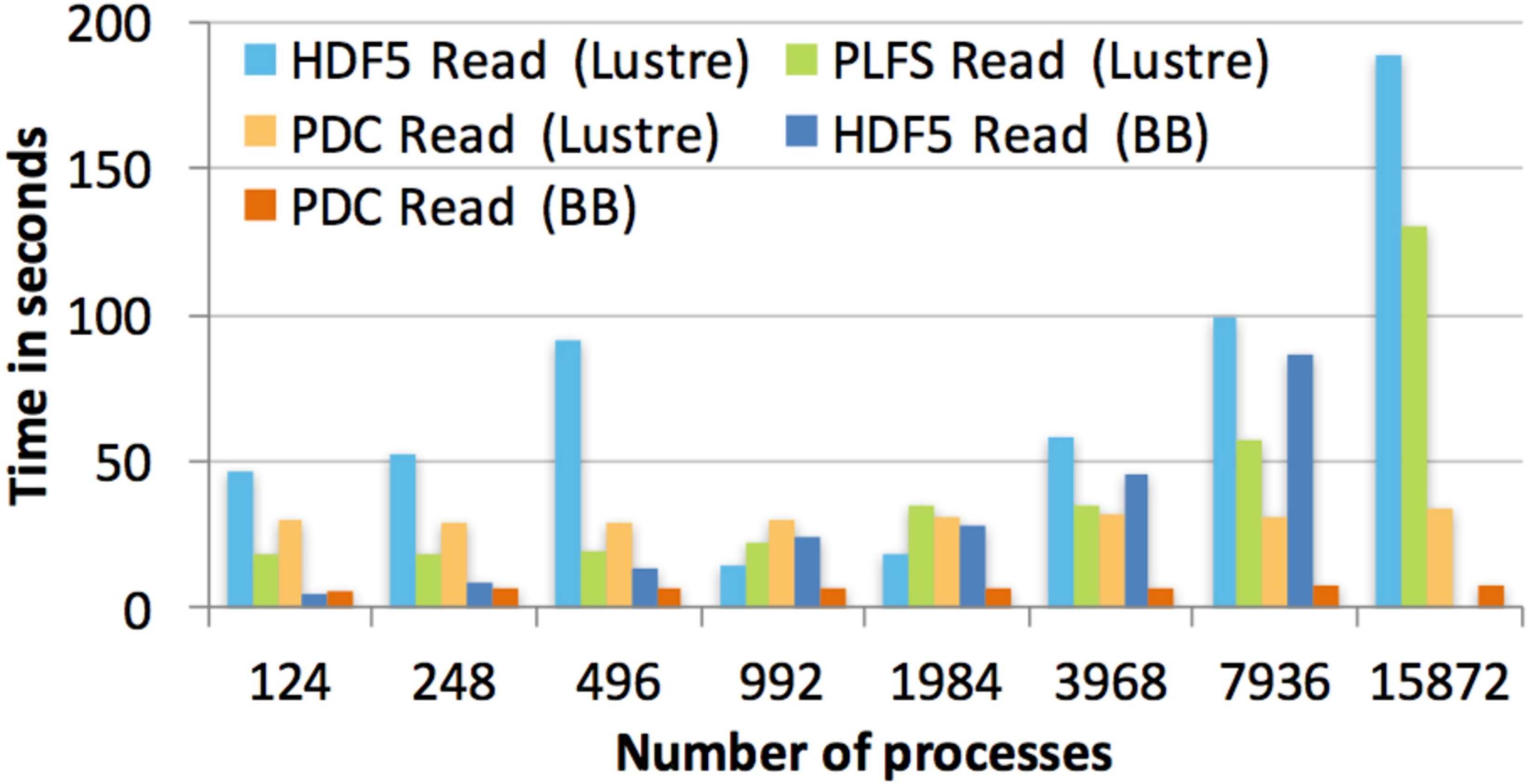

Total time to write 5 timesteps from the VPIC-IO kernel to Lustre and Burst Buffer on Cori. PDC is up to **5x** faster than HDF5 and **23x** over PLFS.

# VPIC-IO Write on Cooley



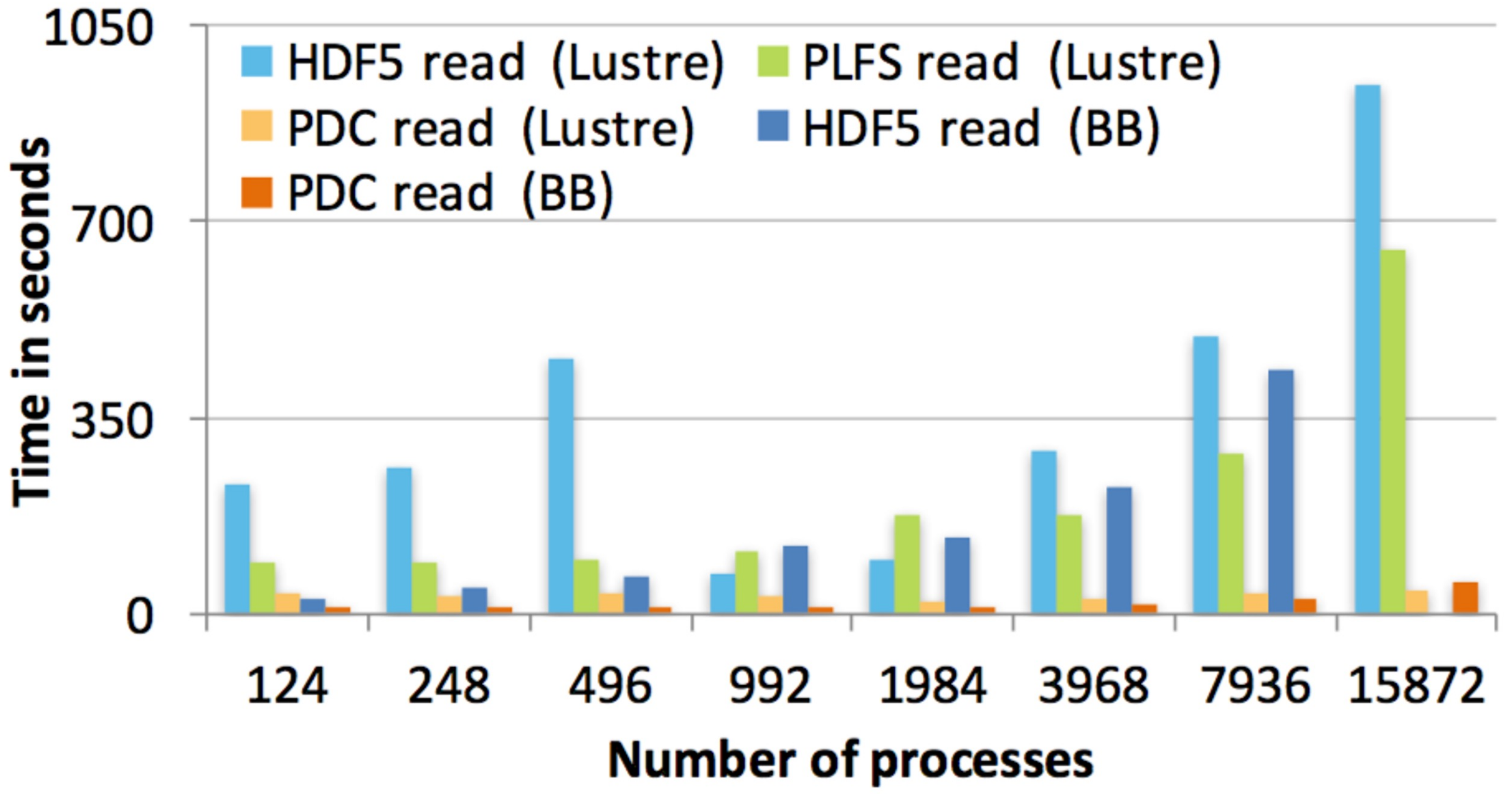

Total time to write 1 and 5 timesteps from the VPIC-IO kernel to the GPFS file system on Cooley. PDC is up to **7x** and **35x** than HDF5 to write 1 and 5 timesteps data.

# BD-CATS-IO (Weak scaling) Single-timestep Read



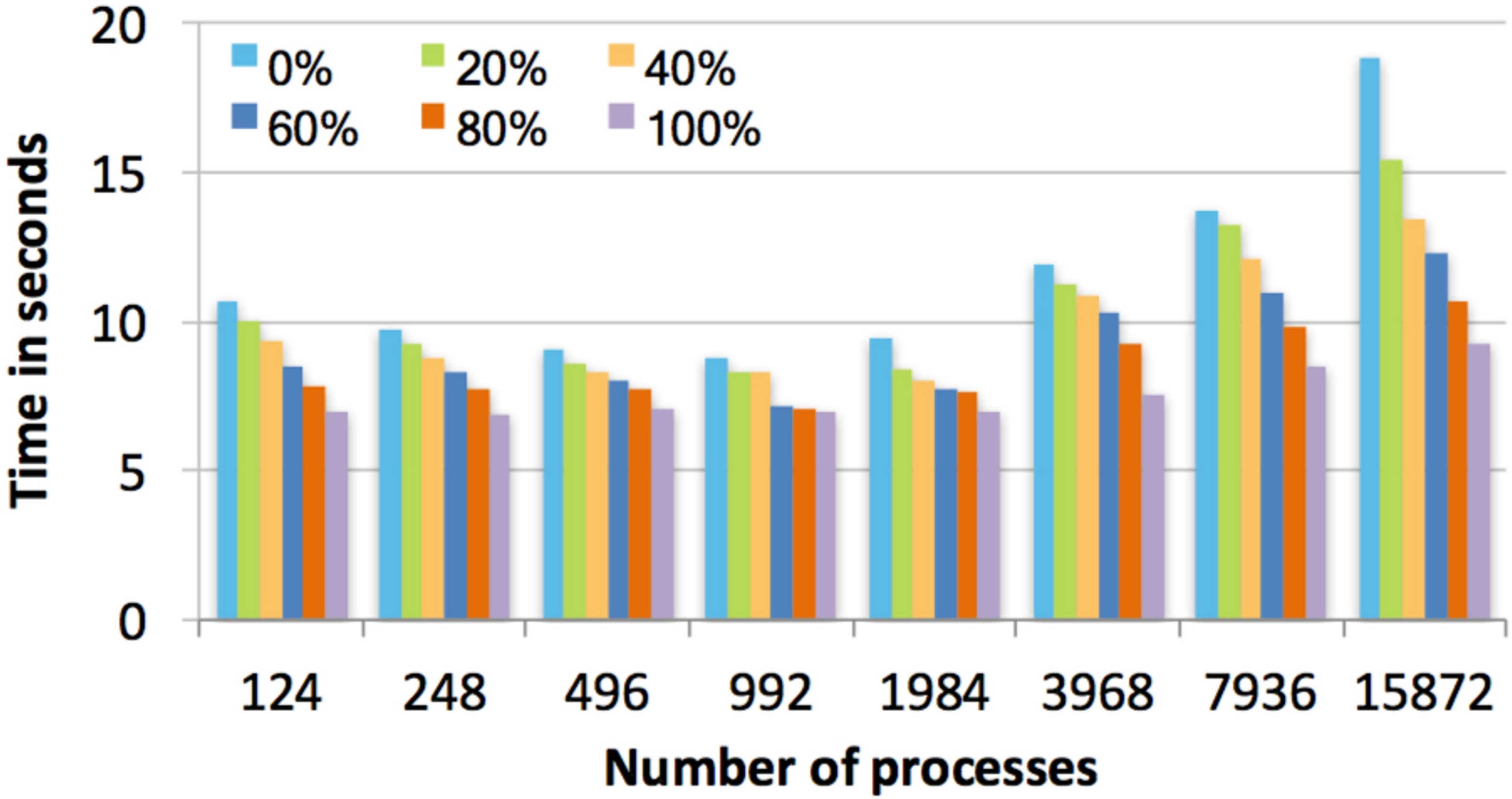

Total time for reading 1 timestep data using the BD-CATS-IO kernel using HDF5, PLFS, and PDC. PDC is up to **5x** and **4x** faster than HDF5 and PLFS.

# BD-CATS-IO (Weak scaling) Multi-timestep Read



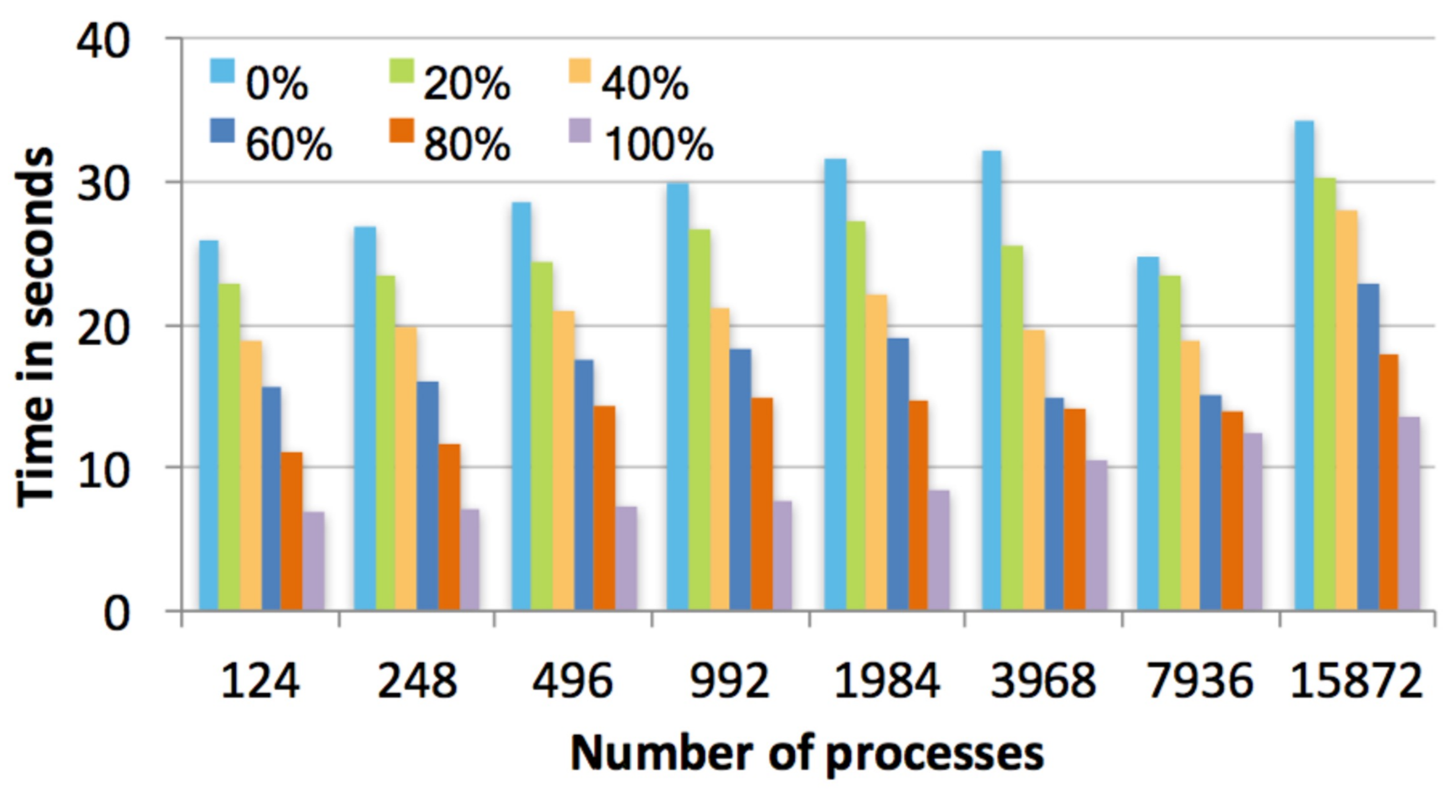

Total time for reading data of 5 timesteps from the BD-CATS-IO kernel from the Lustre and from the burst buffer. PDC is up to **11X** faster than PLFS and HDF5.

# Multi-level Storage Write



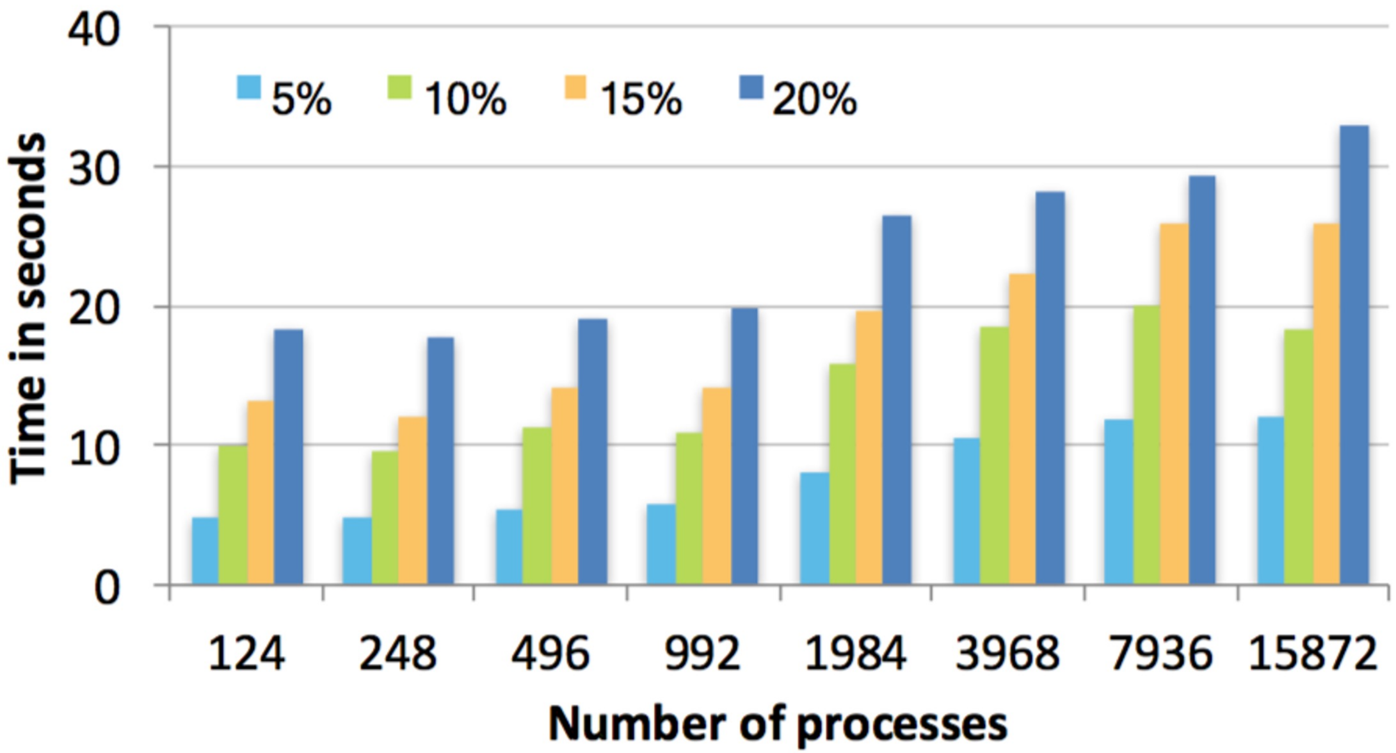

Write time with part of the data written to faster burst buffer and the remaining to slower Lustre file system on Cori.

# Multi-level Storage Read



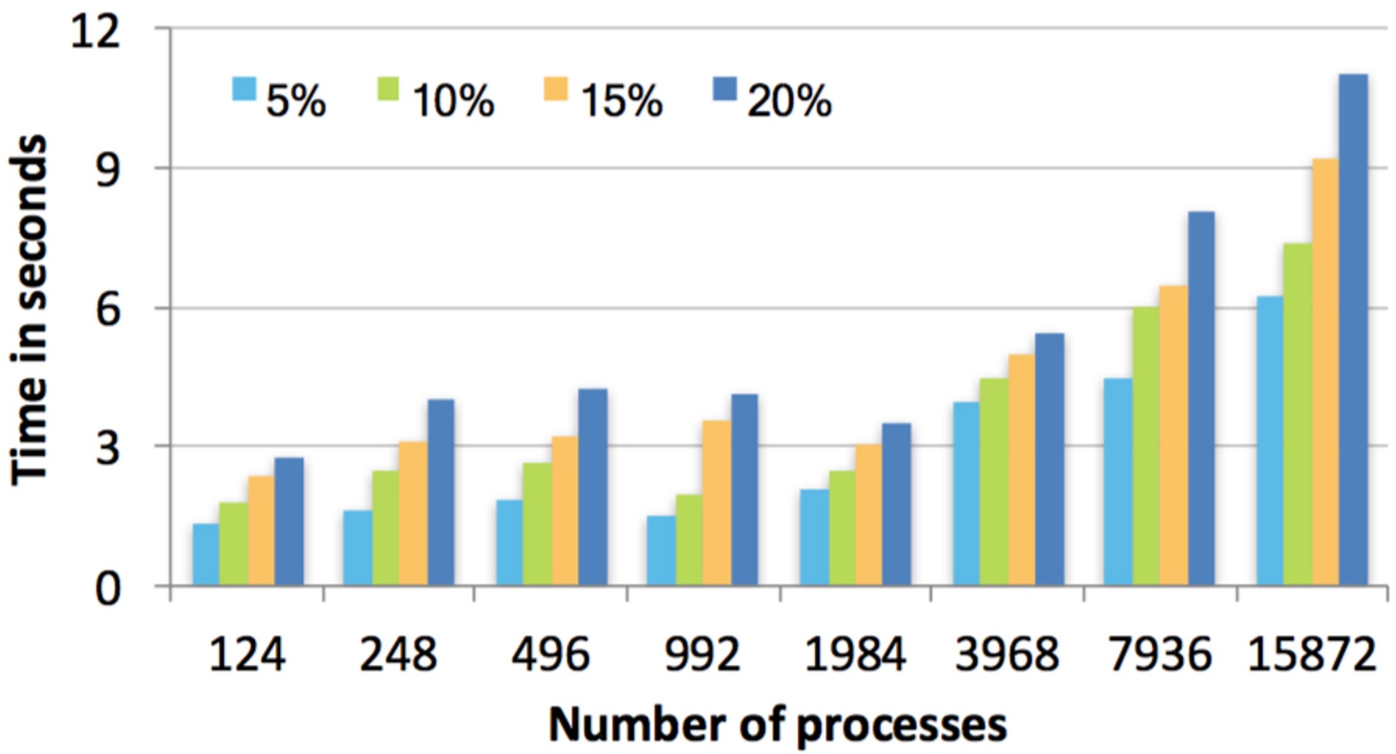

Read time with part of the data written to faster burst buffer and the remaining to slower Lustre file system on Cori.

# Spatial-selection Data Read from Lustre




Time to read various selected object regions specified by the client processes from Lustre on Cori.

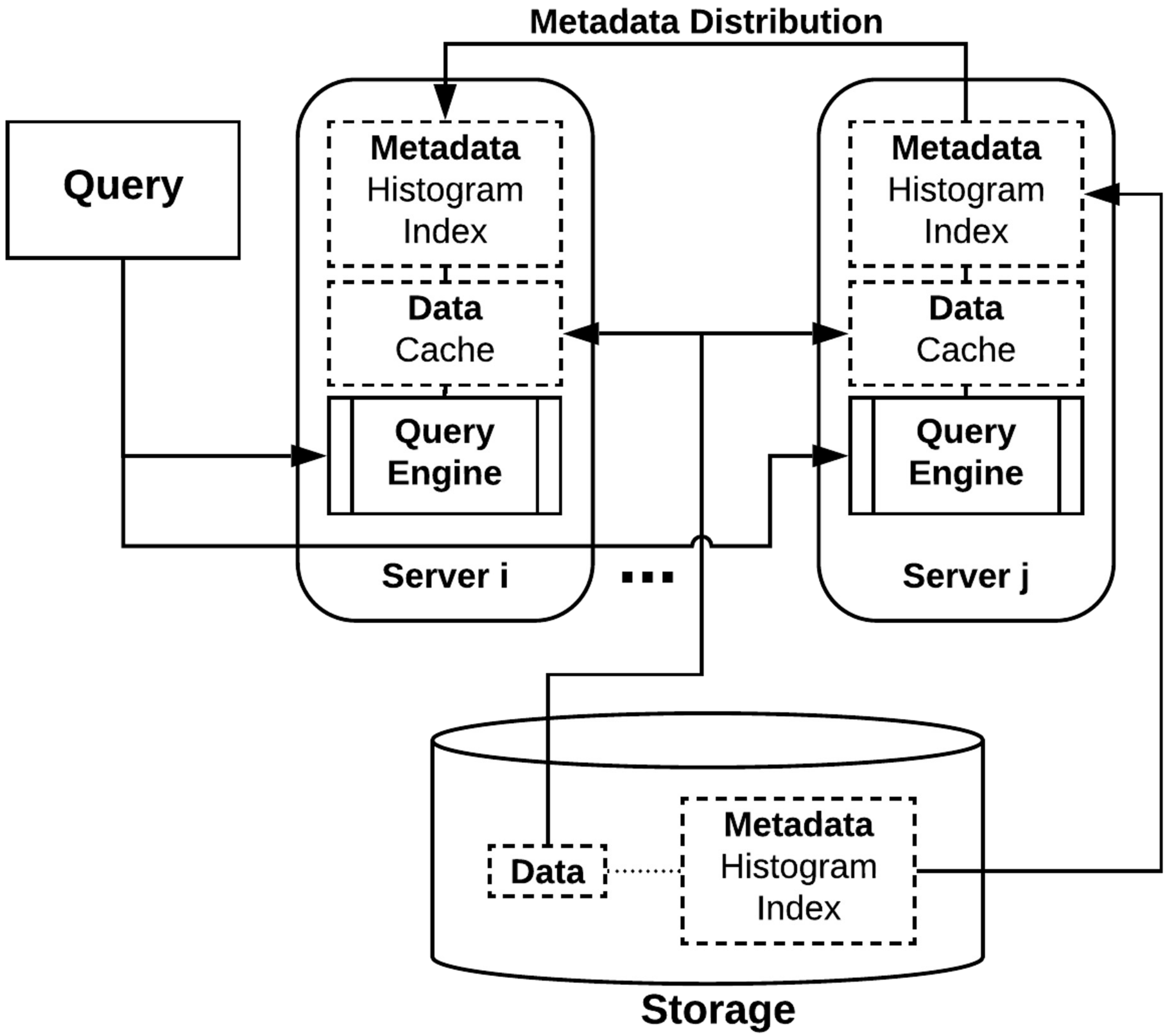
# Spatial-selection Data Read from Burst Buffer




Time to read various selected object regions specified by the client processes from burst buffer on Cori.

# Queries in PDC

- **Metadata query**
  - Previous paper: "SoMeta: Scalable Object-Centric Metadata Management for High Performance Computing"
- **Data query**
  - Single variable
  - Multi variable
  - Get number of hits
  - Get selection
  - Get value



# PDC-query Interface

```
// Create a one-sided data query
pdcquery_t *PDCquery_create(pdcid_t obj_id, pdcquery_op_t op, pdctype_t type,
void *value);
// Combine queries
pdcquery_t *PDCquery_and(pdcquery_t *query1, pdcquery_t *query2);
pdcquery_t *PDCquery_or(pdcquery_t *query1, pdcquery_t *query2);
// Set query region constraint
perr_t PDCquery_set_region(pdcquery_t *query, pdcregion_t *region);
// Query operations
perr_t PDCquery_get_nhits(pdcquery_t *query, uint64_t *n);
perr_t PDCquery_get_selection(pdcquery_t *query, pdcselection_t *sel);
perr_t PDCquery_get_data(pdcid_t obj_id, pdcselection_t *sel, void *data);
perr_t PDCquery_get_data_batch(pdcid_t obj_id, pdcselection_t *sel, uint64_t
batch_size, void *data);
pdchistogram_t *PDCquery_get_histogram(pdcid_t obj_id);
```

# Query Evaluation Strategies

- **Full scan**
  - Straightforward parallel implementation.
  - Go over all elements and check against query condition.
  - Slow for single variable and simple query condition.
- **Data reorganization w/ sorting**
  - Requires data preparation, extra storage.
  - Eliminates the need to go through all elements.
  - Best performance for single variable query.
- **Bitmap index**
  - Requires index building in advance.
  - Go through index instead of data.
  - Best performance if actual values are not required.

# Optimization

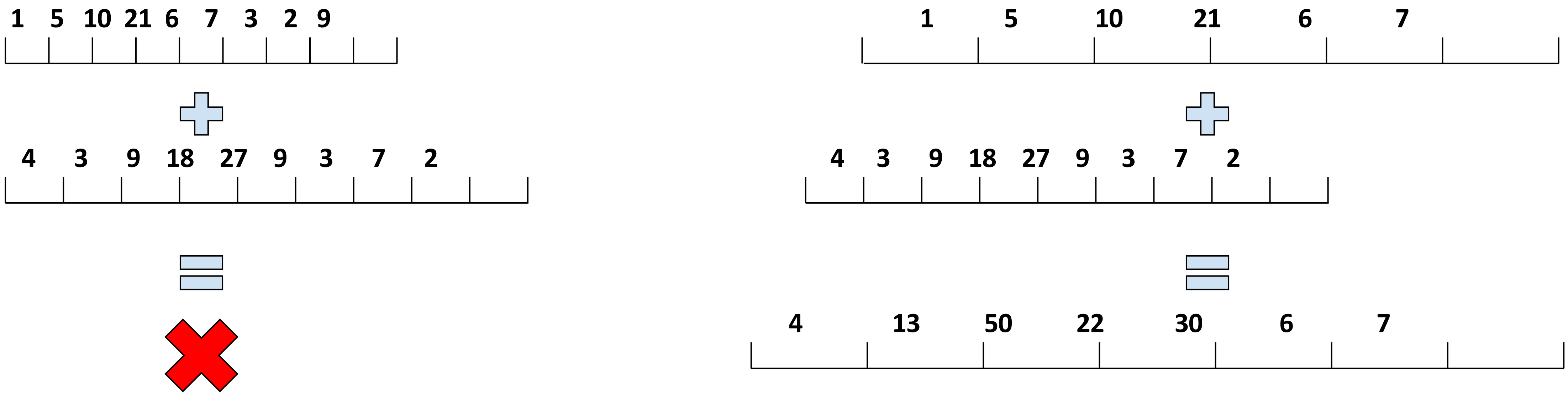
- **Full scan**
  - Skip the inspection of some amount of data?
- **Data reorganization**
  - Speedup the evaluation process for multivariate query conditions?
- **Index**
  - Skip the evaluation of some indexes?
  - Evaluate the highly selective variable first?

# Histogram

- Generate a histogram for each PDC region
  - Done at data creation time or during server “free” time - *asynchronously*
- Use histogram to get max/min value of a region
- Use histogram to estimate the selectivity of each variable
  - Re-order the query evaluation, prune as many regions as possible.

- Generating a global one is costly, and needs coordination for updates.
  - Can we generate local region-specific histograms that can be easily merged into a global one?

# Mergeable Histogram

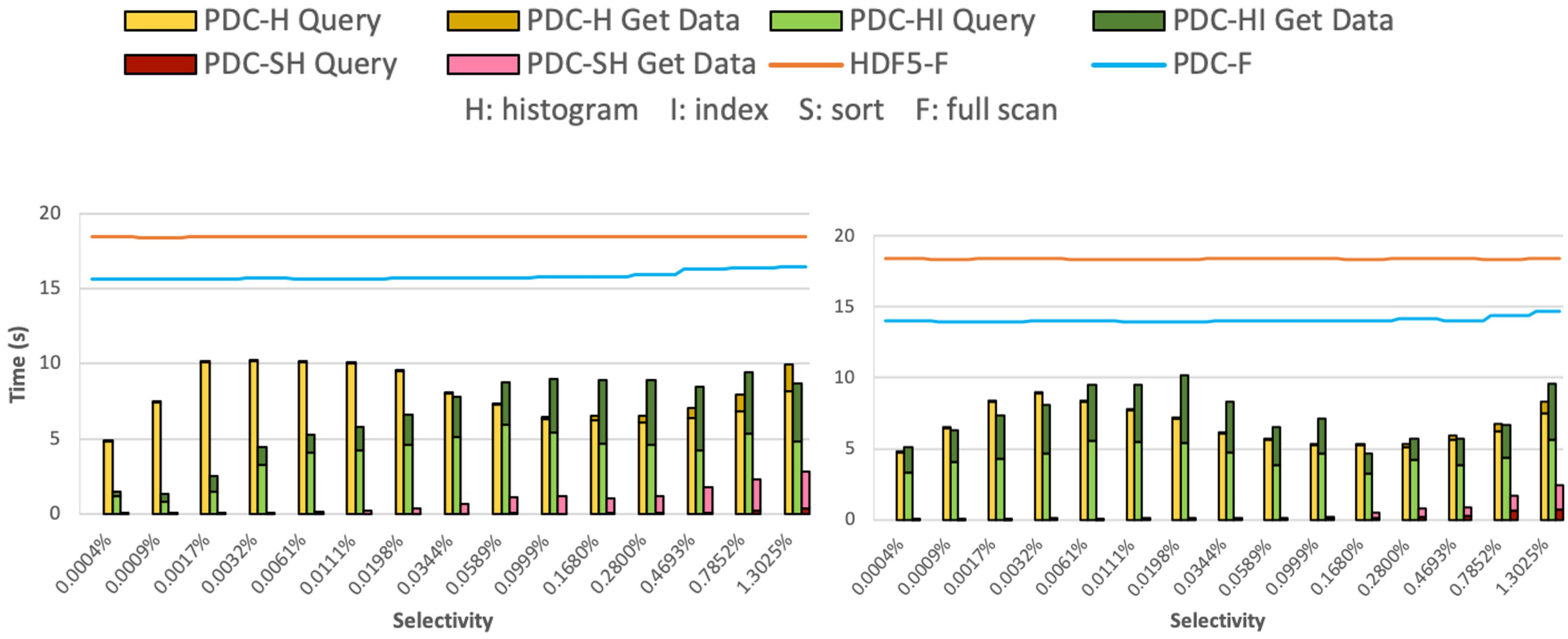



The bin width of different histograms must be same or divisible.

Use random sampling to get approximate min/max and make them aligned with bin boundaries of other histograms.

Both use values from pre-defined sets, $2^n$ and $N \pm 2^n$.

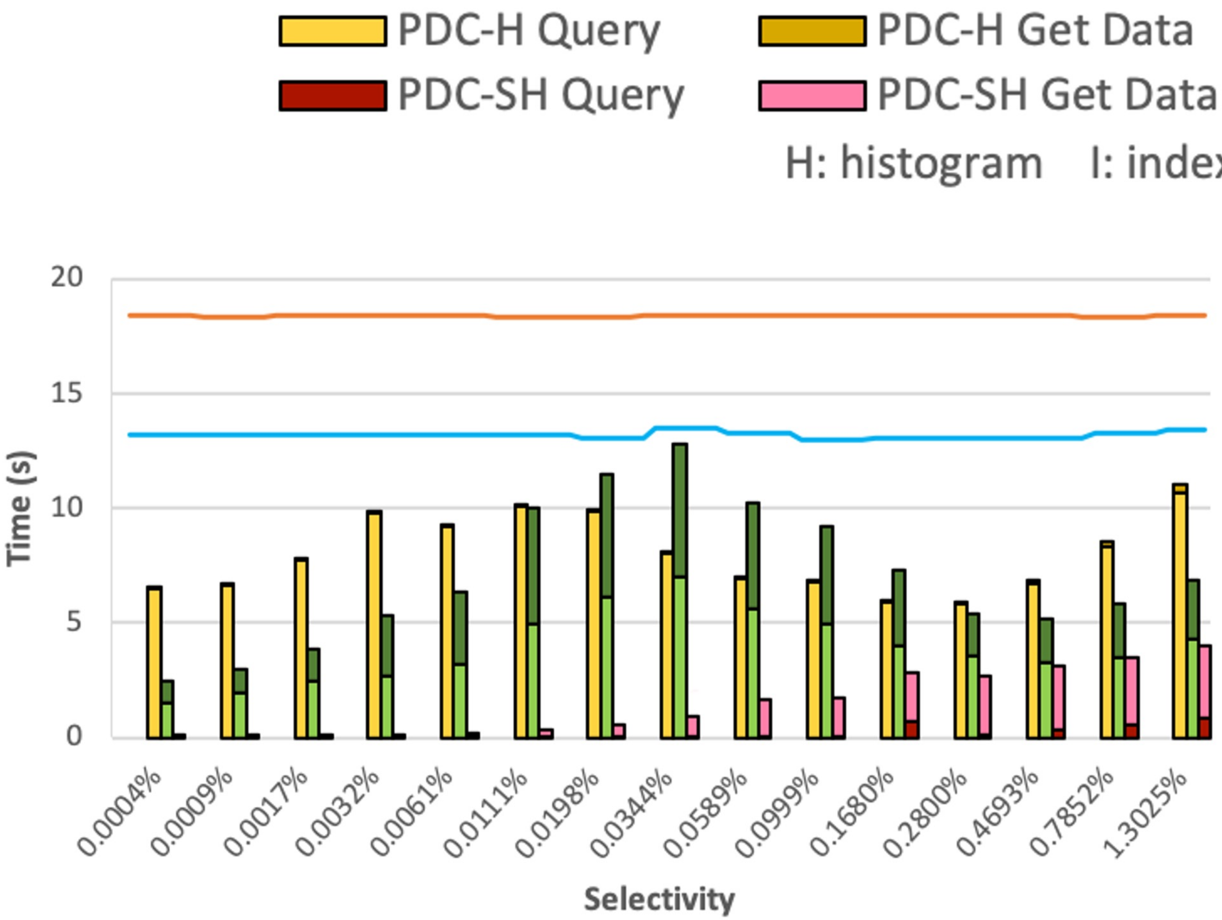
# Region Size

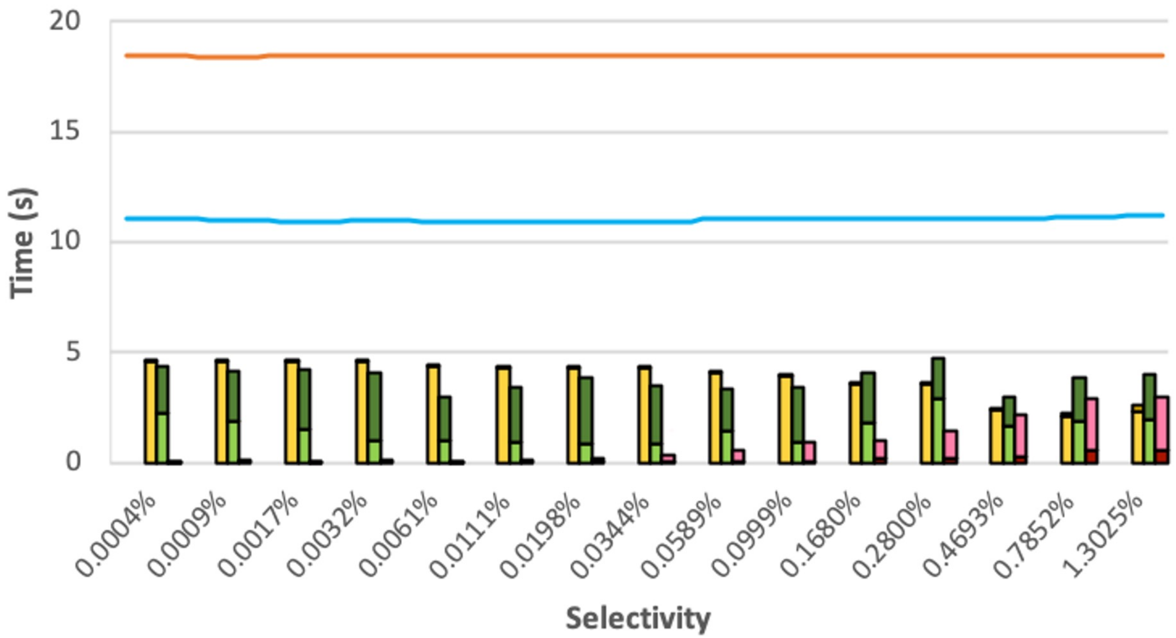



4MB

8MB

# Region Size

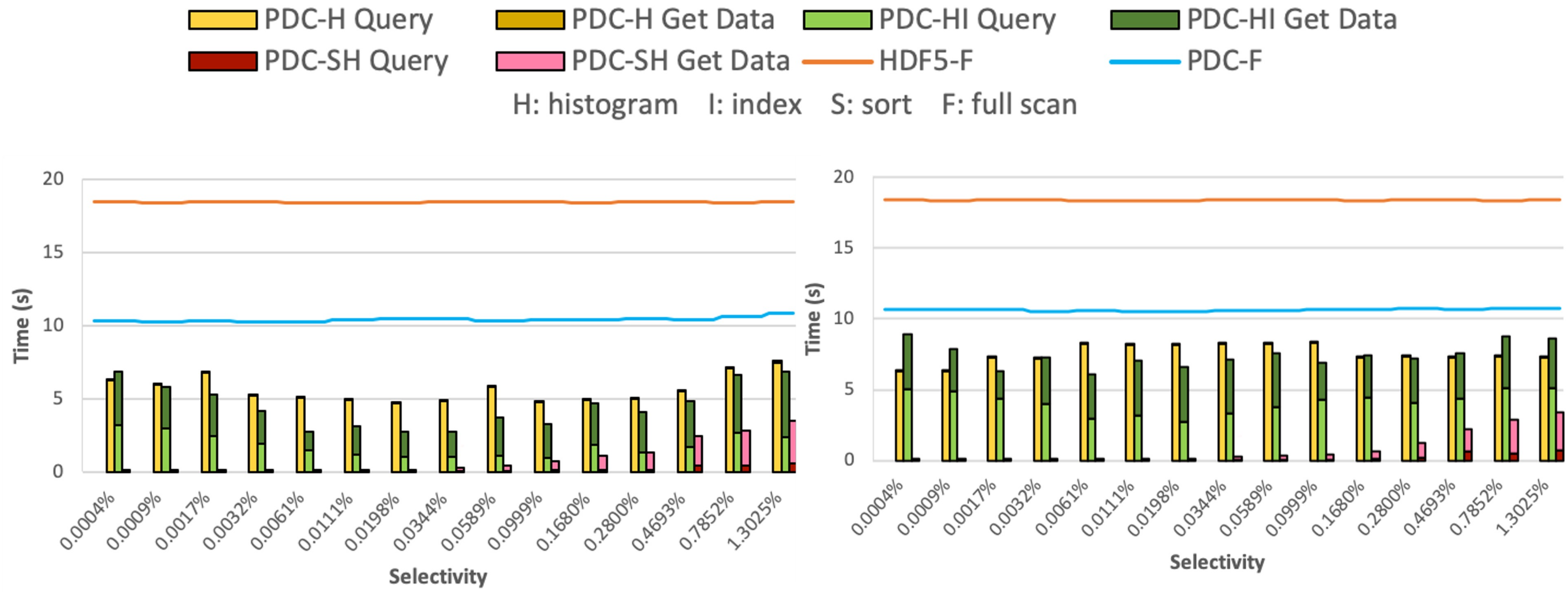



16MB

32MB

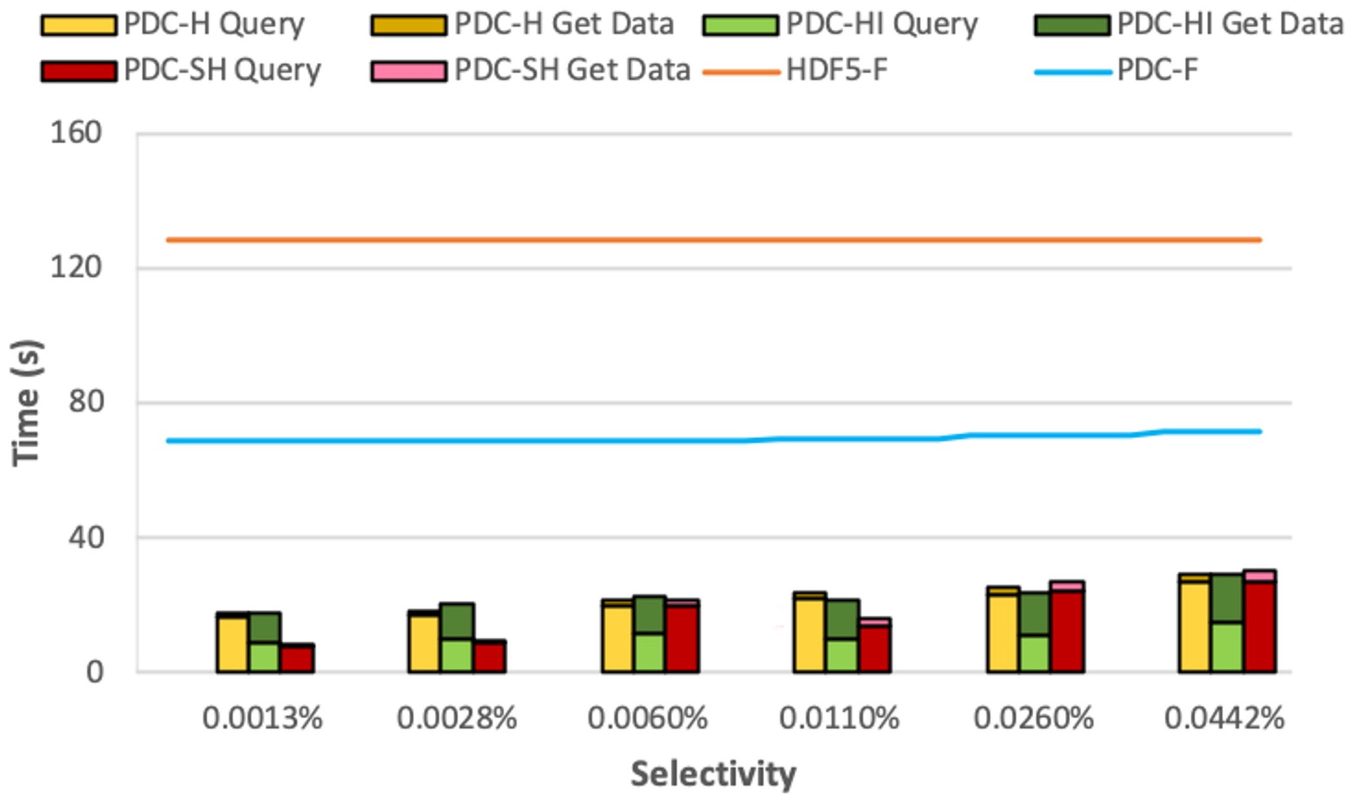
# Region Size




64MB

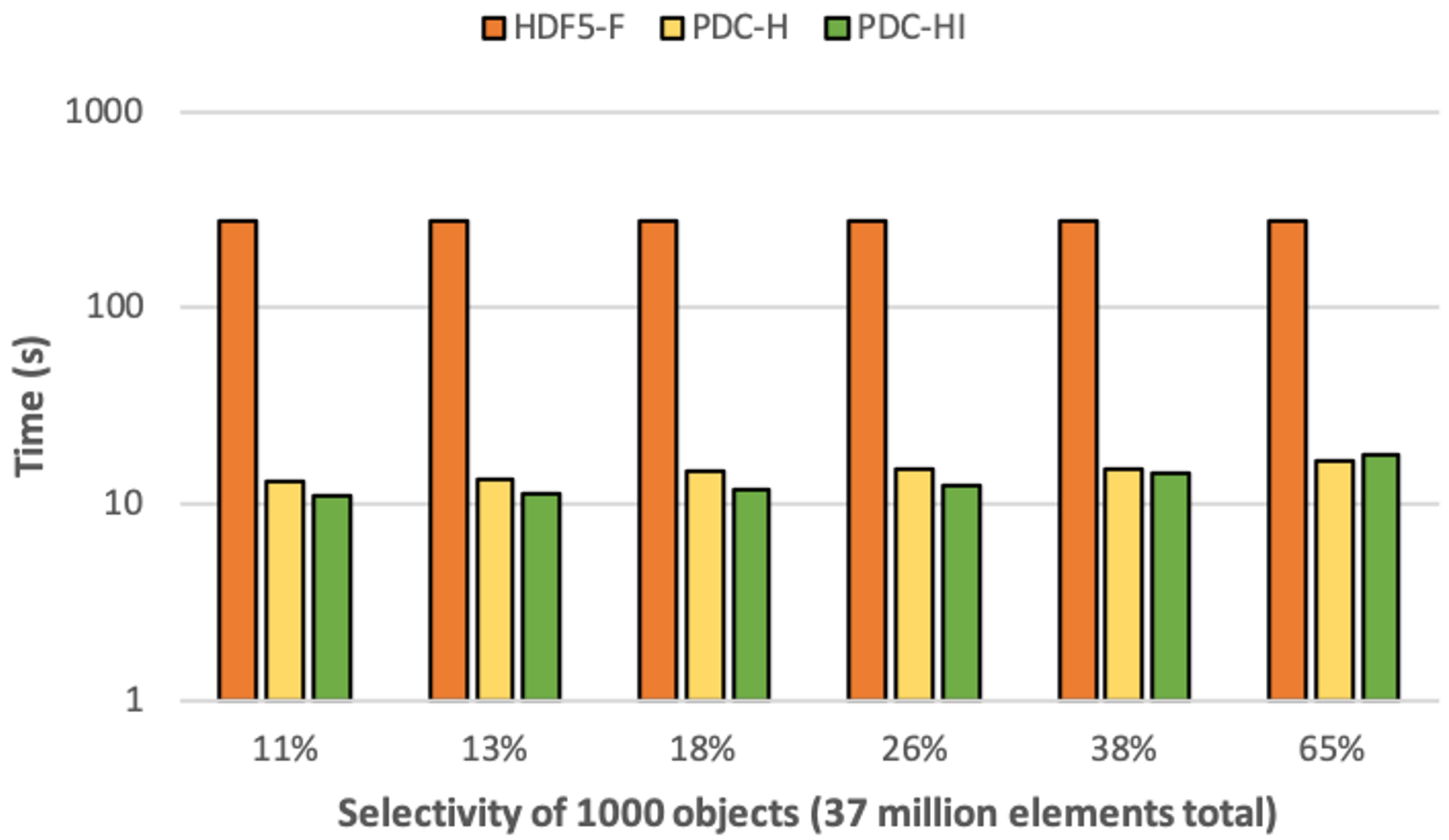
128MB

# Results - Multivariate Query



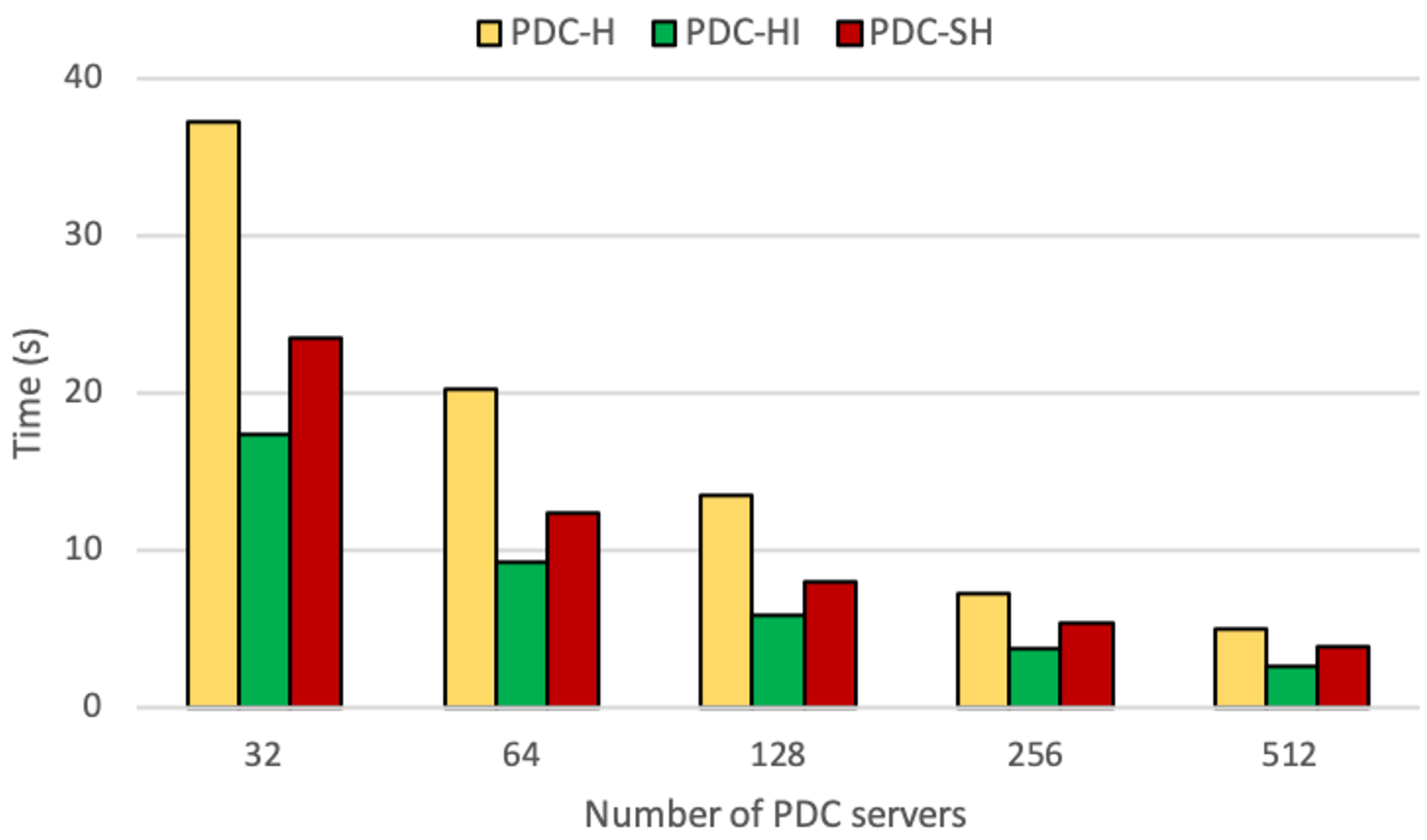

**PDC-H**: PDC with **H**istogram only, **PDC-HI**: PDC with **Hi**stogram and Fastbit **I**ndex, **PDC-SH**: PDC with **S**orted data (sorted by the 'energy' object) and **H**istogram. **HDF5-F**: amortized time to evaluate the 6 queries with HDF5 **F**ull scan. **PDC-F**: amortized time to evaluate the 6 queries with PDC **F**ull scan.

# Results - Metadata + Data Queries




Comparison of queries with both metadata (fixed selectivity on 1000 objects) and data conditions (varied selectivity from 11% to 65%) on the H5BOSS dataset.

# Results - Multivariate Scaling




Query time comparison for a multi-object query condition with 0:011% selectivity using different number of PDC servers.

# Summary of today's class

- PDC data and query services

- Next Class – Project

- Class project –
  - ~~Status update on Apr $11^{th}$~~
  - Project presentation on Apr $20^{th}$
  - Final exam on Apr $25^{th}$