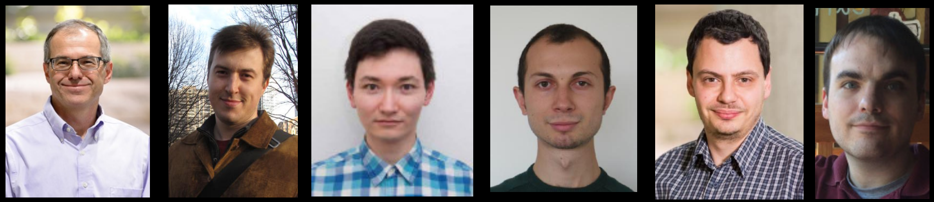
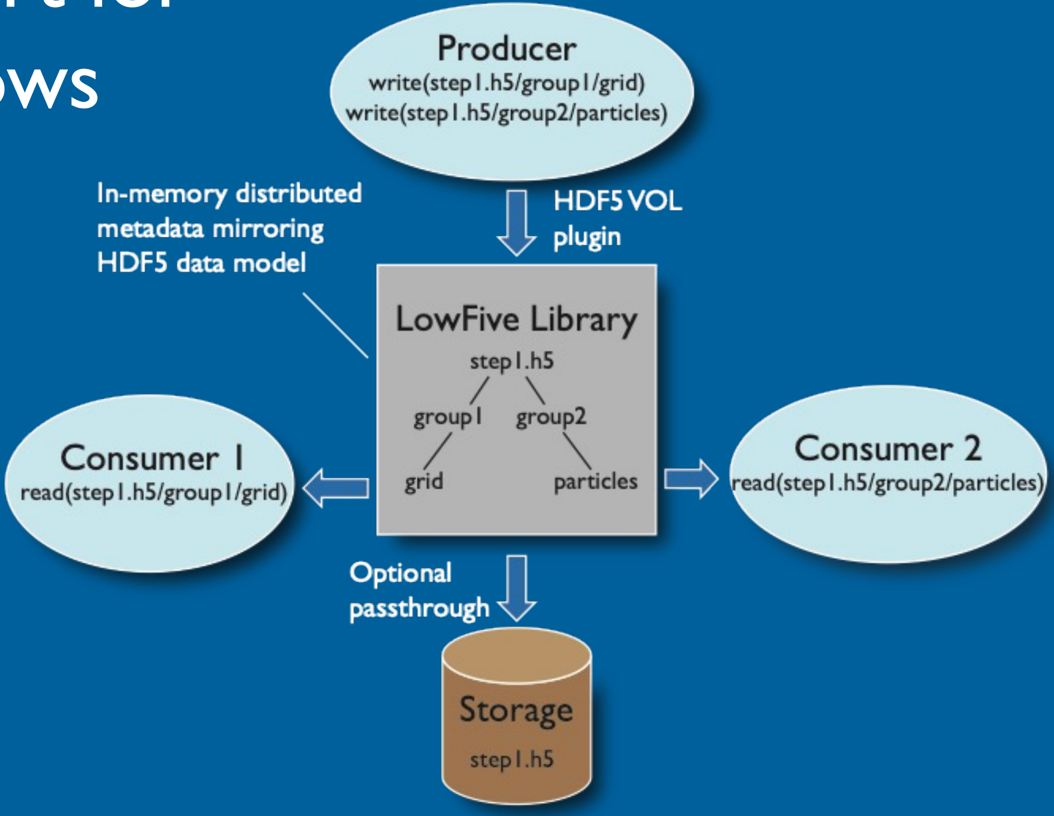


github.com/diatomic/LowFive

LowFive: In Situ Data Transport for High-Performance Workflows

Tom Peterka,	ANL
Dmitriy Morozov	LBNL
Arnur Nigmetov	LBNL
Orcun Yildiz	ANL
Bogdan Nicolae	ANL
Philip Davis	U. Utah



An example of three tasks coupled through the LowFive in situ data transport library.

"Somewhere, something incredible is waiting to be known."

–Carl Sagan

HDF5 Users Group Meeting '23
August 17, 2023

Tom Peterka
tpeterka@mcs.anl.gov
Mathematics and Computer Science Division

Design Choices

A balance between user's view of data (productivity) and the workflow's efficient movement of data (performance)

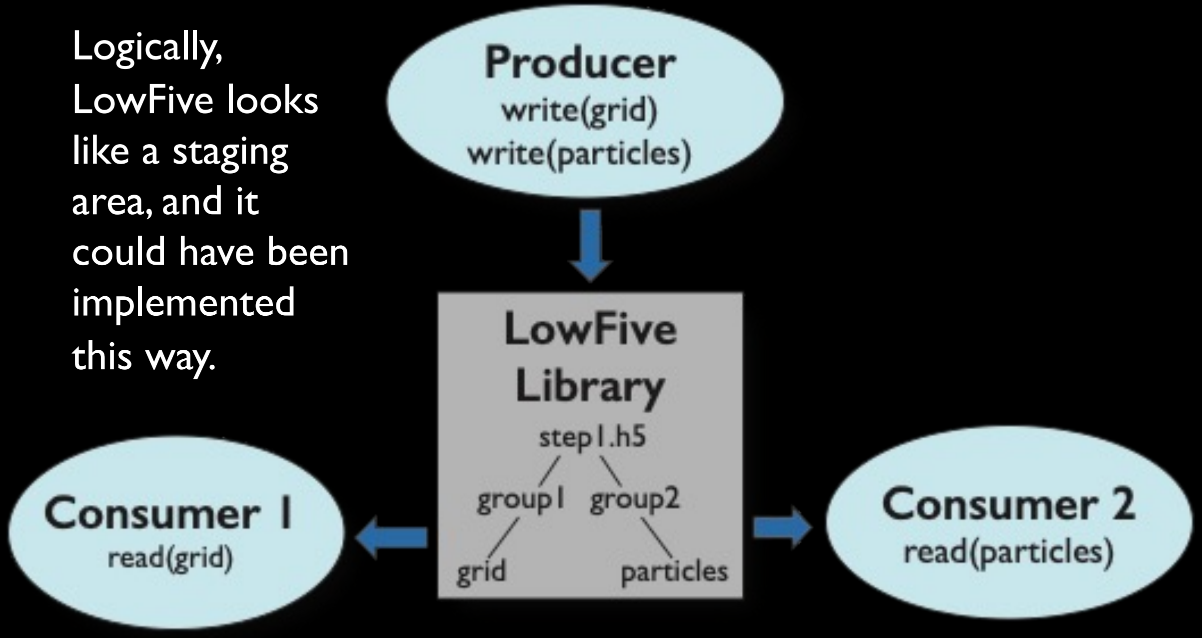
Design Criteria	LowFive Choices
User's view of data (model or schema)	HDF5 data model
In situ transport mechanism (direct, staging)	Direct, parallel, MPI point to point messages
Software stack intercept location	High-level HDF5 metadata
Software design	Standalone HDF5 VOL plugin

In Situ Data Transport Mechanism

Staging

- Dedicated resources for transport
- Decouple producer from consumer (could allow overlap)
- May require launching a separate service
- Shared access (could also involve locking)

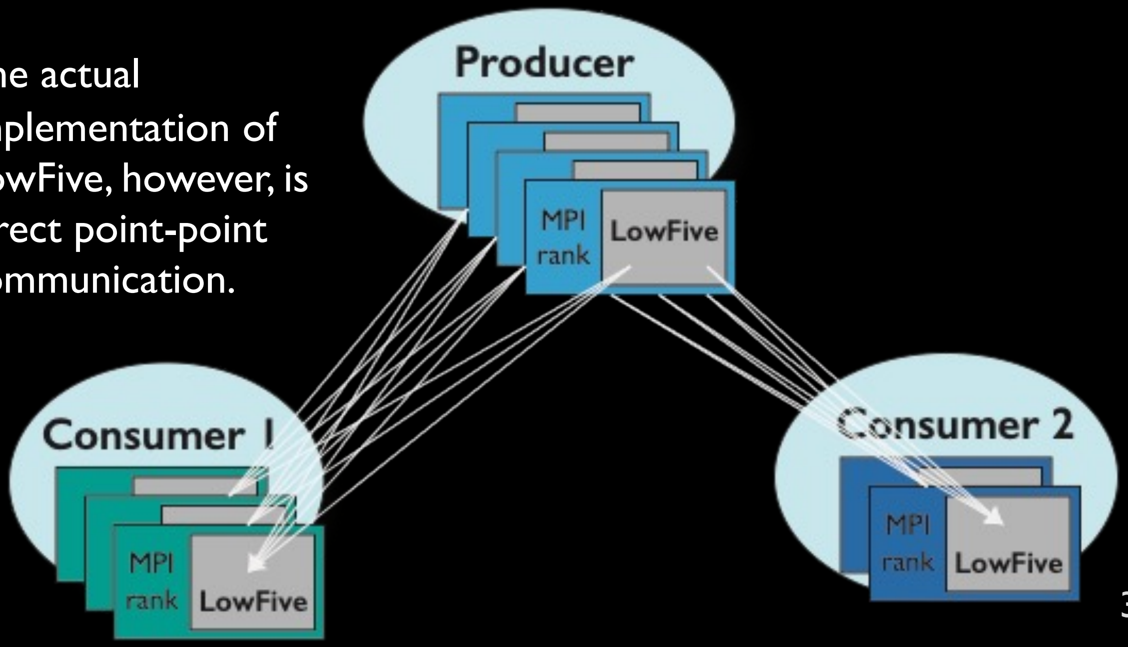
Logically, LowFive looks like a staging area, and it could have been implemented this way.



Direct

- No additional resources or services
- Simple, point-to-point communication
- Tightly coupled producer and consumer (synchronous)
- A staging area could still be a producer/consumer task

The actual implementation of LowFive, however, is direct point-point communication.



Software Stack Intercept Location

- POSIX level (Burst buffer systems)
  - Catch all I/O
  - No metadata

Software Stack Intercept Location

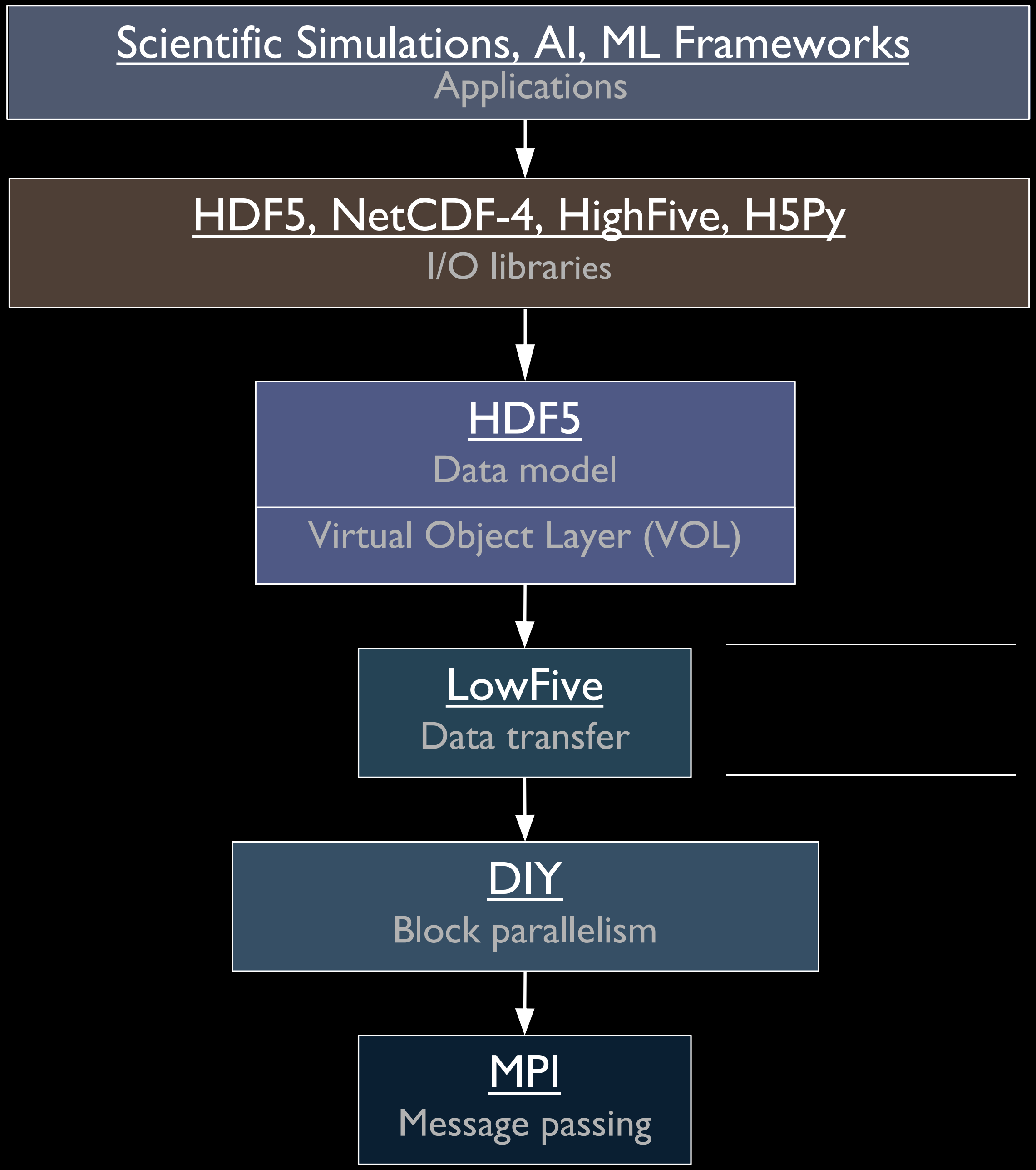
- Application level (Conduit, Bredala)
  - All metadata
  - Change user code

- POSIX level (Burst buffer systems)
  - Catch all I/O
  - No metadata

Software Stack Intercept Location

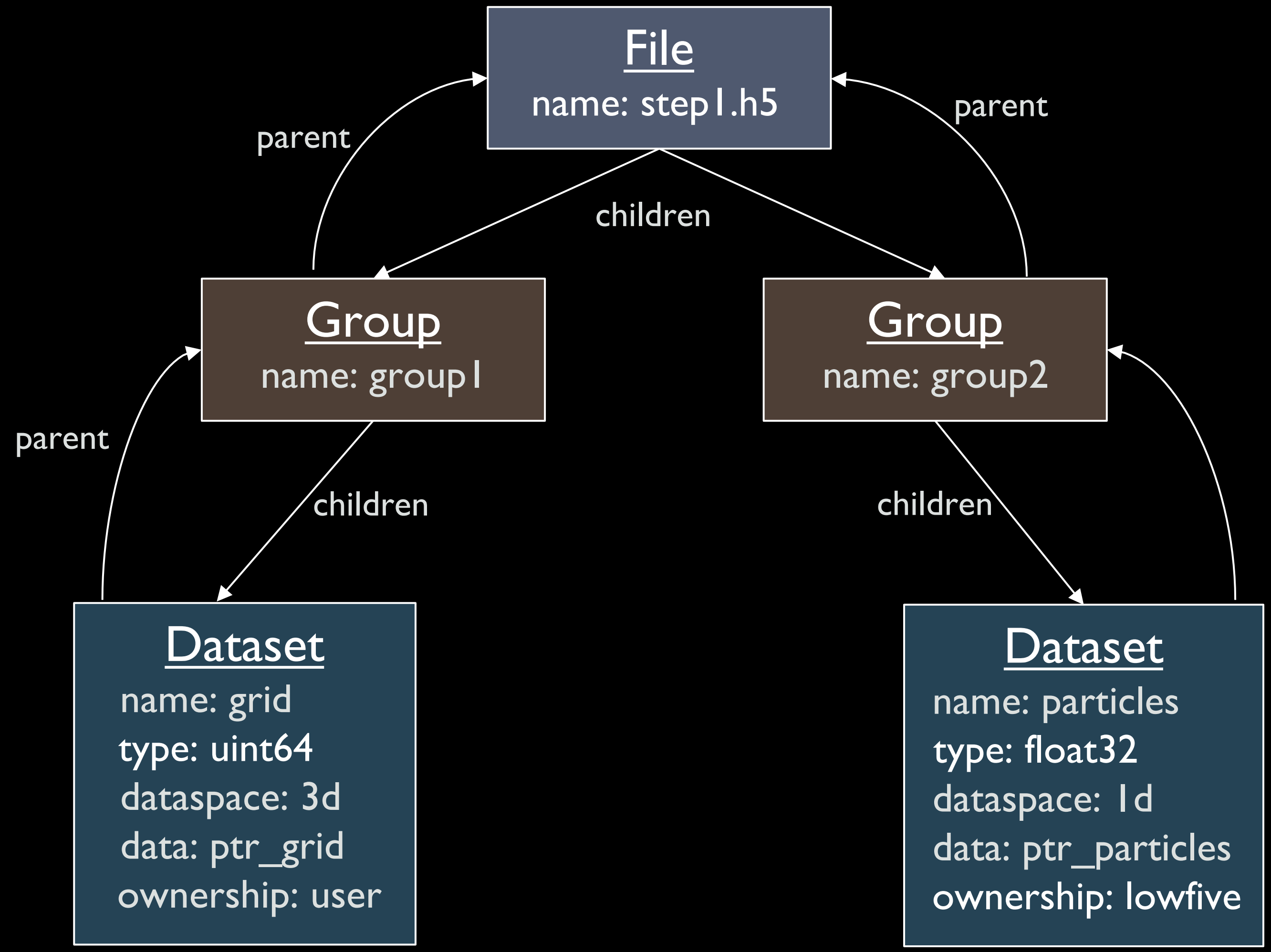
- Application level (Conduit, Bredala)
  - All metadata for data transport
  - Change user code
- High-level I/O API (LowFive)
  - Rich metadata for data transport
  - Little/no change to user code
- POSIX level (Burst buffer systems)
  - Catch all I/O
  - No metadata

Software Stack



- DistMetadataVOL
- MetadataVOL
- VOLBase

LowFive Metadata Tree



Our own LowFive in-memory replica of HDF5 data model.

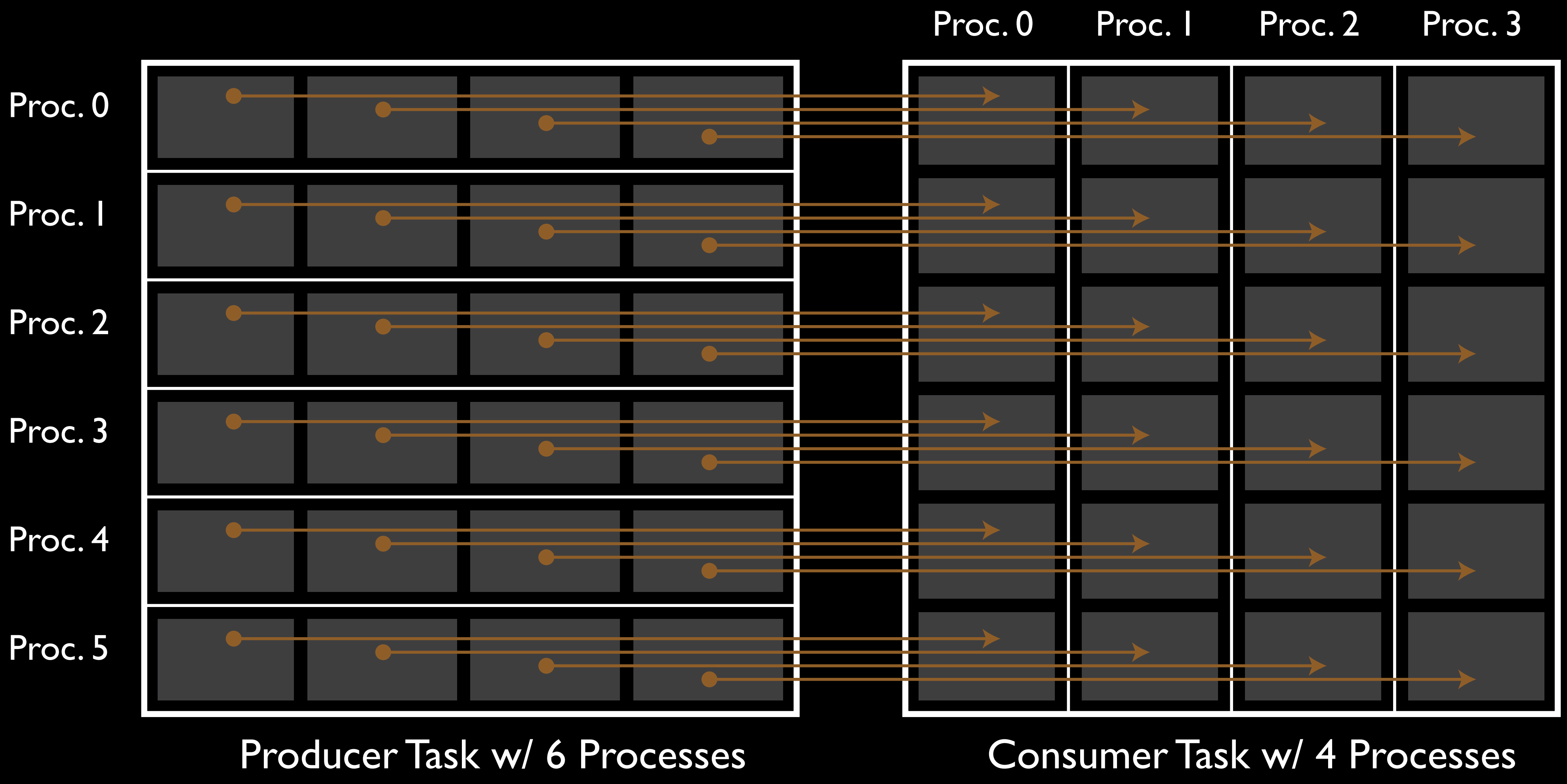
HDF5 Data Model

- Hierarchical data model much like a UNIX file system
- Root is the file
- Internal nodes are groups
- Leaves are datasets or other objects (e.g., attributes)

LowFive Data Model

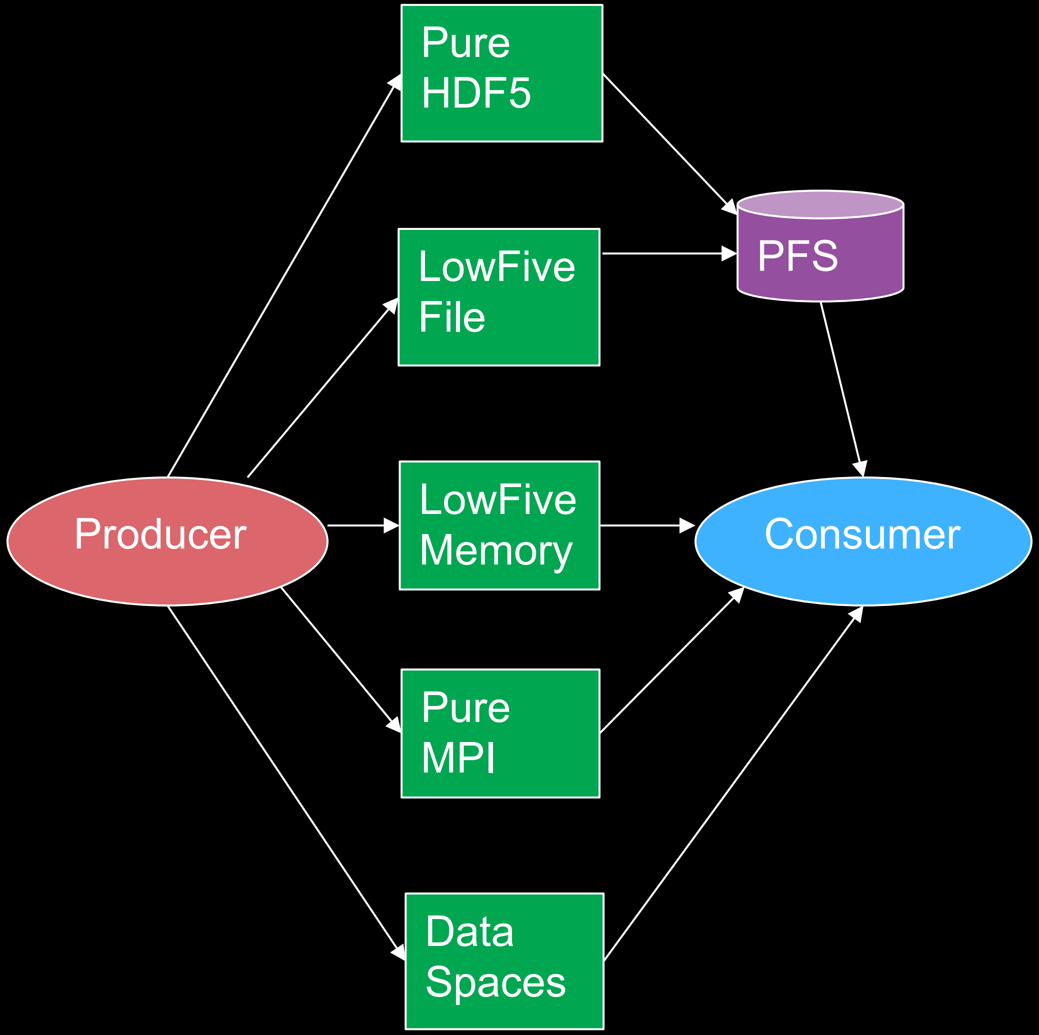
- Our in-memory replica of HDF5 metadata
- One object for every HDF5 object
- Shallow or deep data pointer or copy

Data Redistribution



Example of data redistribution from a producer task with 6 processes decomposed row-wise to a consumer task with 4 processes decomposed column-wise. The problem is that neither the producer nor the consumer task knows anything about the other's decomposition.

Synthetic Benchmarks

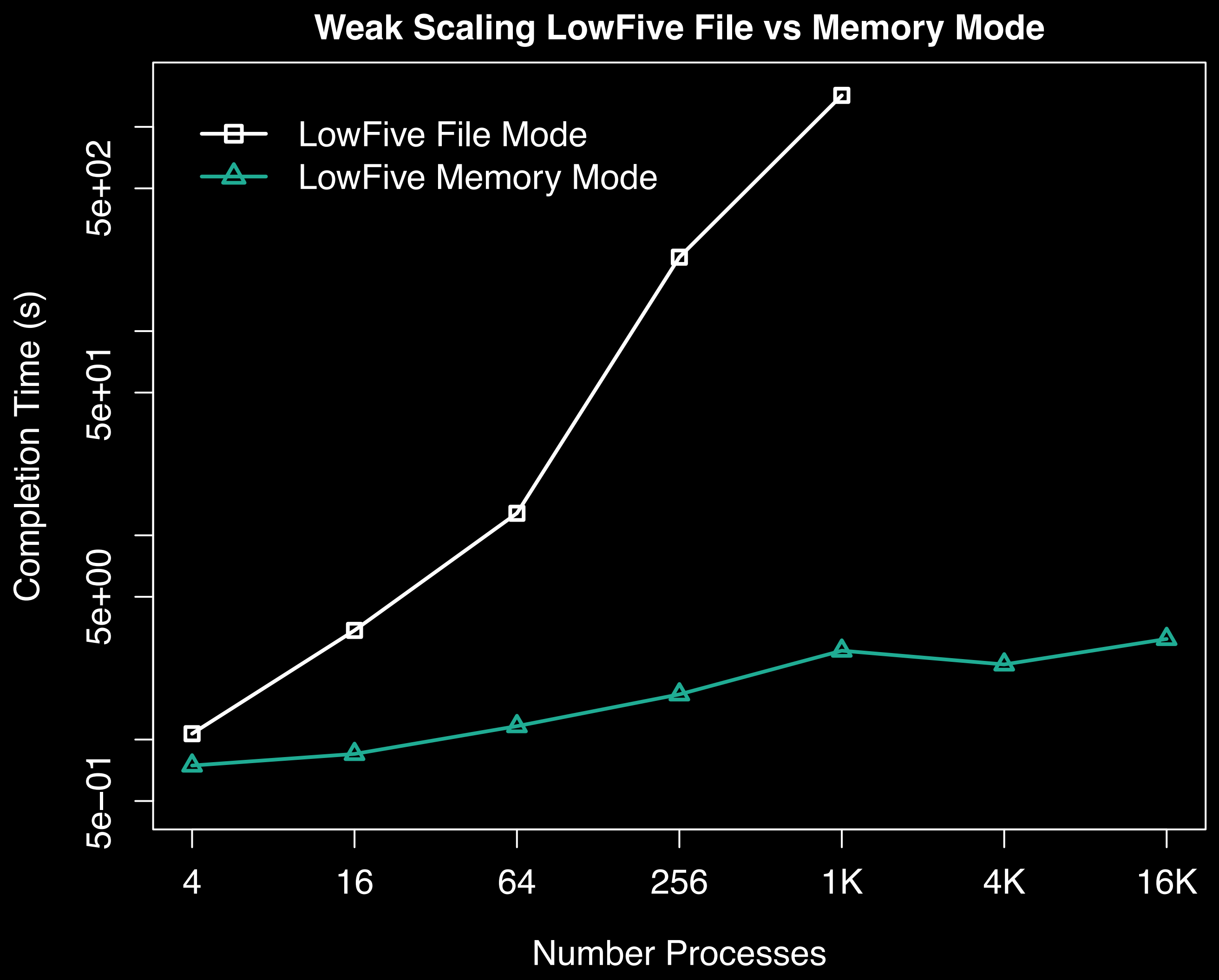


Different experiment scenarios

Total # MPI Procs.	# Producer Procs.	# Consumer Procs.	Total # Grid Points	Total # Particles	Total Data Size (GiB)
4	3	1	3.0e6	3.0e6	0.06
16	12	4	1.2e7	1.2e7	0.22
64	48	16	4.8e7	4.8e7	0.99
256	192	64	1.9e8	1.9e8	3.54
1024	768	256	7.7e8	7.7e8	14.34
4096	3072	1024	3.0e9	3.0e9	55.88
16384	12288	4096	1.2e10	1.2e10	223.51

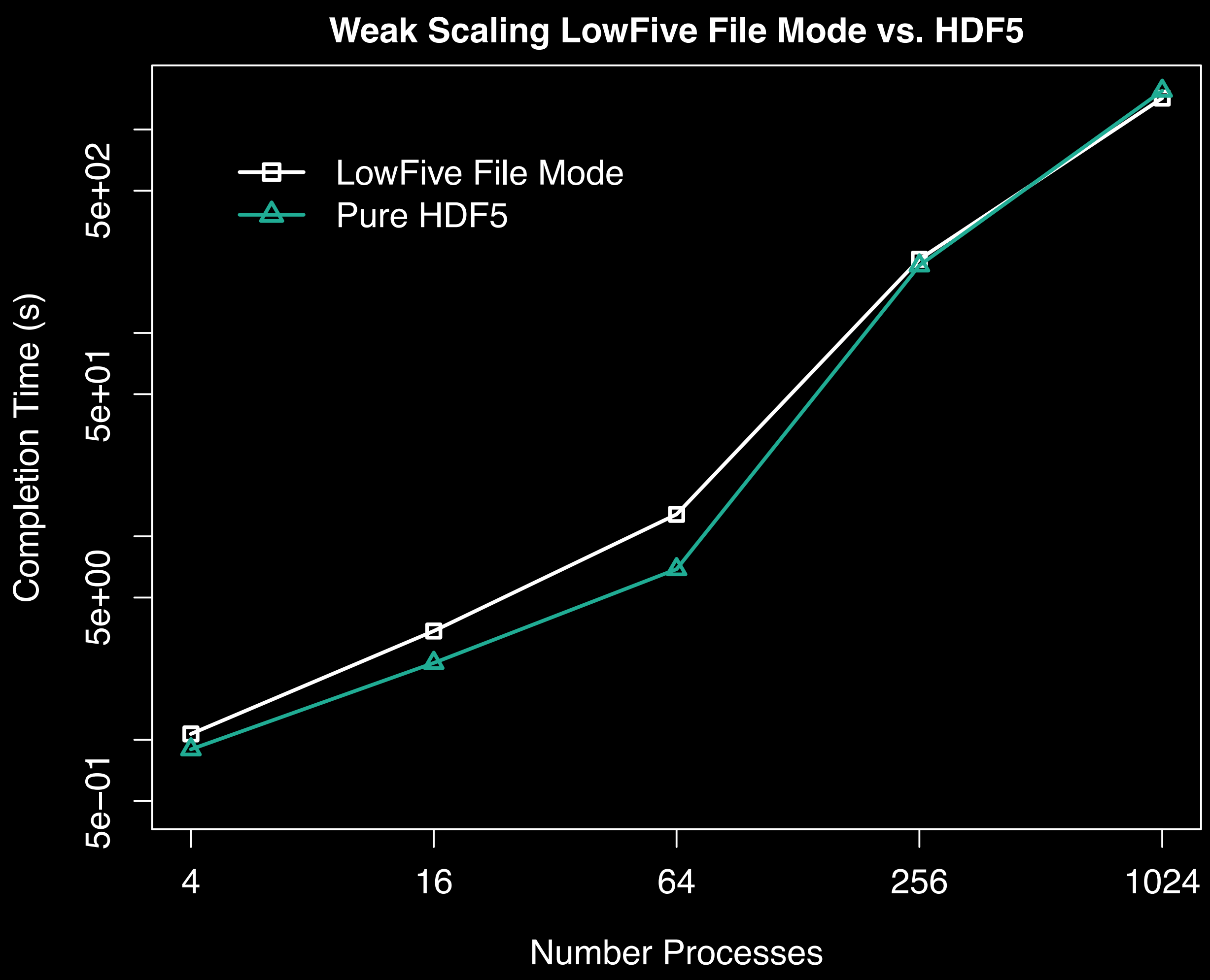
Number of processes and data sizes for synthetic benchmark experiments

Synthetic Benchmarks: In Situ vs. Storage



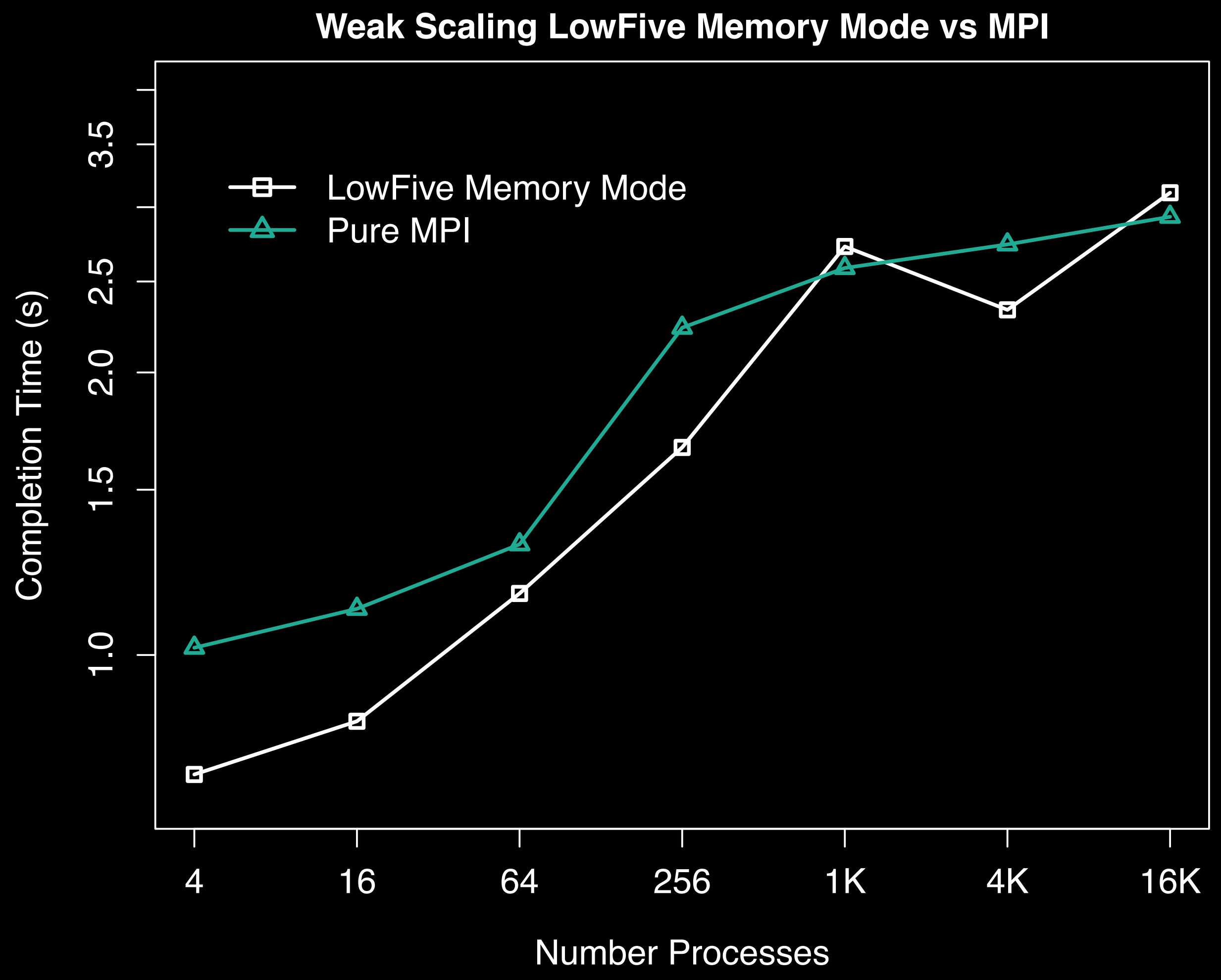
Time to write/read grid and particles between 1 producer task and 1 consumer task, comparing LowFive file and memory modes, in a weak scaling regime.

Synthetic Benchmarks: Overhead of Using LowFive vs. Pure HDF5 for File I/O



Time to write/read grid and particles, comparing LowFive file mode with pure HDF5 file, in a weak scaling regime.

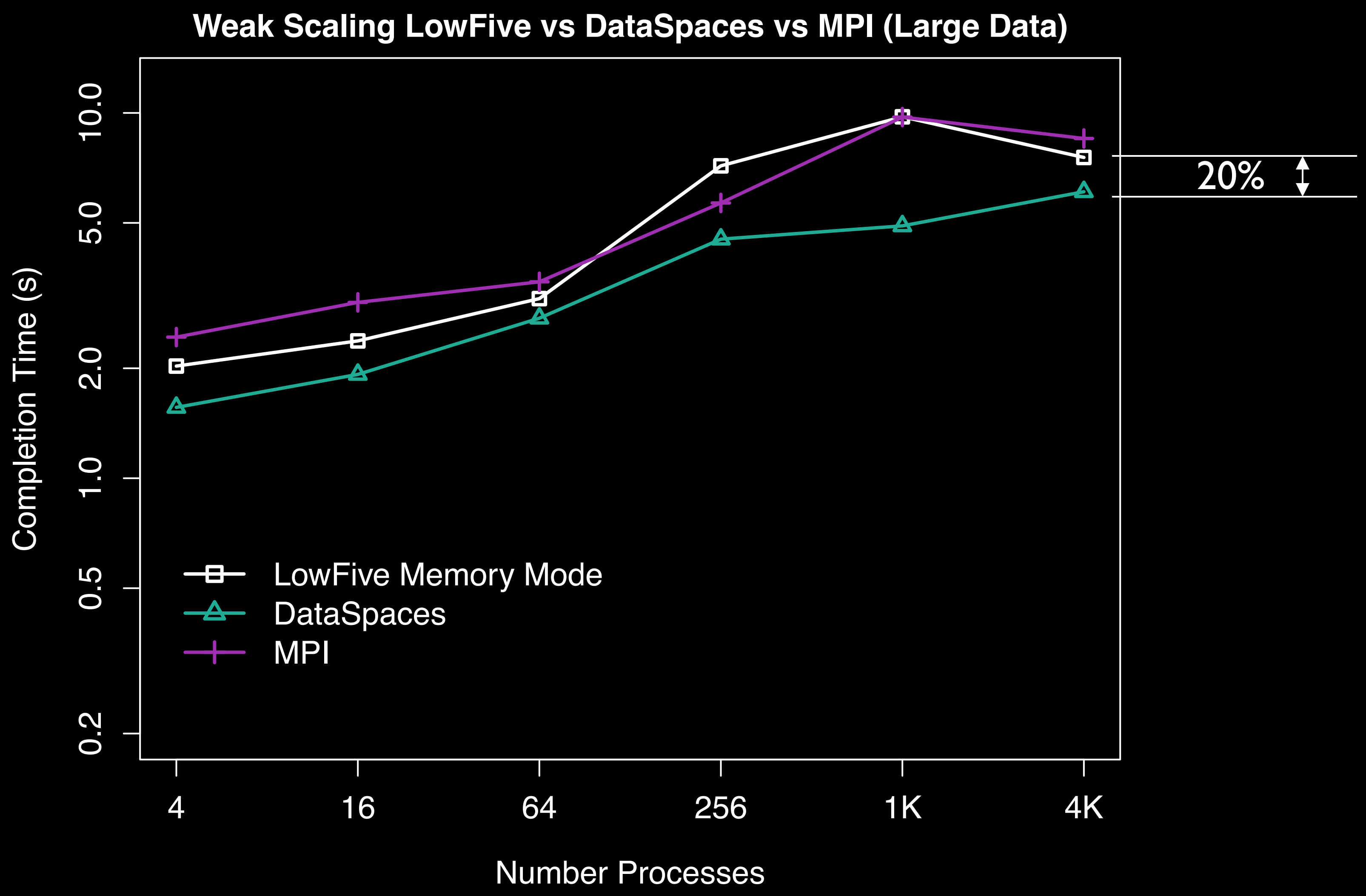
Synthetic Benchmarks: Overhead of Using LowFive vs. Pure MPI for Message Passing



Time to write/read grid and particles comparing LowFive memory mode, with pure MPI communication, in a weak scaling regime.

Synthetic Benchmarks: 10X Data Size

- 10^7 regularly structured grid points + 10^7 particles per producer process
- 190 MiB of data per producer process
- 0.55 GiB of data per consumer process (3:1 producer:consumer procs)
- Total data size at the largest scale tested is 0.55 TiB.

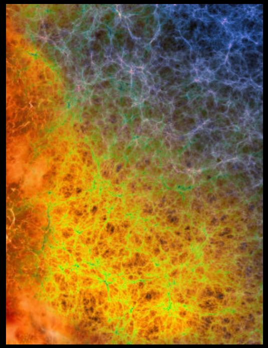


Time to write/read large size grid and particles, comparing LowFive memory mode, DataSpaces, and pure MPI, in a weak scaling regime

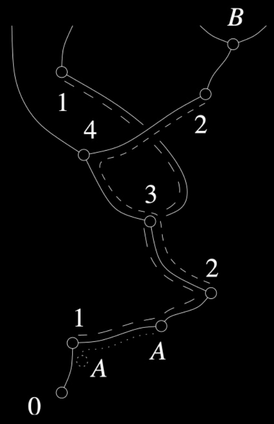
Science Workflow: Cosmology

Nyx → Reeber

Both Nyx and Reeber were used "off the shelf" with no modifications to use LowFive (in the Henson workflow system)



Dark matter particles
Image: https://crd.lbl.gov 2021



Merge tree
Image: Agarwal et al. 2004

Data Size	LowFive Write Time	LowFive Read Time	HDF5 Write Time	HDF5 Read Time	Plotfiles Write Time	LowFive vs HDF5	LowFive vs Plotfiles
256^3	2.87	0.106	5.46	0.37	4.42	1.9	1.54
512^3	2.00	0.287	104.20	0.69	18.10	52.01	9.03
1024^3	2.87	0.628	920.44	3.02	35.00	320.00	12.17
2048^3	7.69	3.205	x	x	154.52	x	20.09

Time to write/read data between Nyx and Reeber using LowFive memory mode, HDF5 files, and AMReX plotfiles demonstrates that LowFive in situ data transport is 20X faster at scale than the best I/O solution (AMReX plotfile format).

Recap

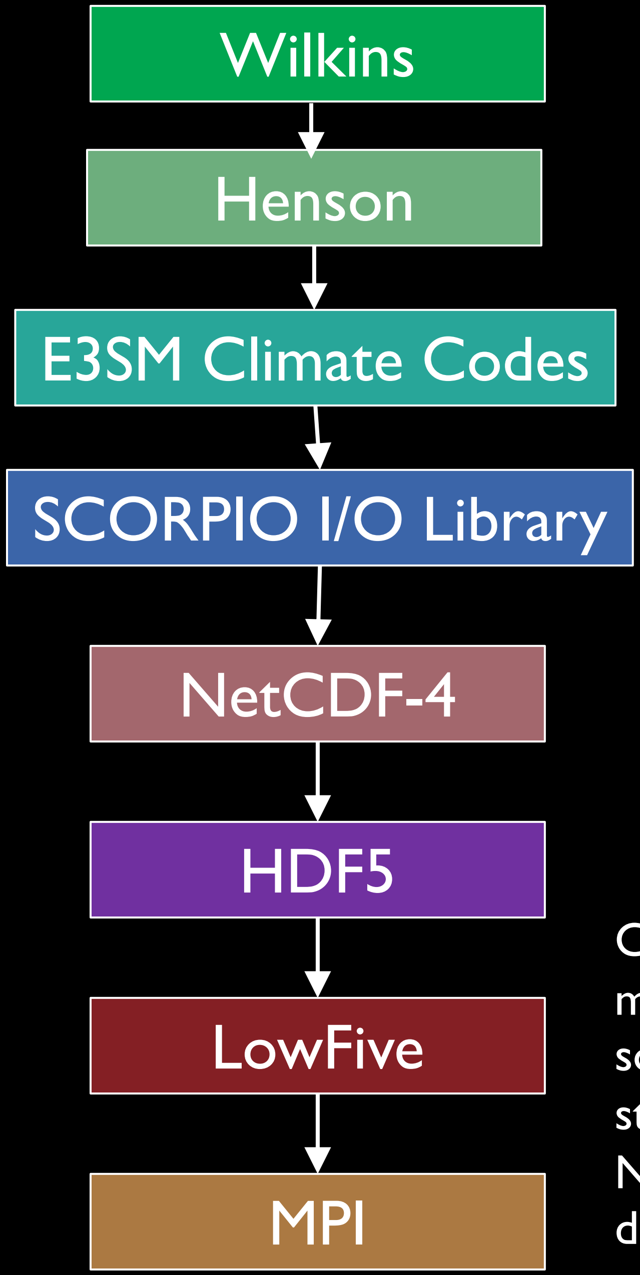
LowFive

- In situ data transport layer for workflows
- HDF5 data model
- Built as an HDF5 VOL plugin
- Allows bypassing storage and sending data over MPI
- Redistributes data between producer and consumer tasks
- Standalone software library that workflow systems can use

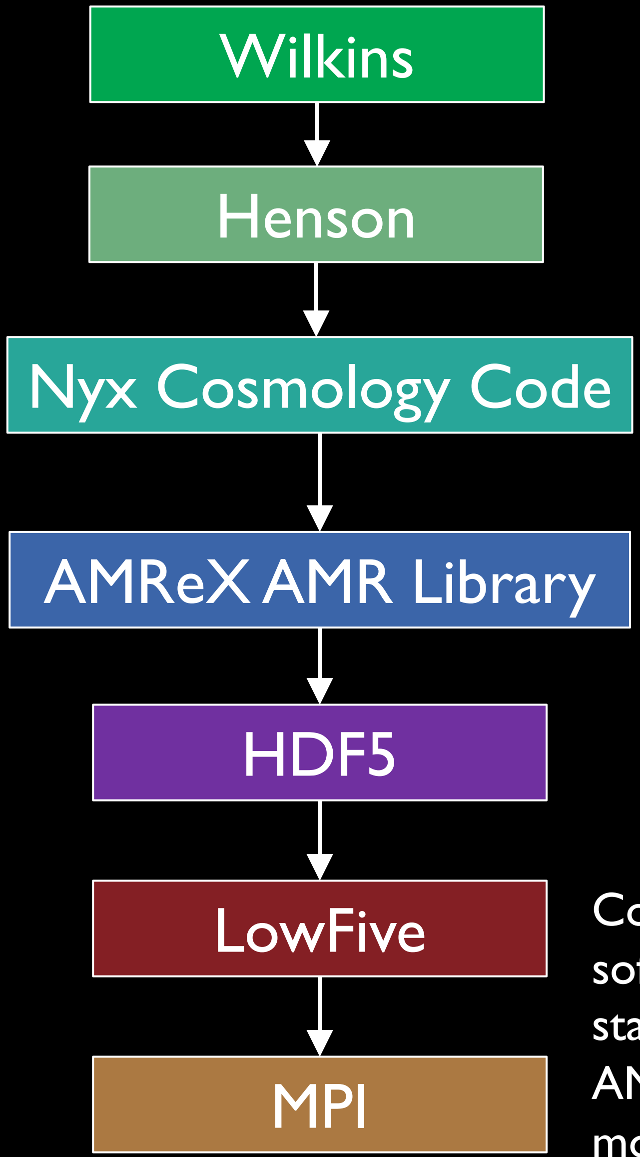
Next Steps

- Finish implementing missing functions in our metadata
- Continue to test on applications and their software stacks
- Producer – consumer synchronization and flow control
- Integrate in workflow systems driving further development
  - Henson can use LowFive (Nyx + Reeber use case)
  - We are also developing a new workflow system---Wilkins---on top of Henson and LowFive

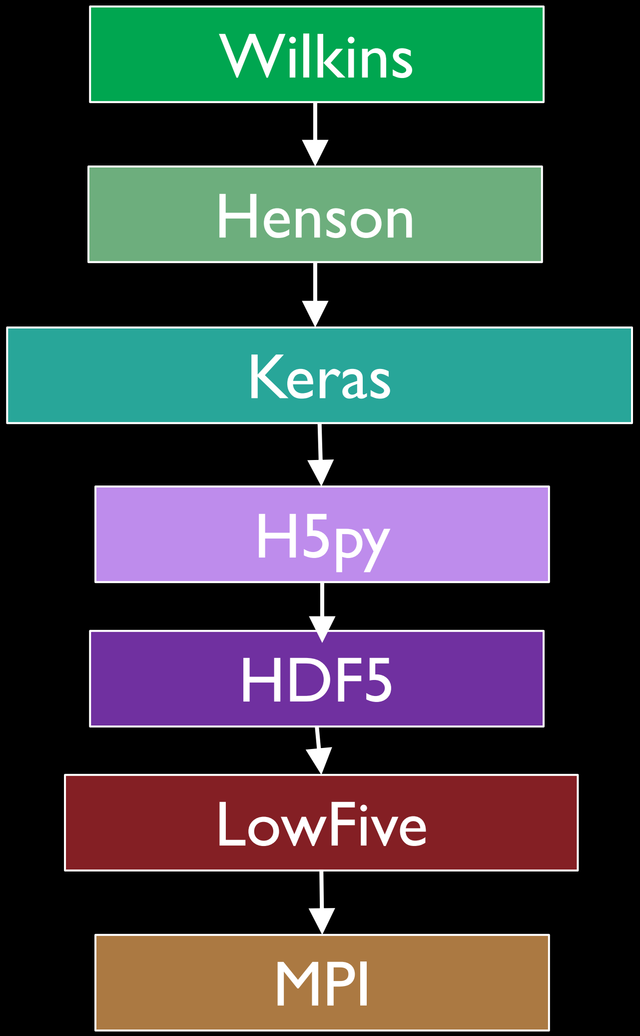
Use Cases and Deeper Software Stacks



Climate modeling software stack using NetCDF data model



Cosmology software stack using AMR data model



AI software stack using tensor data model github.com/diatomic/LowFive

Acknowledgments

Facilities

Argonne Leadership Computing Facility (ALCF)
Argonne Laboratory Computing Resource Center (LCRC)
Oak Ridge Leadership Computing Facility (OLCF)
National Energy Research Scientific Computing Center (NERSC)

Funding

DOE ASCR Research Program
Margaret Lentz

People

Tom Peterka, Dmitriy Morozov, Arnur Nigmetov, Orcun Yildiz, Bogdan Nicolae, Philip Davis

Tom Peterka
tpeterka@mcs.anl.gov
Mathematics and Computer Science Division