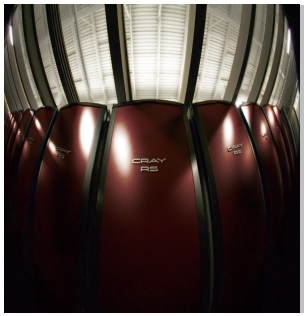


Revisiting Databases for Scale Up Data Management

Jay Lofstead, Margaret Lawson, Ashleigh Ryan, John Mitchell

SAND2017-12103 C, SAND2018-12554 C, SAND2018-13008 C, SAND2019-2166 PE

April 23, 2019

Sandia National Laboratories

Exceptional service in the national interest

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Motivation 1

Sandia National Laboratories

- SIRIUS project
- US DOE Exascale Computing Project Annual

Q: How do we make it easier for scientists to select a data set for deep analysis once it is written?

A: Add user-defined tags and advanced searching capability based on the tags and the data itself.

Margaret Lawson (then Dartmouth undergrad, now UIUC PhD student) took on this challenge

Problems Faced

Sandia National Laboratories

- Simulations with 100s TB per output, run every few minutes
  - Ex. XGC1, Square Kilometer Array Radio Telescope (SKA)
- Storage devices too slow to sift through all output to find "interesting data"
- Scientists have specific data they want to retrieve
  - Ex. "blob" in fusion reactor or a phenomenon in astronomy

EMPRESS' Solution

Sandia National Laboratories

- Allow users to label data and retrieve data based on labels

- Features:
  - Robust, standard per-process metadata
  - User-created metadata that is fully customizable at runtime
  - Programmatic query API to retrieve data contents based on metadata

Previous Solutions

Sandia National Laboratories

- HDF5 and NetCDF – rudimentary attribute capabilities, basic metadata
- ADIOS – per-process metadata

None of these address efficient attribute searching

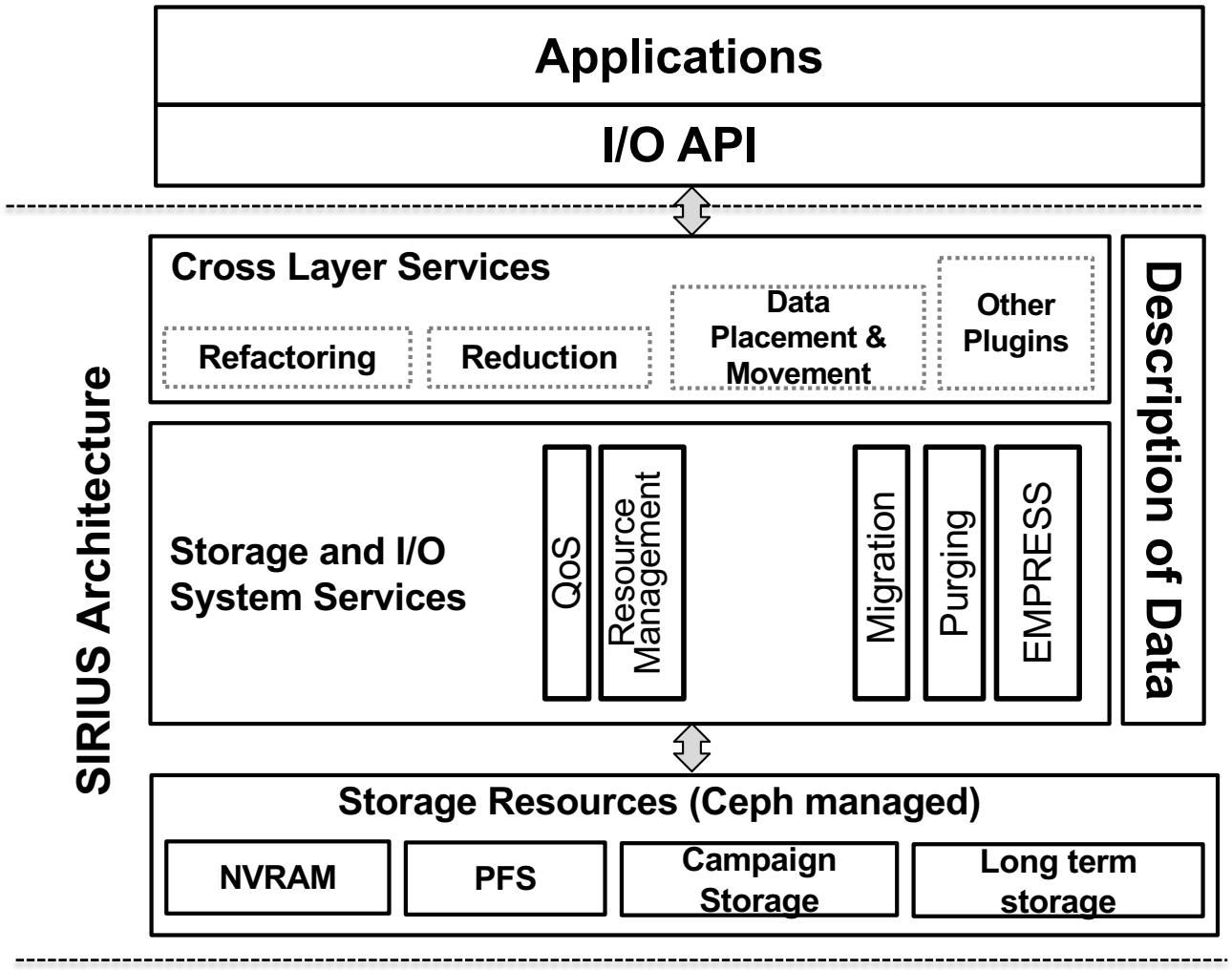
- FastBit – offers data querying based on values, but very limited support for spatial queries and attributes

Why not use a Key-Value Store?

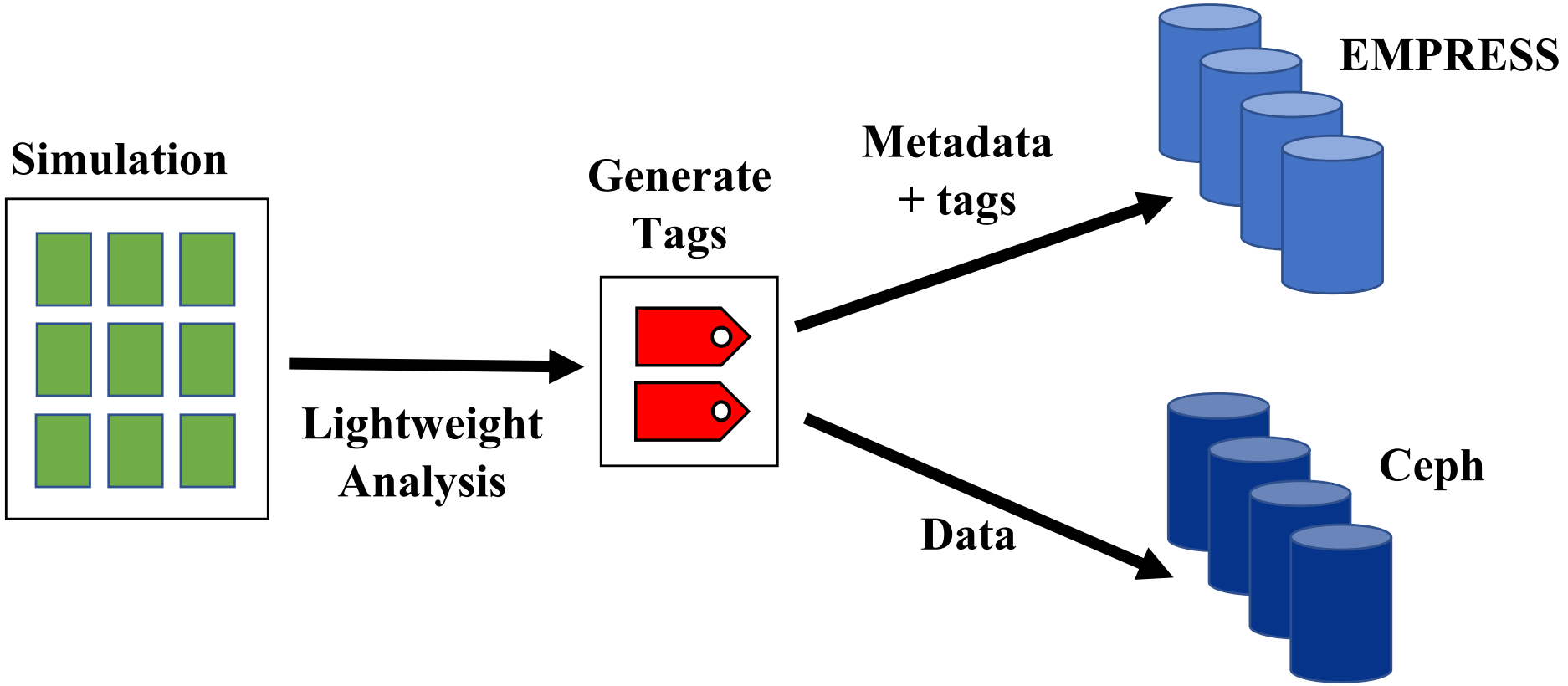
Sandia National Laboratories

- Custom keys can go a long way, but not far enough
- Two Problems:
  - Inexact matches
  - Custom Metadata
- Relational databases with indices are radically faster at searching like this

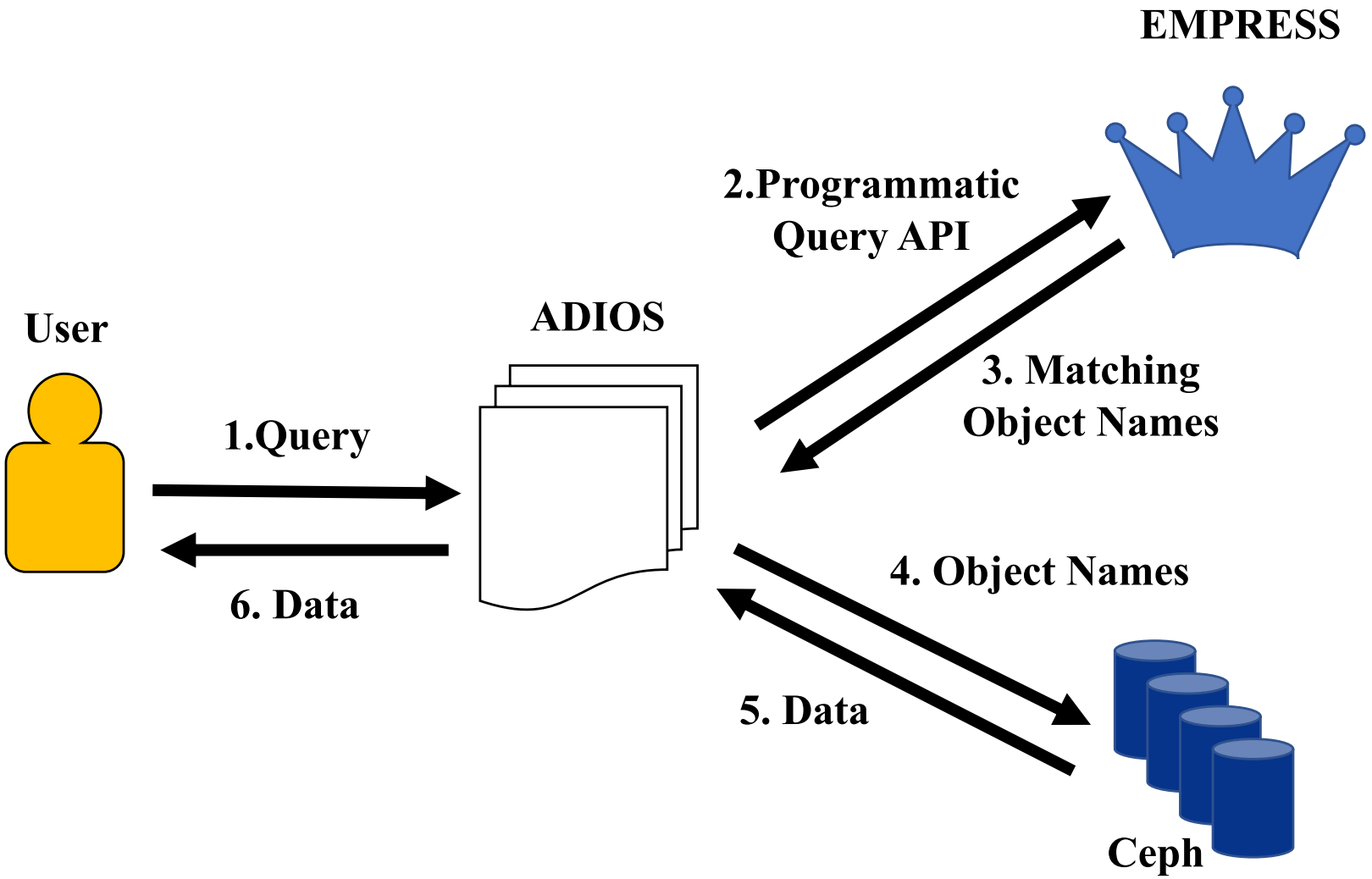
SIRIUS Architecture



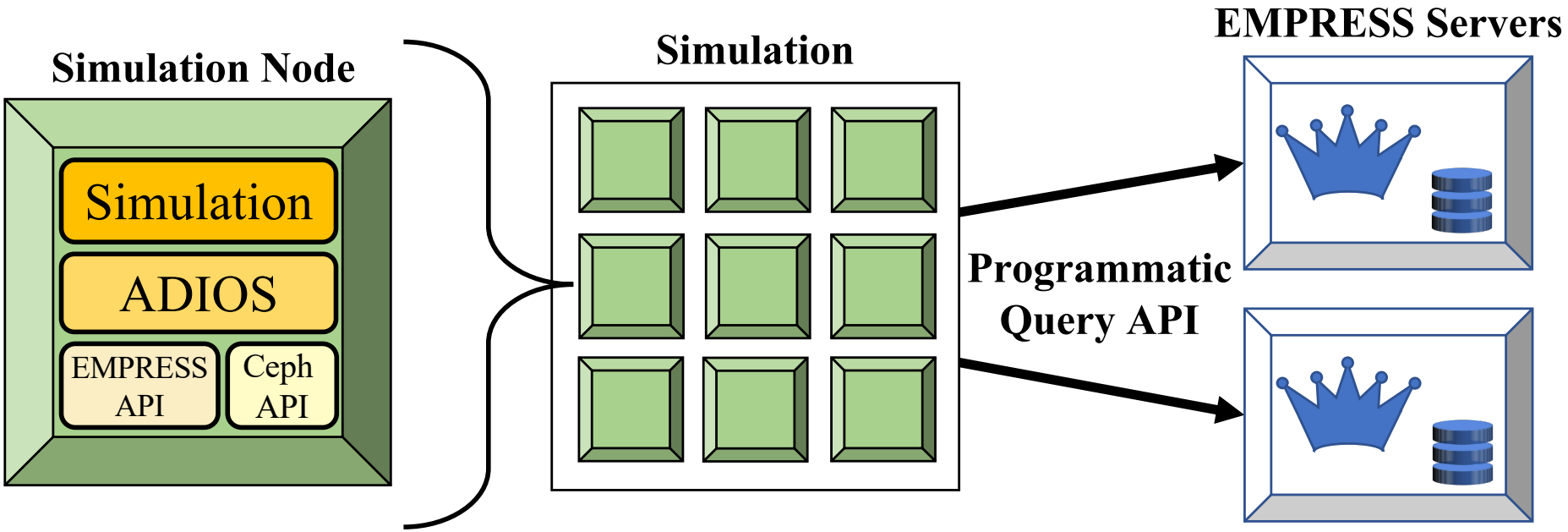
SIRIUS Workflow – Write Process



SIRIUS Workflow – Read Process

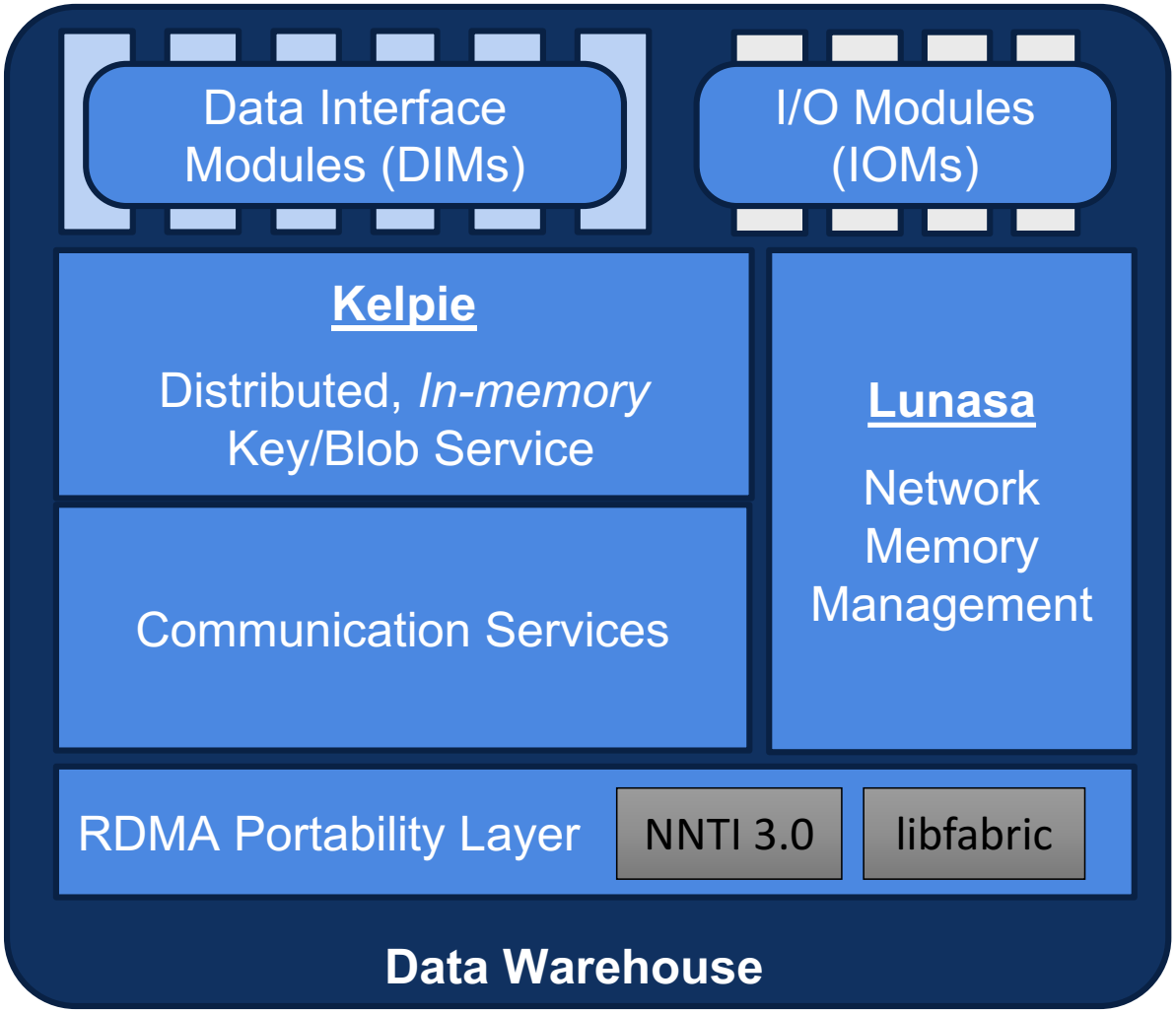


High Level Design



Faodail

Sandia National Laboratories

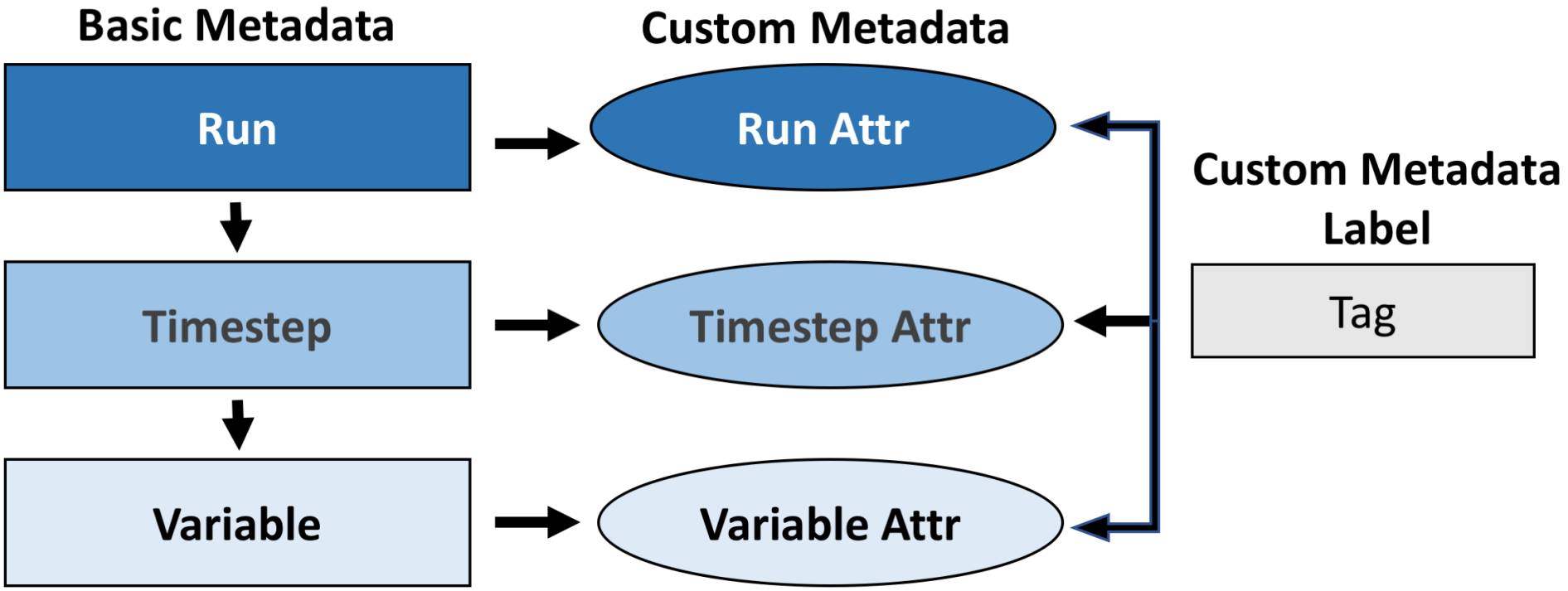


Storage - Tracked Metadata

Sandia National Laboratories

- Dataset information
  - Application, run, and timestep information
- Variable information
  - Catalogs types of data stored for an output operation
- Variable chunk information
  - Subdivision of simulation space associated with a particular variable
- Custom metadata class
  - Metadata category the user adds for a particular dataset
  - Ex. Max
- Custom metadata instance
  - Ex. Flag for chunk or a bounding box spanning chunks

Metadata Model



Atomic Operations

Sandia National Laboratories

- Control data visibility until it is complete and correct
  - Enable better workflow integration
- Low overhead transactions
  - Still atomic, but less rigidly implemented
  - Using D^2T system of Doubly Distributed Transactions

Portability

Sandia National Laboratories

- Need metadata to be independent of the storage layer so it can travel with data should it move to a different storage system

- Storing names can be problematic
  - Using name generator idea explored with UCSC

Testing Goals 1

Sandia National Laboratories

- Scalable?
  - Number of client processes: 1024-2048
- Effect of client to server ratio
  - Ratios tested: 32:1 – 128:1
- Overhead of including a large number of custom metadata items
  - Number of custom metadata classes: 0 or 10
  - On average 2.641 custom metadata instances per chunk

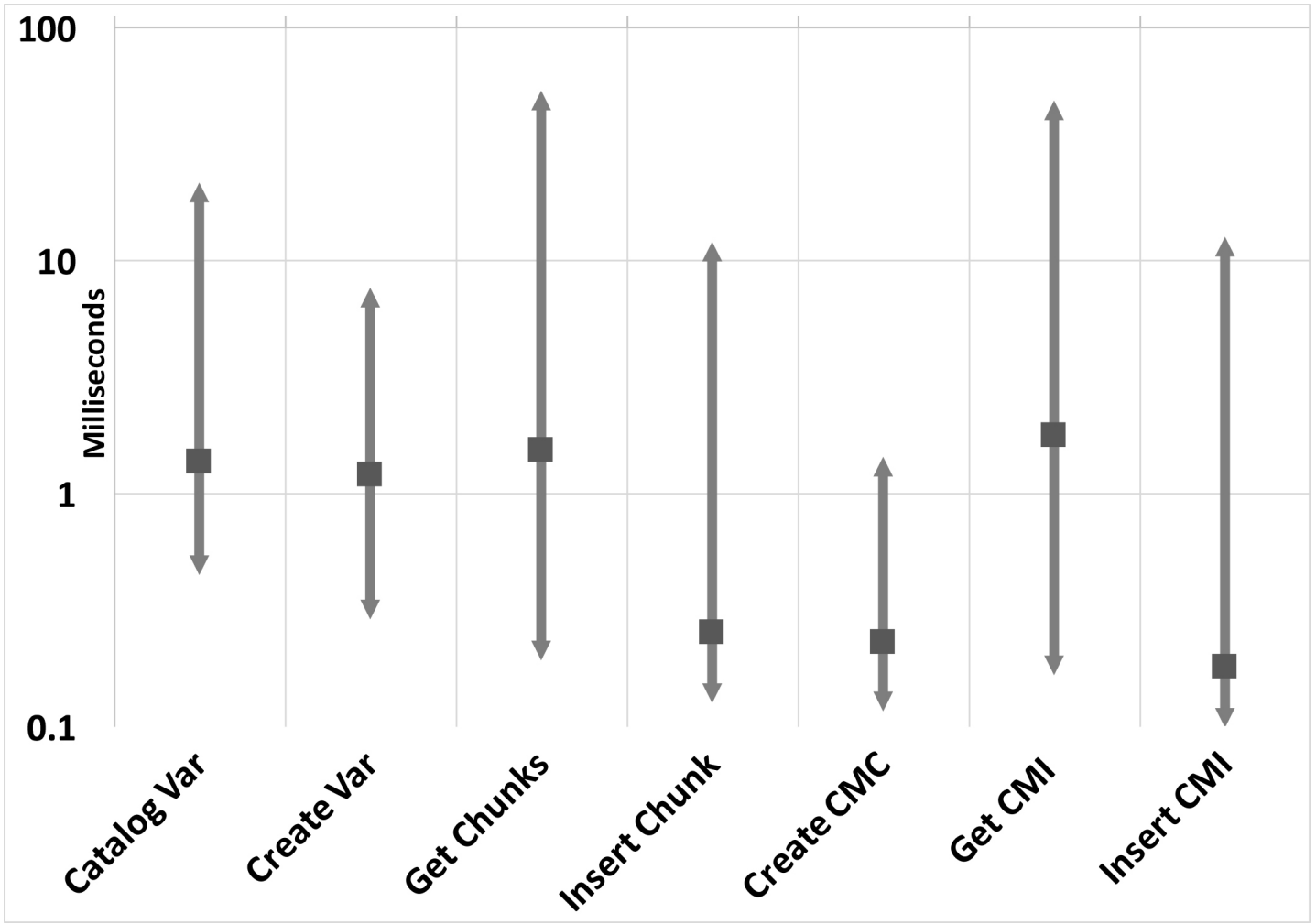
Testing Goals 1 (Continued)

Sandia National Laboratories

- Proof of concept, can EMPRESS efficiently support:
  - Common writing operations
    - 2 datasets written, each with 10 globally distributed 3-D arrays
  - Common reading operations
    - 6 different read patterns that scientists frequently use (Lofstead, et al. “Six Degrees of Scientific Data”)
  - A broad range of custom metadata
    - 10 custom metadata classes including max, flag, bounding box (two 3-D points)
- Scientific validity
  - A minimum of 5 runs per configuration on 3 computing clusters:
    - Serrano (total nodes: 1122)
    - Skybridge (total nodes: 1848)
    - Chama (total nodes: 1232)

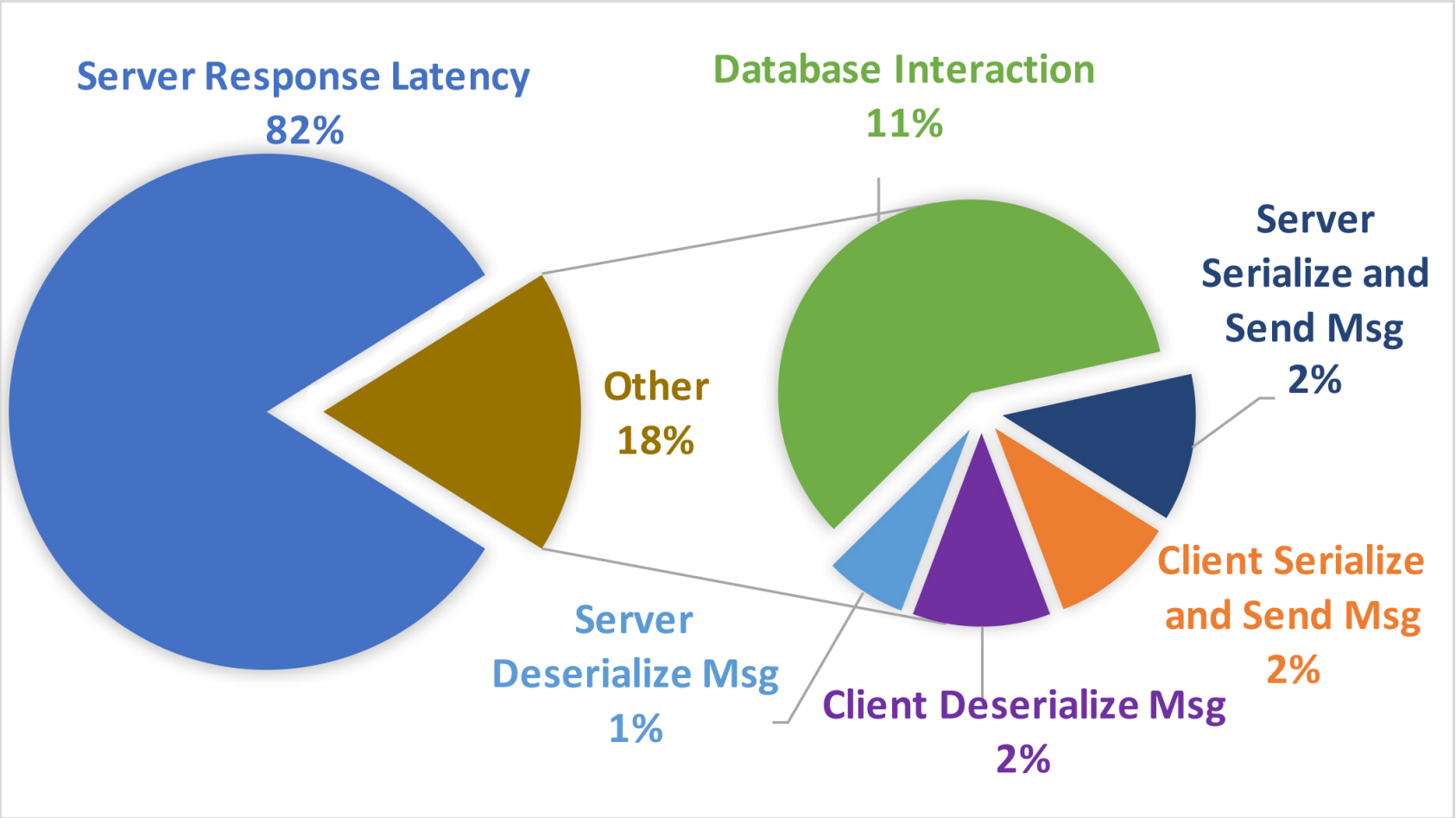
Testing – Query Times

Sandia National Laboratories



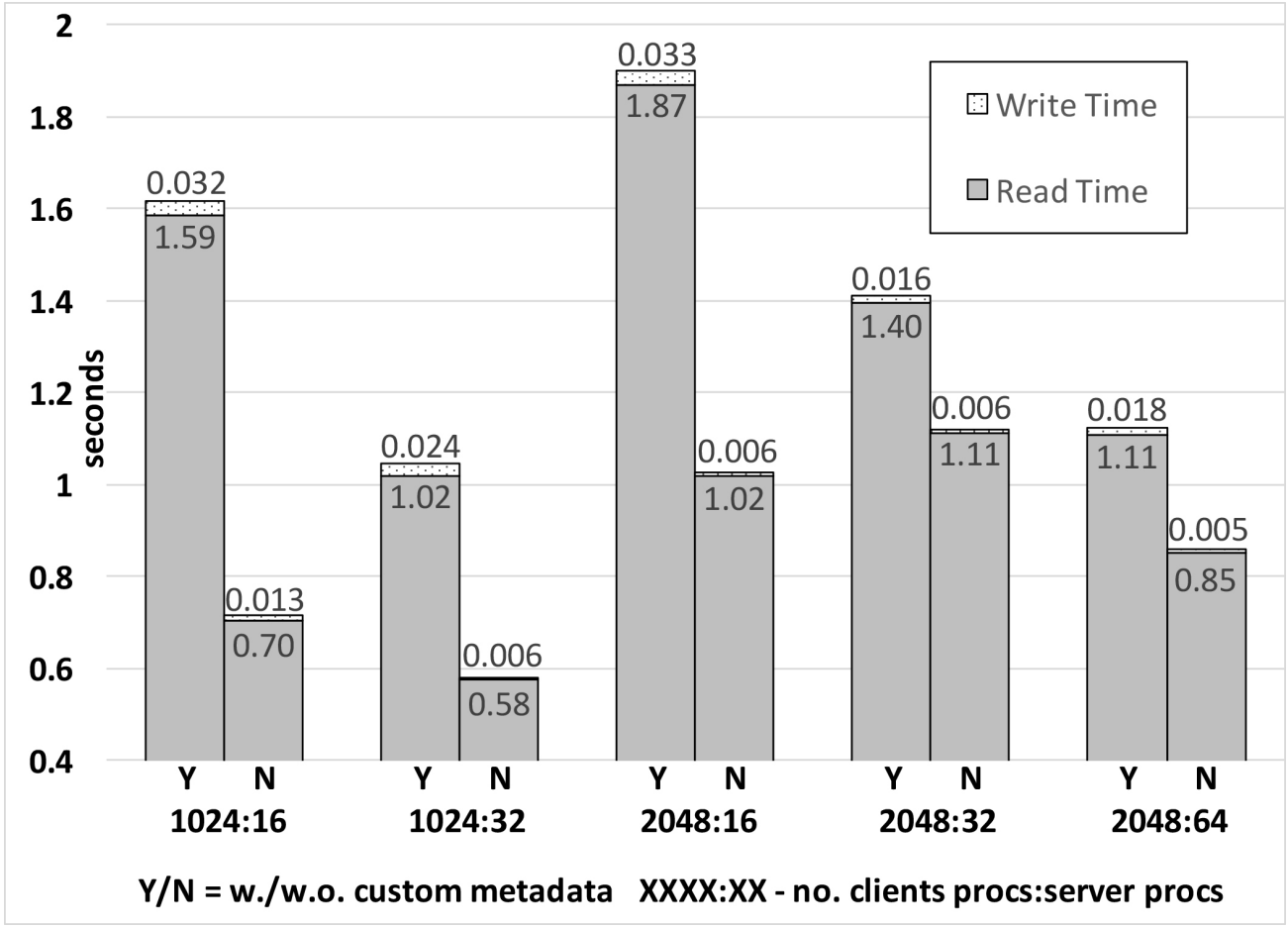
- EMPRESS efficiently supports a wide variety of operations including custom metadata operations

Testing – Chunk Retrieval Time



- Most time is spent waiting for the server to respond
  - Room for improvement in the Faodail infrastructure

Testing – Writing and Reading Time



- Good scalability for fixed client-server ratio
- No significant overhead for adding custom metadata
- Client-server ratio greatly affects performance

Evaluation 2 Compare with HDF5

Sandia National Laboratories

Test Type	# Write Procs	# Read Procs	# Metadata Servers
EMPRESS 2.0 + HDF5	1000	100	1
EMPRESS 2.0 + HDF5	2000	200	2
EMPRESS 2.0 + HDF5	4000	400	4
HDF5	1000	100	N/A
HDF5	2000	200	N/A
HDF5	4000	400	N/A

Evaluation: Write Process

Sandia National Laboratories

- Run structure:
  - One application run, three timesteps, ten 3-D variables
- Data
  - Each process writes 0.4GB of data (10% of RAM) per timestep
- Custom metadata:
  - 10 different tags of varying frequency
  - On average, each process writes 26 attributes per timestep (2.6 per variable)

Evaluation: Read Process

Sandia National Laboratories

- 6 common read patterns are performed including
  - An entire variable
  - A plane and partial plane in each dimension
  - A 3-D subspace
- Custom metadata is used to identify potential features of interest and the associated data is read in

(using Lofstead, et al., “Six Degrees of Scientific Data” patterns)

Evaluation: Writing

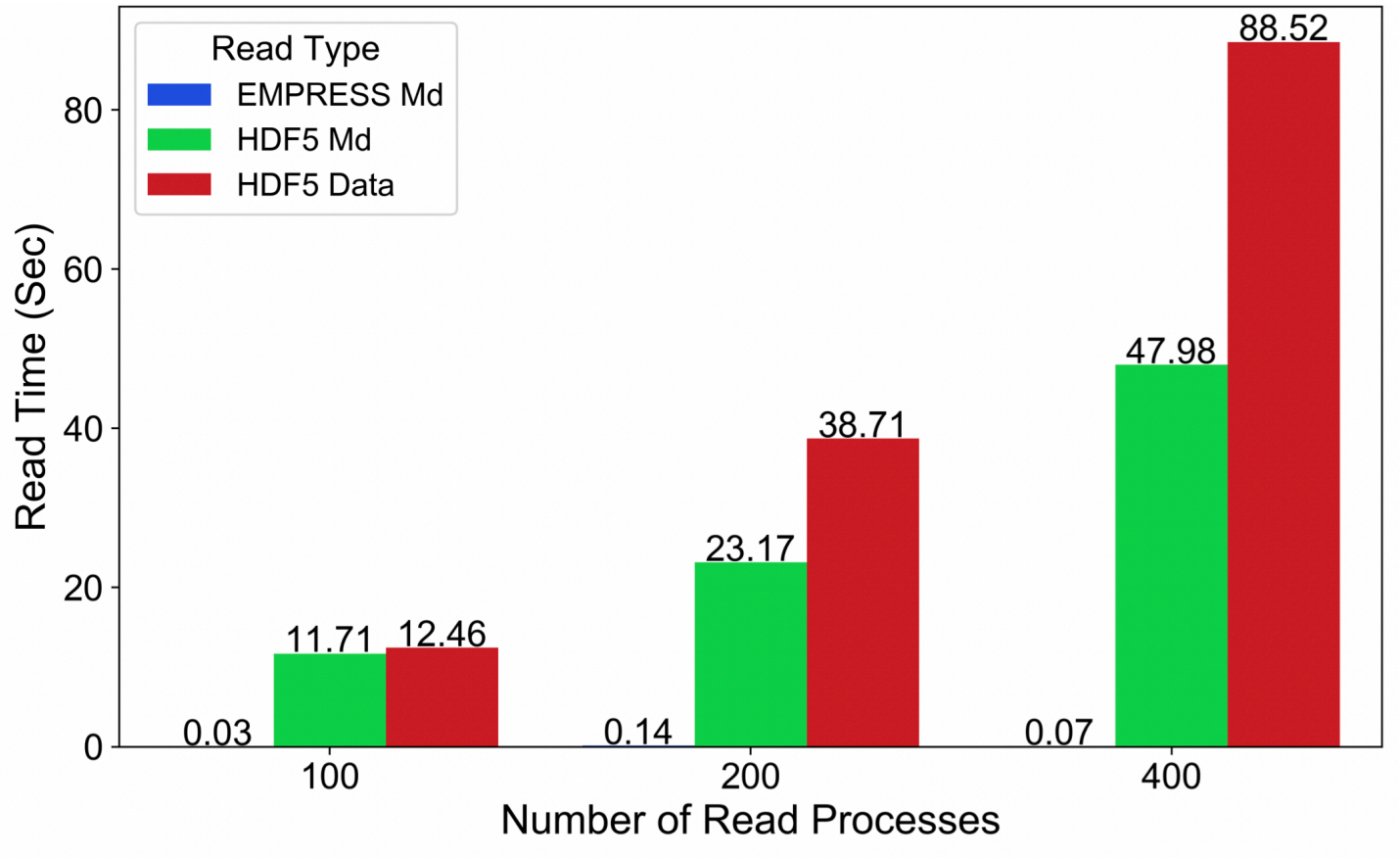
Sandia National Laboratories

# Write Procs	Data Write	EMPR Md Write	EMPR Md Overhead	HDF5 Md Write	HDF5 Md Overhead
1000	1753s	1.53s	0.09%	0.66s	0.04%
2000	3852s	1.65s	0.04%	1.80s	0.05%
4000	7406s	1.56s	0.02%	3.22s	0.04%

- Both can do efficient metadata writes at the evaluated scales
  - But EMPRESS can scale out to achieve constant performance

Evaluation: Read

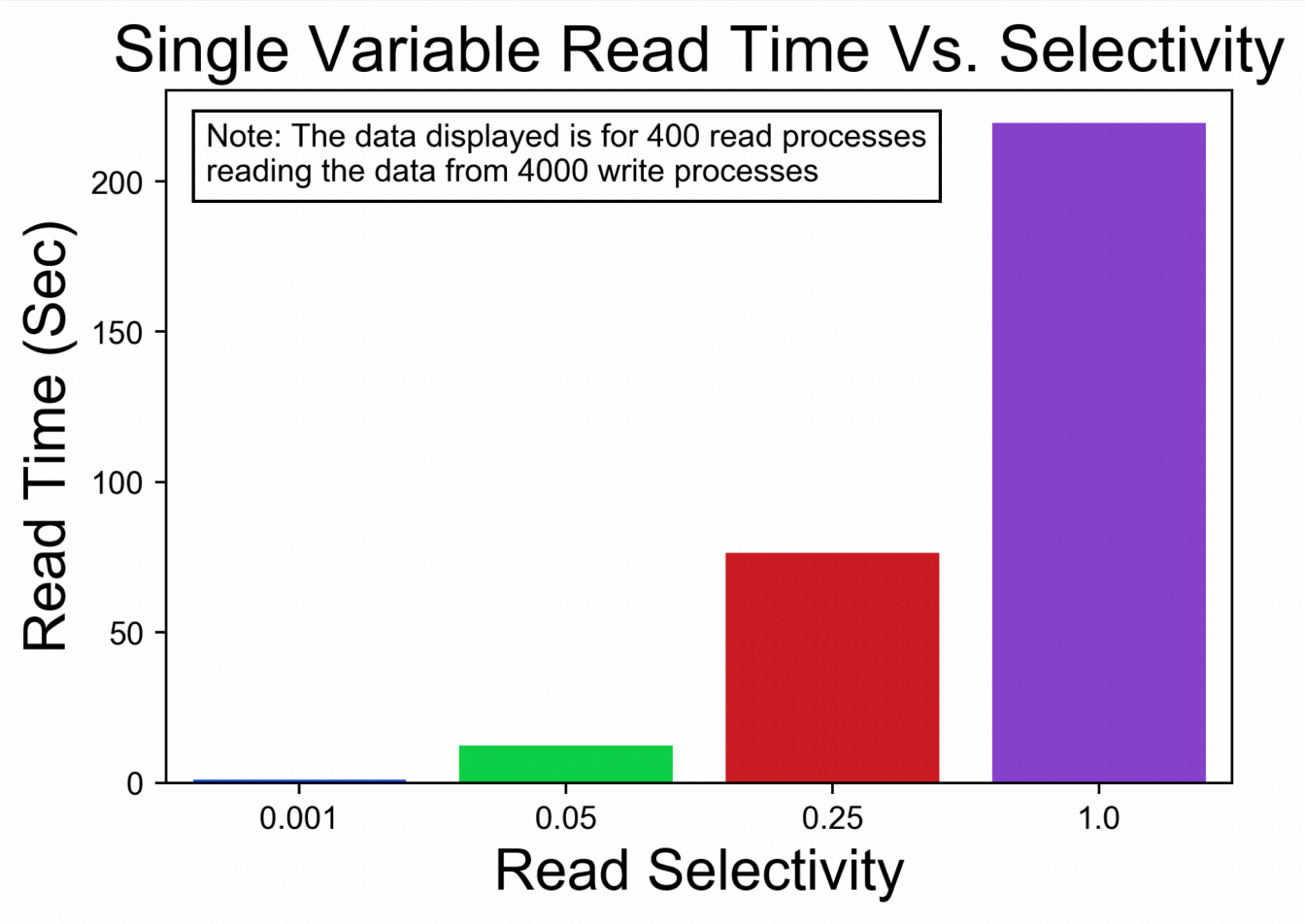
- HDF5 takes almost as long to do the metadata query as it does to read the data



Evaluation: Accelerating Data Reads

Sandia National Laboratories

- EMPRESS can significantly accelerate data reads by limiting the scope to data of interest



Relational is Nice, But Limited

Sandia National Laboratories

- Columnar databases allow adding arbitrary additional columns with typed data you can query against (I think). Will this work with performance?

How did I get here?

Sandia National Laboratories

- Fall 2018 intern Ashleigh Ryan (GT) comes to the rescue
  - Agreed to explore this seemingly simple question
  - Did a tremendous job exploring the deadly terrain that is NoSQL databases-–particularly with the intersection of an HPC cluster on a restricted network.
- Primary database choice: Cassandra
  - Secondary: Hbase
- Other NoSQL databases are either key-value, document stores, or otherwise can’t work
  - Software requirements that can’t be met solely as an end user, it must be run as an OS service, or other non-starter requirements.

What else is comparable?

Sandia National Laboratories

- SoMeta
- MDHIM
- HDF5, ADIOS, PnetCDF, NetCDF

None of these address the desired flexibility and performance for managing metadata, and in particular, data tagging and searching.

What NoSQL databases

Sandia National Laboratories

- We (Ashleigh) investigated several options

Database	FOS	Columnar/ KeyValue	Fault Tolerance	General Language
Apache Accumulo	✓	Columnar	Write Ahead + HDFS	Java
Cassandra	✓	Columnar	Replication	Cassandra Query Language
Druid	✓	Columnar	Replication + HDFS	JSON over HTTP
Dynamo	✗	KV	S3	AWS API
Hbase	✓	Columnar	HDFS	Java API
Vertica	✗	Columnar	Varies	SQL

Let's Battle it Out

Sandia National Laboratories

- Four rounds of problems that have to be solved
  - What has the right features to be worth testing
  - What is it going to take to get it working at all
  - Can we make our queries work with any performance
  - Battle scars and lessons for our next battle against scale out computing tools

Round 1. Fight!

Sandia National Laboratories

- NoSQL assumption: running on a cluster with full Internet access from every node with no code deployment restrictions and full control over the storage infrastructure.
  - We fail this test miserably.

- HDFS requirement is a hard “no”
  - We have our storage infrastructure defined already with burst buffer and parallel storage deployed
- Full Internet access from the compute nodes is a hard “no”
  - Getting TCP/IP to work properly is hard enough
- Not getting full source code is a soft “no”
  - We’ll pay for support, but we need source to get it to work on our messy hardware and software.

Round 1. Result

Sandia National Laboratories

- Two potentially viable choices
  - Cassandra – no HDFS and a C-based query API available
  - Hbase – Java-based API and HDFS, but might work
- Chose Cassandra for the potential and active user community for support

Round 2. Fight!

Sandia National Laboratories

- Now to get Cassandra to work, just on a desktop on the restricted network
- Problem 1: What do you mean I need to install a bunch of packages from the Internet?
- Problem 2: What do you mean I have to run it either as a service or as a separate process I interact with?
- Problem 3: What do you mean I have replication? Can't I just do a single storage node?

Round 2. Problem 1 Result

Sandia National Laboratories

- Dependencies are a nightmare
  - This is not unique to Cassandra
- Let's spend a couple of weeks just generating a list of the dependencies and figuring out how to build them from source, in order, so I can build Cassandra.
- Ultimately it worked out, but it is a pain.

Other systems, like Ceph, can be dramatically worse for their list of dependencies. Open source packages are nice, but dozens to hundreds of dependencies is a recipe for disaster.

Round 2. Problem 2 Result

Sandia National Laboratories

- Cassandra wants to be your storage interface and therefore look like a storage service
- How do you put a TCP/IP-based service on an IB network when TCP/IP isn't the native protocol?
  - Yes, it can work, but configuration isn't straightforward
- You can run it in a window and access that service from another, but it isn't a general solution.
- That C-based API from DataStax works great!

Round 2. Problem 3 Result

Sandia National Laboratories

- Cassandra REALLY wants to use replication—even if you don't
- Running on a single node was easy enough
- Deploying to the cluster and suddenly it wouldn't run anymore
  - Even though the configuration set the replica count to '1'

Result: Even though the configuration said 1 replica, it had to be explicitly set in code to '1' again to make it work properly. The error messages were baffling making debugging a series of trial and error.

Round 3. Fight!

Sandia National Laboratories

- Now that we have something working-ish, how about those fancy queries we want to do?
- Problem 1: Ok, adding a column is straightforward. What do you mean querying has limited functionality against that new column?
- Problem 2: Wow, that is super limited, is there a way to avoid doing table scans regularly?

Round 3. Problem 1 Result

Sandia National Laboratories

- Cassandra's core setup allows only special columns to have an index on them. These are the only ones on which a query can execute efficiently.

- Result: This is completely against what we were trying to do. Is anything else (Hbase?) better?
  - Nope. NoSQL is REALLY limited to get the performance
  - Lots of Internet people say when asked about this, "why would you do that? That isn't how this is supposed to work."
  - Just do table scans for nearly all queries we want to perform.

This is a near fatal blow to the idea. Fortunately, there are still options.

Round 3. Problem 2 Result

Sandia National Laboratories

- Table scans. Really? What can be done?

- Enter: Map-Reduce
- It is possible to embed Spark into Cassandra
  - Wait, that is another set of software dependencies and assorted ugliness.
  - But it can work!
  - Should we switch to Hbase instead because of the built-in Hadoop?

Spark can cache Map-Reduce queries to mimic what we want, but then it is memory bound—and won't tell you when it throws away cached results. It just runs slower when it is out of memory.

Round 4. Fight!

Sandia National Laboratories

- Battle scars and lessons learned
- Problem 1: Can this work at all with performance
- Problem 2: Can this software stack co-exist on scale up platforms
- Problem 3: Is there a way to get what we want?

Round 4. Problem 1 Result

Sandia National Laboratories

- Is there a performant option?
- Bottom line: NO
- Issue: NoSQL is set to query against a limited set of pre-known parameters with the additional columns coming along for the ride.
  - Corollary: Doing a query for the presence or absence of a value for a column is impossible (except for using Map-Reduce).

Rigid relational models are just a better fit for performance, but it is hard to be the right kind of flexible.

Round 4. Problem 2 Result

Sandia National Laboratories

- What about installing this software?
- Bottom Line: Maybe to yes.
- Issue: Lots of dependencies including a heavy assumption of TCP/IP generates terrible network performance and a big software footprint. Getting machine admins to agree to install the behemoth will be difficult.
- This is solvable, but will take a lot of effort. The constant talking to the Internet requires pre-caching things or intercepting version checking. Not impossible, but annoying.

Round 4. Problem 3 Result

Sandia National Laboratories

- Can we make this work?
- Bottom Line: Not using any existing software.
- Issue: relational databases are rigid for performance scalability with arbitrary queries against a fixed schema.
- Issue: NoSQL databases are rigid for performance against a small set of know values to then check if some other attributes have been added. Looking based on attribute isn't a supported model.

Decision

Sandia National Laboratories

- NoSQL has interesting ideas, but we can't use the tools as is on restricted clusters
  - There are caveats that make it possible, but an end-user will find it difficult to impossible making a general library infeasible.
- We have to start over if we want this to work. There is no payoff for industry (or it would be done already) and it is a high-risk prospect for research (cheaper, easier, and more sure to face the relational complexity).
  - HDF5 is the de-facto standard with NetCDF and ADIOS filling in the gaps. Replacing any of these will require solid buy-in from a community (Klasky, as a physicist promoting the tool, was the key for ADIOS to make that jump).
  - But that doesn't mean I won't try ☺
  - I have a summer 2019 student to tilt at this windmill.

Motivation 2

Sandia National Laboratories

- SPPARKS Kinetic Monte Carlo simulation engine welding example
- Problem: vast majority of simulation domain static between output steps. What is a more efficient approach to data management than text files or traditional IO libraries?
- Solution: Stitch

Approach

Sandia National Laboratories

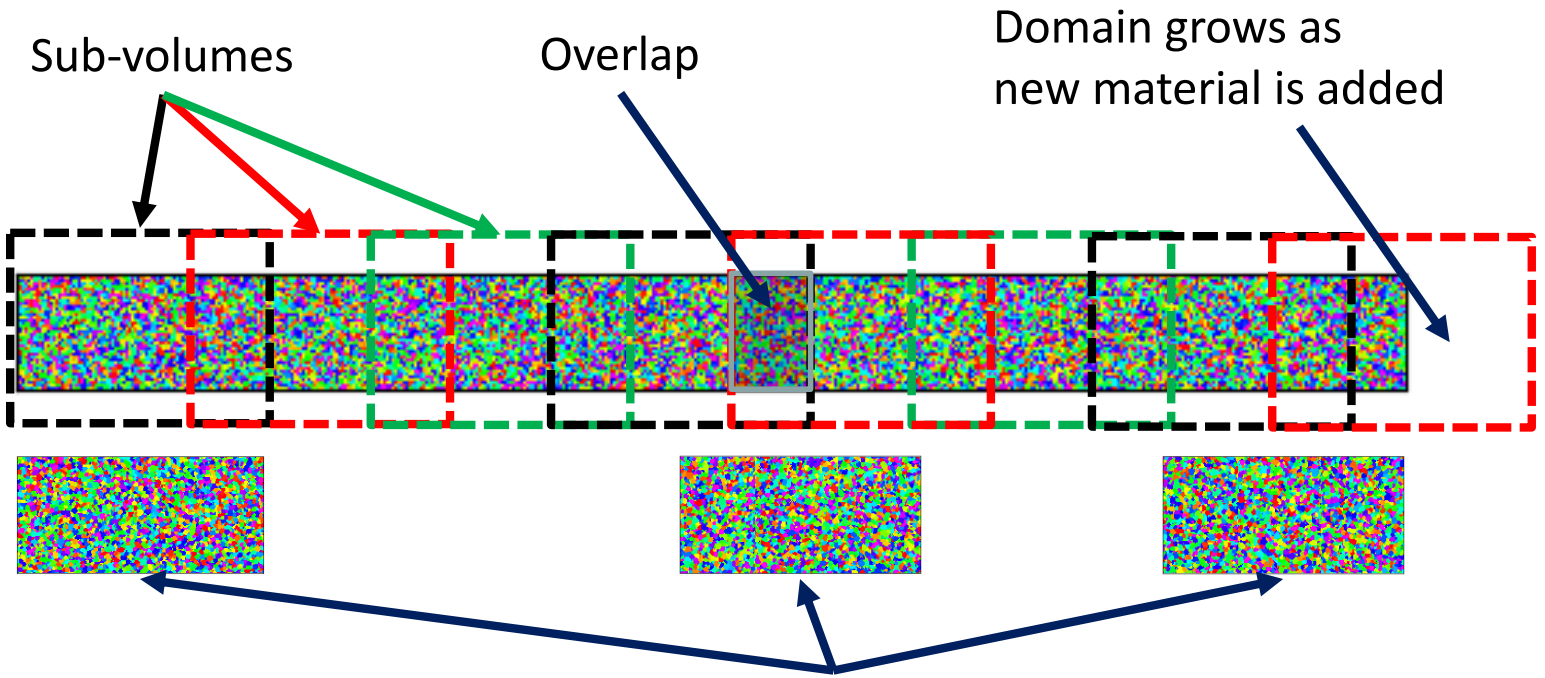
- Lazy
  - Only track what has been seen so far (i.e., we don't care about the size of the simulation domain)
- Minimal
  - Only write was has changed since last output
- Eventually Consistent
  - Rely on the output to eventually "make sense"

- Reading specifies an arbitrary region and a time; Stitch assembles ('stitches') the region state together from various pieces using the newest for every point

Illustrative Example (SPPARKS)

Sandia National Laboratories

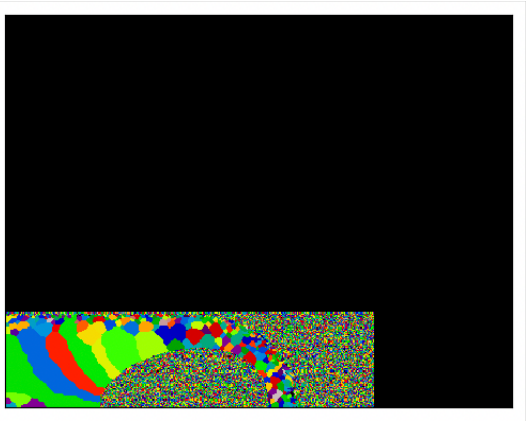
Grain growth across a large domain is simulated using a series of smaller overlapping sub-volumes.

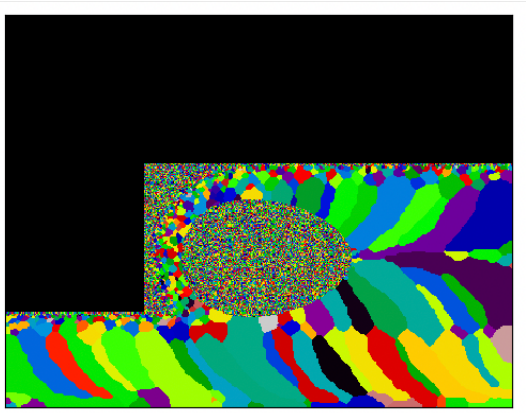


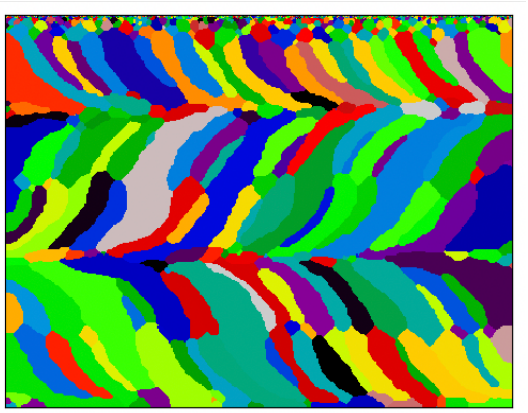
Post-process, visualize and analyze on arbitrary sub-volumes and arbitrary times

2-D Full Domain Example

Sandia National Laboratories



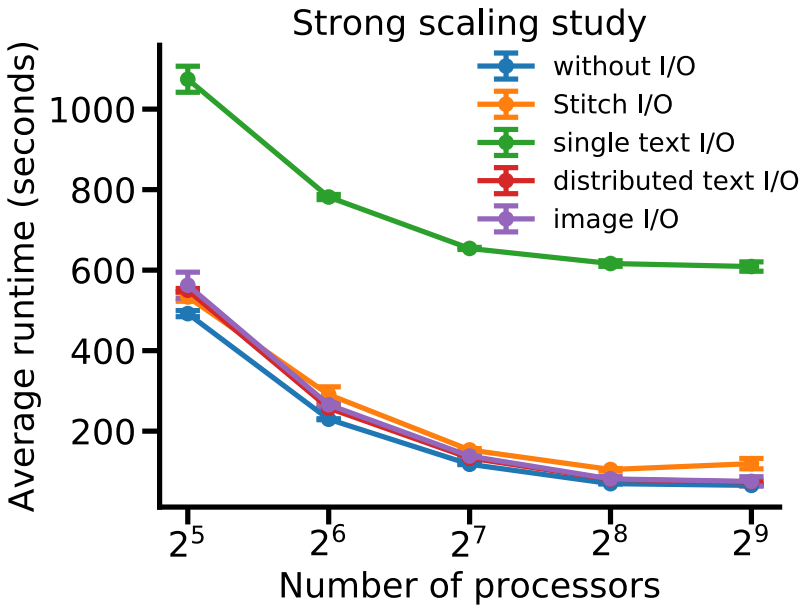


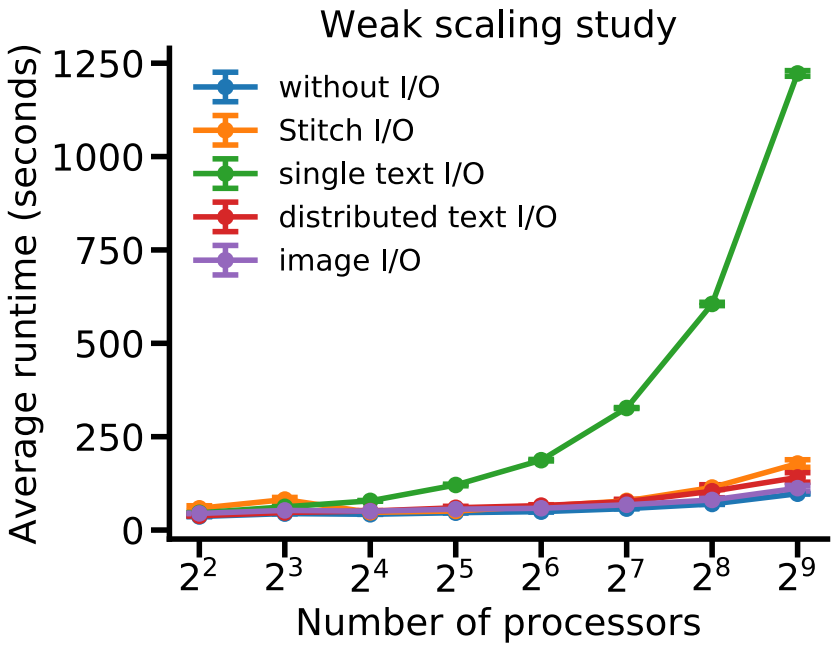


Strong Scaling Study

Sandia National Laboratories

- Database approach scales well compared to others





File Size Comparisons

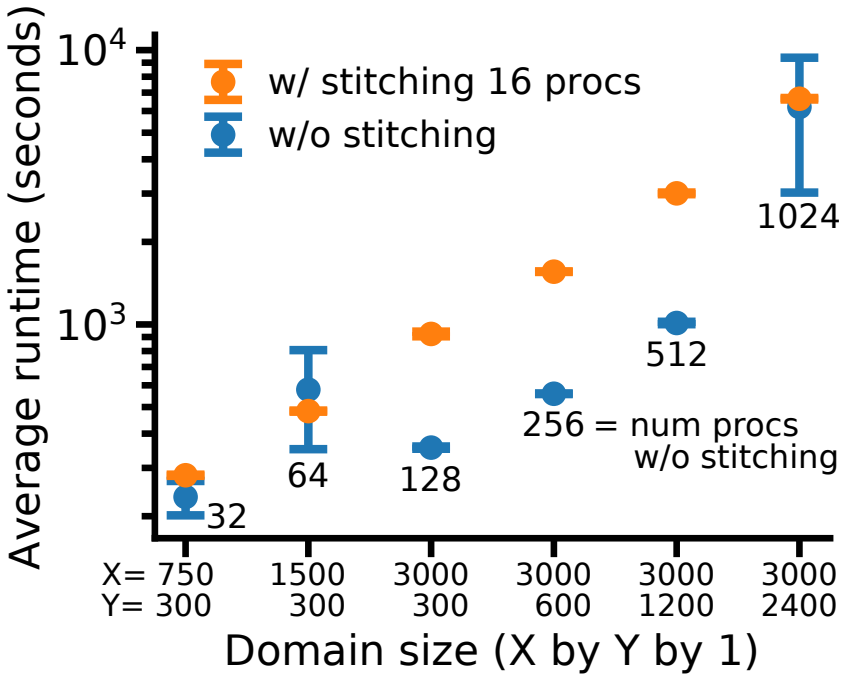
Sandia National Laboratories

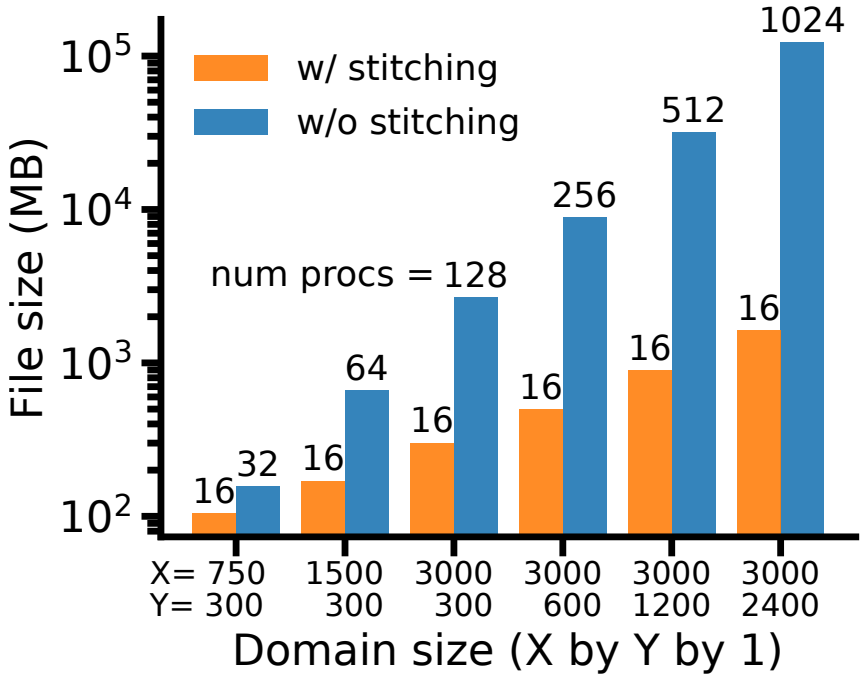
Num procs	Domain size	Num sites	Stitch file size	Text file size
4	$150 \times 150 \times 88$	1980000	23 MB	59 MB
8	$216 \times 216 \times 88$	4105728	47 MB	137 MB
16	$300 \times 300 \times 88$	7920000	91 MB	277 MB
32	$426 \times 426 \times 88$	15969888	186 MB	608 MB
64	$600 \times 600 \times 88$	31680000	363 MB	1.26 GB
128	$860 \times 860 \times 88$	65084800	753 MB	2.67 GB
256	$1200 \times 1200 \times 88$	126720000	1.42 GB	5.53 GB
512	$1700 \times 1700 \times 88$	254320000	2.91 GB	12.1 GB

Table 3.1: Parameters used in weak scaling performance study. The number of sites per process was kept constant at approximately 5×10^5 sites/process. The *stitch* file size was 3 to 4 times smaller than the size of the corresponding text file.

Runtime with Stitching

- With the process count reduction, things look really good





Benefits and Challenges

Sandia National Laboratories

- Move from 1024 process to 16
- Move data size to 1/64th size per output
- Wall clock time the same or slightly smaller (less output time)

- SQLite has issues that prevent full scalability
  - BerkeleyDB may help, but probably not enough

- LOTS more features we can talk about offline
- Open Source (LGPL) release approved (look at my github [Jay Lofstead (gflofst)] once paper accepted somewhere)
- Full paper coming H1 2019 (on track for Cluster)

Questions?

Sandia National Laboratories

gflofst@sandia.gov