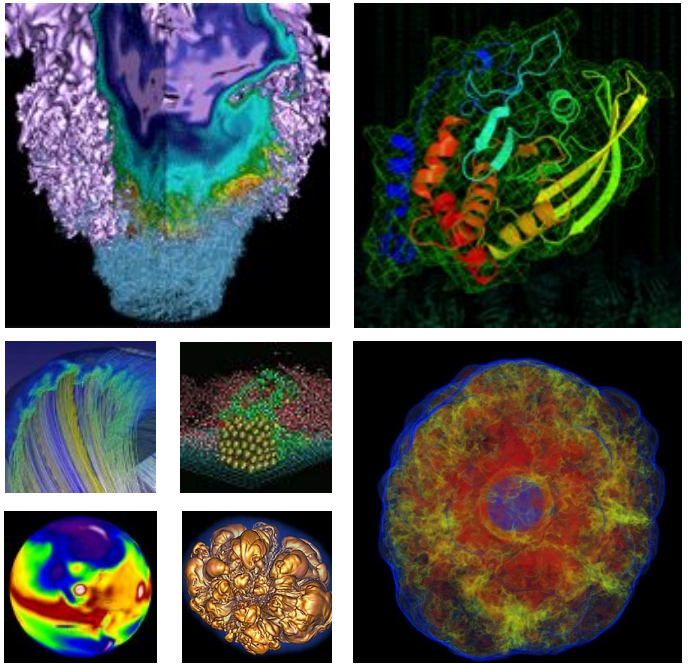
Parallel IO



Jialin Liu
Data Analytics & Service Group
NERSC

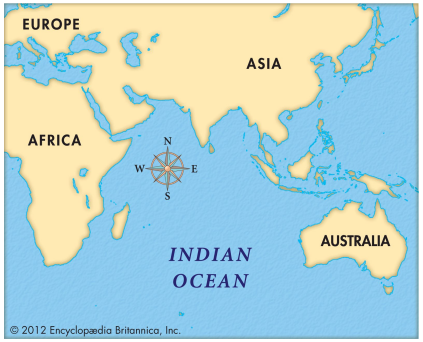
March 21, 2018

New User Training

Outline

- IO and Common IO Issues
- HPC I/O Stack
- Performance vs. Productivity
- Case Study: Athena's Mysterious IO

What is IO?



github.io – Rank: 521
codepen.io – Rank: 3271
soup.io – Rank: 3910
filecloud.io – Rank: 7022
intercom.io – Rank: 14449
binbox.io – Rank: 15369
laravel.io – Rank: 16374
redis.io – Rank: 16418
pen.io – Rank: 17889
put.io – Rank: 21159
icomoon.io – Rank: 21373
imm.io – Rank: 23274
purecss.io – Rank: 23484
media.io – Rank: 27293
ubiquity.io – Rank: 27481
emmet.io – Rank: 28681
galleria.io – Rank: 30307
c9.io – Rank: 30347
torquemag.io – Rank: 33209
gamechanger.io – Rank: 34510
plan.io – Rank: 37505
brackets.io – Rank: 37508
filepicker.io – Rank: 39491
kraken.io – Rank: 41207
sidebar.io – Rank: 42285
dashboard.io – Rank: 46205

Why do I Care about IO

- Bandwidth
  - "The peak bandwidth is 700 GB/s, why I could only get 1% of that?"
- Scalability
  - "I have used more IO processes, why the performance is not scalable?"
- Metadata
  - "File closing is so slow in my test…"
- Pain of Productivity
  - "I like to use Python/Spark/Tensorflow, but the I/O seems slow"

Complex HPC I/O Stack

Productive Interface builds a thin layer on top of existing high performance I/O library for productive big data analytics

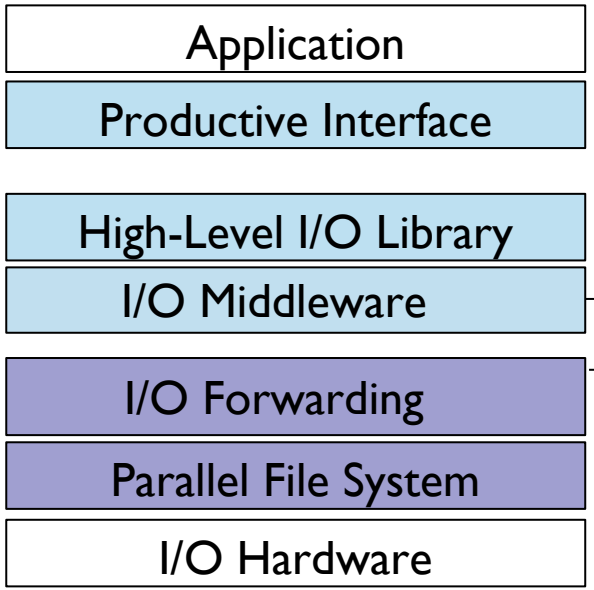
H5py, H5Spark, Julia, TF, Fits

High Level I/O Libraries map application abstractions onto storage abstractions and provide data portability.

HDF5, Parallel netCDF, ADIOS

Parallel file system maintains logical file model and provides efficient access to data.

PVFS, PanFS, GPFS, Lustre



I/O Middleware organizes accesses from many processes, especially those using collective I/O.

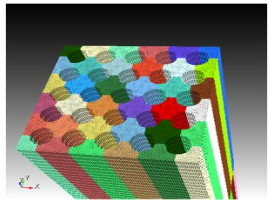
MPI-IO, GLEAN, PLFS

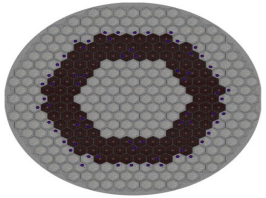
I/O Forwarding transforms I/O from many clients into fewer, larger request; reduces lock contention; and bridges between the HPC system and external storage.

IBM ciod, IOFSL, Cray DVS, Cray Datawarp

Complex Scientific Data Representation

- Applications use advanced data models to fit the problem at hand
  - Multidimensional typed arrays, images composed of scan lines, …
  - Headers, attributes on data





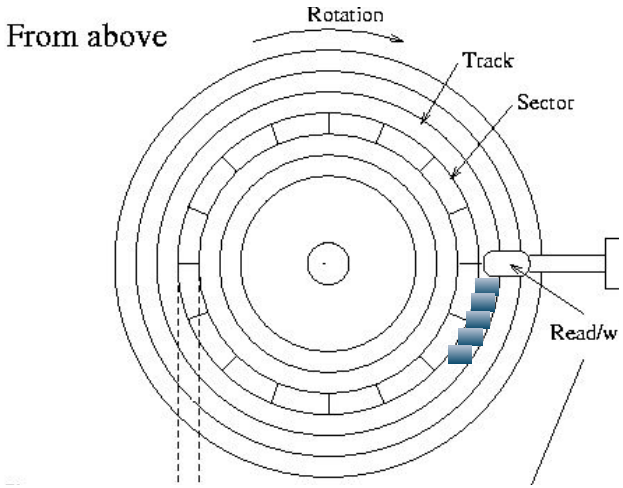
Model complexity:
Spectral element mesh (top) for thermal hydraulics computation coupled with finite element mesh (bottom) for neutronics calculation.

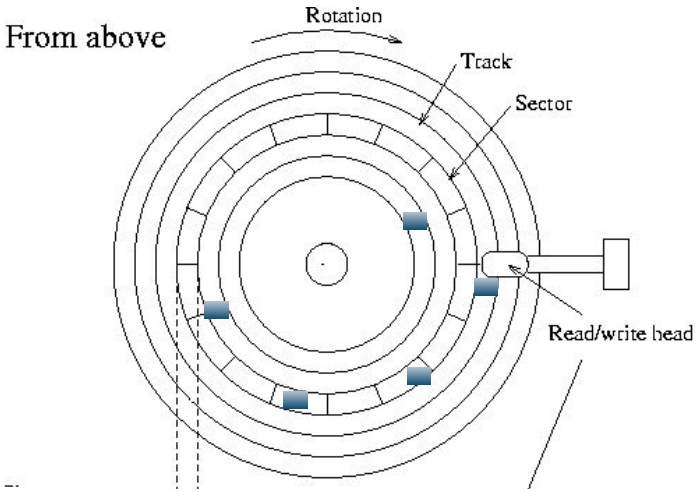
Scale complexity:
Spatial range from the reactor core in meters to fuel pellets in millimeters.

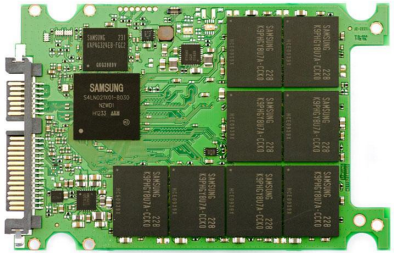
Images from T. Tautges (ANL) (upper left), M. Smith (ANL) (lower left), and K. Smith (MIT) (right).

The Secret of Slow IO

Application
Productive Interface
High-Level I/O Library
I/O Middleware
I/O Forwarding
Parallel File System
I/O Hardware

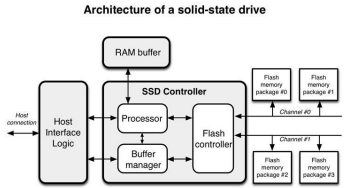






Contiguous IO
- read time, 0.1 ms

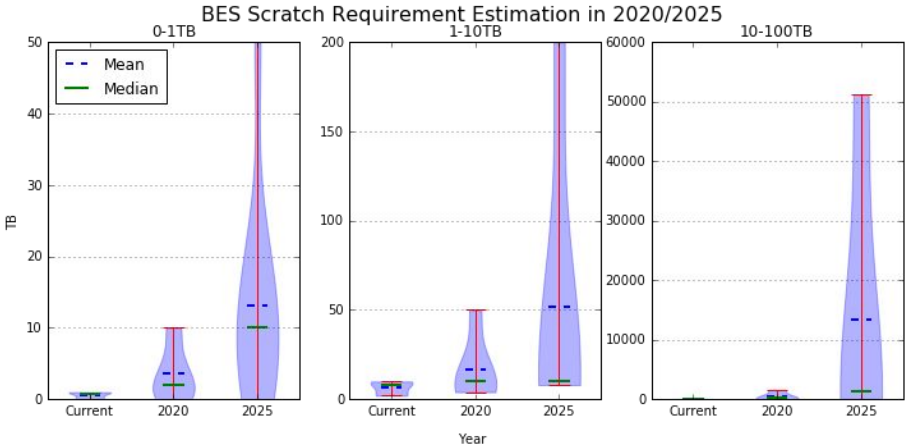
Noncontiguous IO
- seek time, 4ms
- rotation time, 3ms
- read time, 0.1 ms



SSD: No moving parts

I/O Challenges in 2020/2025

- Scientific applications/simulations generate massive quantities of data.
  - Example, BES: Basic Energy Science, Requirement Review, 2015
  - 19 projects review
  - Example projects: Quantum Materials, Soft Matters, Combustion,



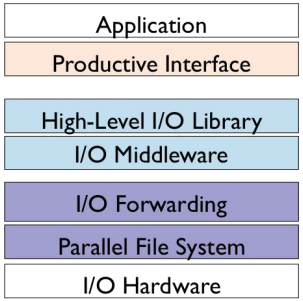
Current (TB)	2020/Current	2025/Current
0-1	7.5	15
1-10	2	4
10-100	3.5	22.5

Average Increasing Ratio

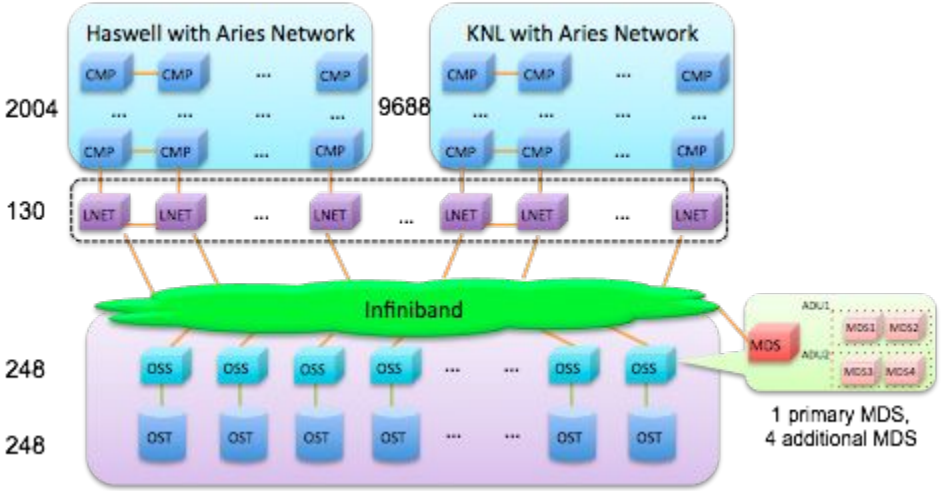
Storage 2020:
http://www.nersc.gov/news-publications/nersc-news/nersc-center-news/2017/new-storage-2020-report-outlines-future-hpc-storage-vision/

Parallel File System

- Store application data persistently
  - Usually extremely large datasets that can't fit in memory
- Provide global shared-namespace (files, directories)
- Designed for parallelism
  - Concurrent (often coordinated) access from many clients
- Designed for high performance
  - Operate over high speed networks (IB, Myrinet, Portals)
  - Optimized I/O path for maximum bandwidth
- Examples
  - Lustre: Most leadership supercomputers have deployed Lustre
  - PVFS-> OrangeFS
  - GPFS-> IBM Spectrum Scale, Commercial & HPC



Parallel File System: Cori FS

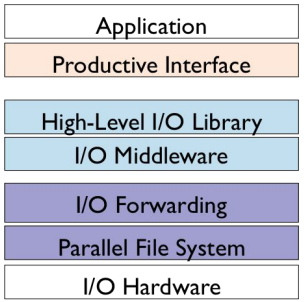


[1] Lustre Striping Recommendation on Cori:
http://www.nersc.gov/users/storage-and-file-systems/i-o-resources-for-scientific-applications/optimizing-io-performance-for-lustre/

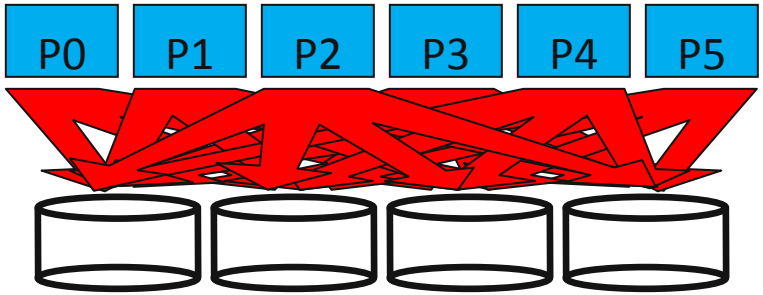
stripe_large .
lfs getstripe .

I/O Middleware

- Why additional I/O Software?
  - Additional I/O software provides improved performance and usability over directly accessing the parallel file system.
  - Reduces or (ideally) eliminates need for optimization in application codes.
- MPI-IO
  - I/O interface specification for use in MPI apps
  - Data model is same as POSIX: Stream of bytes in a file
- MPI-IO Features
  - Collective I/O
  - Noncontiguous I/O with MPI datatypes and file views
  - Nonblocking I/O
  - Fortran bindings (and additional languages)
  - System for encoding files in a portable format (external32)



Independent and Collective I/O

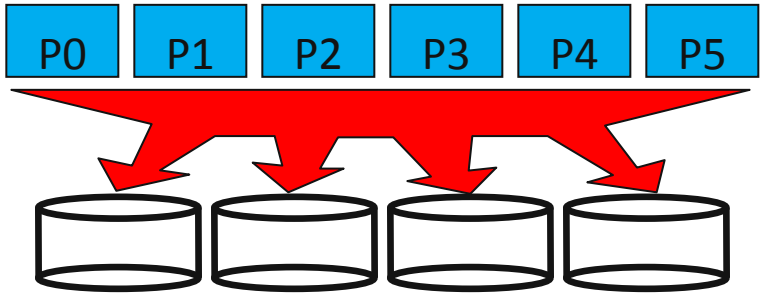


Independent I/O

- Independent I/O operations specify only what a single process will do
  - Independent I/O calls do not pass on relationships between I/O on other processes

- Why use independent I/O
  - Sometimes the synchronization of collective calls is not natural
  - Sometimes the overhead of collective calls outweighs their benefits
    - Example: very small I/O during metadata operations

Independent and Collective I/O

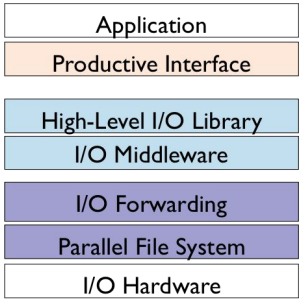


Collective I/O

- Collective I/O is coordinated access to storage by a group of processes
  - Collective I/O functions are called by all processes participating in I/O
- Why use collective I/O
  - Allows I/O layers to know more about access as a whole, more opportunities for optimization in lower software layers, better performance
  - Combined with non-contiguous accesses yields highest performance

High Level I/O Libraries

- Take advantage of high-performance parallel I/O while reducing complexity
  - Add a well-defined layer to the I/O stack
  - Allow users to specify complex data relationships and dependencies
  - Come with machine-independent data formats, self-describing, suitable for array-oriented scientific data



- Examples
  - HDF5: HDF group, since 1989, top 5 libraries at NERSC
  - Parallel netCDF: NWU, ANL, since 2001
  - ADIOS: ORNL, since 2009

High Level I/O Libraries: HDF5

- A parallel HDF5 program has a few extra calls than a serial one

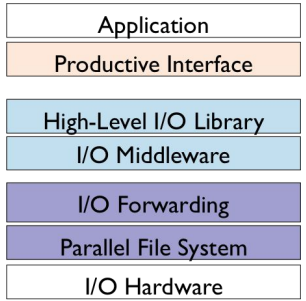
```
MPI_Init(&argc, &argv);

fapl_id = H5Pcreate(H5P_FILE_ACCESS);
H5Pset_fapl_mpio(fapl_id, comm, info);
file_id = H5Fcreate(FNAME,…, fapl_id);
space_id = H5Screate_simple(…);
dset_id = H5Dcreate(file_id, DNAME, H5T_NATIVE_INT,
                    space_id,…);
xf_id = H5Pcreate(H5P_DATASET_XFER);
H5Pset_dxpl_mpio(xf_id, H5FD_MPIO_COLLECTIVE);
status = H5Dwrite(dset_id, H5T_NATIVE_INT, …, xf_id…);
MPI_Finalize();
```

Productive I/O Interface

- Big Data Analytics Stack
  - Spark
  - Tensorflow
  - Caffe
- Science data needs to be loaded efficiently into the engine.
  - H5py
  - H5Spark
  - Fitsio
  - Tensorflow_IO

module load h5py-parallel



H5Py: http://www.nersc.gov/users/data-analytics/data-management/i-o-libraries/hdf5-2/h5py/
H5Spark: http://www.nersc.gov/users/data-analytics/data-management/i-o-libraries/hdf5-2/h5spark/
TensorFlow_IO: https://www.tensorflow.org/api_guides/python/reading_data
TensorFlow_HDF5: To be released by Thorsten Kurth @ NERSC

Productive I/O Interface: H5py

- **Parallel H5py**

```
1 from mpi4py import MPI
2 import h5py
3 fx=h5py.File('output.h5','w',driver='mpio',comm=MPI.COMM_WORLD)
```

```
dset[start:end,:]=temp
```

Independent IO

```
1 with dset.collective:
2  dset[start:end,:]=temp
```

Collective IO

Coding Efforts

```python
from mpi4py import MPI
import numpy as np
import h5py
import time
import sys
comm =MPI.COMM_WORLD
nproc = comm.Get_size()
comm.Barrier()
timefstart=MPI.Wtime()
f = h5py.File(filename, 'w', driver='mpio', comm=MPI.COMM_WORLD)
dset = f.create_dataset('test', (length_x,length_y), dtype='f8')
comm.Barrier()
timefend=MPI.Wtime()
f.atomic = False
length_rank=length_x / nproc
length_last_rank=length_x -length_rank*(nproc-1)
comm.Barrier()
timestart=MPI.Wtime()
start=rank*length_rank
end=start+length_rankL
if rank==nproc-1: #last rank
    end=start+length_last_rank
temp=np.random.random((end-start,length_y))
comm.Barrier()
timemiddle=MPI.Wtime()
if colw==1:
        with dset.collective:
                dset[start:end,:] = temp
        dset[start:end,:] = temp
comm.Barrier()
timeend=MPI.Wtime()
f.close()
```

```c
#include "stdlib.h"
dataspace_id2 = H5Screate_simple(2, dims2, NULL);
dset_id2 = H5Dcreate(file_id2,dataset, H5T_NATIVE_DOUBLE, ...)
H5Sclose(dataspace_id2);
MPI_Barrier(comm);
double t00 = MPI_Wtime();
result_offset[1] = 0;
result_offset[0] =  (dims_x / mpi_size) * mpi_rank;
result_count[0] = dims_x / mpi_size;
result_count[1] = dims_y;
if(mpi_rank==mpi_size-1)
result_count[0] = dims_x / mpi_size + dims_x % mpi_size;
result_space = H5Dget_space(dset_id2);
H5Sselect_hyperslab(result_space, H5S_SELECT_SET, result_offset, ...);
result_memspace_size[0] = result_count[0];
result_memspace_size[1] = result_count[1];
result_memspace_id = H5Screate_simple(2, result_memspace_size, NULL);
  else{
   H5Dwrite(dset_id2, H5T_NATIVE_DOUBLE, result_memspace_id,...);
  }
  MPI_Barrier(comm);

  double t1 = MPI_Wtime()-t0;
  free(data_t);
  double tclose=MPI_Wtime();
  H5Sclose(result_space);
  H5Sclose(result_memspace_id);
  H5Dclose(dset_id2);
  H5Fclose(file_id2);
  tclose=MPI_Wtime()-tclose;
  MPI_Finalize();
}
```

H5py vs. HDF5 Performance

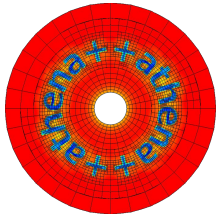
H5Py Performance / HDF5 Performance

Questions: When you gain the productivity, how much performance you can afford to lose?

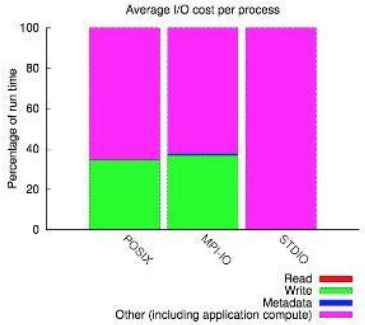
		Single Node	Multi-nodes
Metadata	1k File Creation	63.8%	
	1k Object Scanning	60.0%	
Independent I/O	Weak Scaling	97.8%	100%
	Strong Scaling	100%	97.1%
Collective I/O	Weak Scaling	100%	90%
	Strong Scaling	98.6%	87%

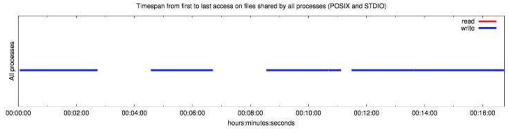
HDF5 vs. H5py: http://www.nersc.gov/users/data-analytics/data-management/i-o-libraries/hdf5-2/h5py/

Case Study: Athena's IO



Athena is an astrophysics code, used in wide range of problems: interstellar medium, star formation, etc. https://princetonuniversity.github.io/athena/





"I made the changes you suggested and did the test. It solved my problem! Previously, **the I/O can take 40% of the time. Now the I/O time is basically 0**.

Thank you very much for your help. This is really useful."

IO Analysis with Darshan

darshan-job-summary.pl darshan_log output.pdf

Darshan: http://www.nersc.gov/users/software/performance-and-debugging-tools/darshan/
HDF5: http://www.nersc.gov/users/data-analytics/data-management/i-o-libraries/hdf5-2/

Thank You

Email me: http://www.nersc.gov/users/getting-help/consulting-services/

[1] J.L. Liu, Q. Koziol, H.J. Tang, F. Tessier, W. Bhimji, B. Cook, B. Austin, S. Byna, B. Thakur, G. Lockwood, J. Deslippe, Prabhat, Understanding the IO Performance Gap Between Cori KNL and Haswell, CUG'17