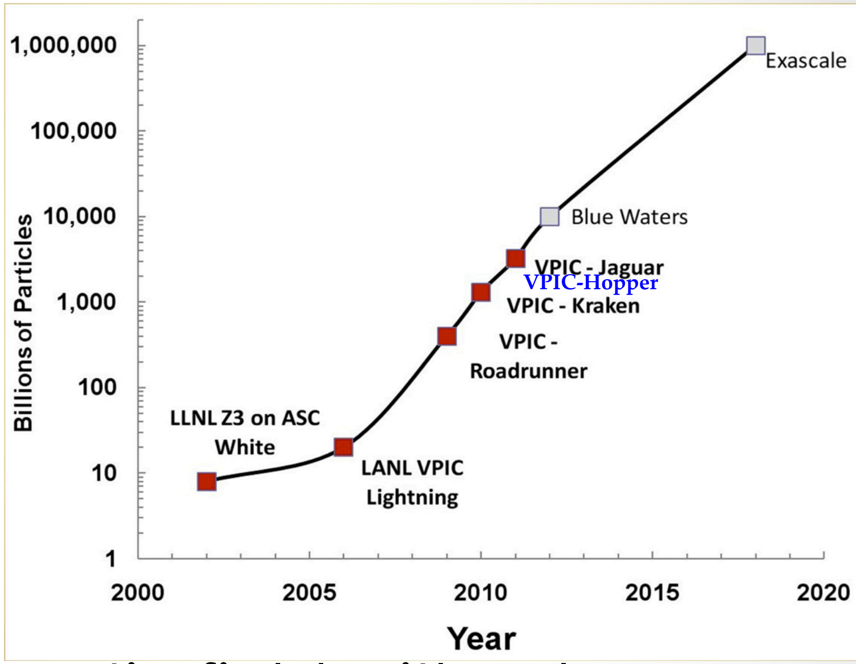
Trillion Particles, 120,000 cores and 350TBs: Lessons Learned from a Hero I/O Run on Hopper*

Suren Byna, **Prabhat**, Andrew Uselton,
David Knaak, Helen He

*CUG 2013 Best Paper Award

VPIC for Plasma Physics

- Vector Particle in Cell simulation code
- Bill Daughton (LANL), Homa Karimabadi (UCSD)
- State-of-the-art 3D electromagnetic relativistic plasma physics simulation
- Simulate space weather



  - Interaction of Earth's magnetic field with solar particles
  - Satellite communications, ISS
  - Magnetic Reconnection, Turbulence
  - Accurate 3D simulation requires $O(10^{12})$ particles

VPIC: BIG Data

- 2 Trillion particles simulated
  - checkpoint/restart
- 1 Trillion electrons used for analysis
  - 8 variables per electron
  - 30TB to 43TB per timestep
- Simulated for ~10,000 timesteps
- 10 timesteps dumped, ~350TB
- 150TB checkpoint data produced

Challenges

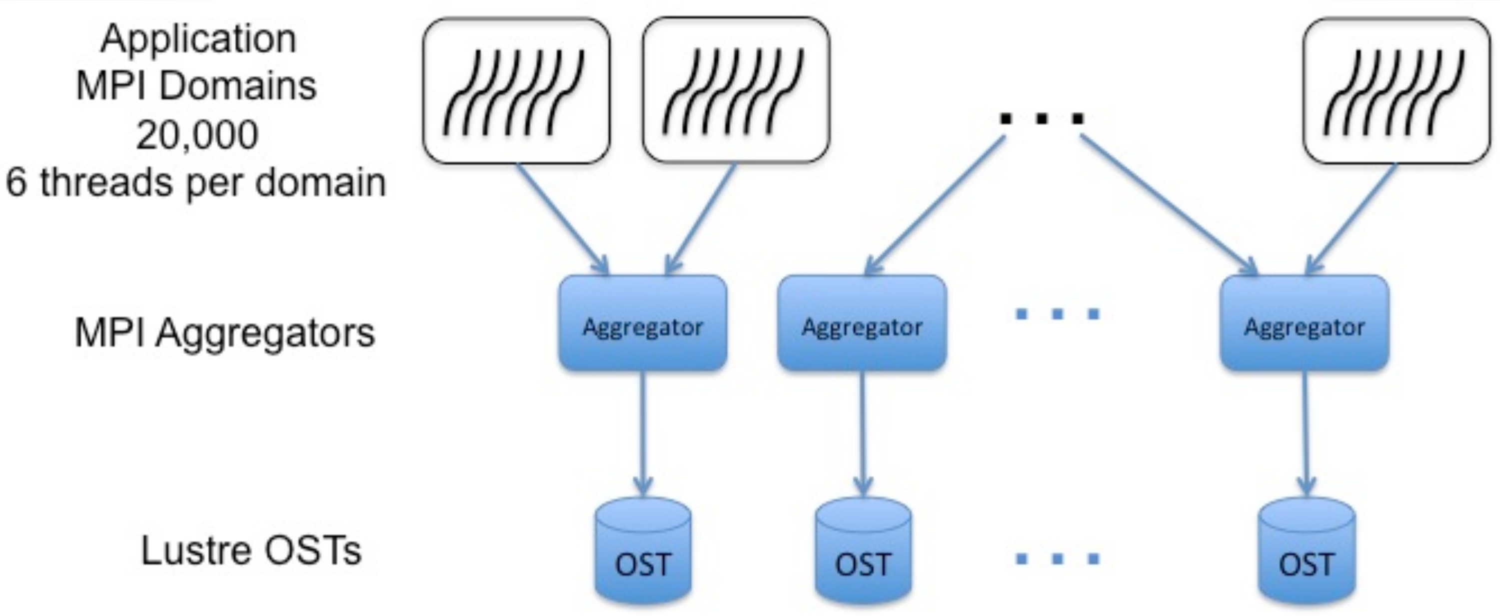
- Scalable I/O strategy?
  - In situ works well if analysis tasks are known a priori
  - Storing data is required for exploratory analysis
- Scalable Analysis strategy?
  - Sift through large amounts of data
- Visualization strategy?
  - Only display “relevant” information

Solutions

- Scalable I/O strategy?
  - H5Part provides a highly productive interface for particle I/O
  - Parallel I/O with HDF5 on a production hardware + software stack
  - Obtain peak performance 35GB/s on Lustre, 80% sustained bandwidth
- Scalable Analysis strategy?
  - Hybrid parallel version of FastQuery
  - 10 minutes to index single variable, 3 seconds to execute range queries
- Visualization strategy?
  - Query based visualization in VisIt

S. Byna, J. Chou, et al., "Parallel I/O, Analysis, and Visualization of a Trillion Particle Simulation", in Proc. of the ACM/IEEE Supercomputing Conference (SC'12)

VPIC: Hopper Configuration



- Hopper:
  - Cray XE6 system, 1.28 PF, ~150,000 cores
  - Node: Two 12-core AMD Magny-Cours, 32GB memory
  - Lustre filesystem with 156 OSTs, 35 GB/s peak bandwidth
- Lustre aware MPI-IO implementation
  - MPI collective buffer size = stripe size
  - # MPI aggregators = stripe count

(Expected) Challenges

- NERSC has a broad user base and high system utilization
  - Excellent staff and vendor support

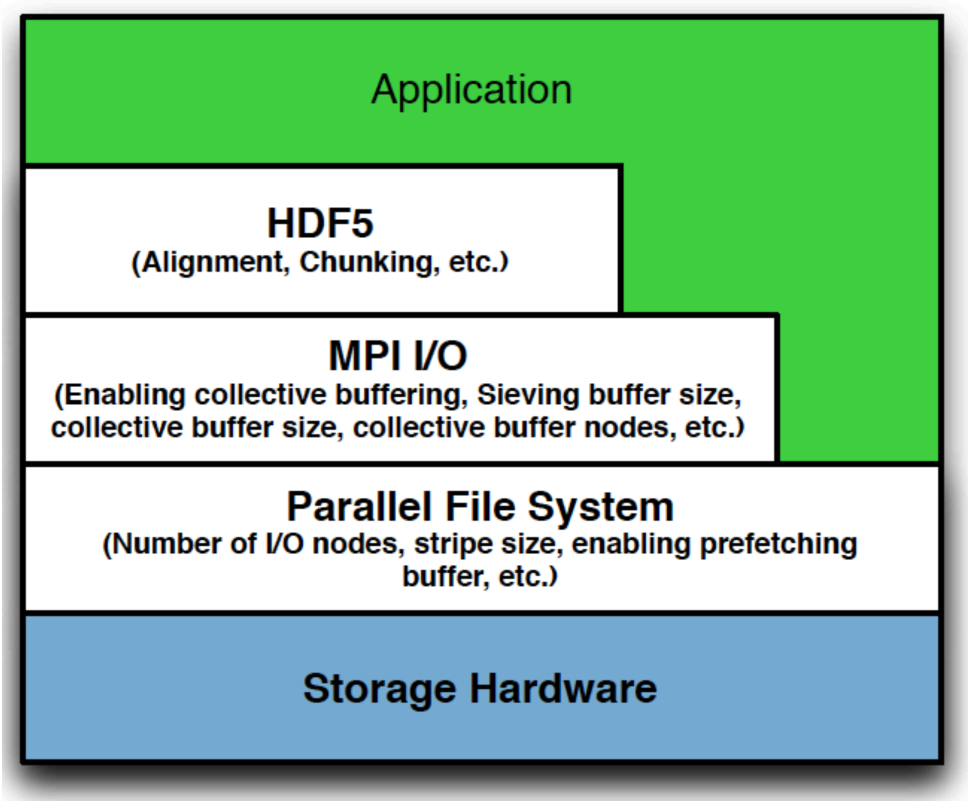
- Disk Quota
  - 500TB over a period of 6 months

- Scheduling Runs
  - 120,000 cores, all available memory on nodes
  - 24-36 hour runtime
  - Initial runs required monitoring
  - Reg_xbig queue turns on at 9pm on Friday
  - Schedule checkpoints carefully

Lessons Learned

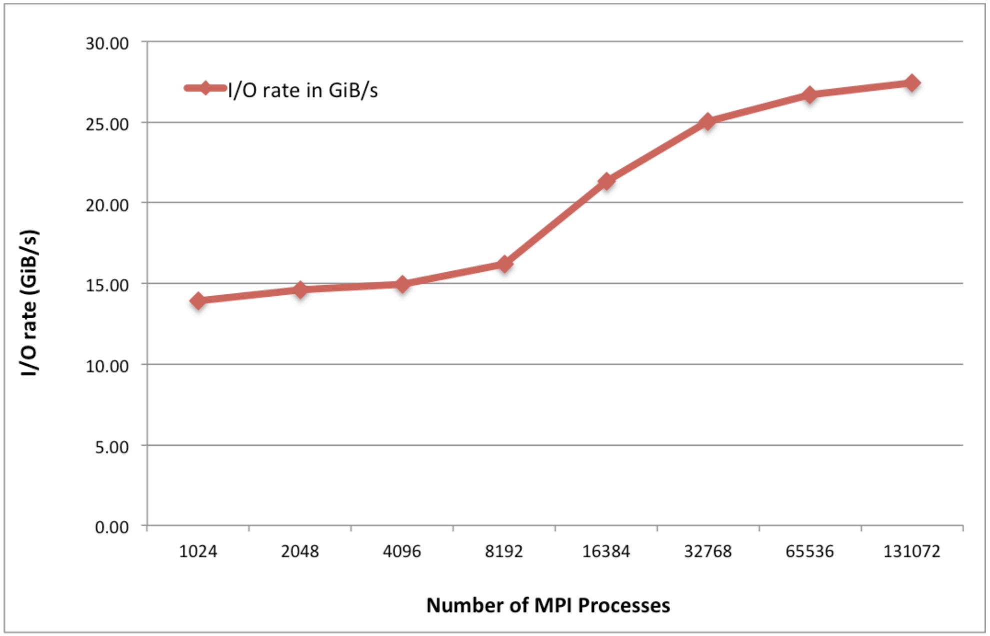
1. Tuning multiple layers of parallel I/O subsystem is important (and out of the reach of most users)
2. Collective writes to single shared HDF5 file can work as well as file-per-process
3. Advance verification of file system hardware is critical for obtaining peak performance
4. Advance verification of available resources for memory intensive applications is important
5. Scalable tools are required for diagnosing software and hardware problems at large scale (100K+ cores)

Lesson 1: I/O tuning is important



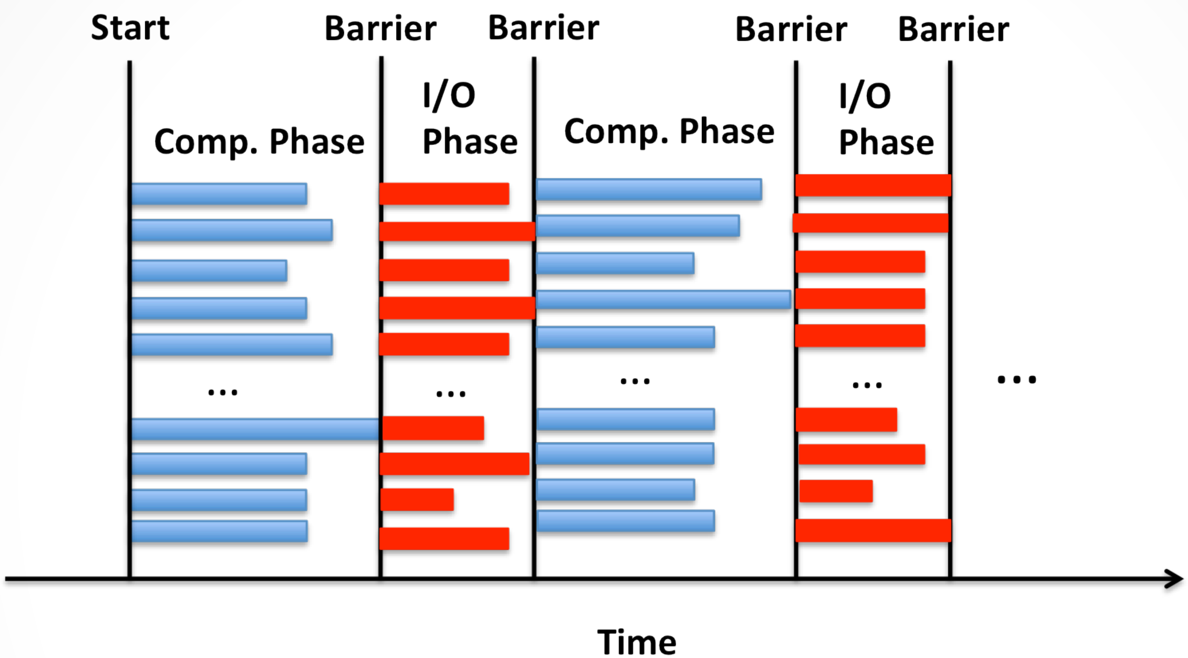
- Users cannot be expected to understand tunable parameters and their interactions
- System defaults might be sub-optimal

Manual tuning with VPIC-IO



- Lustre stripe count and stripe size
  - Varied stripe count from 64 to 156 and stripe size from 1MB to 1GB
  - Chose stripe count of 144 and stripe size of 64MB
- Lustre-aware MPI-IO collective buffering on Hopper uses CB2 algorithm
  - Number of collective buffering aggregator nodes is equal to the stripe count
  - Size of collective buffer is equal to the stripe size

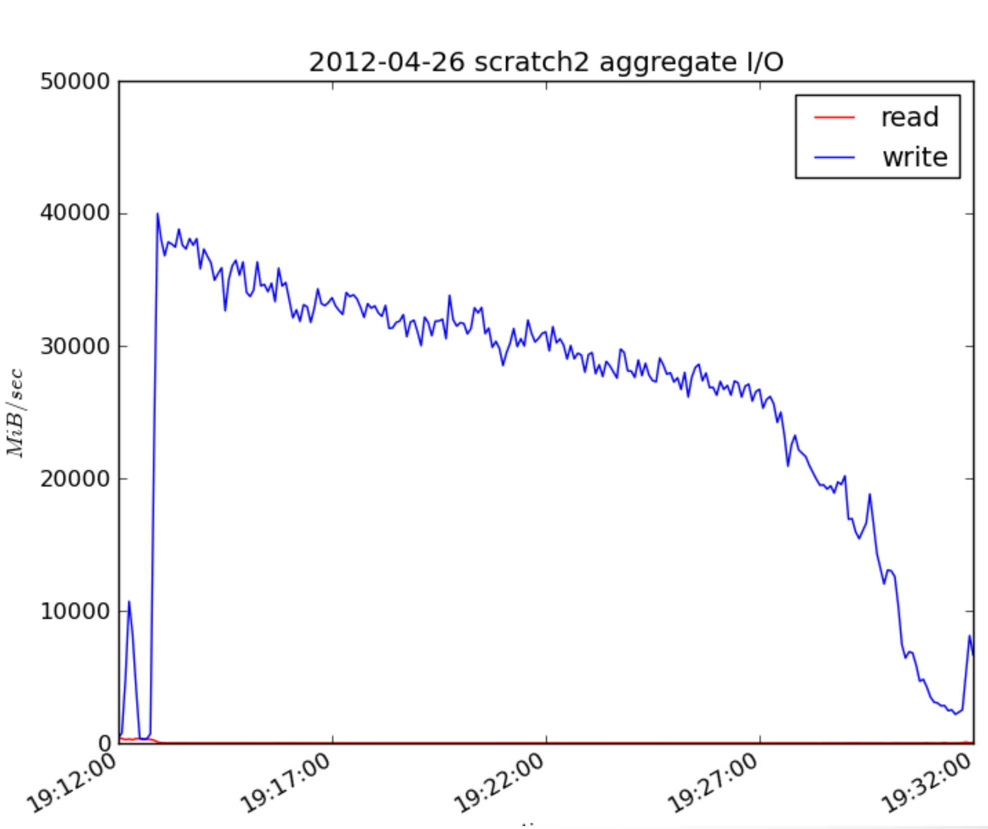
Lesson 2: Parallel HDF5 works well



- I/O of VPIC follows a banded pattern
- Two file writing strategies
  - File per process model
  - Shared file with HDF5 and H5Part

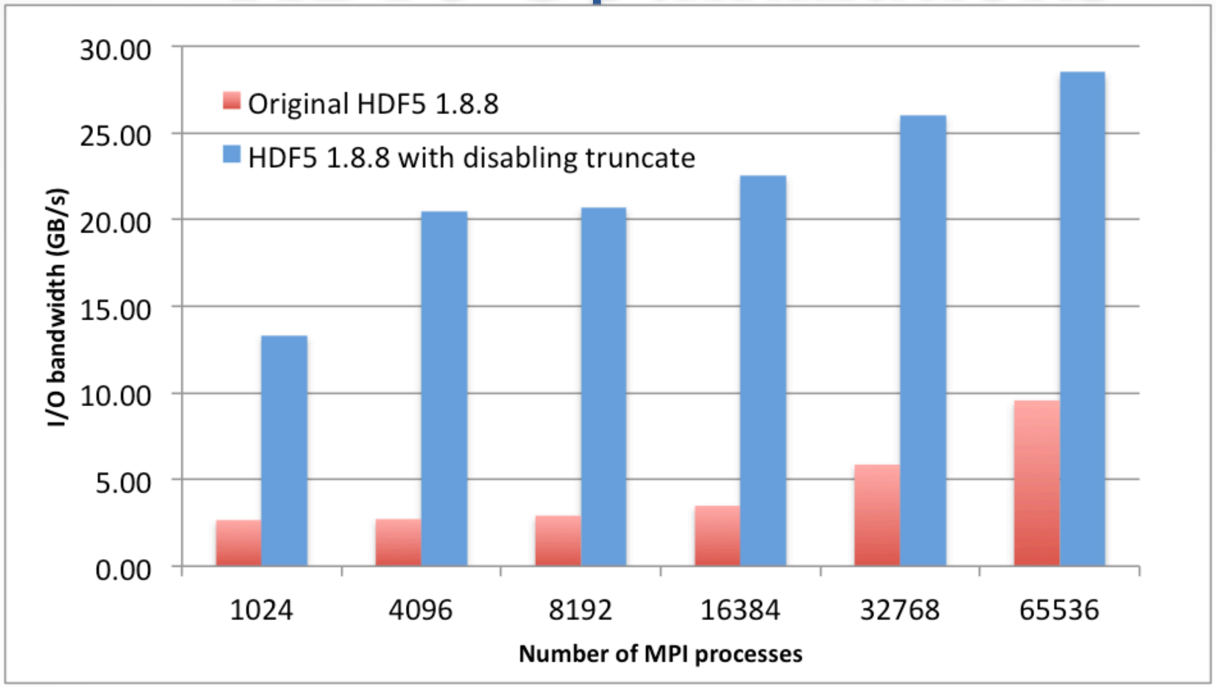
File-per-process Performance

- Performance of 20,000 files with a combined size of ~30TB
- Load imbalance and the slowest OST determine performance
- I/O rate: ~27GB/s
- Create a challenge for downstream vis and analysis tools



Lustre Monitoring Tool (LMT) plots

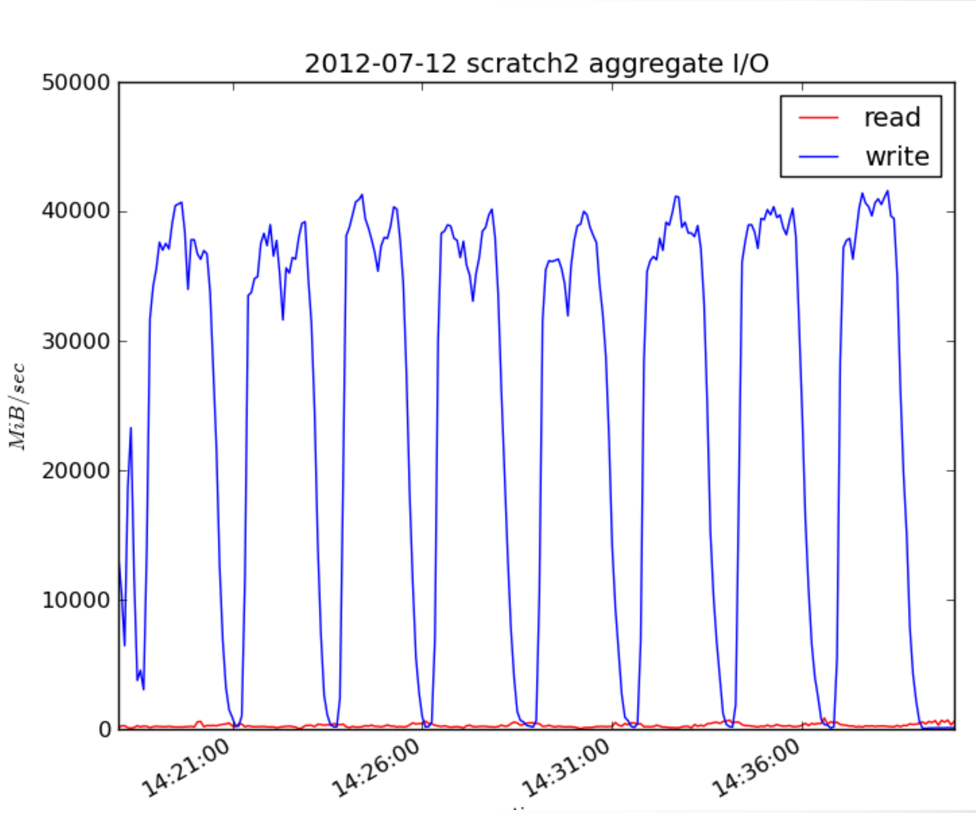
HDF5 Optimizations



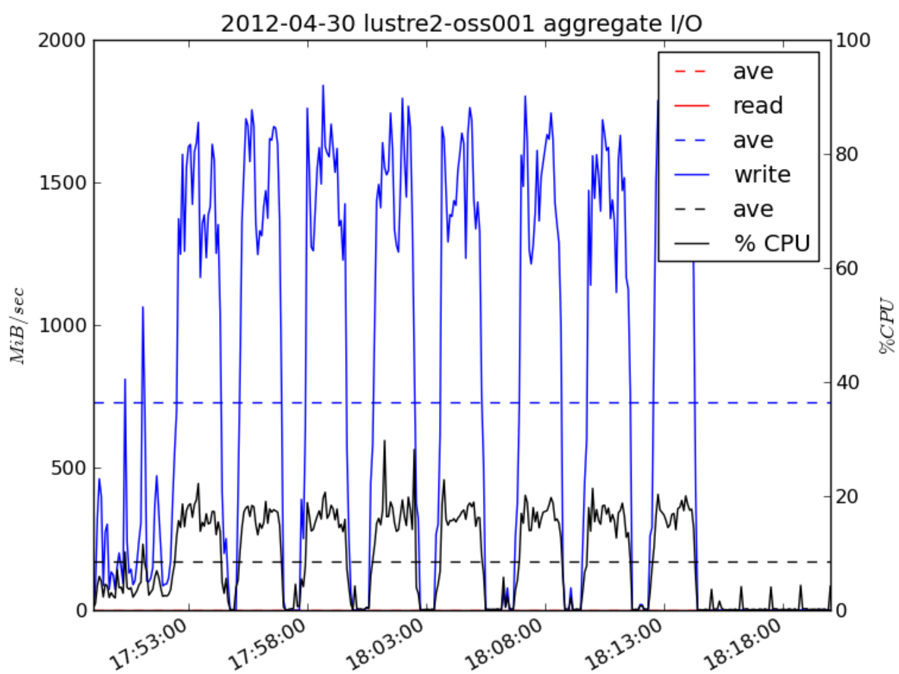
- HDF5 file close function verifies the size of the file matching with its allocated size to detect any external modification or corruption
- This is an expensive operation because of its collective nature
- Modified HDF5 to disable this "truncate" operation and achieved 3-5X performance improvement

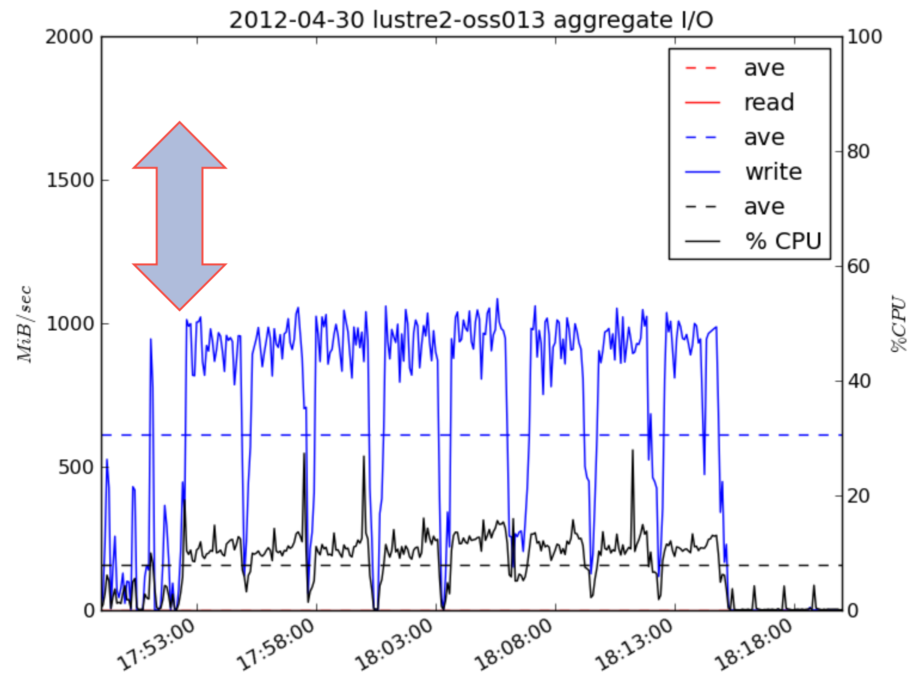
Parallel HDF5 Performance

- Performance of writing one ~31 TB particle file
- I/O rate: ~27 GB/s
- Need for rendezvous after writing each variable, due to H5Part and HDF5 interactions



Lesson 3: Advance verification of filesystem hardware is important





- Early runs were obtaining 60% of peak bandwidth
- LMT logs at the OST and OSS level were critical
- Placed sub-optimal OSTs in read-only mode for the Hero run

Lessons 4: Advance verification of resources for memory-intensive apps is important

- Hopper has 32GB memory on most nodes
  - Some nodes have 64 GB
  - Total memory of 5,000 nodes: ~156 TB
- VPIC memory footprint: ~142 TB
  - ~29GB on each node
  - ~90% of memory used on each node
- Significant memory pressure (share with lightweight OS tasks)
- OOM error from a single node killed one job instance

Lessons 4: Advance verification of resources for memory-intensive apps is important

- Used a combination of tools to verify memory availability before each run and after dumping large particle data
- Node Health Checker (NHC)
  - Free Memory Check to verify the available free memory
  - “Admindown” nodes with less than 29 GB free memory
- Developed a Perl script that reads the free memory information from /proc/buddyinfo on all the nodes in allocation
  - Manually sorted and verified free memory

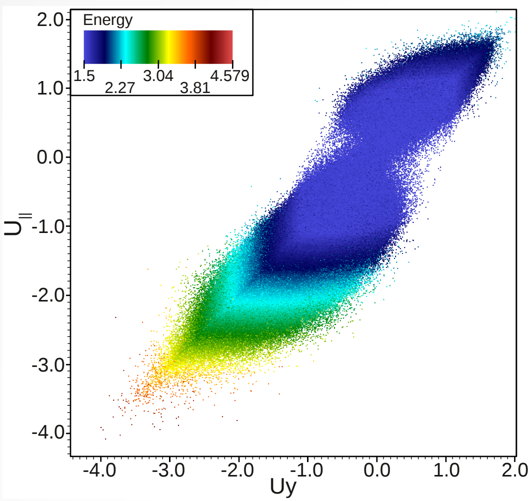
Lesson 5: Scalable tools are required for diagnosing software and hardware problems at large scale

- It can be time consuming and tedious for users to verify system health prior to large scale runs
- Tools need to be streamlined to facilitate verification
  - Node Health Checker, 'xtprocadmin', custom perl scripts
- I/O runtime monitor:
  - Sluggish OST can drag performance for the job
  - LMT was very helpful; but used in a post-mortem fashion. Need better pro-active solutions

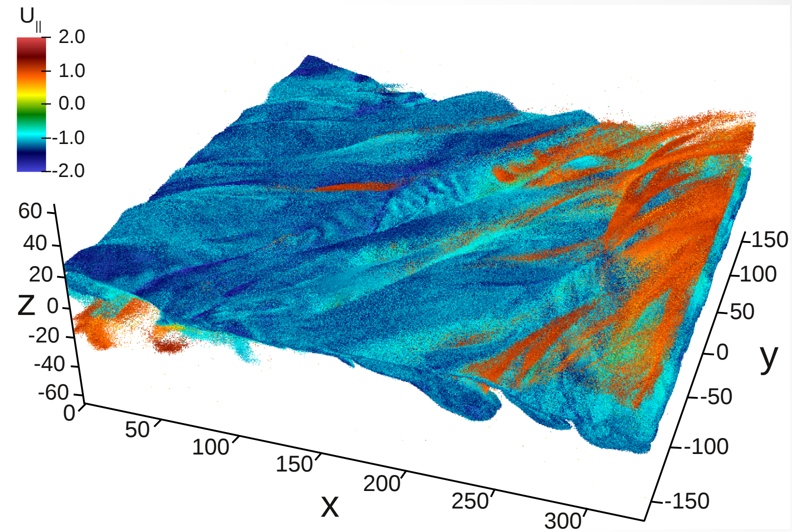
Summary

- Parallel HDF5 obtained peak 35GB/s performance on 120,000 Hopper cores
  - Sustained 80% peak bandwidth
  - Production stack on NERSC platform
- 350 TBs generated and analyzed over a period of 6 months
- Close co-ordination between NERSC, Cray and CRD staff was critical

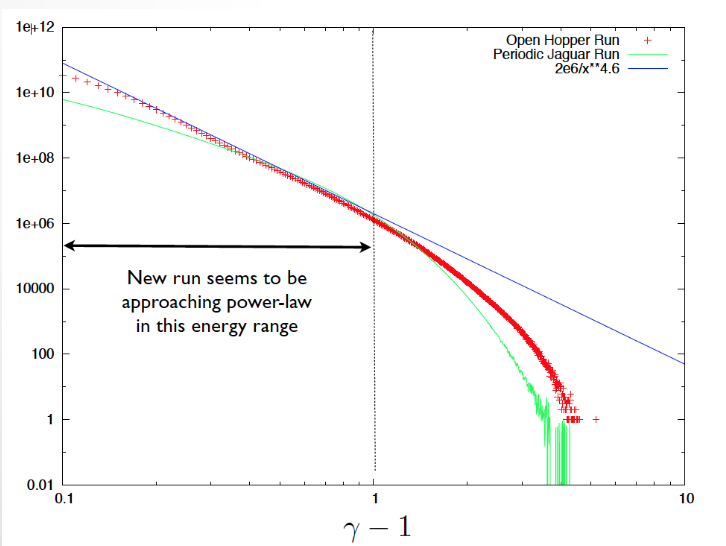
VPIC: Science Results



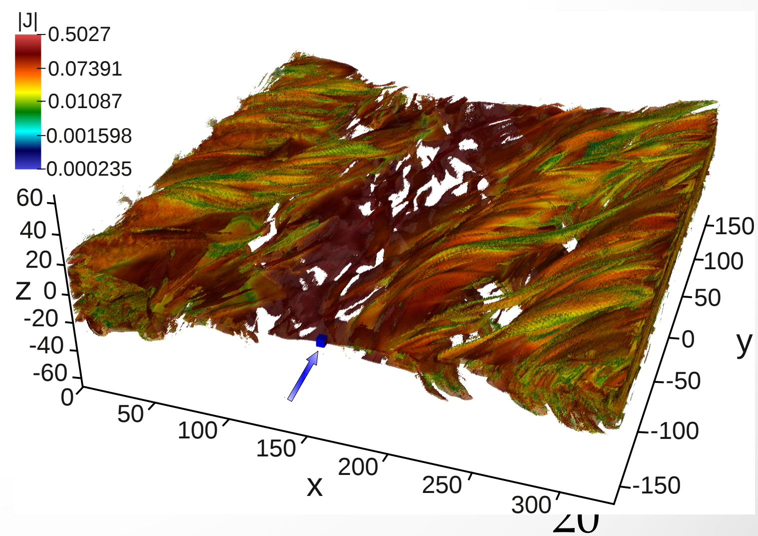
Preferential acceleration along magnetic field



Energetic particles are correlated with flux ropes

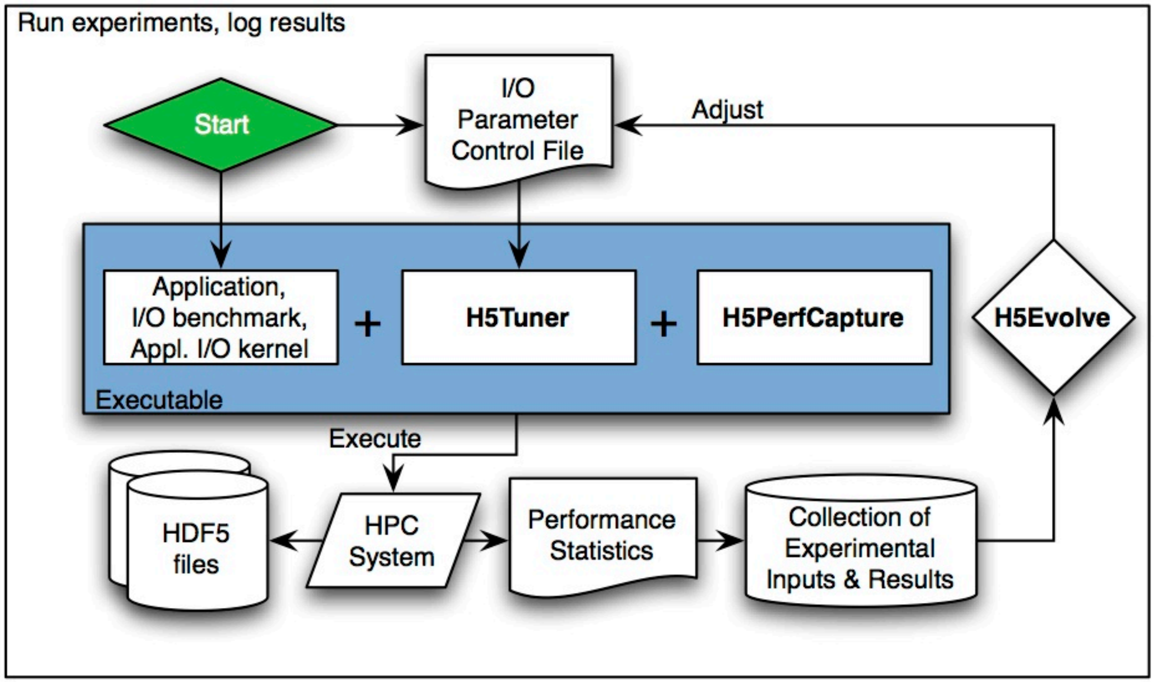


Discovered power-law distribution in energy spectrum



Discovered agyrotropy near the reconnection hot-spot

Future Directions: HDF5 Auto-Tuning



- Combination of genetic algorithms and statistical models to explore tunable parameter space
- Tested on 4 applications on Intrepid, Stampede, Hopper
- 2-40x performance improvement for HDF5 applications

Future Directions: HDF5 Scaling

- 10 Trillion particle run being planned on Blue Waters
- Exploring Burst Buffer Hardware on Cori

- HDF5 Multi-dataset write calls
  - Large writes: VPIC
  - Small writes: Chombo/AMR

- HDF5 Subfiling support
  - File-per proc (n-n) and write to shared file (n-1) are extreme ends of the spectrum
  - "n" files create a major data management challenge
  - Moving "1" file around (archiving, copying) is a major challenge
  - n-m

Acknowledgments

- **DOE/ASCR** (PM: Lucy Nowell): Support for the ExaHDF5 project
- **Domain Scientists**: Homa Karimabadi, Bill Daughton, Vadim Roytershteyn
- **Collaborators**: Jerry Chou, Oliver Rubel, John Wu, Wes Bethel, Arie Shoshani
- **NERSC**: Tina Butler, Katie Antypas, Francesca Verdier, Woo-Sun Yang, Harvey Wasserman
- **Cray**: Steve Luzmoor, Terence Brewer, Randell Palmer, Bill Anderson, Mark Pagel, Steven Oyanagi

Thanks!

- Questions?

H5Part

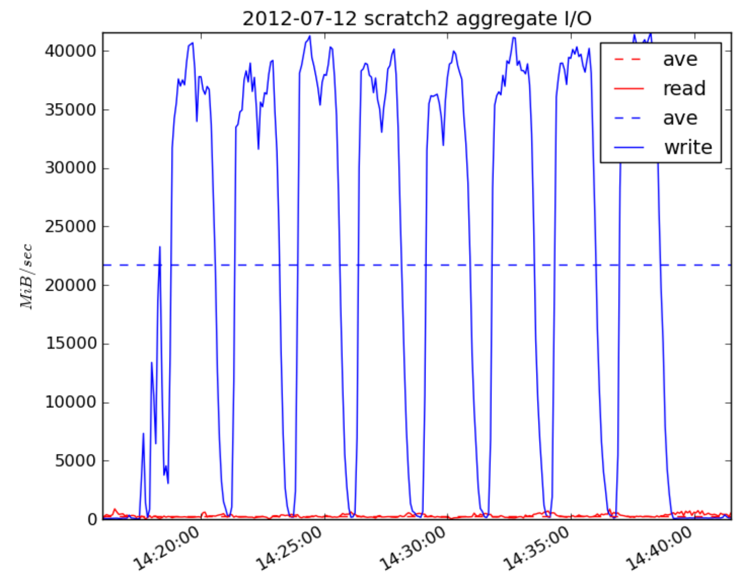
```
h5pf = H5PartOpenFileParallel (fname, H5PART_WRITE |
                    H5PART_FS_LUSTRE, MPI_COMM_WORLD);
H5PartSetStep (h5pf, step);
H5PartSetNumParticlesStrided (h5pf, np_local, 8);

H5PartWriteDataFloat32 (h5pf, "dX", Pf);
H5PartWriteDataFloat32 (h5pf, "dY", Pf+1);
H5PartWriteDataFloat32 (h5pf, "dZ", Pf+2);
H5PartWriteDataInt32   (h5pf, "i",  Pi+3);
H5PartWriteDataFloat32 (h5pf, "Ux", Pf+4);
H5PartWriteDataFloat32 (h5pf, "Uy", Pf+5);
H5PartWriteDataFloat32 (h5pf, "Uz", Pf+6);
H5PartWriteDataFloat32 (h5pf, "q", Pf+7);

H5PartCloseFile (h5pf);
```

Parallel I/O and Analysis of a Trillion Particle Simulation

- Objectives: Support I/O and analysis needs for a state-of-the-art PIC plasma physics code
- Accomplishments:
  - Ran Trillion particle simulation on 120,000 hopper cores
  - Parallel HDF5 obtained peak 35GB/s I/O rate and 80% sustained bandwidth (top figure)
  - Developed hybrid parallel FastQuery using FastBit to utilize multicore hardware
  - FastQuery took 10 minutes to index and 3 seconds to query energetic particles (bottom figures)
  - SC12 paper, XLDB 2012 poster
- Impact
  - Proved efficient storage and analysis of files of size greater than 40TB with HDF5



#cores	500	1,250	2,500	5,000	10,000
MPI-alone	1704s	935s	572s	423s	280s
hybrid	1660s	850s	587s	347s	256s

#cores	scan	MPI-alone	hybrid
250	975	10.1	10.8
500	532	8.6	5.5
1250	266	4.1	2.7

A comparison of FastQuery MPI-alone and hybrid parallel versions. Top table compares indexing time and the bottom compares querying time