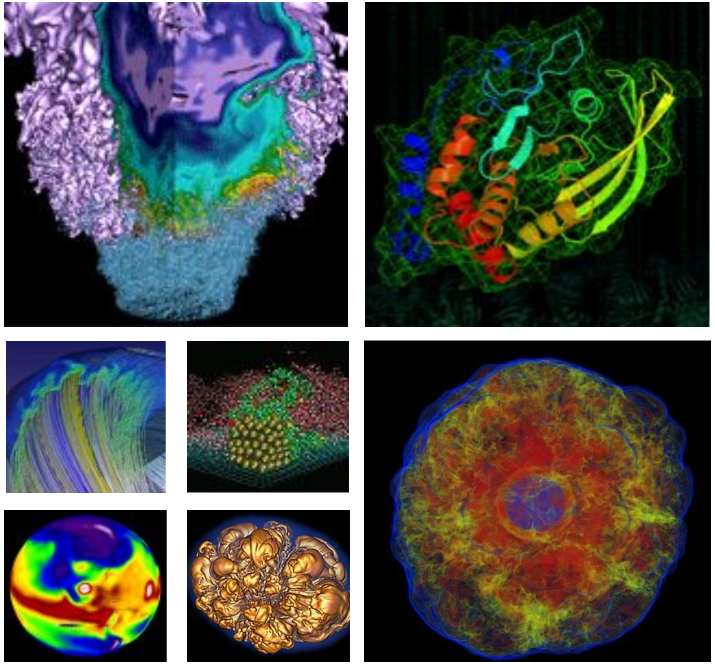
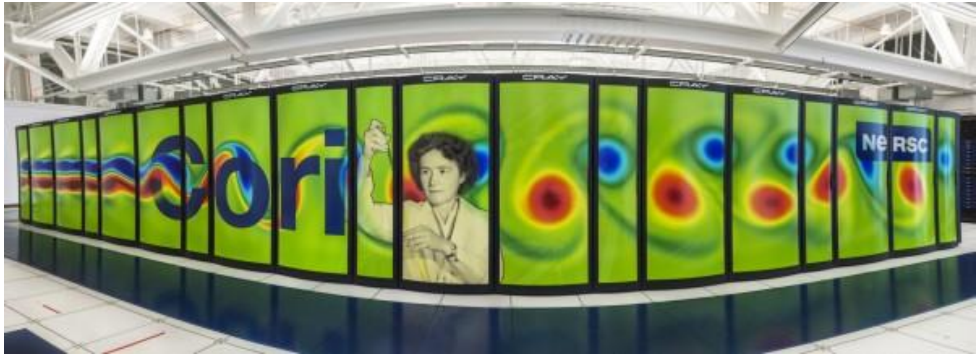
Advancing physics simulation and analysis workflows from customized local clusters to Cori

Jan Balewski, Matthew Kramer, Rei Lee, Mustafa Mustafa, Jeff Porter, Vakho Tsulaia



NERSC

CHEP 2019



U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

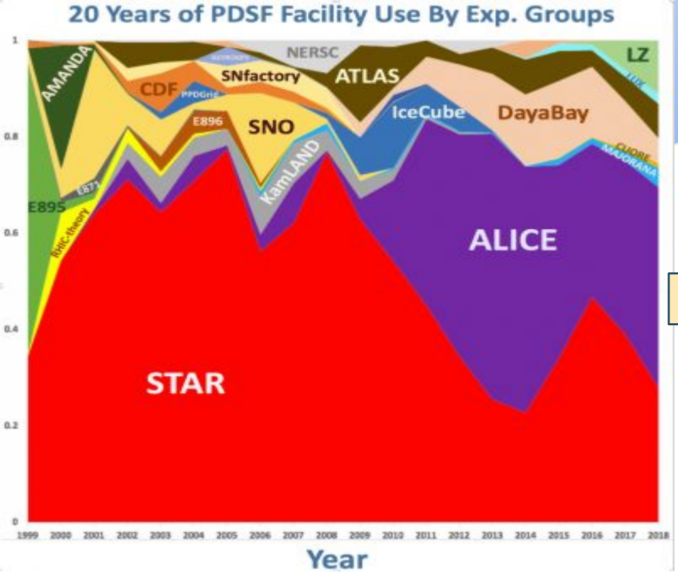
NERSC Systems

2016

NERSC-8: Cori, 30PFs, 4MW
2k Haswell + 6k KNL

NERSC-7: Edison
2.5 PFs
Multi-core CPU
3MW

2013



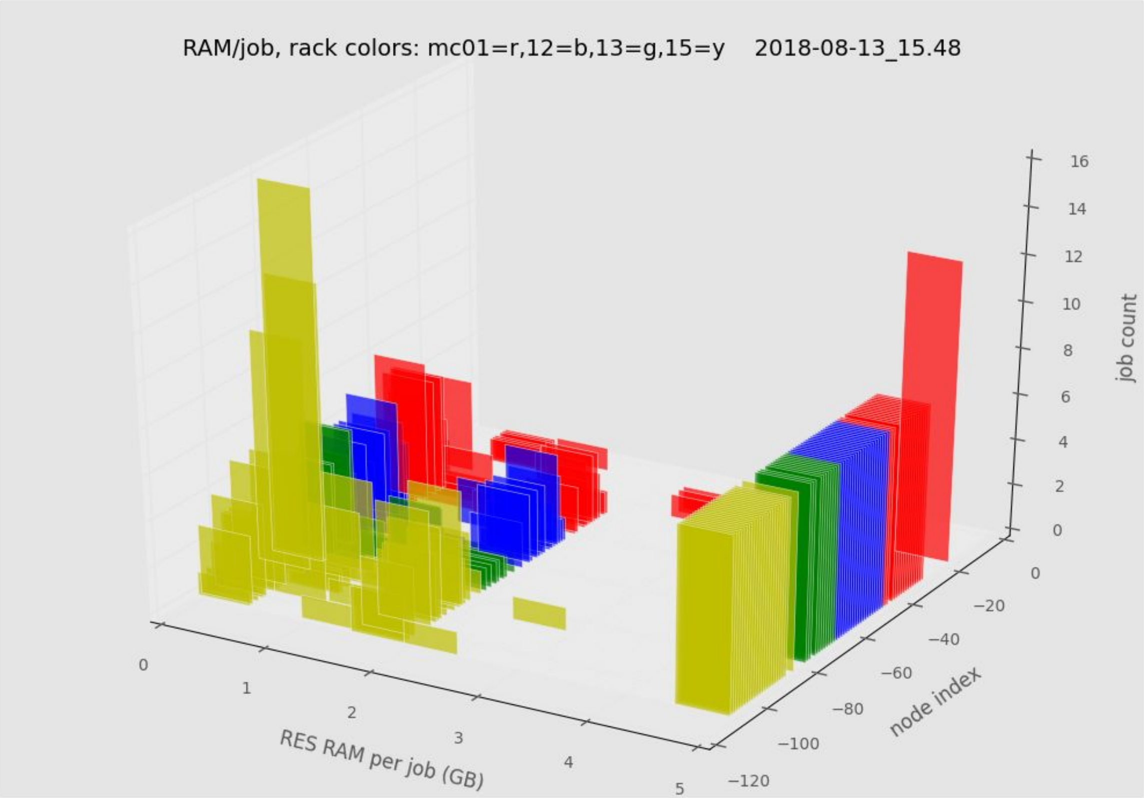
PDSF
70 Haswell nodes
32 cores, 128 GB RAM
Local storage

2019

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

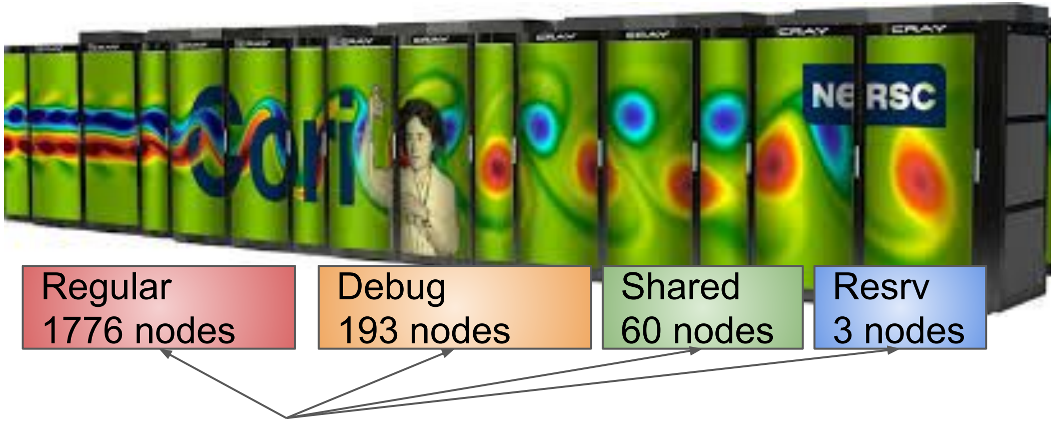
PDSF load - RAM profile of running jobs



123 GB/32 CPUs =3.8 GB/task

Diversity of jobs (and users) allows for better utilization of nodes

Cori - NERSC CRAY Workhorse



Regular
1776 nodes

Debug
193 nodes

Shared
60 nodes

Resrv
3 nodes

Partitions (aka queues)

2,004 Xeon "Haswell" nodes

- 32 cores (2x hyper-thread)
- 120 GB RAM

9,300 Xeon Phi "Knight's Landing" nodes (KNL)

- 68 cores (4x hyper-thread)
- 90 GB RAM

User Home:40 GB
(GPFS)

/project/projectdirs/star,...
(GPFS)

$SCRATCH
20TB/user
(Luster)

Running on Cori at scale(1) - highway analogy

Interactive usage : salloc

Throughput: ~10 CPU hours/day

- code debugging

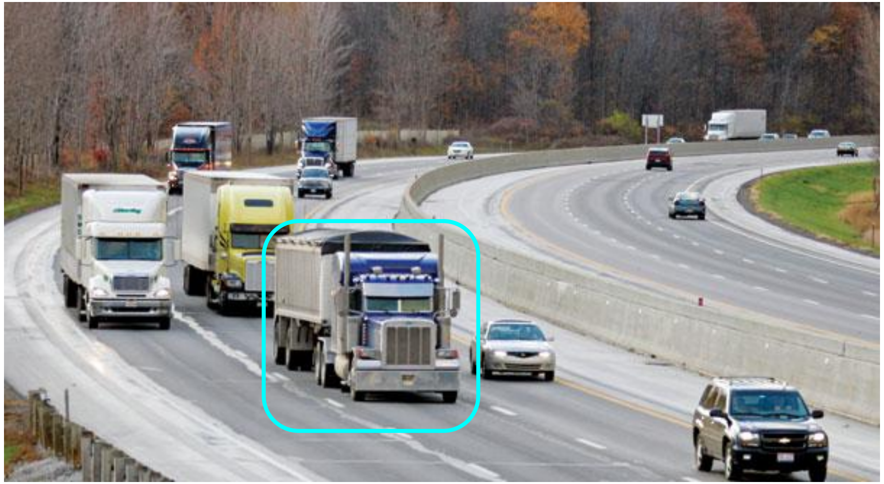
Submit 1-core job(s) to **shared** queue

Throughput: ~5k CPU hours/day
10 nodes* 30 tasks *20 h

- Management of 10k jobs is non-trivial
- Only 60 nodes accessible (3% of Cori)

Running on Cori at scale(2) - highway analogy

Full node jobs: 30 to 50 tasks/node, **regular** queue



Multi-node jobs w/ ephemeral DBs

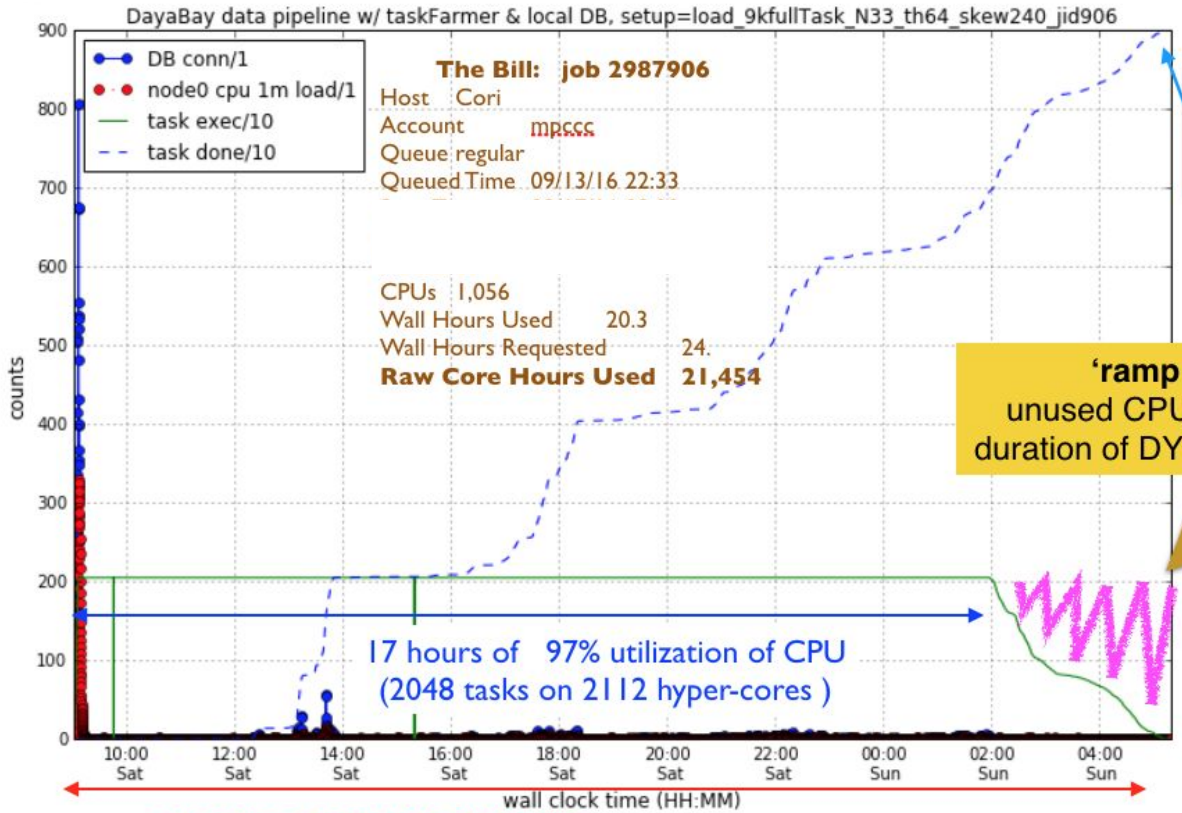
Throughput: ~100k CPU hours/day
200 nodes* 30 tasks *20 h

- 90% of Cori is (potentially) accessible
- IO bottleneck - need optimization
- External DBs not able to handle concurrency

Throughput: ~1M CPU hours/day
2000 nodes* 30 tasks *20 h

- HPC compute power
- Single 30-nodes job w/ local DB creates
- Requires expert understanding of Cori

DayaBay 20-h 2000-tasks as 1 Slurm job



The Bill: job 2987906

Host Cori

Account mpccc

Queue regular

Queued Time 09/13/16 22:33

CPUs 1,056

Wall Hours Used 20.3

Wall Hours Requested 24.

Raw Core Hours Used 21,454

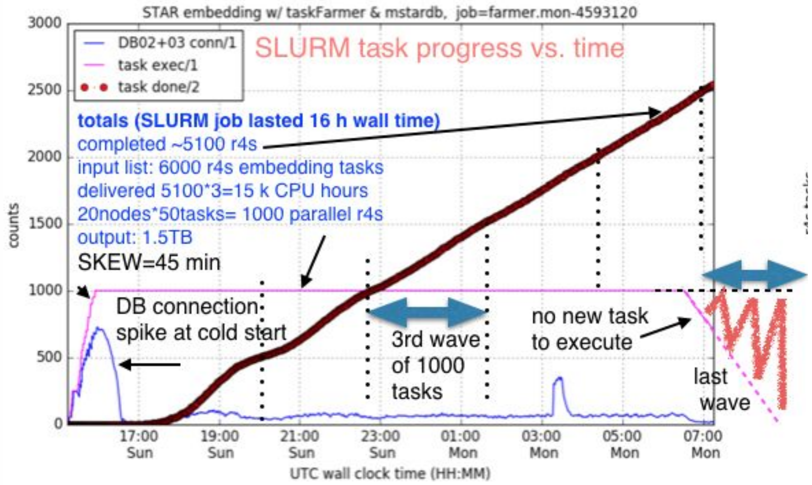
17 hours of 97% utilization of CPU
(2048 tasks on 2112 hyper-cores)

allocation charged for 20 hours

All 9k tasks completed

'ramp down'
unused CPU due to fixed duration of DYB tasks of ~4 h

Example: 20-node 1000 root4star Slurm job



SLURM task progress vs. time

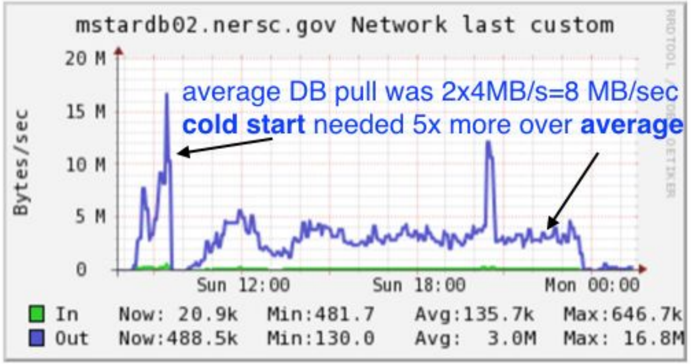
totals (SLURM job lasted 16 h wall time)
completed ~5100 r4s
input list: 6000 r4s embedding tasks
delivered 5100*3=15 k CPU hours
20nodes*50tasks= 1000 parallel r4s
output: 1.5TB
SKEW=45 min

DB connection spike at cold start

3rd wave of 1000 tasks

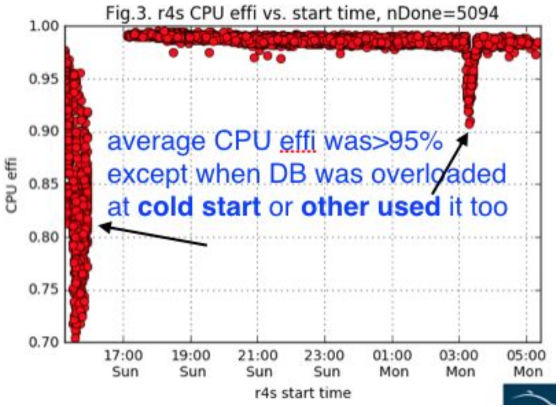
no new task to execute

last wave



average DB pull was 2x4MB/s=8 MB/sec
cold start needed 5x more over **average**

```
■ In   Now: 20.9k   Min:481.7   Avg:135.7k   Max:646.7k
■ Out  Now:488.5k   Min:130.0   Avg:  3.0M   Max: 16.8M
```



average CPU effi was>95%
except when DB was overloaded
at **cold start** or **other used** it too

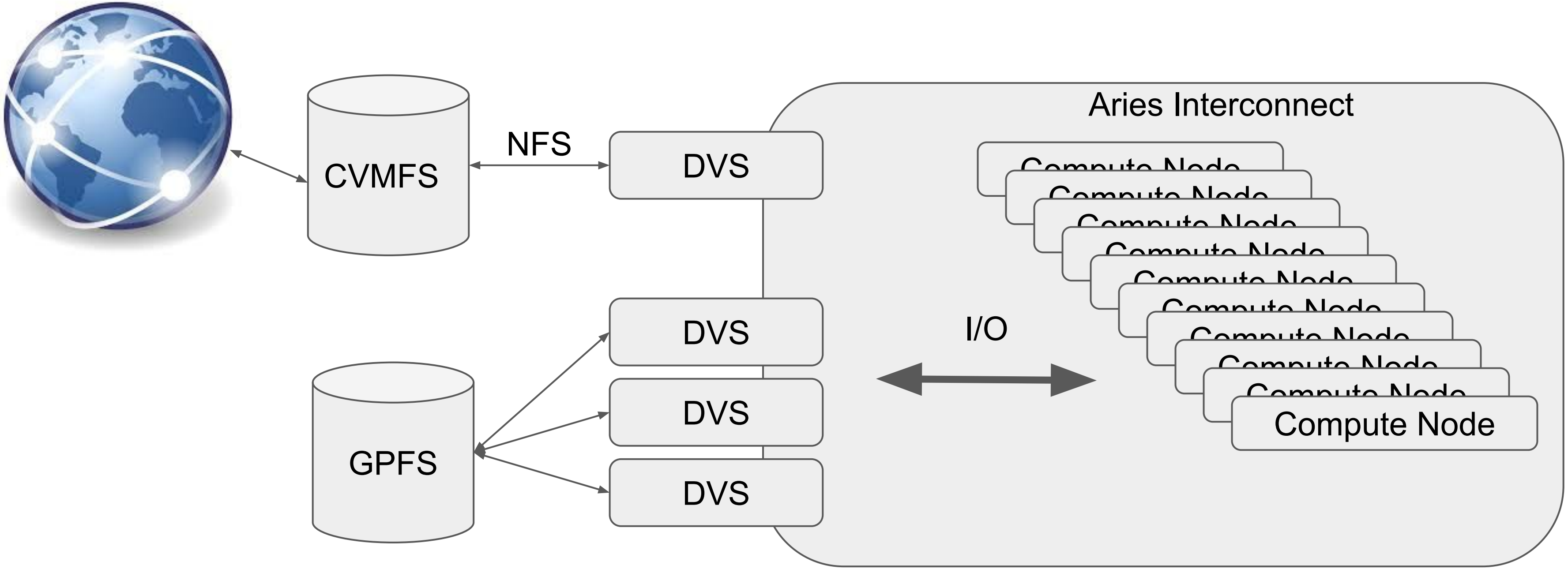
Use taskFramer for BFC management

Multiple ‘waves’ of BFC in one job

Duration 16 wall hours

NO local DB → lower utilizat

CVMFS on Cori



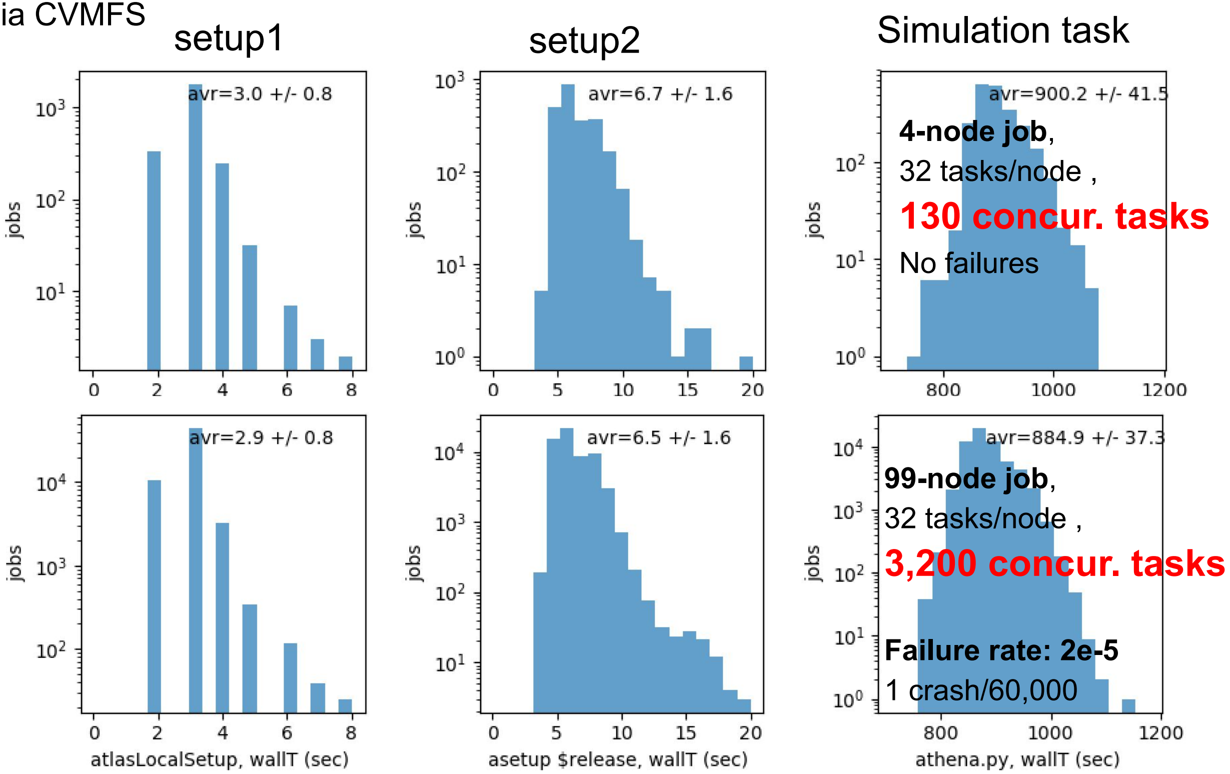
DVS does I/O forwarding and caching data
Cori has 32 DVS servers, 4 of those are dedicated to CVMFS

Scalable CVMFS on Cori - ATLAS workflow

Atlas user source 2 scripts at the start of any ATLAS job

- software and condition-DB delivered via CVMFS
- Test duration : 6h wall time
- It was simulation task

1. **atlasLocalSetup**: finds base code on CVMFS , takes ~3 seconds
2. **Asetup**, scans CVMFS tree for specific version of libs, takes ~7 seconds
3. Run simulation (**athena.py**). 3 events/simu, 15 min/simu, 60,000 simu tasks per 1 slurm job



setup1

setup2

Simulation task

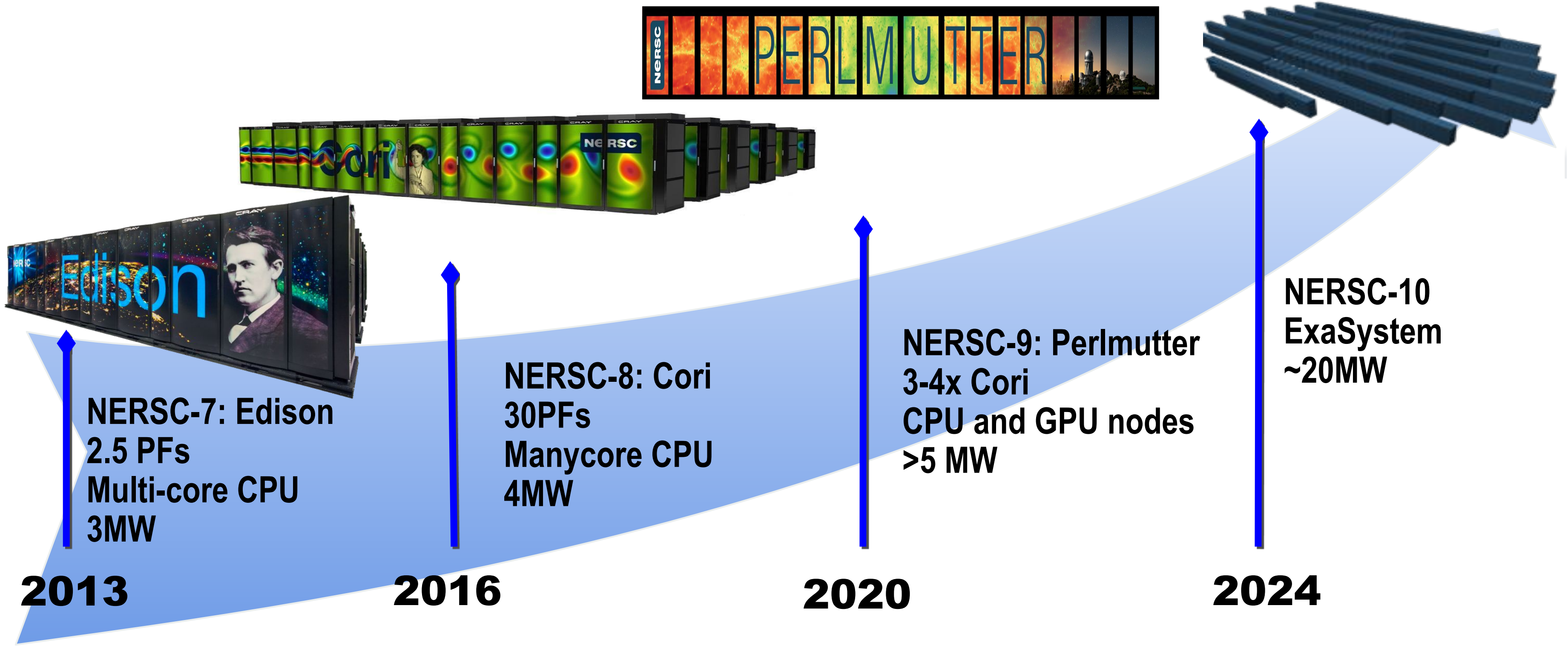
4-node job, 32 tasks/node , **130 concur. tasks** No failures

99-node job, 32 tasks/node , **3,200 concur. tasks**

Failure rate: 2e-5 1 crash/60,000

Problem: ‘find’ used by asetup pulls meta-data from CVMFS which are not cached by DVS

NERSC Systems Roadmap



NERSC-7: Edison
2.5 PFs
Multi-core CPU
3MW

NERSC-8: Cori
30PFs
Manycore CPU
4MW

NERSC-9: Perlmutter
3-4x Cori
CPU and GPU nodes
>5 MW

NERSC-10
ExaSystem
~20MW

2013

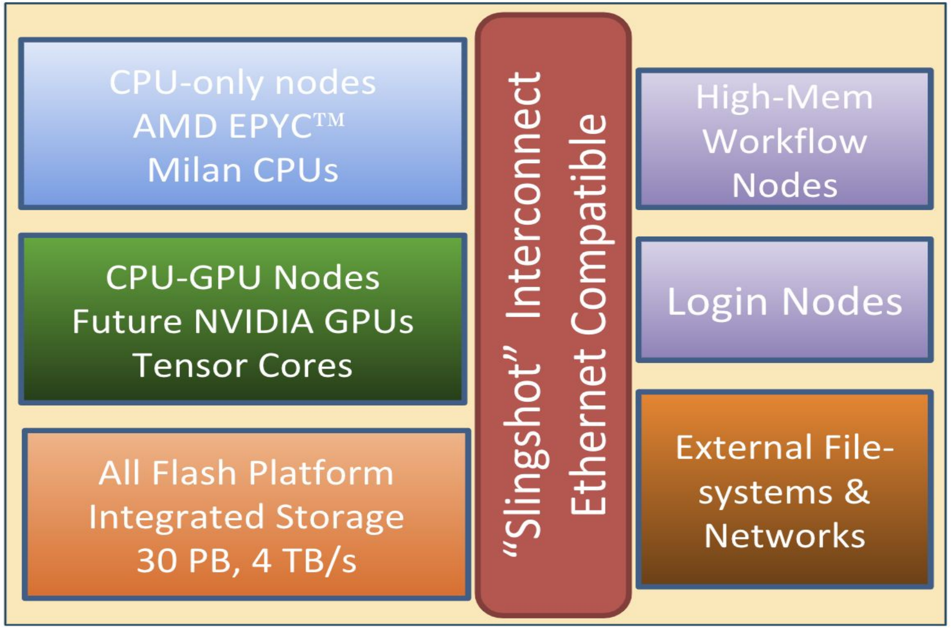
2016

2020

2024

Perlmutter: A System Optimized for Science

- GPU-accelerated and CPU-only nodes meet the needs of large scale simulation and data analysis from experimental facilities
- Cray “Slingshot” - High-performance, scalable, low-latency Ethernet-compatible network
- Single-tier All-Flash Lustre based HPC file system, 6x Cori’s bandwidth
- Dedicated login and high memory nodes to support complex workflows



5 ECP Apps to Integrated into NESAP

- ECP funded; selection occurred in partnership with ECP in Fall 2018.
- 15 Apps Applied, Reviewed by NERSC and ECP Staff. Priority given to apps beginning to or actively porting to GPUs
- Participation in NESAP funded by ECP HI Apps Integration at Facilities
- There will be additional overlap with codes that are part of ECP, but focus will be different from ECP efforts

PI Name	Institution	Application name	objective	Category
Yelick	LBNL	ExaBiome	DNA analysis of bio-communities	Data
Perazzo	SLAC	ExaFEL	real time, free-electron lasers	Data
Voter	LANL	EXAALT	fusion and fission materials on atomistic level	Simulation
Bhattacharjee	PPPL	XGC1, GENE	confined fusion plasma	Simulation
Vay, Almgren	LBNL	WarpX, AMReX	advanced particle accelerators	Simulation

Summary

NERSC Computing systems evolve with time

- RAM/CPU ratio will shrink
- Total available power imposes limitations on total compute
- New, energy efficient accelerators will dominate computing at scale
- Software/workflows will evolve to utilize new hardware