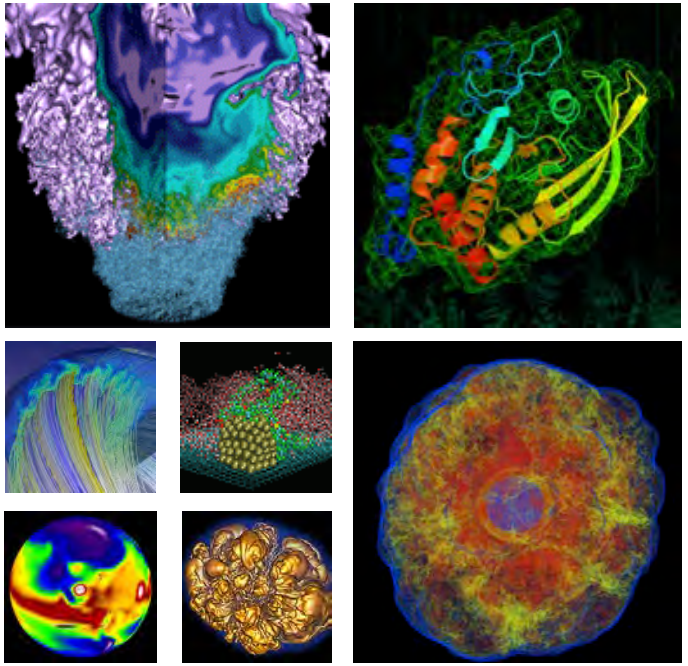
Parallel I/O at NERSC: Today and Tomorrow



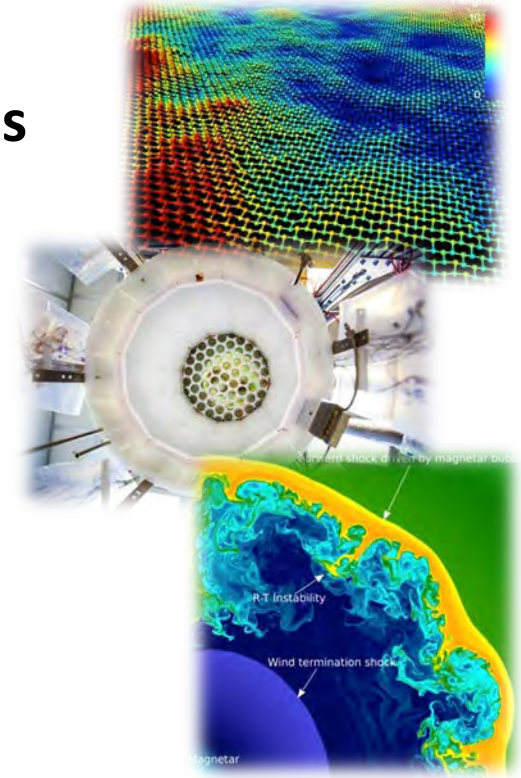
Glenn K. Lockwood, Ph.D.

Advanced Technologies Group

May 17, 2017

I/O at NERSC: Diverse

- **NERSC supports the broad mission needs of the six DOE Office of Science program offices**
  - > 6,900 active users
  - > 750 projects (> 1/3 involve experimental/observational data)
  - > 650 unique applications
- **Our users are productive**
  - > 2,000 referred publications in 2015
  - 2015 Nobel prize in physics supported by NERSC systems and archive



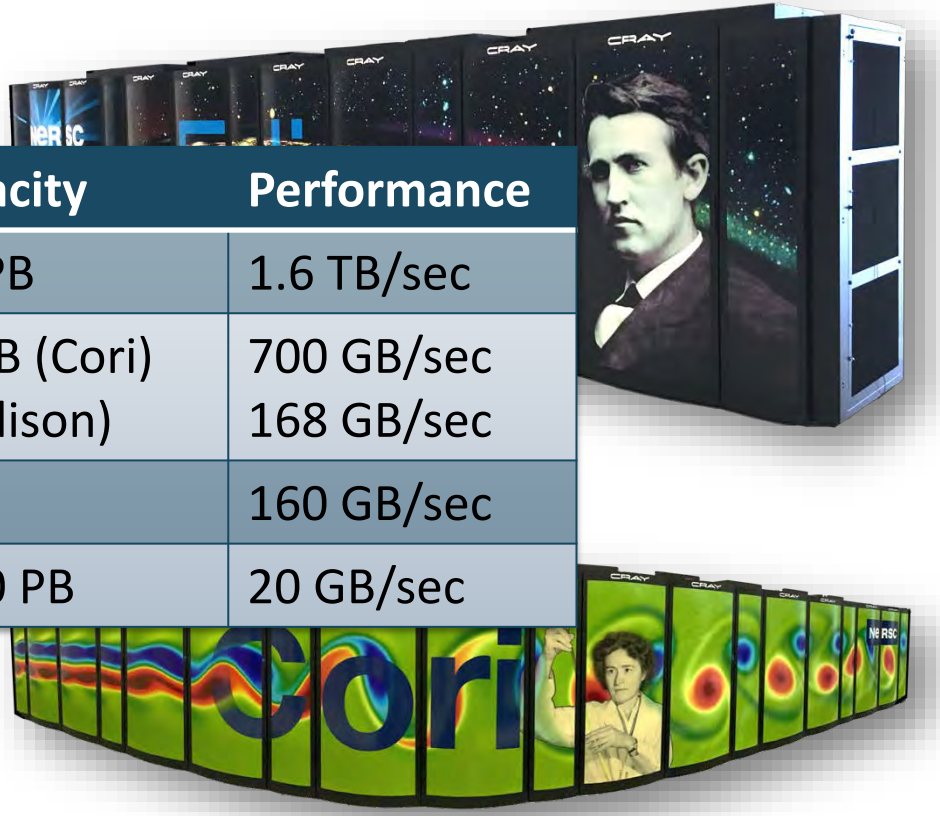
I/O at NERSC: Extreme-Scale

Very diverse sources of I/O

- 5,576 + 12,076 native Lustre clients
- > 15,000 jobs/day
- > 90% average utilization
- *all* job sizes

I/O at NERSC: Extreme-Scale

Tier	Technology	Capacity	Performance
Burst Buffer	DataWarp	1.8 PB	1.6 TB/sec
Scratch	Lustre	28 PB (Cori) 8 (Edison)	700 GB/sec 168 GB/sec
Projects	GPFS	7 PB	160 GB/sec
Archive	HPSS	>100 PB	20 GB/sec



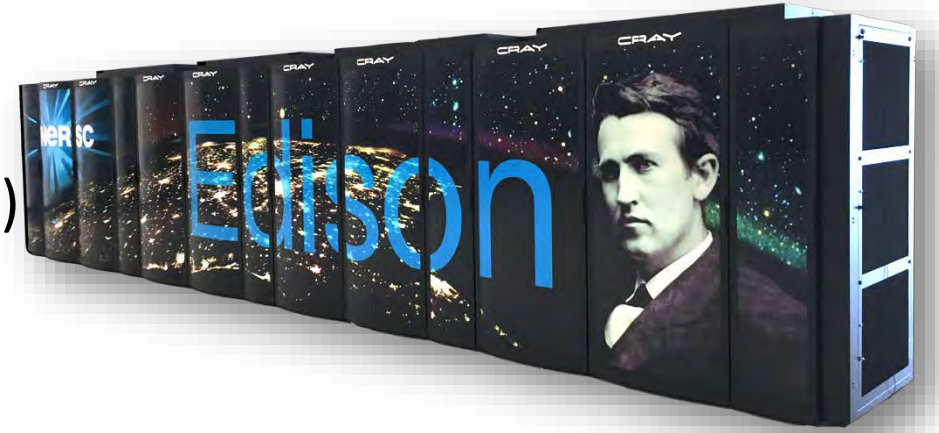
I/O at NERSC: Extreme-Scale

Lots of data movement (e.g., 5/14/2017)

- 900 TiB written (Cori scratch)
- 1,830 TiB read (Cori scratch)
- 181 TiB written (HPSS)
- 31 TiB read (HPSS)

Steady State

- 50/50 read/write (Lustre)
- 60/40 write/read (HPSS)
- > 40 years of data on HPSS



NERSC I/O vs. Hyperscale I/O

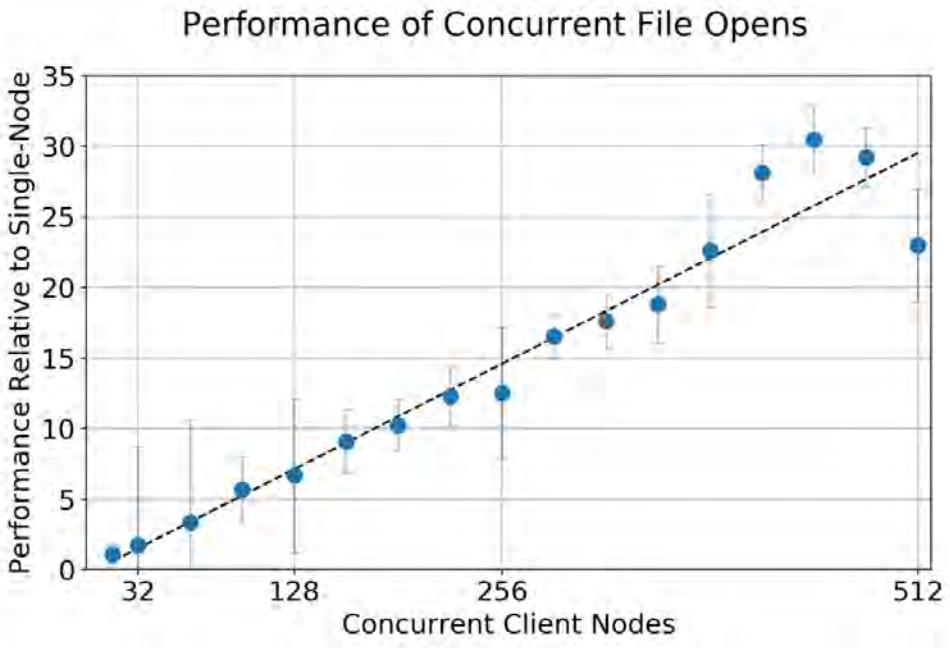
- **We are not WORM (50/50, 60/40 w/r ratios)**
  - SMR hasn't found its place yet
  - We do a *lot* more than checkpoint/restart
- **Primarily batch-oriented, so initial workflow latency is (usually) tolerable**
  - Rehydrating from tape can be OK; don't need to serve up the entire Netflix library on-demand
  - $/GB of tape still attractive for archive
- **Every day is Black Friday**
  - Intense bursts are common (as is incoherent parallel I/O)
  - Require low latency, high bandwidth, scalable metadata
- **Enterprise solutions are often an imperfect fit for HPC**

Where do we go for the future?

- **Trade POSIX for performance**
- **Specifically trade *POSIX consistency semantics* for lower latency**
- **Keep *POSIX API* for compatibility**
- **Assume that I/O workloads are already well behaved**

Trade POSIX for Performance

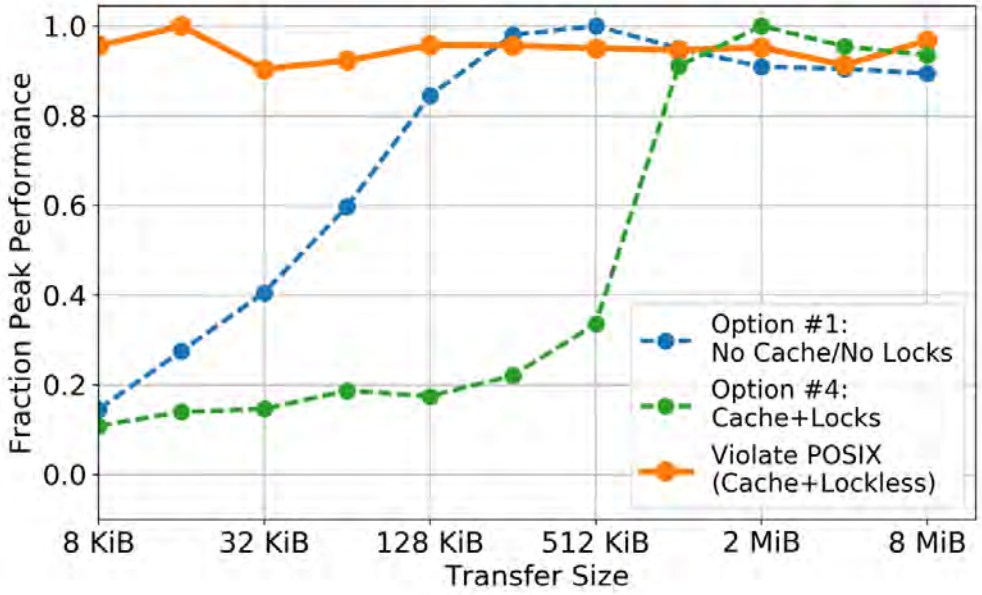
- **Throw out stateful I/O under the hood**
- **Stateless PUTs/GETs**
- **Use on userspace I/O to drive down latency**



Trade POSIX for Performance

- **Don't guarantee *POSIX consistency semantics* by default**
- **Bad code can use compatibility to get bad performance (cf. KNL ISA compatibility)**
- **Don't make good code pay for bad codes' sins**

Performance Impact of Strong Consistency



Thank you – thoughts?