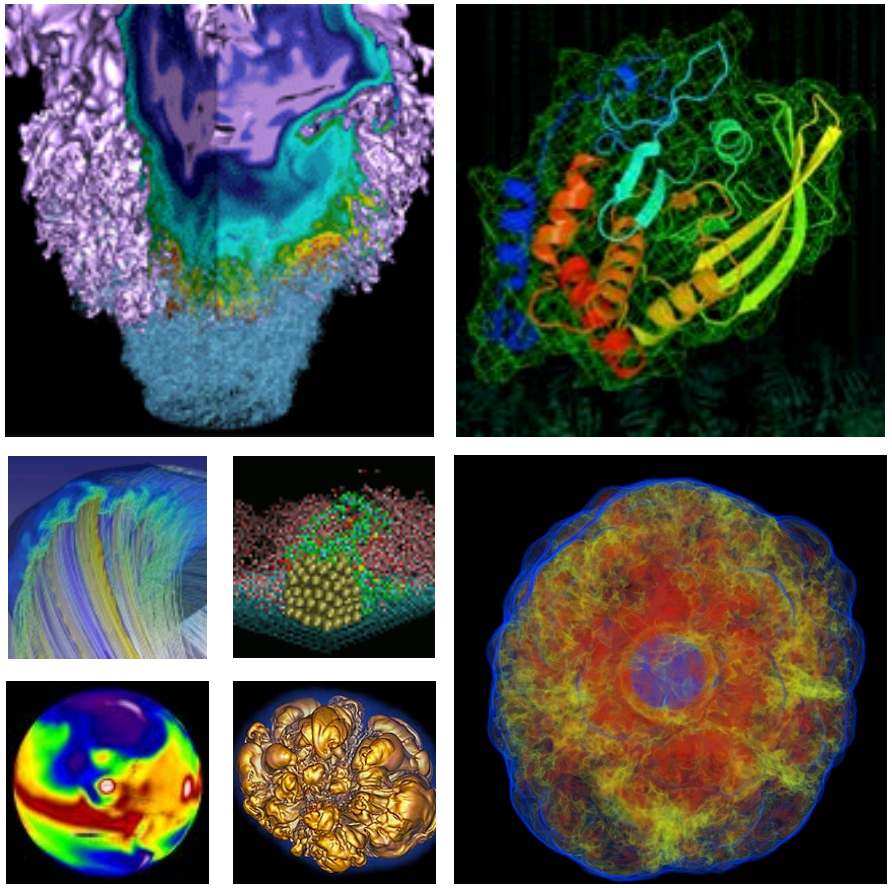
Containers in HPC



Shane Canon
IDEAS
Data & Analytics Group, NERSC

February 5, 2019

Contents

- What are containers
- Why Containers
- Containers versus VMs
- What is Docker
- What is an image?
- Creating images

The Struggles

- **My software doesn't build on this system…**
- **I'm missing dependencies…**
- **I need version 1.3.2 but this system has version 1.0.2..**
- **I need to re-run the exact same thing 12 months from now…**
- **I want to run this exact same thing somewhere else…**
- **I want my collaborators to have the same exact software as me…**
- **I've heard about these Containers, can I just run that?**
- **Can I run docker on this HPC system?**

Solution - Containers

What are Containers?

- **Uses a combination of Kernel “cgroups” and “namespaces” to create isolated environments**
- **Long history of containers Solaris Zones (2005), LXC(2008), LMCTFY/Google and then Docker(2013)**
- **Docker provided a complete tool chain to simplify using containers from build to run.**
- **Entire ecosystem has grown around containers especially around orchestration.**

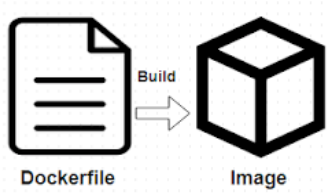
Docker Basic's

Build

Ship

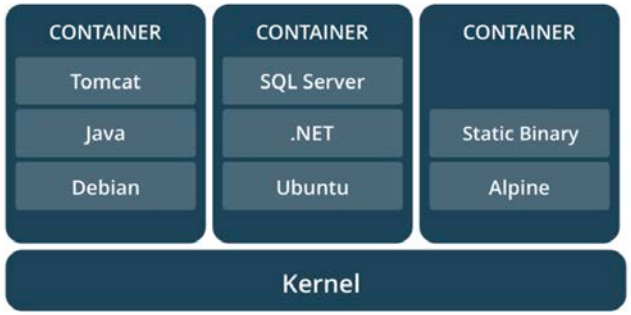
Run

- **Build images that captures applications requirements.**
- **Manually commit or use a recipe file.**
- **Push an image to DockerHub, a hosted registry, or a private Docker Registry.**
- **Share Images**
- **Use Docker Engine to pull images down and execute a container from the image.**



Why Containers?

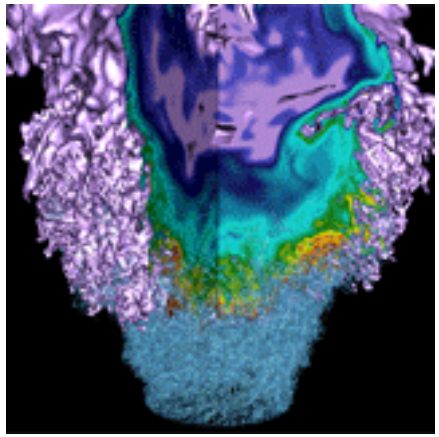
- **Light weight, executable piece of software that contains everything you need to run it**
  - Code, system libraries and tools, environment, settings
- **All software and processes are isolated from their surroundings**
- **Portable**
- **Typically used for single instance programs**

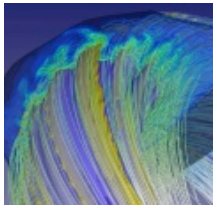


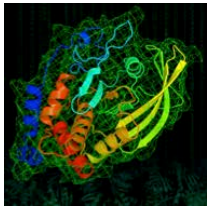
Containers and Science

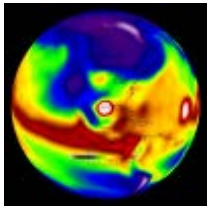
- **Reproducibility**
  - Everything you need to redo a scientific analysis
  - Image manifest contains all information about environment
    - Scripts, portable input files can be managed with version controller for greater control
- **Portability**
  - Runs on every system
- **Reduction of Effort**
  - Compile takes 10 hours? Just do it once and share it with everyone
  - System doesn't have the right library version? Yum install it yourself in the container

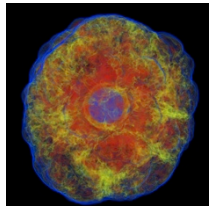
Containers in Action - Demo

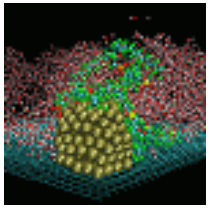




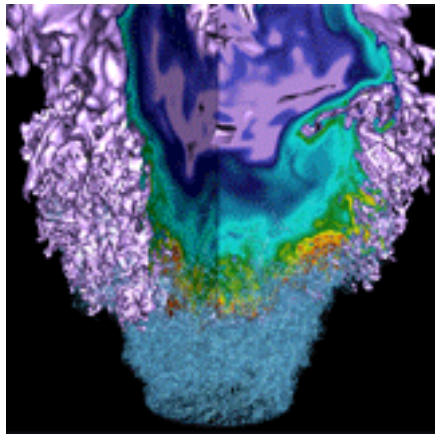


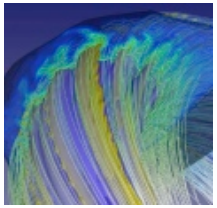


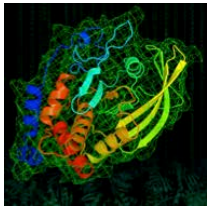


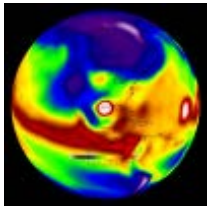


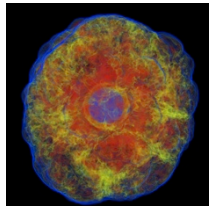
HPC Container Runtimes

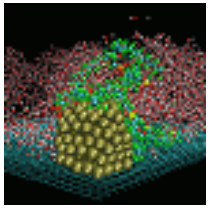






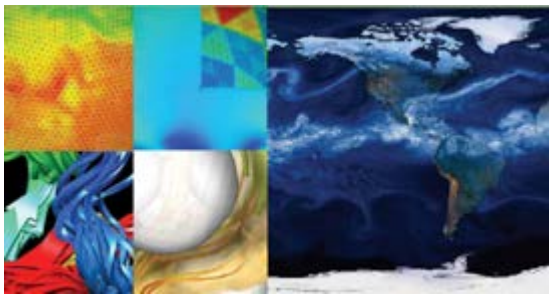


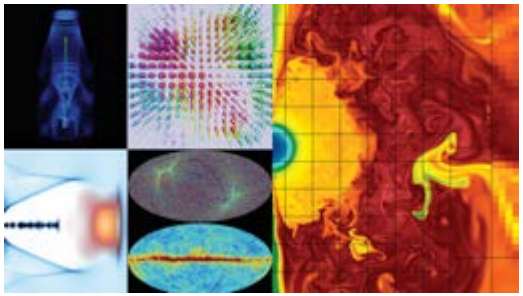


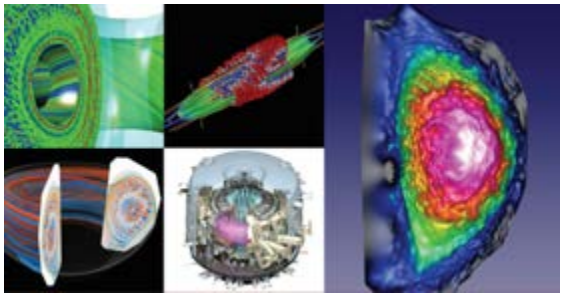


Why Containers at NERSC

- **NERSC deploys advanced HPC and data systems for the broad Office of Science community**
- **Approximately 6000 users and 750 projects**
- **Growing number of users around Analyzing Experimental and Observational Data, "Big Data" Analytics, and Machine Learning**
- **Shift towards converged systems that support traditional modeling and simulation workloads plus new models**







Why not just run Docker

- **Security: Docker currently uses an all or nothing security model. Users would effectively have system privileges**

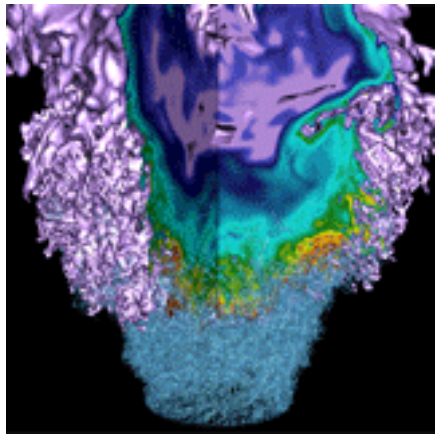
```
> docker run -it -v /:/mnt --rm busybox
```

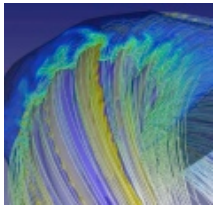
- **System Architecture: Docker assumes local disk**
- **Integration: Docker doesn't play nice with batch systems.**
- **System Requirements: Docker typically requires very modern kernel**
- **Complexity: Running real Docker would add new layers of complexity**

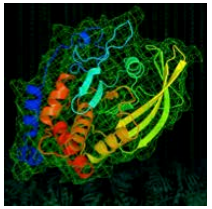
Solution: Shifter

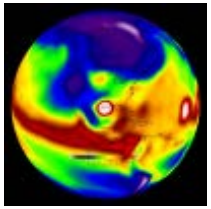
- **Design Goals:**
  - User independence: Require no administrator assistance to launch an application inside an image
  - Shared resource availability (e.g., file systems and network interfaces)
  - Leverages or integrates with public image repos (i.e. DockerHub)
  - Seamless user experience
  - Robust and secure implementation
- **Hosted at GitHub:**
  - https://github.com/nersc/shifter

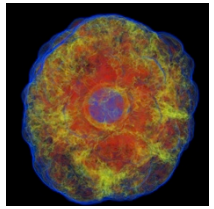
Implementation

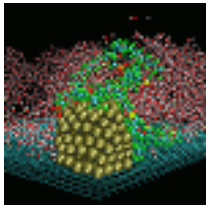






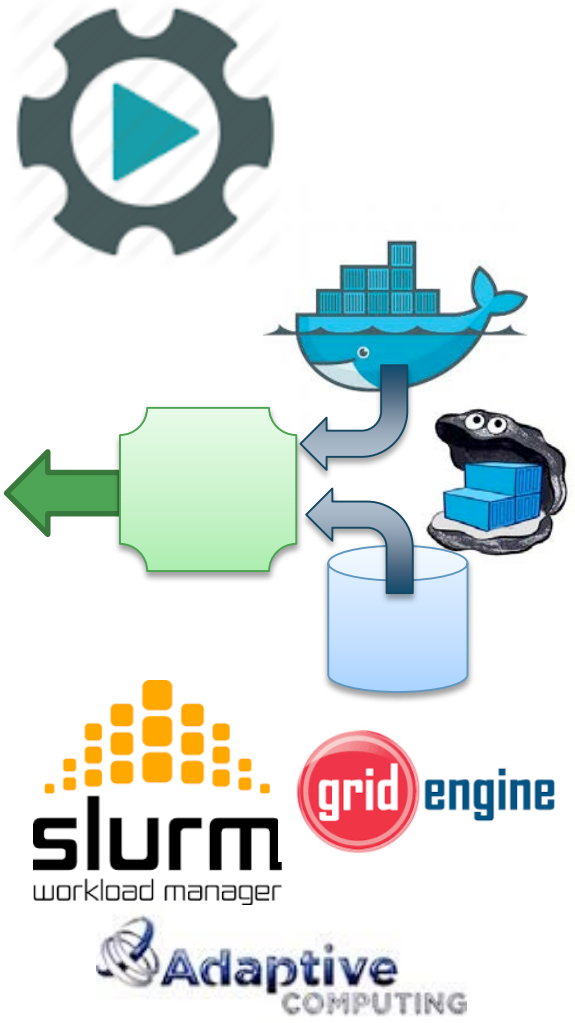




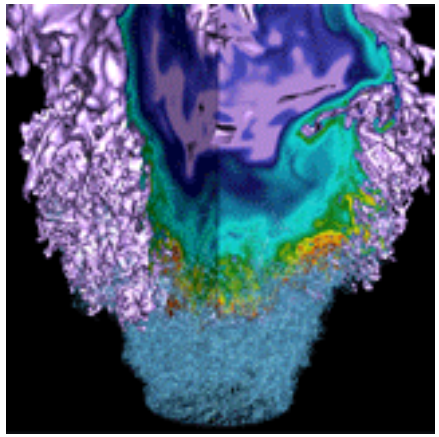


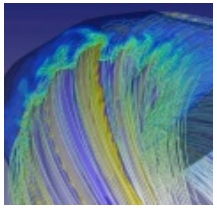
Shifter Components

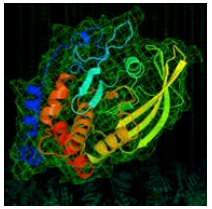
- **Shifter Image Gateway**
  - Imports and converts images from DockerHub and Private Registries
- **Shifter Runtime**
  - Instantiates images securely on compute resources
- **Work Load Manager Integration**
  - Integrates Shifter with WLM

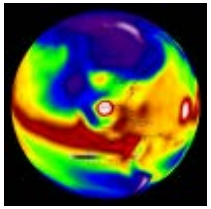


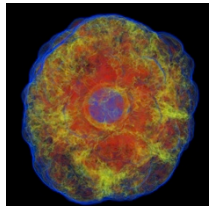
Shifter in Action

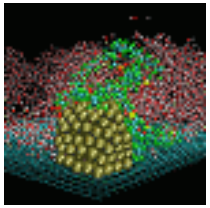












Usage slide

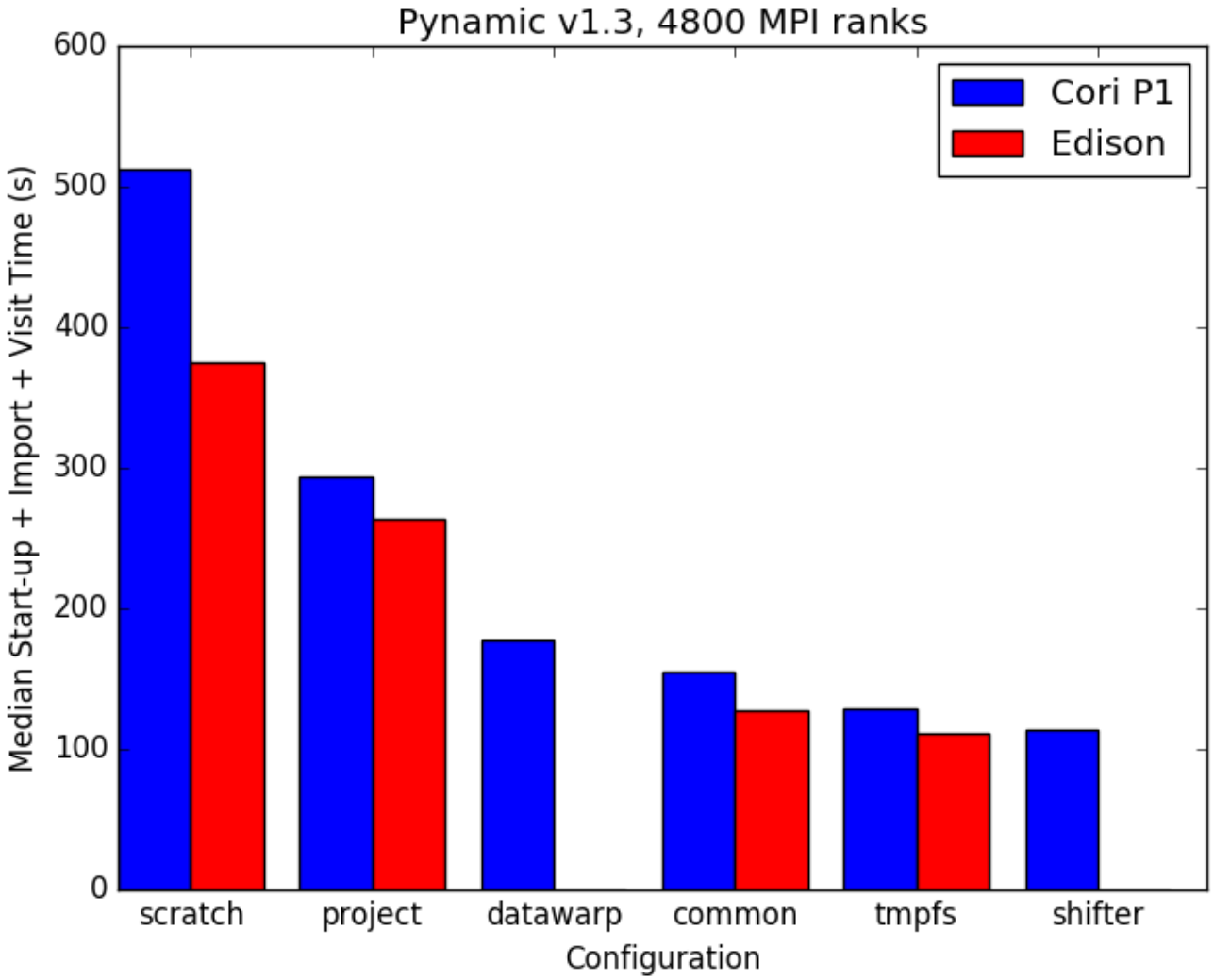
- **Use shifterimg pull to pull images from a registry**
  - Only do this once or after an update

```
> shifterimg pull ubuntu:14.04
```

- **Use shifter command to run a container with an image**

```
> shifter --image=ubuntu:14.04 bash
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.5 LTS
Release:  14.04
Codename:     trusty
```

Shifter accelerates Python Apps



Why?

- **Python must walk through the python libraries to construct the namespace**
- **Python must load up (read) any dynamic libraries that are required**
- **The loader must traverse the LD_LIBRARY_PATH to find the libraries to load**
- **Result: Lots of metadata accesses which put a load on the file system Metadata server**

Shifter and MPI

- **In Image**
  - Add required libraries directly into image.
  - Users would have to maintain libraries and rebuild images after an upgrade.
- **Managed Base Image (Golden Images)**
  - User builds off of a managed image that has required libraries.
  - Images are built or provided as part of a system upgrade.
  - Constrained OS choices and a rebuild is still required.
- **Volume Mounting**
  - Applications built using ABI compatibility.
  - Appropriate libraries are volume mounted at run time.
  - No rebuild required, but may not work for all cases.

Running an MPI Job – Building Image

```
FROM nersc/mpi-ubuntu:14.04

ADD . /app
RUN cd /app && \
    mpicc –o hello helloworld.c
```

Dockerfile

```
> docker build –t scanon/hello .
> docker push scanon/hello
```

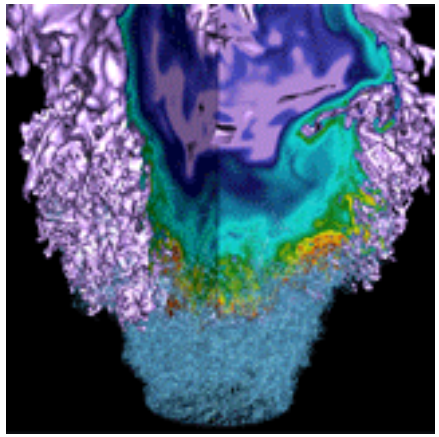
Running an MPI Job – Submit and run

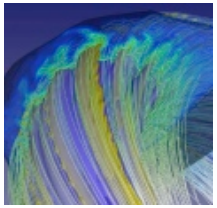
submit.sl

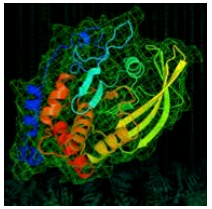
```
#!/bin/sh
#SBATCH --image=scanon/hello
srun -np 10 shifter /app/hello
```

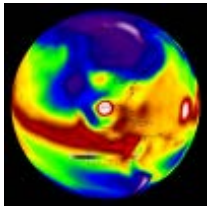
```
> sbatch submit.sl
```

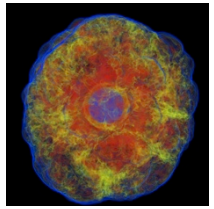
Related Work and Discussion

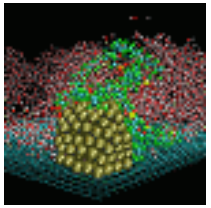












Other HPC Container Solutions

- **Singularity**
  - Very popular
  - Easy Installation
  - Runtime similar to Shifter
  - Native Image format in addition to Docker
  - Commercial company (Sylabs) now developing it
- **CharlieCloud**
  - Very light-weight
  - No special privileges required
  - Separate tools to unpack Docker images

How does Shifter differ from Docker?

Most Noticeable

- **Image read-only on the Computational Platform**
- **User runs as the user in the container – not root**
- **Image modified at container construction time (e.g. additional mounts)**

Less Noticeable:

- **Shifter only uses mount namespaces, not network or process namespaces**
- **Shifter does not use cgroups directly (integrated with the Workload Manager)**
- **Shifter uses individual compressed filesystem files to store images, not the Docker graph (slows down iterative updates)**
- **Shifter starts some additional services (e.g. sshd in container space)**

Why Users will like Shifter

Enables regular users to take advantage of Docker on HPC systems at scale.

This enables users to:

- **Develop an application on the desktop or laptop and easily run it on a cluster or Supercomputer**
- **Solve their dependency problems themselves**
- **Run the (Linux) OS of their choice and the software versions they need**

And…

- **Improves application performance in many cases**
- **Improves reproducibility**
- **Improves sharing (through sites like Dockerhub)**

Acknowledgements

Cray

NERSC – Benchmarks and Use Cases

Lisa Gerhardt, Rollin Thomas, Wahid Bhimji, Debbie Bard

Others – Contributors and Use Cases

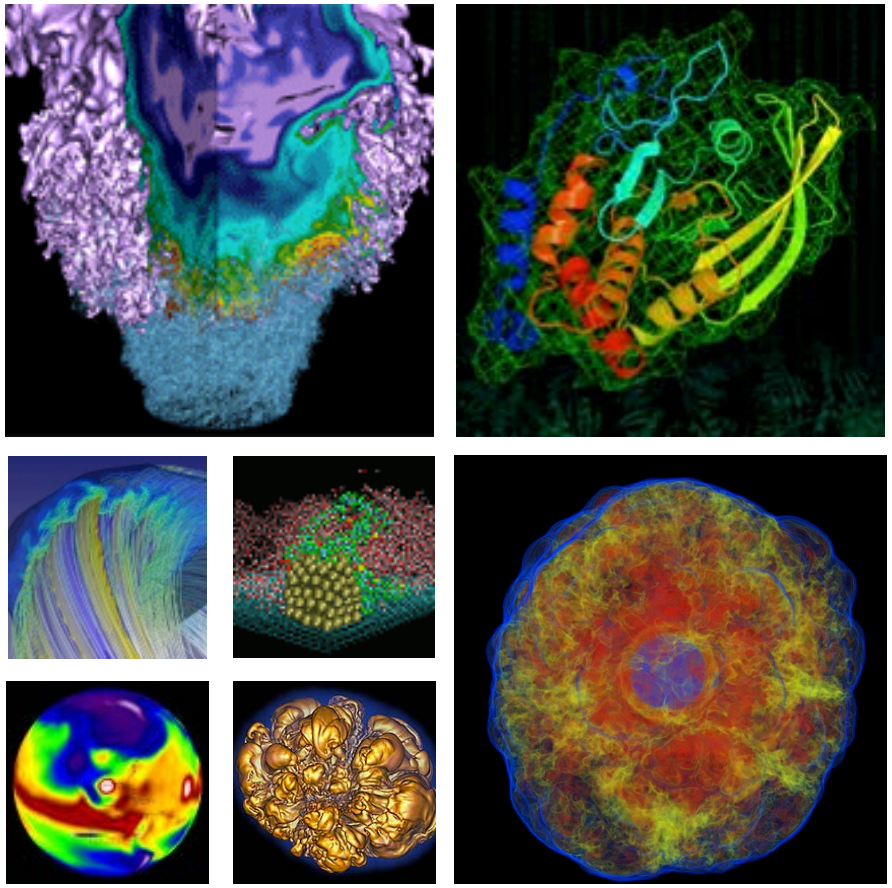
CSCS, Vakho Tsulaia, Ted Kisner

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research of the U.S. Department of Energy under Contract No. **DE-AC02-05CH11231**.

Summary

- **Shifter enables HPC systems to easily and securely run Containers even at large scale**
- **Shifter provides the flexibility of Docker without sacrificing security, scalability or performance.**
- **Shifter opens the door to the many benefits of Docker including easy sharing of images, reproducibility, etc.**

Shifter – Advanced Usage



Shane Canon

NERSC Data and Analytics Services

January 14, 2019

Volume Mounts

- **Volume Mounts provide a way to map external paths into container paths.**
- **This allows paths in the container to be abstracted so it can be portable across different systems.**
- **Basic syntax is:**

```
-volume <external path>:<container path>
```

- **Shifter places some constraints on what paths can be mapped and where they can be mapped for added security.**

Using Volume Mounts

```
anon@cori06:~> ls $SCRATCH/myjob
config   data.in

canon@cori06:~> shifter --image=ubuntu --volume=$SCRATCH/myjob:/data bash
canon@cori06:~$ ls /data/
config   data.in

canon@cori06:~$
```

PerNode Write Cache

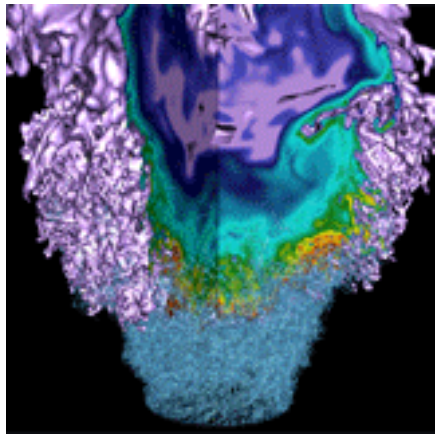
- **PerNodeWrite extends the volume concept to create temporary writeable space that aren't shared across nodes.**
- **These spaces are ephemeral (removed on exit)**
- **These are node local and the size can be adjusted**
- **Performs like a local disk but is more flexible**
- **Basic syntax is**

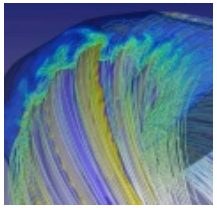
```
--volume <external path>:<container path>:perNodeCache=size=XXG
```

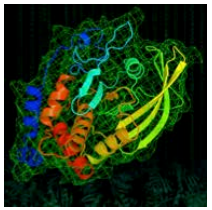
Using Volume Mounts

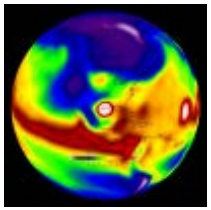
```
canon@cori06:~> shifter --image=ubuntu \
      --volume=$SCRATCH:/scratch:perNodeCache=size=100G /bin/bash
canon@cori06:~$ df -h /scratch/
Filesystem      Size  Used Avail Use% Mounted on
/dev/loop4      100G   33M  100G   1% /scratch
canon@cori06:~$ dd if=/dev/zero bs=1k count=10M of=/scratch/output
10485760+0 records in
10485760+0 records out
10737418240 bytes (11 GB, 10 GiB) copied, 22.2795 s, 482 MB/s
canon@cori06:~$ ls -lh /scratch/output
-rw-r--r-- 1 canon canon 10G Nov  9 23:38 /scratch/output
canon@cori06:~$ exit
exit
canon@cori06:~> shifter --image=ubuntu \
     --volume=$SCRATCH:/scratch:perNodeCache=size=100G /bin/bash
canon@cori06:~$ ls -l /scratch
total 0
```

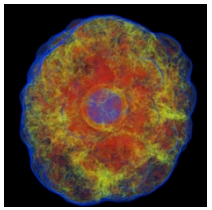
Optimizations

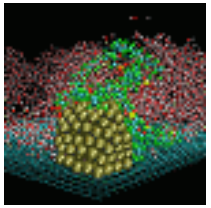












Dockerfile Best Practices

Bad:

```
RUN wget http://hostname.com/mycode.tgz
RUN tar xzf mycode.tgz
RUN cd mycode ; make; make install
RUN rm -rf mycode.tgz mycode
```

Good:

```
RUN wget http://hostname.com/mycode.tgz && \
 tar xzf mycode.tgz && \
 cd mycode && make && make install && \
 rm -rf mycode.tgz mycode
```

Dockerfile Best Practices

Bad:

```
RUN wget http://hostname.com/mycode.tgz ; \
 tar xzf mycode.tgz ; \
 cd mycode ; make ; make install ; \
 rm -rf mycode.tgz mycode
```

Good:

```
RUN wget http://hostname.com/mycode.tgz && \
 tar xzf mycode.tgz && \
 cd mycode && make && make install && \
 rm -rf mycode.tgz mycode
```

Dockerfile Best Practices

Bad:

```
ADD . /src

RUN apt-get update –y && atp-get install gcc

RUN cd /src && make && make install
```

Good:

```
RUN apt-get update –y && apt-get install gcc

ADD . /src

RUN cd /src && make && make install
```

Multi-Stage Builds

- **Added in Docker 17.05**
- **Allows a build to progress through stages**
- **Files can be copied from a stage to later stages**
- **Useful for splitting images between build and run-time to keep image sizes small**
- **Can be used to make public images that make use of commercial compilers**

Dockerfile – Multistage build

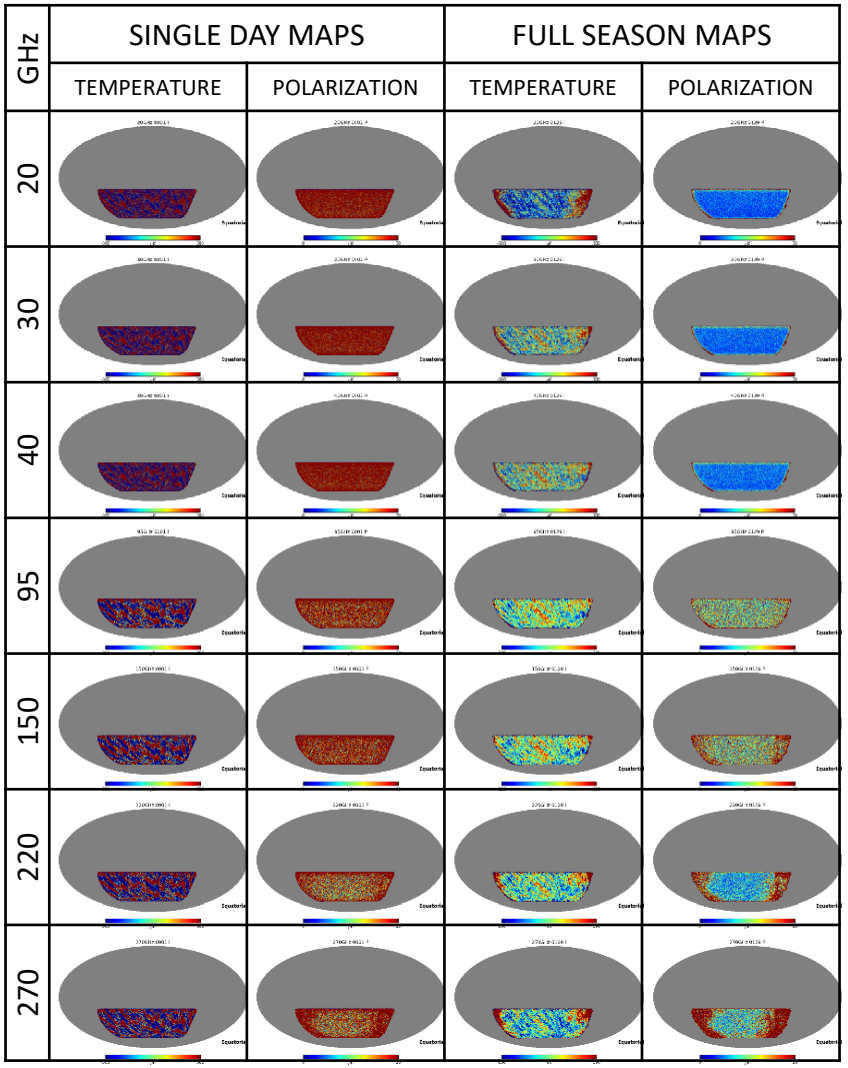
```
FROM centos:7 as build
RUN yum -y install gcc make
ADD code.c /src/code.c
RUN gcc -o /src/mycode /src/code.c

FROM centos:7
COPY --from=build /src/mycode /usr/bin/mycode
```

Other Considerations

- **Avoid very large images (> ~5 GB)**
- **Keep data in $SCRATCH and volume mount into the container if data is large**
- **Use volume mounts for rapid prototyping and testing, then add that into the image after code stabalizes**

Measuring the Composition of the Universe

- **CMB – S4**
  - Ambitious collection of telescopes to measure the remnants of the Big Bang with unprecedented precision
- **Simulated 50,000 instances of telescope using 600,000 cores on Cori KNL nodes.**
- **Why Shifter?**
  - Python wrapped code needs to start at scale



U.S. DEPARTMENT OF ENERGY | Office of Science

COMPUTATIONAL COSMOLOGY CENTER

BERKELEY LAB

Questions?

Create an image with Docker

Dockerfile

```
FROM ubuntu:14.04
LABEL maintainer="Shane Canon scanon@lbl.gov"
# Update packages and install dependencies
RUN apt-update –y && \
    apt-get install -y build-essential

# Copy in the application
ADD . /myapp
# Build it
RUN cd /myapp && \
    make && make install
```

```
> docker build –t scanon/myapp:1.1 .
```

Running an image with Docker

```
> docker run -it –rm scanon/myapp:1.1 /myapp/myapp
…
App Output
….
```

Popular features of a data intensive system and supporting them on Cori

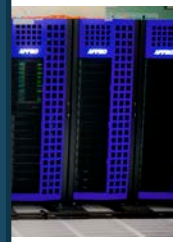
Data Intensive Workload Need	Cori Solution
Local Disk	NVRAM 'burst buffer' and ***Shifter***
Large memory nodes	128 GB/node on Haswell; Large memory login and service nodes
Massive serial jobs	NERSC serial queue
Complex workflows	***Shifter*** CCM mode Large Capacity of interactive resources
Communicate with databases from compute nodes	Advanced Compute Gateway Node
Stream Data from observational facilities	Advanced Compute Gateway Node
Easy to customize environment	***Shifter***
Policy Flexibility	Improvements coming with Cori: Rolling upgrades, CCM, above COEs would also contribute

Converging Data Intensive Systems and HPC

Compute Intensive

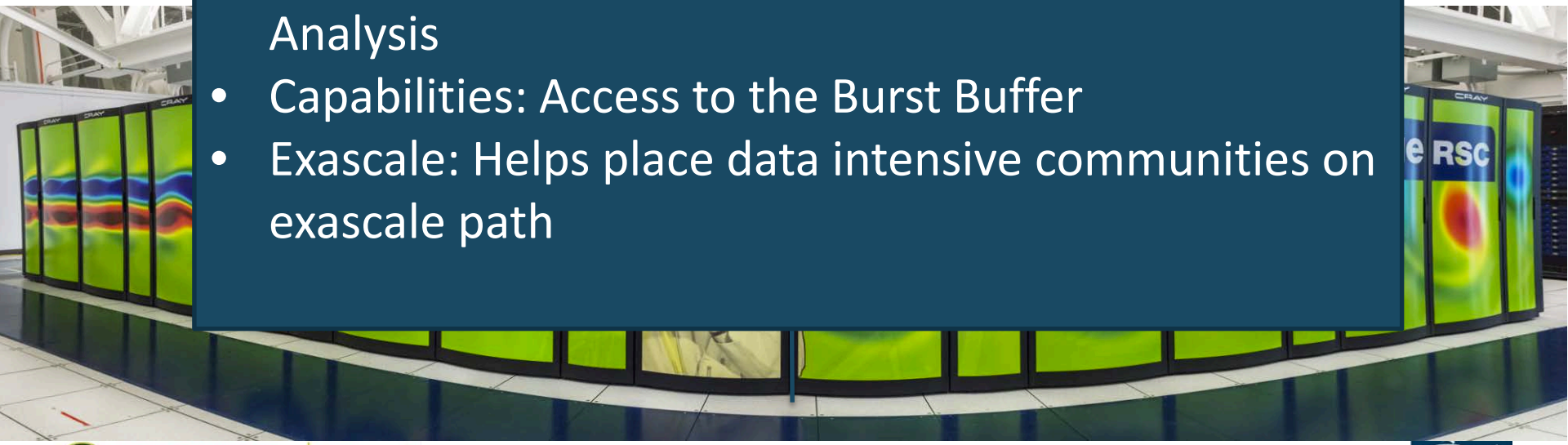
Data Intensive

Carver



Why Convergence?

- Scale: Cori will have the scale needed to tackle current and emerging data challenges
- Coupling: Increasing Need to Couple Simulation and Analysis
- Capabilities: Access to the Burst Buffer
- Exascale: Helps place data intensive communities on exascale path



What's in an Image

- **Directory tree**
  - Base Linux OS
  - Libraries, binaries, tools, scripts, etc
  - User code
  - Data
- **Run-time Settings**
  - Environment variables
  - Working Directory
  - Default execution and parameters
- **Other things**
  - Network-related (e.g. ports)
  - Run User

Singularity

- **Can pull and use Docker images. Converted to a singularity file.**
- **Has its own recipe file and build tools**
- **Images are flat (no layers)**
- **Images can be copied like regular files**

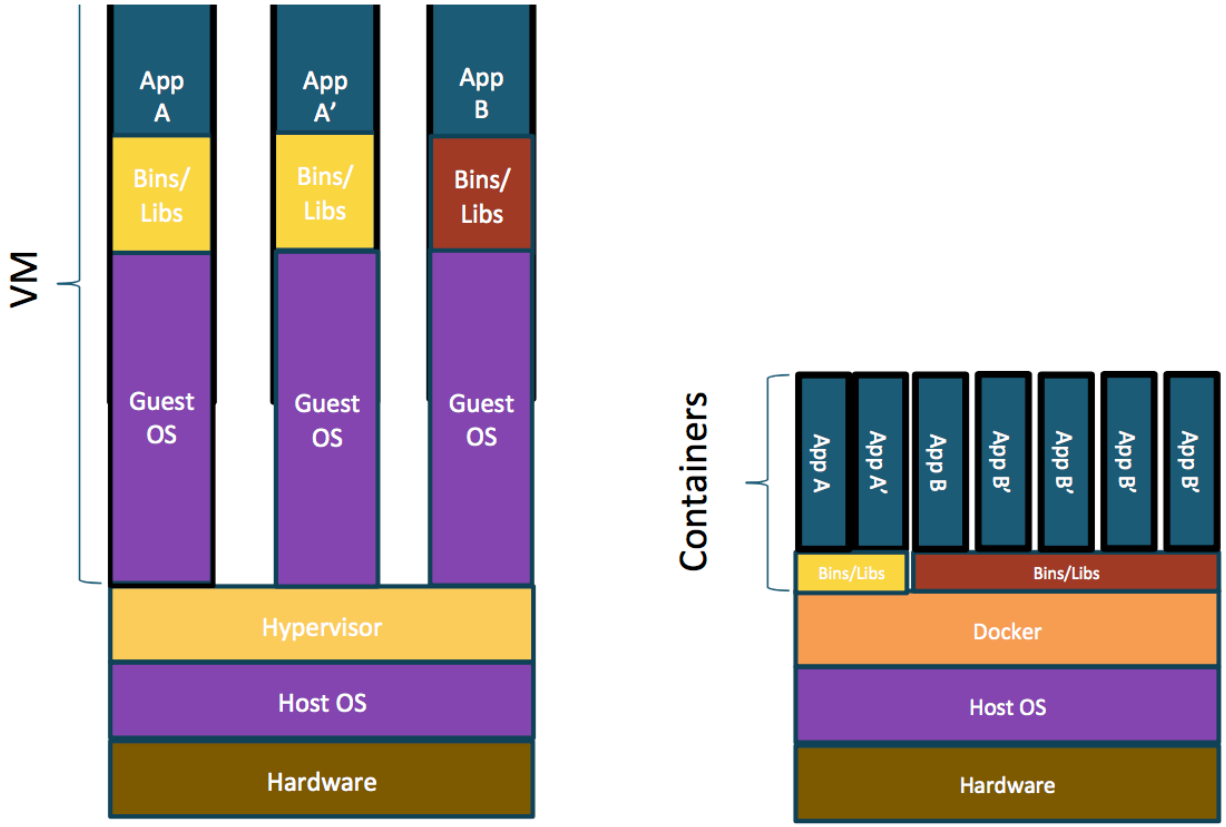
Singularity Recipe File Example

```
Bootstrap: docker
From: ubuntu

%help
Example Singularity Image

%files
    script.sh /script.sh

%labels
    Maintainer I. M. Maintainer
    Version v1.0

%environment
    FOO=bar
    export FOO

%post
    apt-get update -y
    apt-get install -y curl
    echo 'export BAR=blah' >> $SINGULARITY_ENVIRONMENT

%runscript
    exec /script.sh
```

Singularity

```
> singularity build myimage.simg Singularity
```

Containers versus VMs



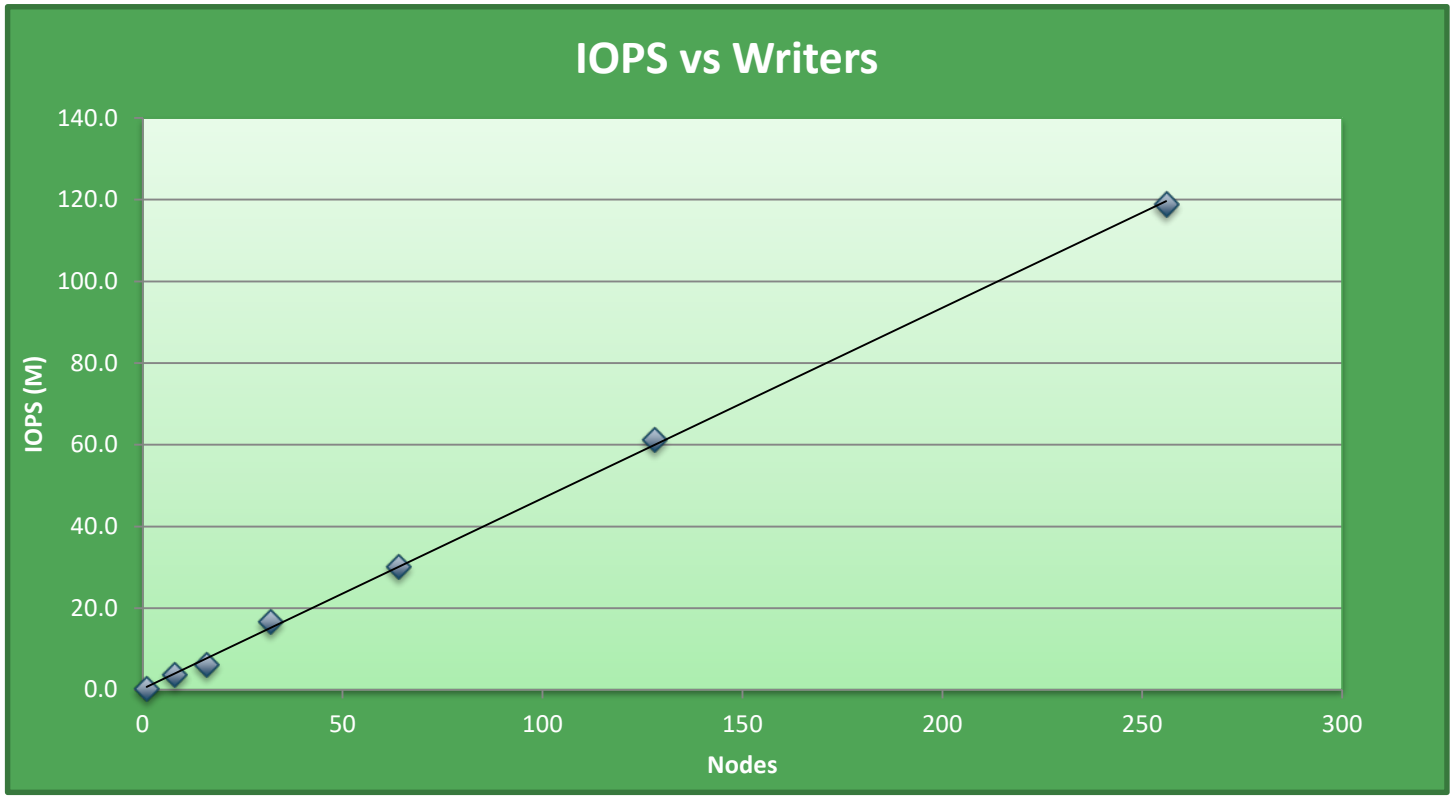
Singularity Execution

```
> singularity shell myimage.simg
Singularity myimage.simg:~>

> singularity run myimage.simg
Hello World

> singularity shell docker://ubuntu:latest
 Singularity ubuntu:~>
```

Charliecloud

- https://github.com/hpc/charliecloud
- **No root privs required for installation**
- **Does require user namespace report (increasingly common now)**
- https://hpc.github.io/charliecloud/index.html

Per-Node Write Cache (IOPS)



Results of an IOR File per-process, 2 tasks per node, 512B transfer size, 2GB write. 100x faster than Lustre at the same scale.