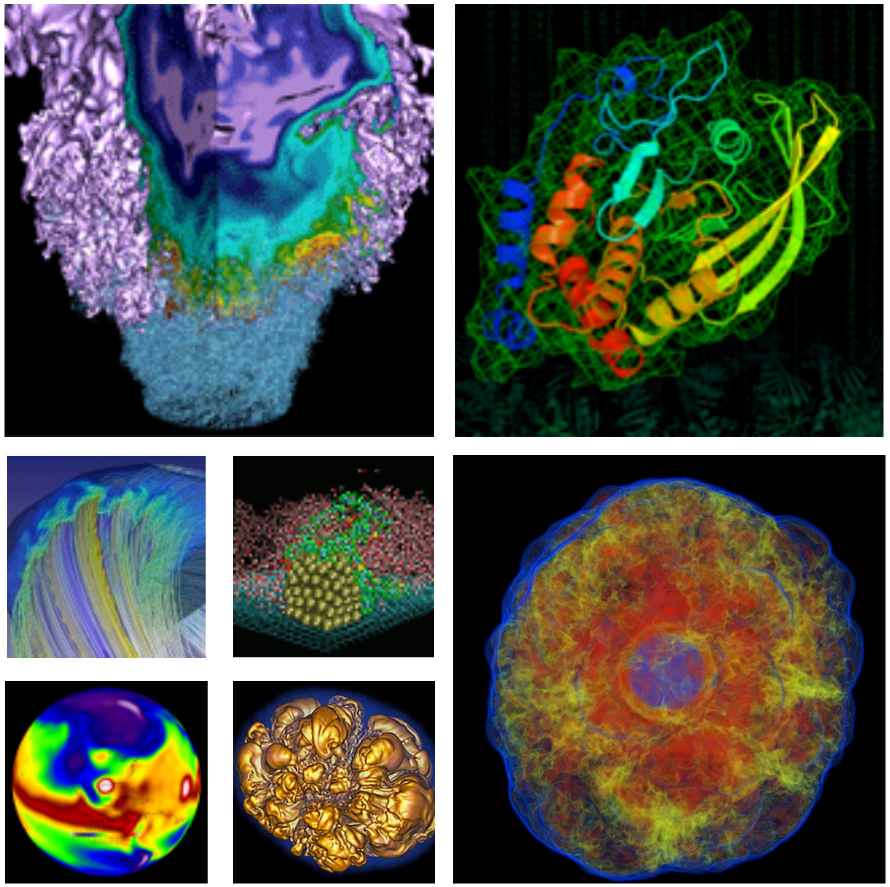
Shifter: Containers for HPC



Shane Canon & Doug Jacobsen
NERSC/Lawrence Berkeley Lab

Cray User Group 2016 May 12, 2016

Agenda

- Motivation and Background
- Shifter Architecture and Design
- Shifter in Action
- Discussion and Future Work

Acknowledgements

Cray – UDI COE and technical input
Dani Conner, Martha Dumler, Rose Olson,
Dave Henseler, Dean Roe, Don Bahls, Bill Sparks
Kitrick Sheets, Alan Mutschelknaus, Andrew Barry

NERSC – Benchmarks and Use Cases
Lisa Gerhardt, Rollin Thomas, Wahid Bhimji, Debbie Bard

Others – Contributors and Use Cases
Miguel Gila (CSCS), Vakho Tsulaia

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**.

Convergence of Disruptive Technology

- Increasing Role of Data
- Converging HPC and Data Platforms
- New Models of Application Development and Delivery

DOE Facilities Require Exascale Computing and Data

Astronomy

Particle Physics

Chemistry and Materials

Genomics

Fusion

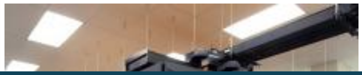
Petascale to Exascale

- **Petabyte data sets today, many growing exponentially**
- **Processing requirements grow super-linearly**
- **Need to move entire DOE workload to Exascale**

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

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

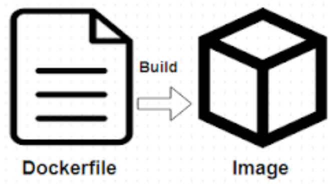
Data Intensive Workload Need	Cori Solution
Local Disk	NVRAM 'burst buffer' and ***Shifter***
Large memory nodes	128 GB/node on Haswell; Large memory login and service nodes
Massive serial jobs	NERSC serial queue
Complex workflows	***Shifter*** CCM mode Large Capacity of interactive resources
Communicate 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

Docker Basic's

- 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.



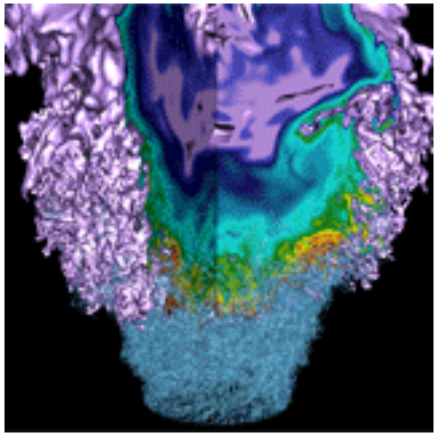
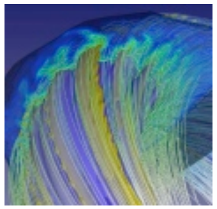
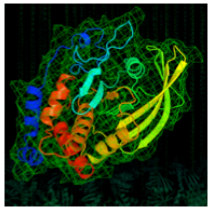
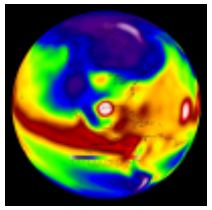
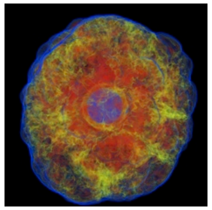
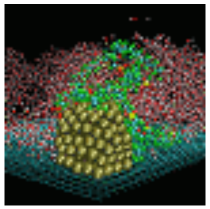
Why not just run Docker

- System Architecture: Docker assumes local disk
- Security: Docker currently uses an all or nothing security model. Users would effectively have system privileges
- 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

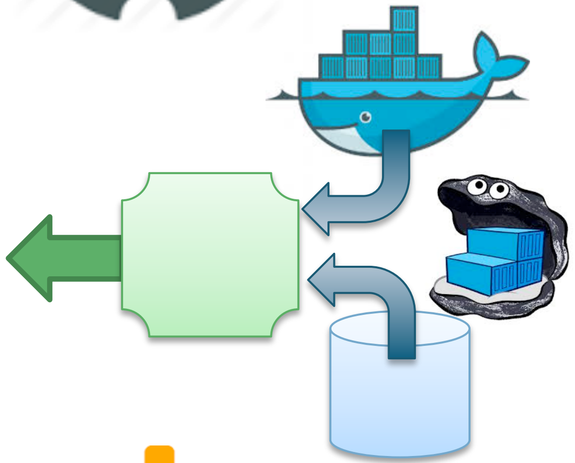
- **Partnership with Cray to design a solution to run containers on an HPC platform.**
- **Design Goals:**
  - User independence: Require no administrator assistance to launch an application inside an image
  - Shared resource availability (e.g., PFS/DVS mounts and network interfaces)
  - Leverages or integrates with public image repos (i.e. DockerHub)
  - Seamless user experience
  - Robust and secure implementation

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 Architecture and Flow

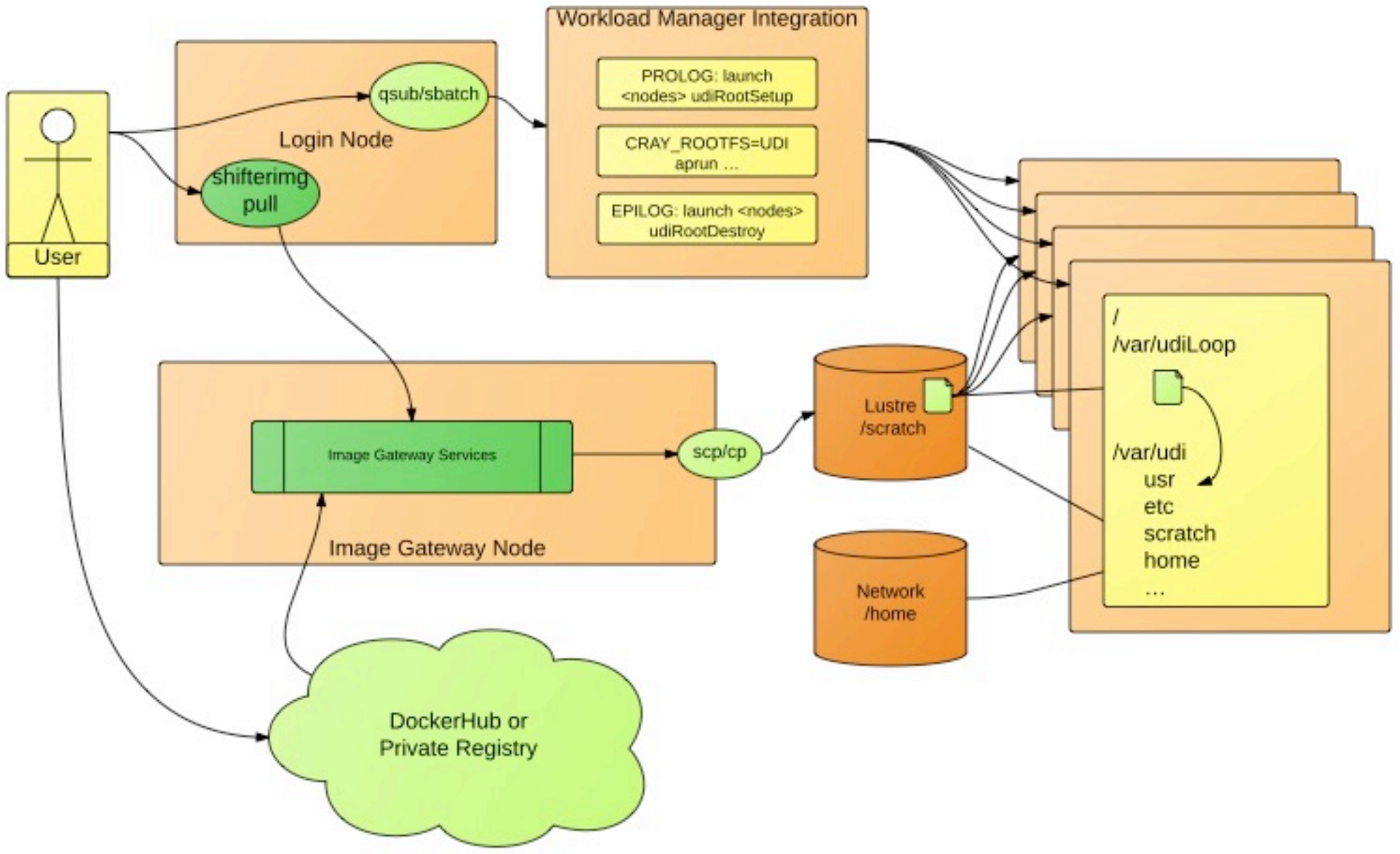
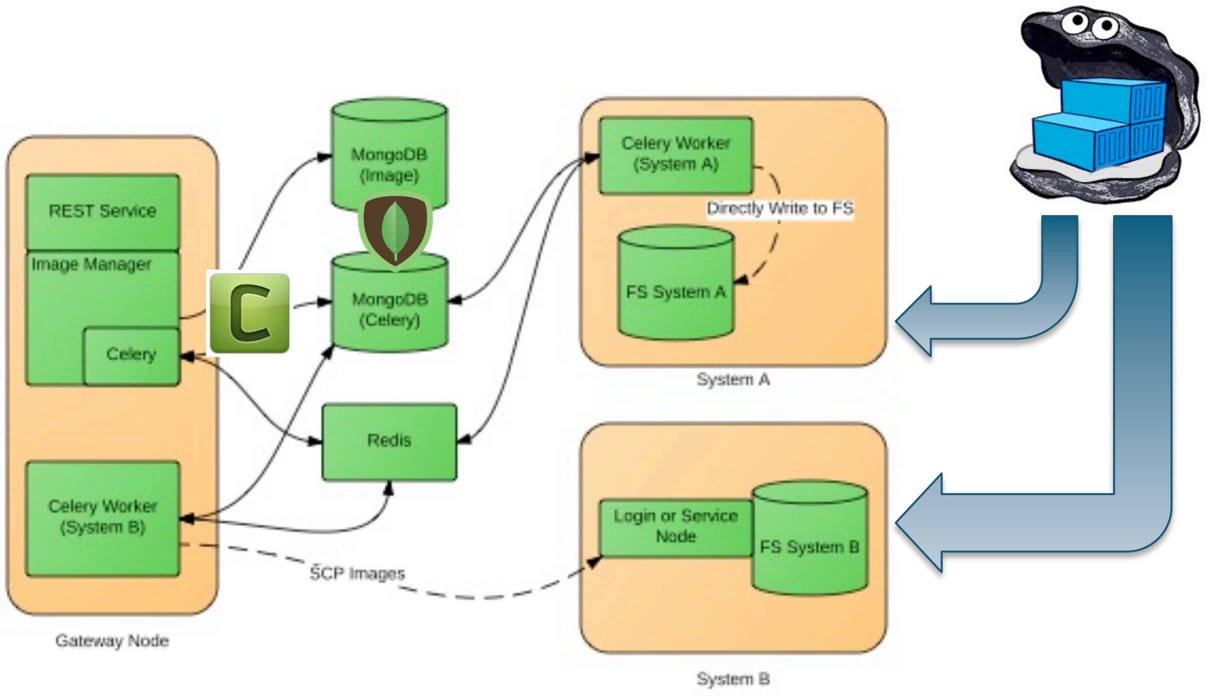


Image Gateway Design

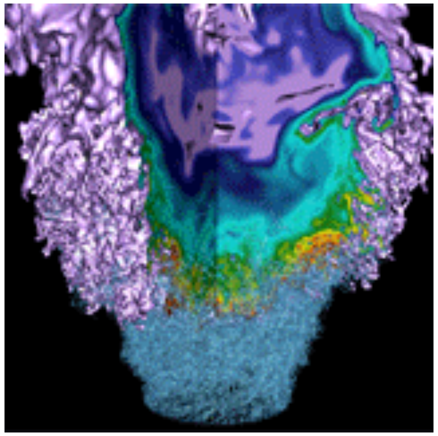


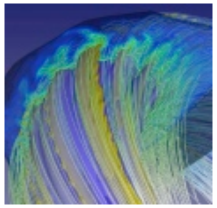
- Python Flask Application provides REST interface
- Mongo Database stores image metadata and provides an index of available images
- Python Celery provides a distributed queueing system
- Celery "Workers" do the actual image manipulation including pulling Docker Images from DockerHub or Registries

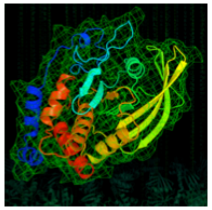
Workload Integration - Slurm

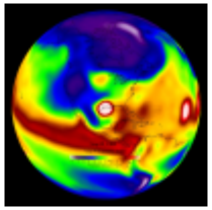
- Custom Plugin using the Spank plugin architecture
- Allows users to specify images, volumes and other options directly in the batch submission.
- Extensions will pre-create a common Shifter area (best for MPI applications).
- See Doug for more details.

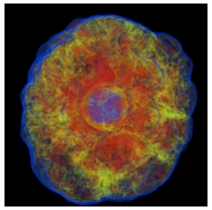
Shifter in Action

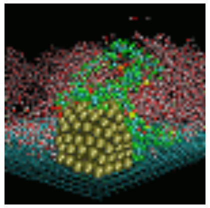












Create an image with Docker

Dockerfile

```
FROM ubuntu:14.04
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 .
> docker push scanon/myapp:1.1
```

Use the Image with Shifter

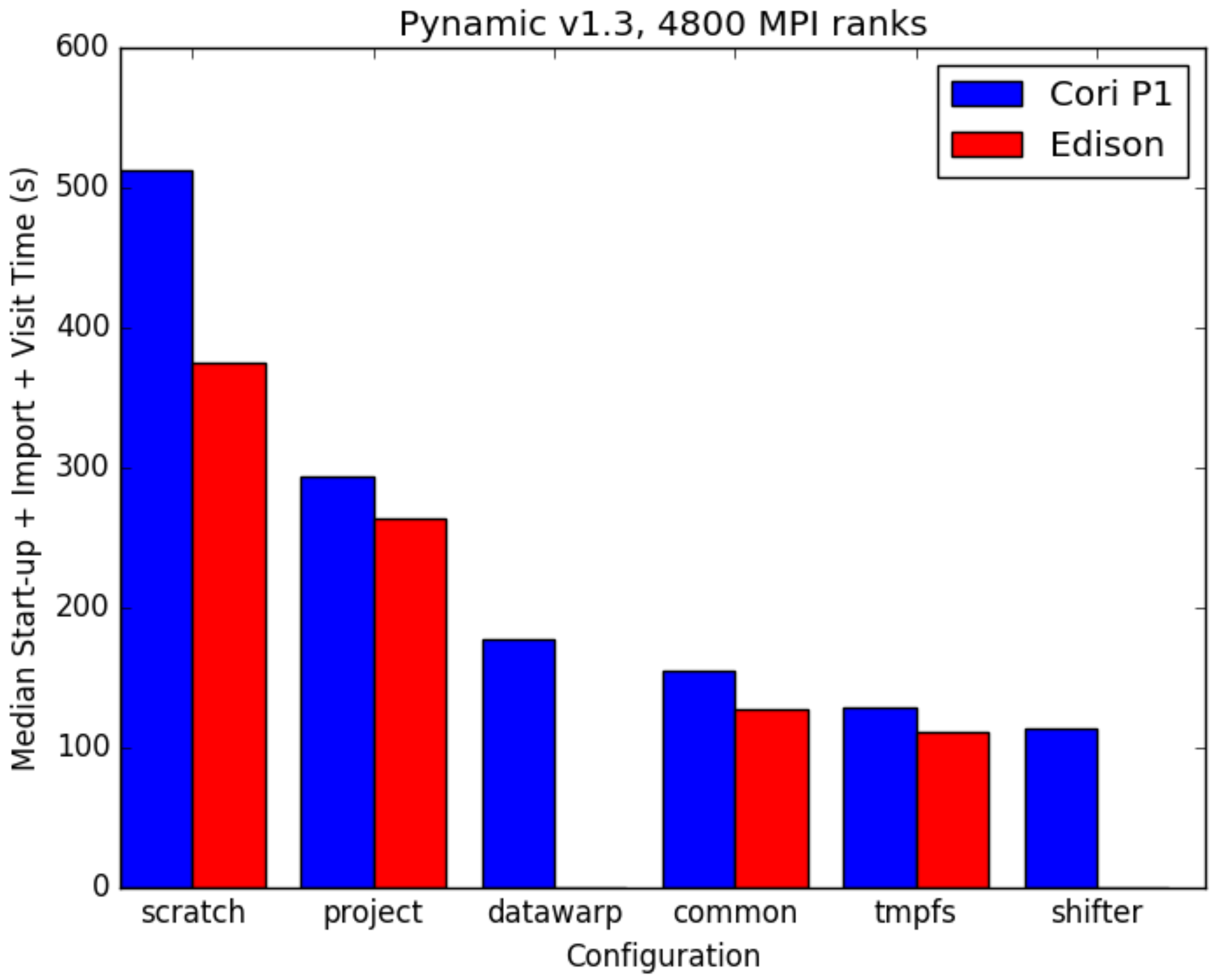
Submit script

```
#!/bin/bash
#SBATCH -N 16 -t 20
#SBATCH --image=docker:scanon/myapp:1.1
#SBATCH --volume=/global/cscratch1/sd/canon/
backingFile:/mnt:perNodeCache=size=100G


module load shifter
export TMPDIR=/mnt
srun -n 16 shifter /myapp/app
```

```
> shifterimg pull docker:scanon/myapp:1.1
> sbatch ./job.sl
```

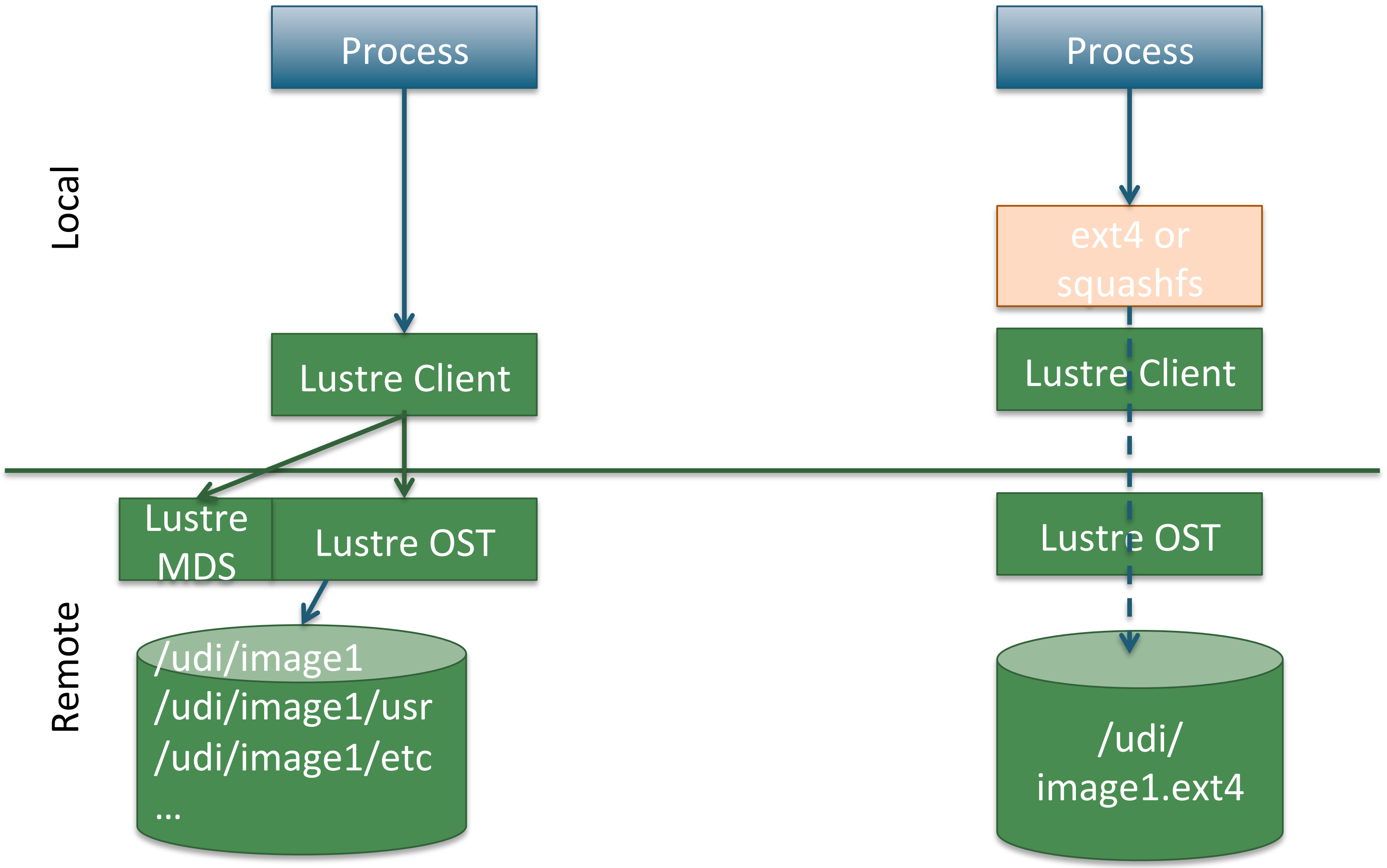
Shifter accelerates Python Apps



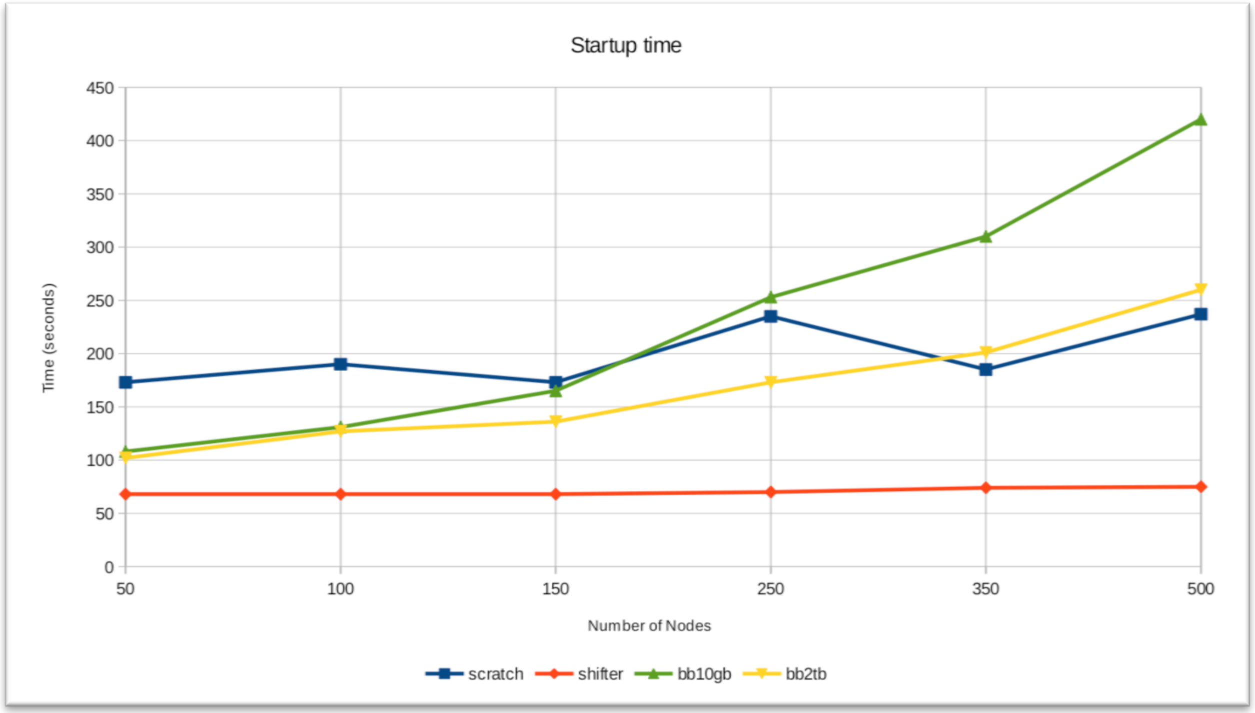
Why?

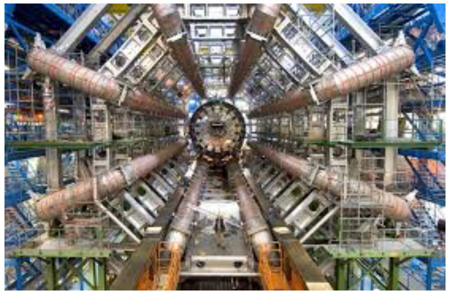
- Python must walk through the python libraries to construct the namespace
- Python must load up any dynamic libraries that are required
- The loader must traverse the LD_LIBRARY_PATH to find the libraries to load

File System flow – Traditional vs Shifter



Shifter and Atlas





- ATLAS software built and maintained by the international collaboration.
- Makes heavy use of "CVMFS" a software distribution system.
- Complete ATLAS CVMFS distro is O(TB) in size.
- Shifter provides linear startup times and requires no additional integration to run on the Cray systems.
- Images range from ~200GB to 3 TB!!

Per-Node Write Cache (New Feature)

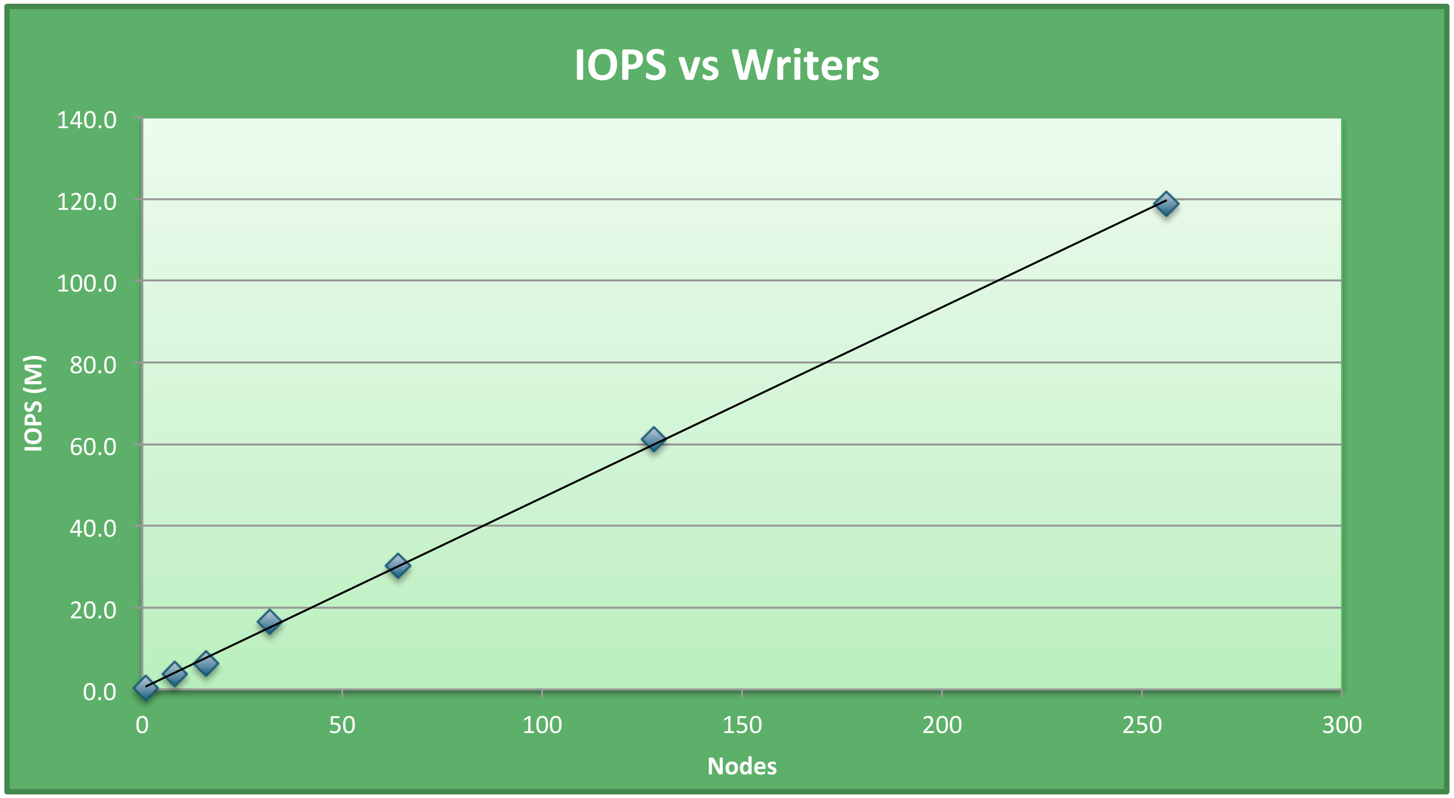
Per-Node Write Cache provides local disk like functionality but is backed by the Parallel File System.

Nodes/Writers per node	Lustre (MB/s) per writer	Shifter (MB/s) per writer	Real Local Disk (MB/s) per writer
1/1	83	594	416
10/10	87	625	416*
10/20	67	616	165*
10/40	55	589	53*
20/40	71	627	165*
20/80	55	588	53*

Results of a simple “dd” test to simulate writing ~5GB of small transaction I/O (`dd if=/dev/zero of=$TARGET bs=512 count=10M`)

* Extrapolated from a single node test

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.

Spark

- “Big Data” high productivity analytics Framework
- Designed around commodity clusters (Ethernet network and local disk)
- Shifter image: lgerhardt/spark-1.6.0
- Uses per-Node write cache for spills and other temporary per-node file caches.
- Tested up to full scale of Cori Phase 1 (1600 nodes) with multiple Spark applications.

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.

See Cray talk at 4:30

Advanced example with MPI support

Dockerfile

```
FROM cern/slc6-lite:latest
## update packages and install dependencies
RUN yum upgrade -y && \
    yum -y install csh tar numpy scipy matplotlib gcc
WORKDIR /
## replace mpi4py with cray-tuned one
ADD optcray_cori.tar /
ADD mpi4py-1.3.1.tar.gz /usr/src
ADD mpi.cfg /usr/src/mpi4py-1.3.1/
RUN cd /usr/src/mpi4py-1.3.1 && \
    chmod -R a+rX /opt/cray && chown -R root:root /opt/cray && \
    python setup.py build && \
    export MPI4PY_LIB=$( rpm -ql $(rpm -qa | grep mpi4py | head -1) | egrep "lib$" ) && \
    export MPI4PY_DIR="${MPI4PY_LIB}/.." && \
    python setup.py install && \
    cd / && rm -rf /usr/src/mpi4py-1.3.1 && \
    echo "/opt/cray/wlm_detect/default/lib64/libwlm_detect.so.0" >>/etc/ld.so.preload && \
    (echo "/opt/cray/mpt/default/gni/mpich2-gnu/48/lib\n/opt/cray/pmi/default/lib64”;\
     echo “/opt/cray/ugni/default/lib64\n/opt/cray/udreg/default/lib64”;\
     echo “/opt/cray/xpmem/default/lib64\n/opt/cray/alps/default/lib64”) \
     >> /etc/ld.so.conf && \
    ldconfig
```

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

Advanced example with Golden Image

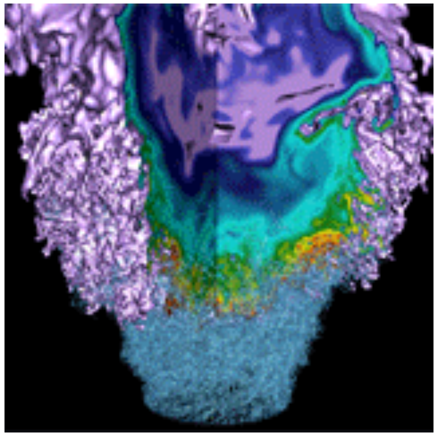
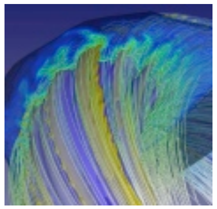
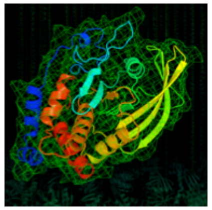
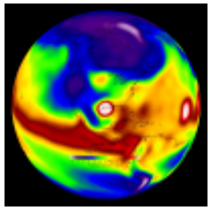
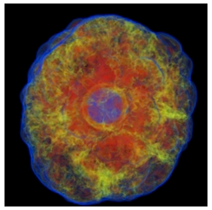
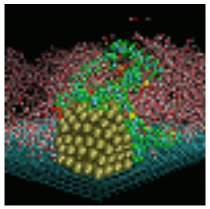
Dockerfile

```
FROM nersc/cori:latest

ADD . /myapp
RUN cd /myapp && \
    make
```

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

Discussion and Future Work

Why Users will like Docker and Shifter

- Develop an application on your desktop or laptop and easily run it on a Cray or other Supercomputer
- Enables the user to solve their dependency problems themselves
- Run the (Linux) OS of their choice and the software versions they need
- Improves application performance in many cases
- Improves reproducibility
- Improves sharing (through sites like Dockerhub)

Roadmap

- **16.05 Release:**
  - Support for RHEL 6/7, SLES 11/12, Rhine/Redwood
  - RPM builds
  - Improved scaling
  - UI Improvements
  - Per-node write cache
  - Bug Fixes
- **16.08 Release**
  - ACL support (private and authenticated images)
  - Image expiry and removal
  - Image usage statistics and metrics
  - Overlayfs support (stretch)
  - Debian packages for Ubuntu LTS

Future Work (beyond the roadmap)

- Continue to simplify installation and increase test coverage
- Expand support for other image types and batch systems (with outside help)
- Create a base image for running MPI applications in a NERSC private Docker registry
- Continue to promote Docker and Shifter within the HPC community to increase access

Conclusions

- Shifter is enabling and improving support for Data Intensive Workloads
- “Shifter” implementation demonstrates that centers can provide 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.

National Energy Research Scientific Computing Center