Singularity

Containers for Science

Presented by:
Gregory M. Kurtzer <gmk@lbl.gov>
HPC Systems Architect and Open Source Developer
Lawrence Berkeley National Laboratory and UC Berkeley

Containers….

In a nutshell:

Containers are encapsulations of system environments

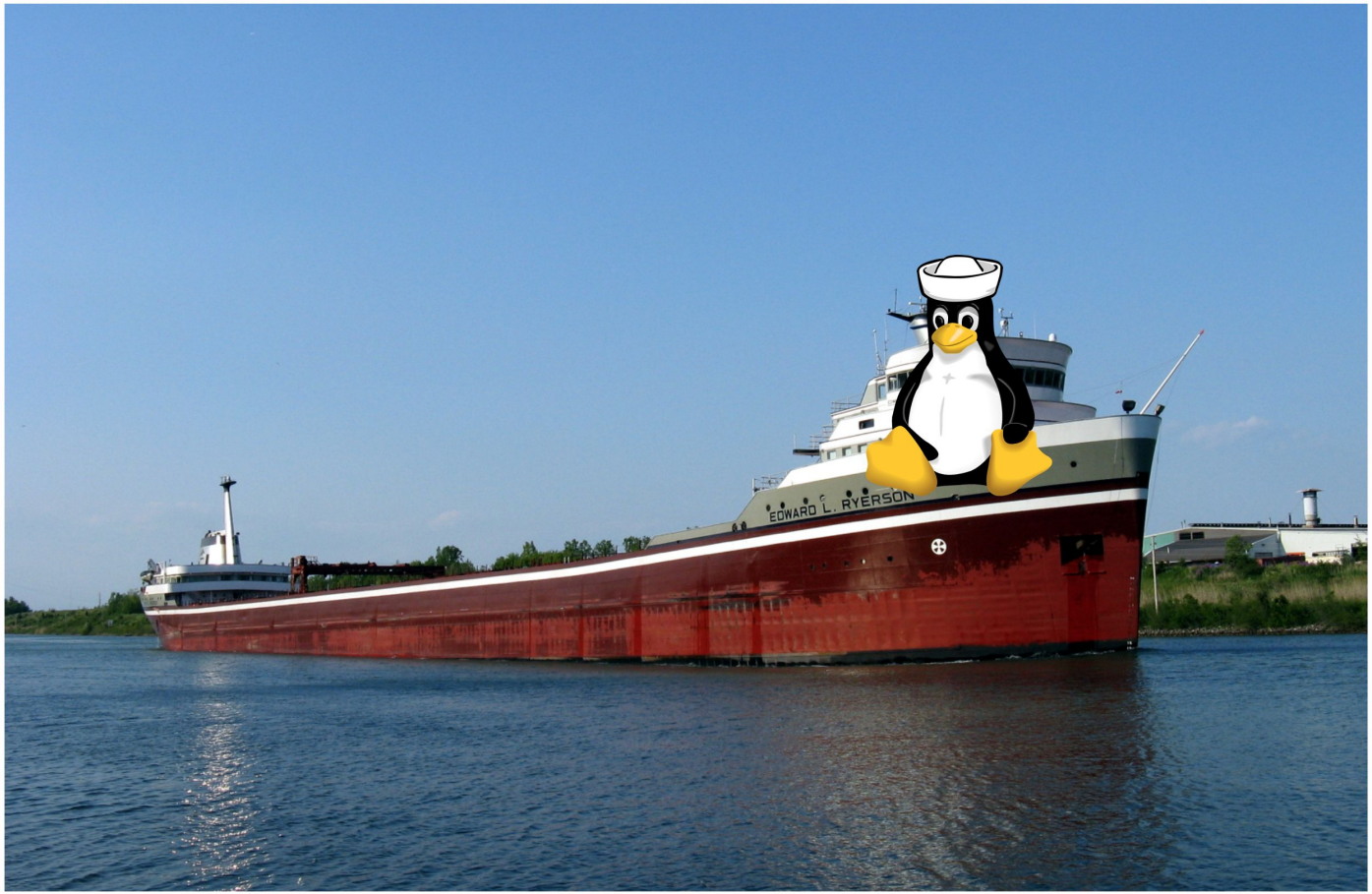
(and of course a means to use it)


1300 556 997


ubuntu
POWERED BY
APACHE
NGINX
http server
PostgreSQL
MariaDB
CentOS

Generally containers technologies have been designed to solve a single primary use case for the enterprise:

micro-service virtualization


EDWARD L. RYERSON

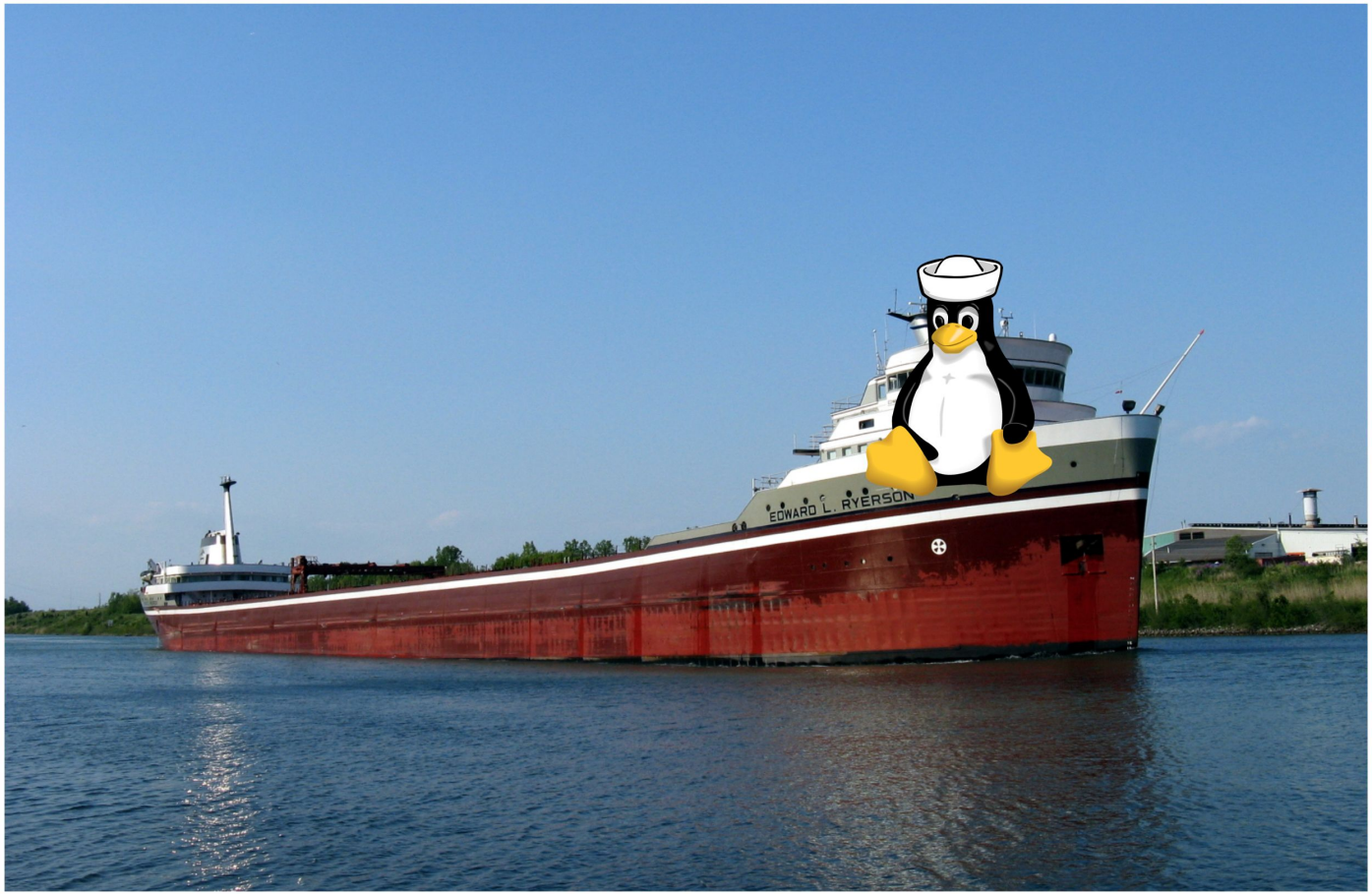

micro-service containers
MAERSK LINE

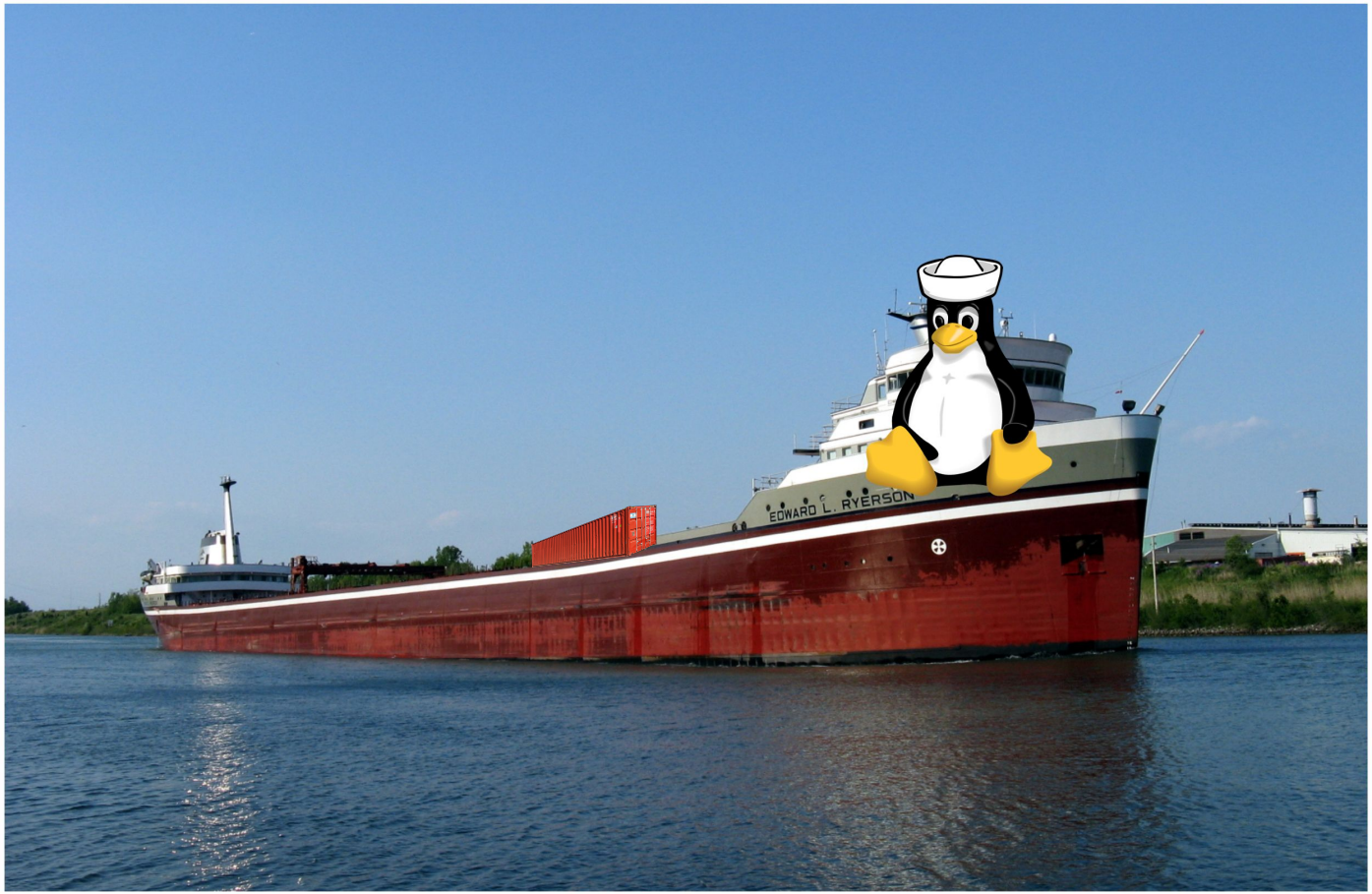

RENA
MONROVIA

This is *NOT* the scientific or HPC use case!

So… what is our use case?

Our use case is the opposite of enterprise computing!


EDWARD L. RYERSON


EDWARD L. RYERSON


tex

Modern day pirates, pillaging resources!

Scientists are like Pirates, pillaging for resources instead of booty!

We want to run our jobs. We want to get results.

And when we find available resources, we need to ensure application and environment compatibility.

This is where containers can be a perfect fit…

But as I mentioned, our use-case and needs are different from enterprise!

So what about Docker!?

- Docker is the most well known and utilized container platform
- Designed primarily for network micro-service virtualization
- Facilitates creating, maintaining and distributing container images
- Containers are kinda reproducible
- Easy to install, well documented, standardized

For these reasons, it appears to be the answer to life, the universe and everything!

And many scientists have jumped on the bandwagon to made use of it

So why not just keep using Docker?

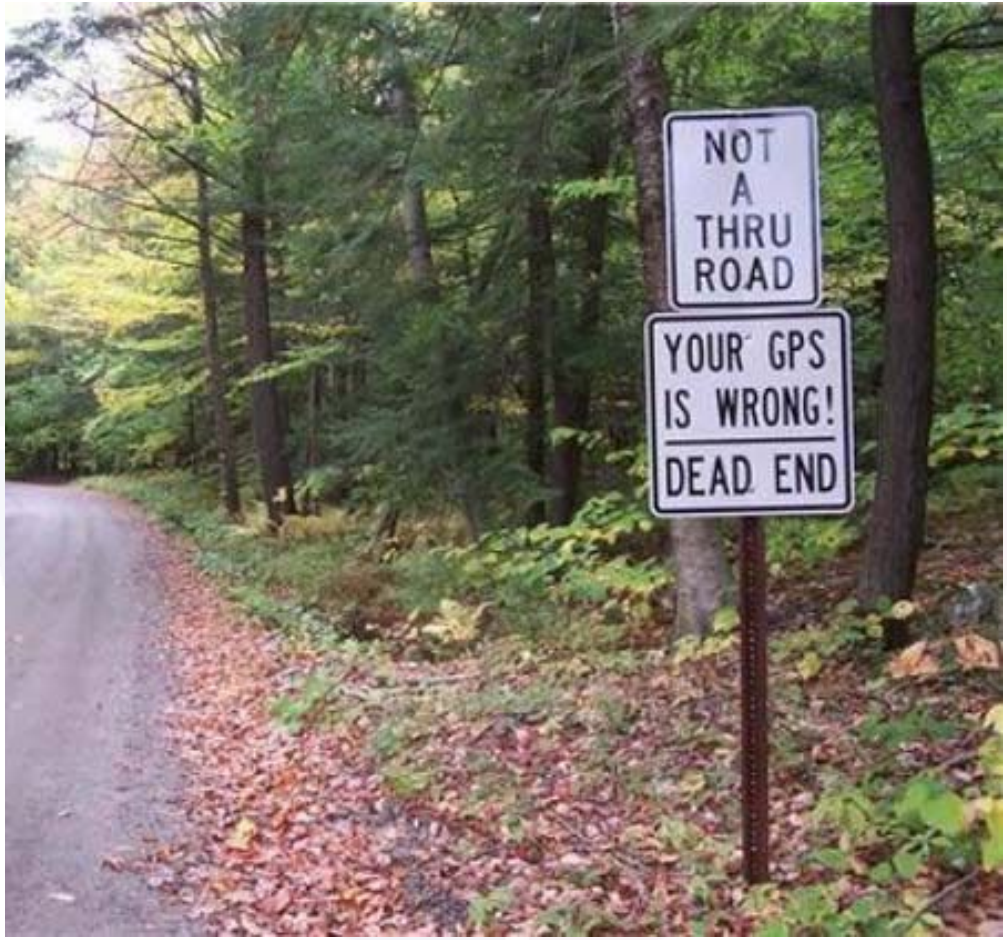
The good news:

You can! It works great for local and private resources. You can use it to develop and share your work with others using Docker-hub.

The bad news:

If you ever need to scale beyond your local resources, it maybe a dead end path! Docker, and other enterprise focused containers, are not designed for, efficient or even compatible with traditional HPC.

No HPC centers allow it!


NOT
A
THRU
ROAD
YOUR GPS
IS WRONG!
DEAD END

Singularity to the rescue!

Singularity: About

- Developed from necessity,.. and demands, threats, and bribes
- First public release in April 2016, followed by a massive uptake
- Created for and by the people who need and use it
  - Scientists, users, HPC engineers, Linux developers
- Tighter integration with other scientific apps (e.g. Nextflow, SLURM, *MPI*)
- Integration with other container technologies
- Various awards, publications (both online and print), and interviews
- HPC Wire Editor's choice: Top Technologies to Watch for 2017

http://www.admin-magazine.com/HPC/Articles/Singularity-A-Container-for-HPC
http://www.admin-magazine.com/Archive/2016/34/Interview-with-the-developer-of-Singularity
http://www.rce-cast.com/Podcast/rce-106-singularity.html
https://www.hpcwire.com/2016/10/20/singularity-containers-easing-scientific-computing/
https://www.hpcwire.com/off-the-wire/hpcwire-reveals-winners-2016-readers-editors-choice-awards-sc16-conference-salt-lake-city/

Singularity: Design Goals

- Architected to support “Mobility of Compute”, agility, BYOE, and portability
- Single file based container images
  - Facilitates distribution, archiving, and sharing
  - Very efficient for parallel file systems
- No system, architectural or workflow changes necessary to integrate on HPC
- Limits user’s privileges (inside user == outside user)
- No root owned container daemon
- Simple integration with resource managers, InfiniBand, GPUs, MPI, file systems, and supports multiple architectures (x86_64, PPC, ARM, etc..)

Overview of Singularity

Singularity: access and privilege

User contexts are always maintained when the container is launched.

When launched by a particular user, the programs inside will be running as that user. Any escalation pathways inside the container are blocked. Thus…

If you want to be root inside the container,

you must first be root outside of the container!

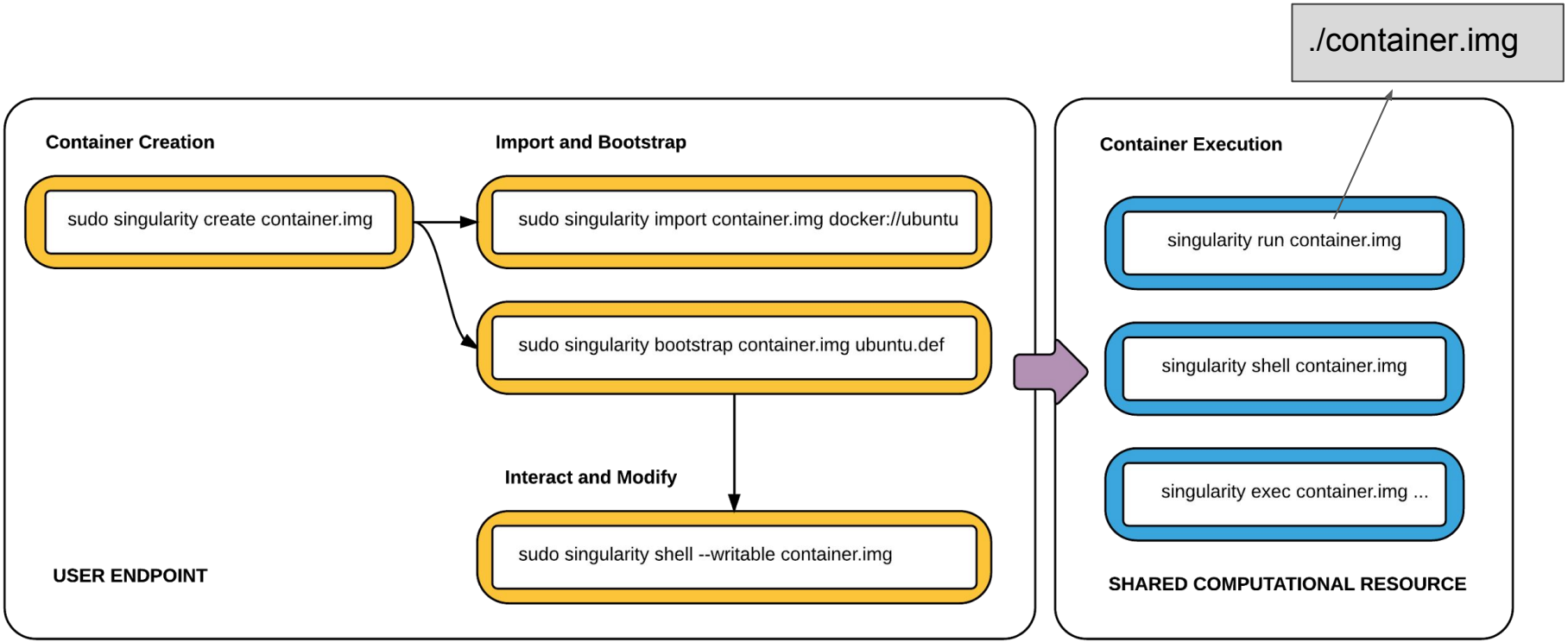
```
[gmk@centos7-x64 ~]$ whoami
gmk
[gmk@centos7-x64 ~]$ singularity shell /tmp/debian.img
Singularity: Invoking an interactive shell within container…

Singularity.debian.img> whoami
gmk
Singularity.debian.img> sudo whoami
sudo: effective uid is not 0, is /usr/bin/sudo on a file system with the
'nosuid' option set or an NFS file system without root privileges?
Singularity.debian.img> ls -l /usr/bin/sudo
-rwsr-xr-x. 1 root root 136808 Aug 17 13:20 /usr/bin/sudo
Singularity.debian.img> exit

[gmk@centos7-x64 ~]$ sudo singularity shell /tmp/debian.img
Singularity: Invoking an interactive shell within container...

Singularity.debian.img> whoami
root
Singularity.debian.img> exit
```

Singularity: Workflow



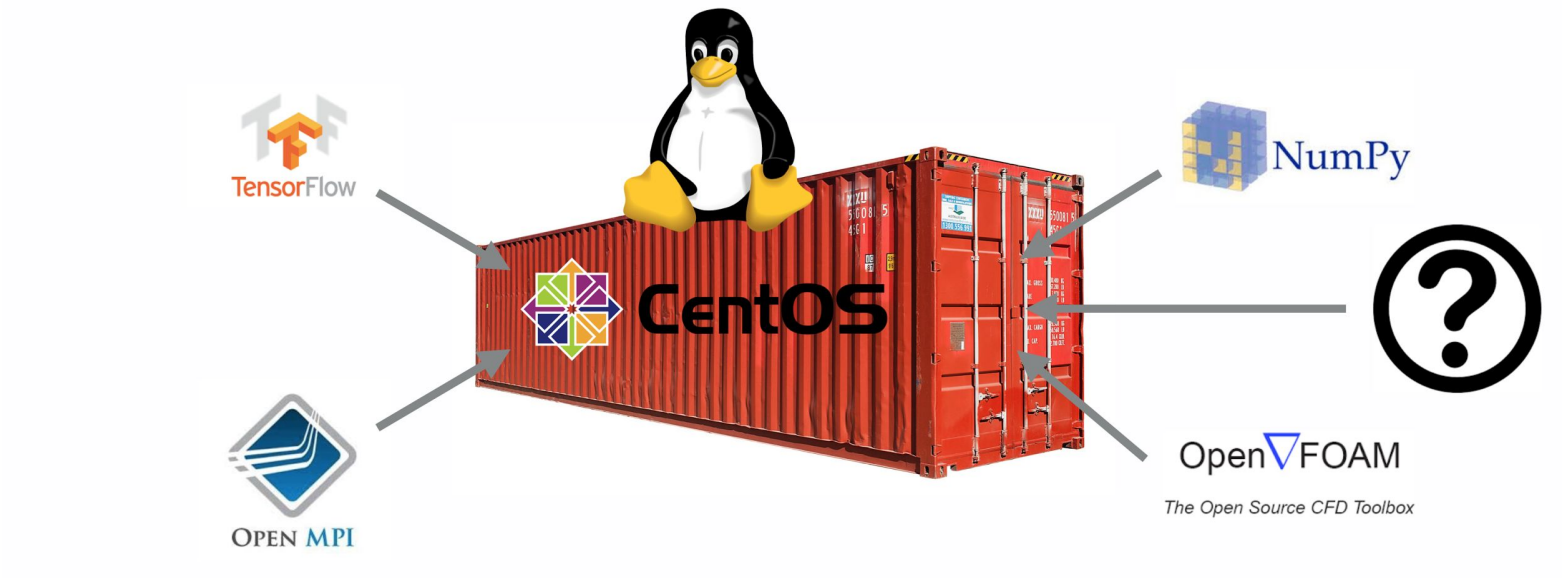
Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific Containers for Mobility of Compute (under review)

Singularity: Creating the container

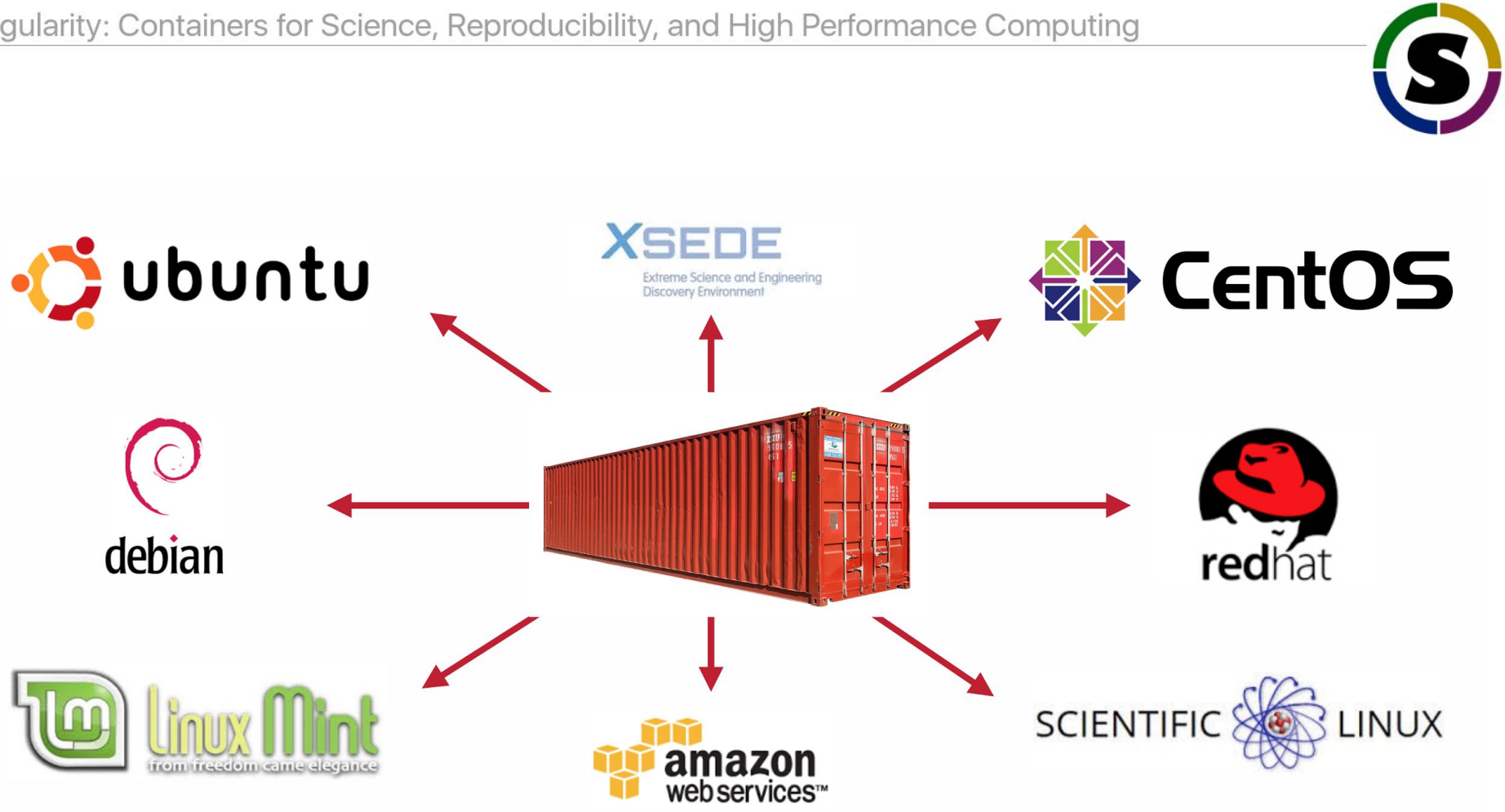
$ sudo singularity create --size 2048 /tmp/Centos-7.img

Singularity: Bootstrapping the container

$ sudo singularity bootstrap /tmp/Centos-7.img centos.def




1300 556 991


ubuntu
XSEDE
Extreme Science and Engineering Discovery Environment
CentOS
debian
redhat
Linux Mint
from freedom came elegance
amazon web services™
SCIENTIFIC LINUX

```
$ singularity shell /tmp/Centos-7.img
Singularity: Invoking an interactive shell within container...

Singularity.Centos-7.img> cat /etc/redhat-release
CentOS Linux release 7.2.1511 (Core)
```

Usage Examples

```
$ singularity shell /tmp/debian.img
Singularity: Invoking an interactive shell within container...

Singularity.debian.img> uname -a
Linux centos7-x64 3.10.0-327.28.2.el7.x86_64 #1 SMP Wed Aug 3 11:11:39 UTC 2016 x86_64
x86_64 x86_64 GNU/Linux
Singularity.debian.img> cat /etc/debian_version
stretch/sid
Singularity.debian.img> whoami
gmk
Singularity.debian.img> cd ~/git/singularity
Singularity.debian.img> ./configure
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
...
checking for gcc... no
checking for cc... no
checking for cl.exe... no
configure: error: in `/home/gmk/git/singularity':
configure: error: no acceptable C compiler found in $PATH
See `config.log' for more details
Singularity.debian.img> exit
```

```
$ python ./hello.py
Hello World: The Python version is 2.7.5

$ sudo singularity exec --writable /tmp/debian.img apt-get install python
…

$ singularity exec /tmp/debian.img python ./hello.py
Hello World: The Python version is 2.7.12

$ cat hello.py | singularity exec /tmp/debian.img python
Hello World: The Python version is 2.7.12
```

```
$ mpirun singularity exec /tmp/Centos7-ompi.img /usr/bin/mpi_ring
Process 0 sending 10 to 1, tag 201 (4 processes in ring)
Process 0 sent to 1
Process 0 decremented value: 9
Process 0 decremented value: 8
Process 0 decremented value: 7
Process 0 decremented value: 6
Process 0 decremented value: 5
Process 0 decremented value: 4
Process 0 decremented value: 3
Process 0 decremented value: 2
Process 0 decremented value: 1
Process 0 decremented value: 0
Process 0 exiting
Process 1 exiting
Process 2 exiting
Process 3 exiting
```

Container Image types supported

- Local images supported:
  - Singularity 2.x default image format
  - SquashFS
  - Tarballs (requires caching)
  - Flat directories (chroots)
- Network images are supported via URIs and all require local caching:
  - docker:// - This will pull a container from Docker Hub
  - http://, https:// - This will pull an image or tarball from the URL, cache and run it
  - shub:// - Pull an image from the Singularity Hub … Wait, what?! Patience, all in due time!

```
$ singularity exec docker://python:latest /usr/local/bin/python hello.py
library/python:latest
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Downloading layer: sha256:e41da2f0bac3da1769ecdac8b0f5df53c1db38603e39b9e261cafd10caf904de
Downloading layer: sha256:75ef15b2048b4cfb06c02f2180f4d89033d02c63f698672d2909b8c9878c4270
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Downloading layer: sha256:45b2a7e03e44b5ea7fad081537134c9cc725bddf94f9093b00e1fa8d8ebbcda1
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Downloading layer: sha256:52f3db4b5710849a53bc2eea0b6f0895c494d751c38c597404d805da82b3f37c
Downloading layer: sha256:76610ec20bf5892e24cebd4153c7668284aa1d1151b7c3b0c7d50c579aa5ce75
Downloading layer: sha256:fce5728aad85a763fe3c419db16885eb6f7a670a42824ea618414b8fb309ccde
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
Downloading layer: sha256:5040bd2983909aa8896b9932438c3f1479d25ae837a5f6220242a264d0221f2d
Hello World: The Python version is 3.6.0
```

```
$ singularity exec docker://tensorflow/tensorflow python -m tensorflow.models.image.mnist.convolutional
tensorflow/tensorflow:latest
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
...
Downloading layer: sha256:6498e51874bfd453352b79b1a3f669109795134b7adcd1a02d0ce69001f4e05b
Downloading layer: sha256:862a3e9af0aeffe79345b790bad31baaa61e9402b6e616bff17babed6b053b54
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting data/train-images-idx3-ubyte.gz
Extracting data/train-labels-idx1-ubyte.gz
Extracting data/t10k-images-idx3-ubyte.gz
Extracting data/t10k-labels-idx1-ubyte.gz
Initialized!
Step 0 (epoch 0.00), 5.1 ms
Minibatch loss: 8.334, learning rate: 0.010000
Minibatch error: 85.9%
Validation error: 84.6%
Step 100 (epoch 0.12), 140.0 ms
Minibatch loss: 3.250, learning rate: 0.010000
Minibatch error: 6.2%
Validation error: 7.6%
…
Step 8500 (epoch 9.89), 134.2 ms
Minibatch loss: 1.618, learning rate: 0.006302
Minibatch error: 0.0%
Validation error: 0.9%
Test error: 0.8%
```

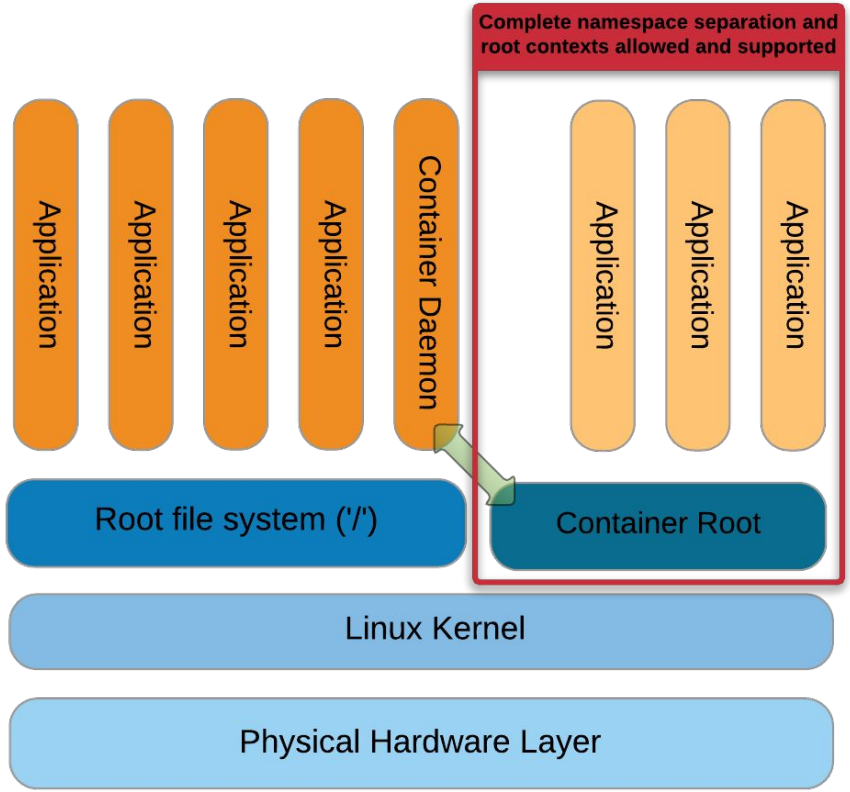
Performance info

General Container Architecture

Container technologies utilize the host's kernel and thus have the ability to run contained applications with the same performance characteristics as native applications.

There is a minor theoretical performance penalty as the kernel must now navigate namespaces.

The initial inception of containers shares a direct lineage from virtual machines and is sometimes considered a more efficient virtual machine technology.

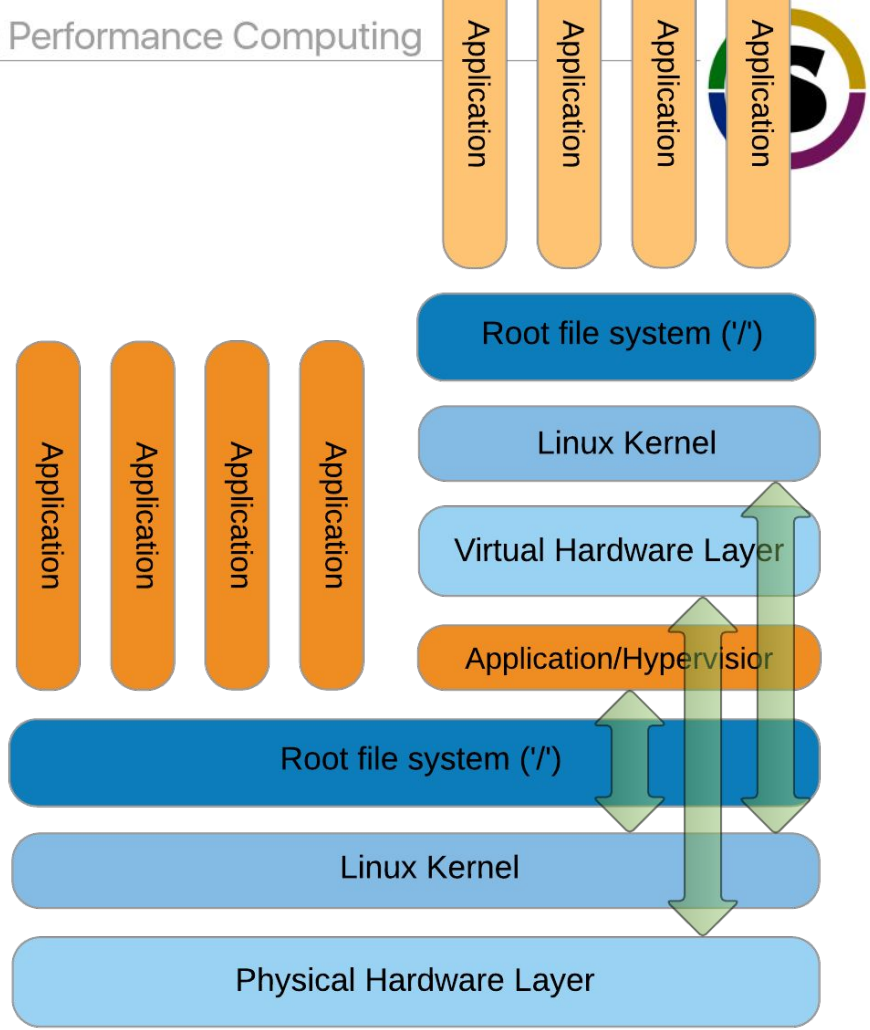


Virtual Machine Architecture

The applications running within the virtual machine have to traverse redundant code pathways and the performance cost of emulation.

Technology has bridged many of the performance gaps, but there is considerable overhead and redundancy when evaluating application performance on a virtual machine.

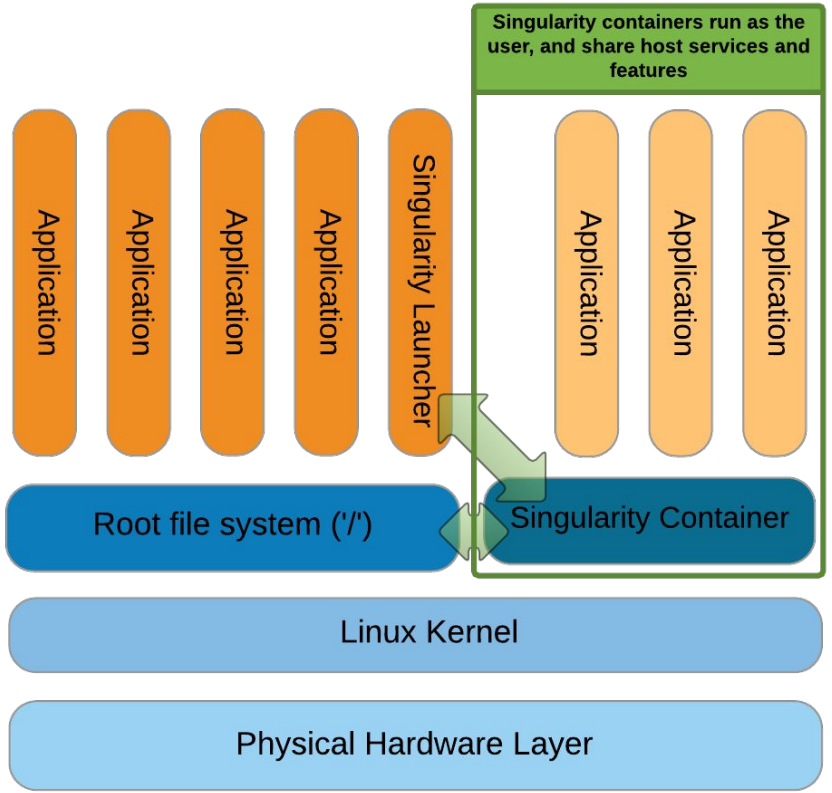
Containers are indeed a much more efficient virtualization mechanism, but may not offer the same confidence when it comes to isolation.



Singularity Architecture

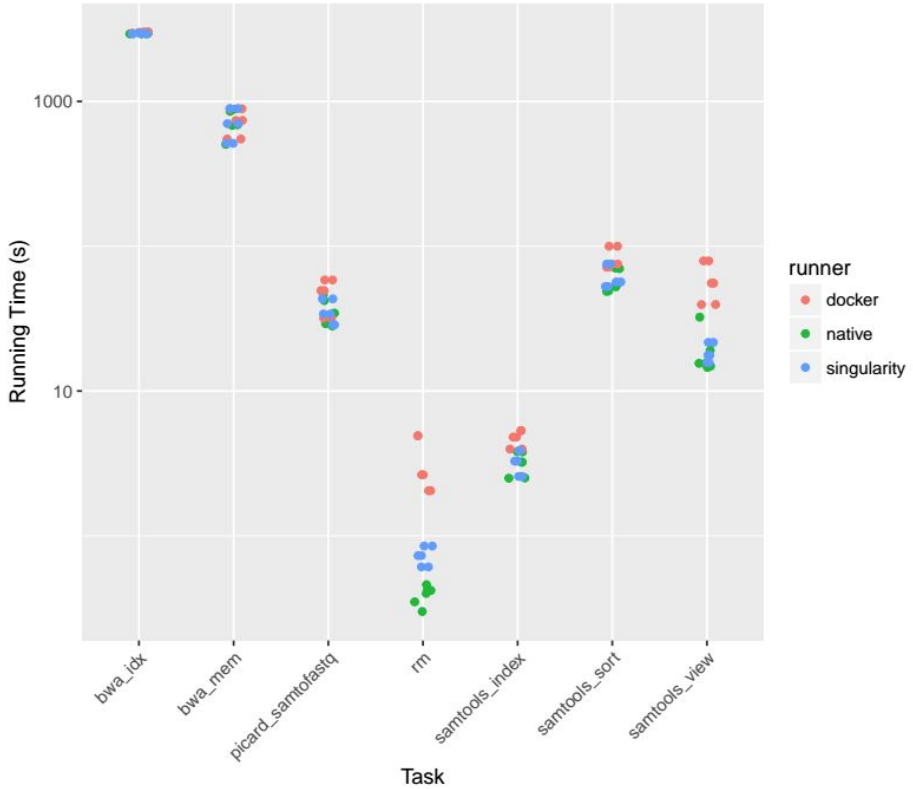
Singularity on the other hand is not designed around the idea of micro-service process isolation. Therefore it uses the smallest number of namespaces necessary to achieve its primary design goals.

This gives Singularity a much lighter footprint, greater performance potential and easier integration than container platforms that are designed for full isolation.



Performance of Bio-apps

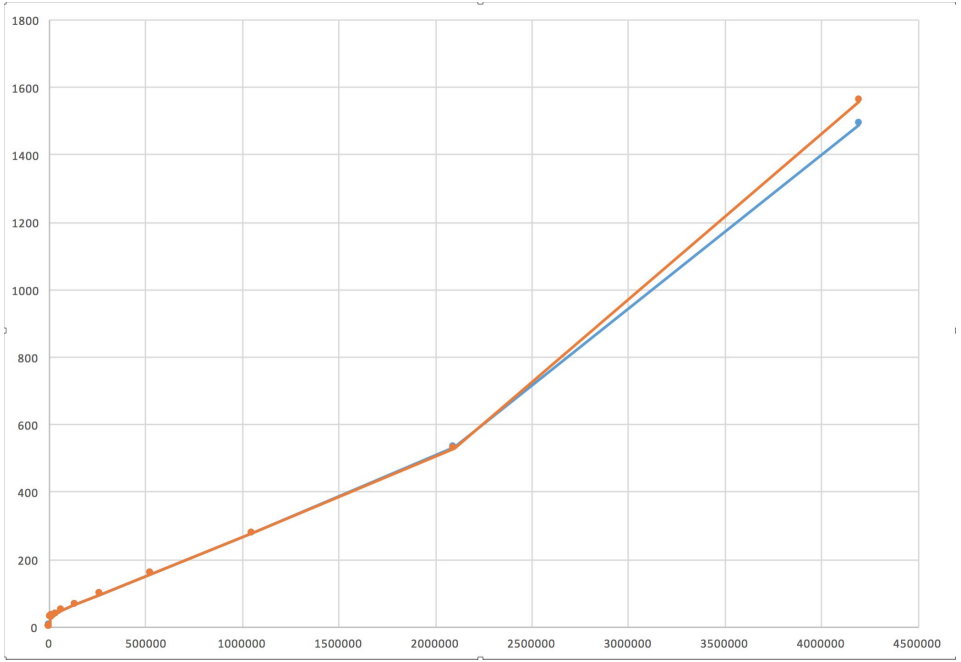
- BWA index and to a lesser extent BWA mem are, in these cases, CPU/memory bound
- Samtools index and sort are partially I/O bound and throughput of large numbers of files (rather than latency)
- Samtools view is IOPS heavy



Data provided by: Jonathan Dursi <Jonathan.Dursi@sickkids.ca>, CanDIG project

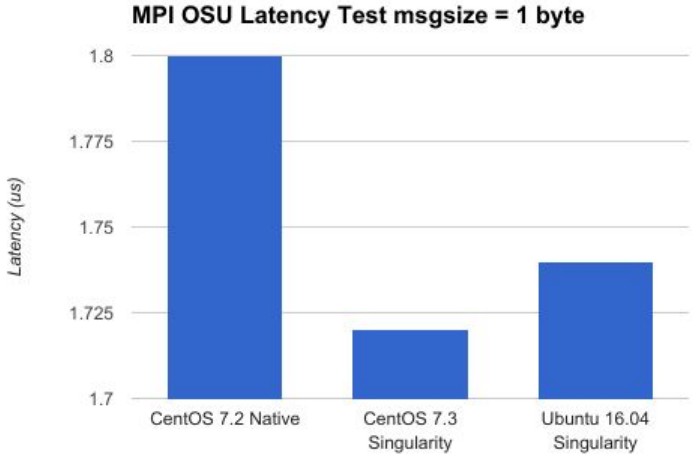
Shared memory MPI latency

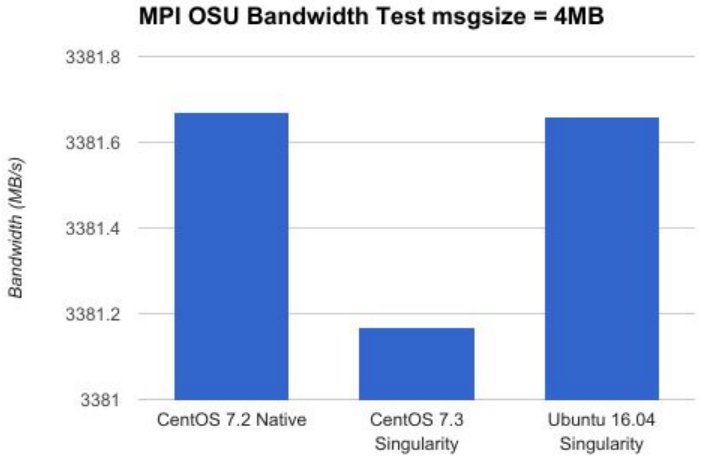
- Same OS image and libraries inside and outside of container
- Tested with Open MPI master branch (2.x)
- Both perform closely and only subtly diverge on large messages



Data provided by: Ralph Castain <rhc@open-mpi.org>

Containerized MPI Latency comparison over IB





- OpenMPI 2.0.1 with OSU Micro Benchmarks 5.3.2 http://mvapich.cse.ohio-state.edu/benchmarks/
- Running over 40Gb/s Infiniband with Singularity v2.2

Note: Centos-7.2 uses libibverbs 1.1.8 and Centos-7.3 is 1.2.1, libmlx4 is (1.0.6 vs 1.2.1), among other OS differences which may account for the differences

Data provided by: Paulo Souza <paulo7@gmail.com>

And a taste of things to come...

Singularity Version > 2.2

- Support for “Data Containers”
- Ability to do more without privilege (e.g. create, import, export, mount, etc)
- Capture data/changes in a container and encapsulate as “output”
- Optional full process isolation (for untrusted containers or testing)
- Support for daemon/background processes
- Revamp of the Singularity image file (>= v3.x, over a year)
  - Support for user space file system
  - Need a FUSE file system developer/contributor!
- How else can we help? You tell us!

Singularity Hub: Container Registry

SINGULARITY Containers About User Guide vsoch

Make containers, run, done.

A collaboration between Stanford University and Lawrence Berkeley National Laboratory

Package your Analysis

Whether you are working from your own computer, or a cluster environment, you can capture your analysis

Singularity Hub

Developed by Vanessa Sochat from Stanford's Research Computing group:

- A container build, archive, and citation platform, designed to facilitate reproducible research, publications, and peer-review
- Integrated with GitHub such that you can trigger builds and automatic containerization of your project/code
- Allows easy reference and citations for scientific publications
- With the Singularity Hub citation in your publication, others can easily replicate your analysis

Singularity Hub: Container Registry

Builds

OFFICIAL LIBRARY | ALL COLLECTIONS

Enter Keywords Here

Name	Builds	Latest
hb-slash/busybox	5	3ef96e2a3657f0c30878ed8a46380d171b0d4cba
researchapps/tensorflow	4	0e759ccd3be631998aa927b7ab8068e1136221c2
nextflow-io/rnatoy	3	1495129c1df7196f250eb52bef4d697084edeefb
singularityhub/debian	3	0e700bacc7399c344f016e49f15324088f421979
vsoch/singularity-images	2	6d3715a982865863ff20e8783014522edf1240e4
singularityhub/ubuntu	2	5a1d85b1268750b862eea4de157f035f37f21e38
singularityhub/nginx	2	04dd95c295244f7e18afbe45867550fe25253c53
lsorrillo/first_project	2	41e2e518f3b3ebe0a53ac61d5b0a7f69161fc6fa
c1t4r/CiTAR-Containers	2	a33514d7bc85c4e9fd509fd898134c8c09cec044
MaxUlysse/compile-beamer	1	732fa35d286b0507282962249ce19ea95d96b4e4
researchapps/optitype	1	6d27c887c81f17765ca7ab9e7aed28957dbee6db
researchapps/polysolver	1	430f46102184e47b6c9b4ca71bf6af3d424951cb
vsoch/singularity-scientific-example	1	037ed6f3d3a6adeeffb358f2ec125387f5475bcc
singularityhub/centos	1	8266a4eba15efefe5951aebb18f675108e3d4cb3

Singularity Hub: Build Details

singularityhub/debian

VIEW ASCIINEMAS | DISCUSSION | MAKE PRIVATE | BRANCHES | EDIT BUILDER | DISABLE

Builds

Id	Tag	Build Date	Status	Version
72	jessie	Jan. 18, 2017, 8:29 p.m.	COMPLETE	0e700bacc7399c344f016e49f15324088f421979
71	sid	Jan. 18, 2017, 8:25 p.m.	COMPLETE	ebe6855ccea39b39f9c982918fd75e4b73d37ab1
70	master	Jan. 18, 2017, 7:42 p.m.	COMPLETE	de93b38c436d86ec971bc98c89e635fbcbc04906

Rows per page: 50 1-3 of 3

DELETE COLLECTION

Singularity Hub: CLI Integration

Build Spec | Files | Plots | **Commands** | Log

For all commands, please use the master (development) version of Singularity for all functionality.

Pull the container to your machine:

```
singularity pull shub://72
singularity pull shub://singularityhub/debian:jessie
```

Shell into the container:

```
singularity shell shub://72
singularity shell shub://singularityhub/debian:jessie
```

Run the container:

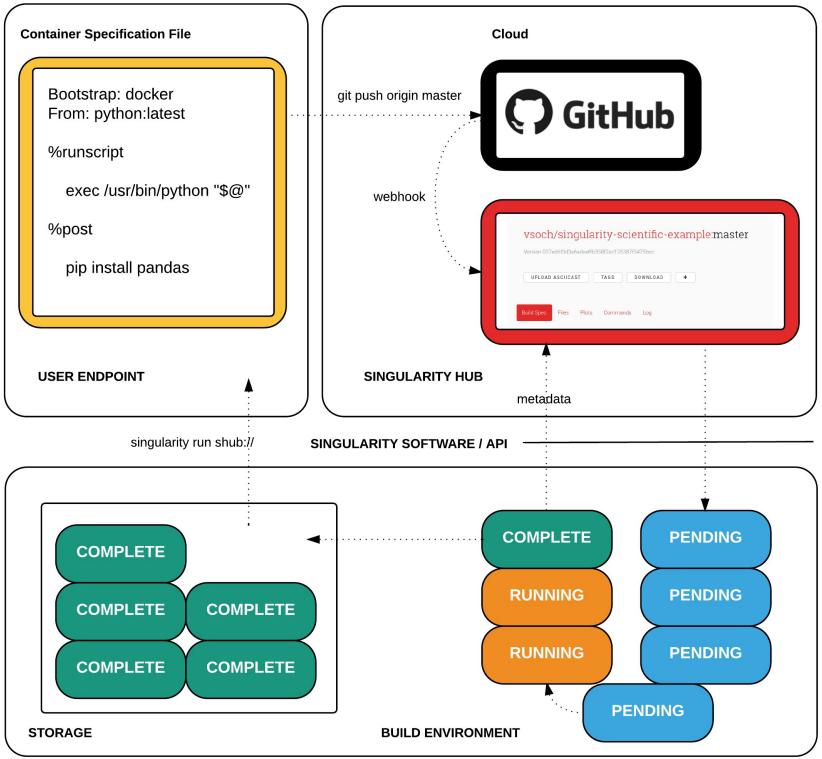
```
singularity run shub://72
singularity run shub://singularityhub/debian:jessie
```

Include container in a bootstrap:

Singularity Hub: Build/Import workflow

- Singularity bootstrap definition is committed and pushed to a GitHub repository (named ‘Singularity’)
- GitHub communicates with Singularity Hub via a web hook, and it gets queued to be built via continuous integration
- Once built the resulting product is stored in Google Cloud and available to be accessed
- The container will be archived and can/should be cited in your scientific publications so others can replicate and leverage your work with:

```
$ singularity shell shub://$UNIQUE_ID
$ singularity run shub://$USER/$CONTAINER:$TAG
```



Sochat V., Prybol C, Kurtzer GM. Singularity Hub: Registry for Reproducible Containers (manuscript in progress)

Contributors to Singularity:

BERKELEY LAB
Lawrence Berkeley National Laboratory

nextflow

HARVARD UNIVERSITY

Stanford University

TACC
TEXAS ADVANCED COMPUTING CENTER

ONTROPOS

UNIVERSITY OF MICHIGAN

Berkeley
UNIVERSITY OF CALIFORNIA

NVIDIA

intel

SDSC
SAN DIEGO SUPERCOMPUTER CENTER

UNIVERSITY OF Nebraska Lincoln

NIH National Institutes of Health

amazon web services

Institut Pasteur

GSI

GHENT UNIVERSITY

INDIANA UNIVERSITY

Dartmouth

Lenovo

Singularity

Containers for Science

Twitter: @SingularityApp
GitHub: http://www.github.com/singularityware/singularity
Slack: http://singularity-container.slack.com (email me for invite)

Bootstrapping

Singularity: Bootstrap definition - RedHat/YUM

```
$ cat examples/centos.def
BootStrap: yum
OSVersion: 7
MirrorURL: http://mirror.centos.org/centos-%{OSVERSION}/%{OSVERSION}/os/$basearch/
Include: yum

%runscript
    echo "This is what happens when you run the container..."


%post
    echo "Hello from inside the container"
    yum -y install vim-minimal
```

- The header defines the core operating system
- %runscript: What should this container do when “run” (singularity run, or ./container.img)
- %post: After the core operating system has been built, this gets executed inside the new container

Singularity: Bootstrap definition - Debian/Ubuntu

```
$ cat examples/debian.def
BootStrap: debootstrap
OSVersion: stable
MirrorURL: http://ftp.us.debian.org/debian/

%runscript
    echo "This is what happens when you run the container..."

%post
    echo "Hello from inside the container"
    apt-get update
    apt-get -y install vim
```

Singularity: Bootstrap definition - Docker

```
$ cat examples/docker.def
BootStrap: docker
From: ubuntu:latest
IncludeCmd: yes

%runscript
    echo "This is what happens when you run the container..."


%post
    echo "Hello from inside the container"
    echo "Install additional software here"
```

Singularity: Bootstrapping

```
$ sudo singularity create /tmp/debian.img
Creating a new image with a maximum size of 768MiB...
Executing image create helper
Formatting image with ext3 file system
Done.

$ sudo singularity bootstrap /tmp/debian.img debian.def
Bootstrap initialization
Checking bootstrap definition
Executing Prebootstrap module
Executing Bootstrap 'debootstrap' module
...
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/ex (ex) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/editor (editor) in auto
mode
Processing triggers for libc-bin (2.19-0ubuntu6) ...
Done.
```