Singularity: Containers for Science, Reproducibility, and High Performance Computing

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

micro-service virtualization
micro-service containers
This is *NOT* the scientific or HPC use case!
So… what is our use case?
Our use case is the opposite of enterprise computing!
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: Containers for Science, Reproducibility, and High Performance Computing
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.rce-cast.com/Podcast/rce-106-singularity.html
https://www.hpcwire.com/2016/10/20/singularity-containers-easing-scientific-computing/
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: Scientific Containers for Mobility of Compute (under review)

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
Singularity: Containers for Science, Reproducibility, and High Performance Computing
$ 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

Hello World: The Python version is 3.6.0
Singularity: Containers for Science, Reproducibility, and High Performance Computing

```bash
$ singularity exec docker://tensorflow/tensorflow python -m tensorflow.models.image.mnist.convolutional
tensorflow/tensorflow:latest
```

```
Downloading layer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4
...Downloaded layer: sha256:6498e51874bfd453352b79b1a3f669109795134b7adcd1a02d0ce69001f4e05b
Downloaded 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 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

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Builds</th>
<th>Latest</th>
</tr>
</thead>
<tbody>
<tr>
<td>busybox</td>
<td>5</td>
<td>3e96e2a3657fd0c30678dec8e46380d171b9d4cb9</td>
</tr>
<tr>
<td>tensorflow</td>
<td>4</td>
<td>0e759cc3b6e631906a9f27b7a8906e113221c2</td>
</tr>
<tr>
<td>nvidia/gpu-driver</td>
<td>3</td>
<td>1d4fd1f196f257d25eb522a8d49d707b84d88f8</td>
</tr>
<tr>
<td>debian</td>
<td>3</td>
<td>0e700baace73993c344016e49f1532436f8f421979</td>
</tr>
<tr>
<td>singularityhub/nginx</td>
<td>2</td>
<td>6d3715a982b6686f8120e878014522e6f1240e4</td>
</tr>
<tr>
<td>singularityhub/centos</td>
<td>2</td>
<td>5a1d85b1268750b862a4a4e1571035373721e38</td>
</tr>
<tr>
<td>singularityhub/centos</td>
<td>2</td>
<td>04d95c2952447c18a45b567550e25223c53</td>
</tr>
<tr>
<td>isomillo/first_project</td>
<td>2</td>
<td>41e2e518a53eebe3a53a6d1b53c0769a1615c66a</td>
</tr>
<tr>
<td>c14r/GITAR-Containers</td>
<td>2</td>
<td>a33514d7b850460f9d08998134abe8c00e044</td>
</tr>
<tr>
<td>MaxOlyse/compile-beamer</td>
<td>1</td>
<td>732a35d28e8b9d672b9262249ce19eaa95db9b4e4</td>
</tr>
<tr>
<td>rechnets/optype</td>
<td>1</td>
<td>6d27c887ca117765c7a7ab47ae2d8997dbbdee6cb</td>
</tr>
<tr>
<td>rechnets/polyolver</td>
<td>1</td>
<td>430f46102184e4b6b84a7b1f0a3d9d24951cb</td>
</tr>
<tr>
<td>vsoch/singularity-examples</td>
<td>1</td>
<td>0376e6d3a2a4ad6f931253877f67547b5c</td>
</tr>
<tr>
<td>singularityhub/centos</td>
<td>1</td>
<td>8226a4e4e8a15eefef50514edbb18b13f75101e3d43c3b</td>
</tr>
</tbody>
</table>
Singularity Hub: Build Details

<table>
<thead>
<tr>
<th>ID</th>
<th>Tag</th>
<th>Build Date</th>
<th>Status</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>jessie</td>
<td>Jan. 18, 2017, 8:29 p.m.</td>
<td>COMPLETE</td>
<td>7f0bacc7399c344b316e4f1f3324f8b4f21979</td>
</tr>
<tr>
<td>71</td>
<td>sid</td>
<td>Jan. 18, 2017, 8:25 p.m.</td>
<td>COMPLETE</td>
<td>6be855cc3a39b69f0c8929f8d75a4b73d37ab1</td>
</tr>
<tr>
<td>70</td>
<td>master</td>
<td>Jan. 18, 2017, 7:42 p.m.</td>
<td>COMPLETE</td>
<td>73b38a4398d1e971b0b8c69e639fddbc04906</td>
</tr>
</tbody>
</table>
Singularity Hub: CLI Integration

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:

  \[
  \begin{align*}
  \text{} & \text{} & \text{} & \text{} \\
  \text{\$ singularity shell shub://$UNIQUE_ID} & \text{} & \text{} & \text{} \\
  \text{\$ singularity run shub://$USER/$CONTAINER:$TAG} & \text{} & \text{} & \text{}
  \end{align*}
  \]

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

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