Containers and GPUs

Mats Rynge (rynge@isi.edu)
Christina Koch (ckoch5@wisc.edu)
GPUS
What is a GPU?

- GPU = Graphical Processing Unit
- Has hundreds to thousands of “cores” that can be used to parallelize work.
GPU Use Cases

- Programs that map well to GPUs include:
  - Deep learning
  - Molecular dynamics
  - Anything with lots of number crunching (like matrix operations) and low(er) data load.
GPUs on the OSG

• Scale: 100s (vs 10,000s of CPUs)
• Variety of available GPU cards

• Same restrictions as always: not sure what you’ll get, jobs can be interrupted

• May take longer to start
Making robust GPU jobs

• Use a software strategy that can run on different GPU types:
  – Container
  – Install inside the job

• OR use job requirements to request certain kind of GPU (more limiting)
Submit File options

- Request GPUs with “request_gpus”
- Can use custom requirements

request_gpus = 1

requirements = (CUDACapability >= 6.0)
CONTAINERS
Returning to Our Analogy…

• Using a container is kind of like bringing along a whole kitchen…

Photo by PunkToad on Flickr, CC-BY
Containers

- Containers are a tool for capturing an entire job “environment” (software, libraries, operating system) into an “image” that can be used again.
Container Motivations

Consistent environment (default images) - If a user does not specify a specific image, a default one is used by the job. The image contains a decent base line of software, and because the same image is used across all the sites, the user sees a more consistent environment than if the job landed in the environments provided by the individual sites.

Custom software environment (user defined images) - Users can create and use their custom images, which is useful when having very specific software requirements or software stacks which can be tricky to bring with a job. For example: Python or R modules with dependencies, TensorFlow

Enables special environment such as GPUs - Special software environments to go hand in hand with the special hardware.

Process isolation - Sandboxes the job environment so that a job can not peek at other jobs.

File isolation - Sandboxes the job file system, so that a job can not peek at other jobs’ data.
Container Types

• Two common container systems:

Docker
https://www.docker.com/

Singularity
https://sylabs.io/

The container itself will always be some version of Linux - but can be run on Linux / Mac / Windows if Docker or Singularity is installed.
Focus on Docker

• Docker has well-established and well-documented ways to build container images. It has huge library of images!

• If you have a Docker image:
  – Can run with Docker
  – Can run with Singularity – Remember this
  – Can convert to a Singularity image
Running Containers

docker run <container> <command>

docker run -it <container> /bin/sh
Docker Hub

- `docker pull`
- `docker push`

Created by Idealog Studio
from Nox Project
Building Containers

Dockerfile → docker build .
# Start with this image as a "base".
# It's as if all the commands that created that image were inserted here.
FROM continuumio/miniconda:4.7.12

# Use RUN to execute commands inside the image as it is being built up.
RUN conda install --yes numpy

# Try to always "clean up" after yourself to reduce the final size of your image.
RUN apt-get update \
&& apt-get --yes install --no-install-recommends graphviz\ 
&& apt-get --yes clean \ 
&& rm -rf /var/lib/apt/lists/*
Containers are defined using Docker
Public Docker Hub

... and executed with Singularity
No direct access to the Singularity command line - that is controlled by the infrastructure

https://github.com/opensciencegrid/cvmfs-singularity-sync
(next slide)
User-defined Container Publishing

Pull request or ticket to register container (one time)

Option 1
- git push
- GitHub
  - Automatic builds on changes

Option 2
- docker build + push
- GitHub
  - docker hub

GitHub
- docker_images.txt
cvmfs-singularity-sync
- Automatic sync on changes

/cvmfs
CVMFS Repositories

/cvmfs/

ams.cern.ch
atlas.cern.ch
cms.cern.ch
connect.opensciencegrid.org
gwosc.osgstorage.org
icecube.opensciencegrid.org
ligo-containers.opensciencegrid.org <-- large project with their own containers
nexo.opensciencegrid.org
oasis.opensciencegrid.org <-- "modules" software
singularity.opensciencegrid.org <-- general containers (next few slide)
snoplus.egi.eu
spt.opensciencegrid.org
stash.osgstorage.org
veritas.opensciencegrid.org
xenon.opensciencegrid.org <-- ~1PB of user published data
OSG stores container images on CVMFS in extracted form. That is, we take the Docker image layers or the Singularity sif files and export them onto CVMFS. For example, `ls` on one of the containers looks similar to `ls /` on any Linux machine:

```
$ ls /cvmfs/singularity.opensciencegrid.org/opensciencegrid/osgvo-el7:latest/
cvmfs host-libs proc sys anaconda-post.log lib64
dev media root tmp bin sbin
etc mnt run usr image-build-info.txt singularity
home opt srv var lib
```

Result: Most container instances only use a small part of the container image (50-150 MB) and that part is cached in CVMFS!