HPS

https://hps.vi4io.org

Azat Khuziyakhmetov

Running Containers with Singularity

Containerization for HPC

Learning Objectives

- Understand the reason for using containers in HPC
- Gain insight into Singularity
- Become aware of pitfalls when using Singularity

Table of contents

Container

- Term *software container* mostly coined by Docker
- Container refers to:
  An application that is bundled with all its dependencies into a single package, which is then executed by the container software.
- Due to dependency bundling, application can run on any machine on which the container management software can run

Benefits

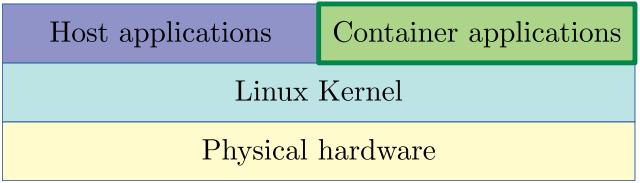
Why containers?

Management: Easy to deploy software in HPC

Environment: You can install a software stack you want

Security: Containers can be isolated from host OS

Lightweight: Compared to other virtualization techniques



Docker vs. Singularity

Why Singularity for HPC instead of Docker?

	Docker	**Singularity**
Rooted daemon in Host OS	Yes	No
Root in the container	Yes	No
Host OS filesystems	Restricted	Mounted
Network	NS & NAT	no isolation

- Docker needs additional steps to ensure security such as *user namespaces*
- Without extra security steps, running user defined containers is not safe

Singularity

Architecture

Important features for HPC

- Since v3 is written in Go (i.e. compilation and dependency resolving)
- Can be installed in shared FS, metadata is on local FS
- Minimum isolation, network and devices are accessible in containers
- Processes run as users without ns (batch systems can account them)

Security

- Only root from Host OS can be root in a container
- Container FS mounted with `nosuid`
- Processes are executed with `PR_SET_NO_NEW_PRIVS`

Singularity

Use cases

Running applications in Singularity can be easier than installing it in HPC

Tensorflow Containers with matching versions of CUDA driver, cuDNN, Tensorflow and native support of GPU

Docker Docker has a large repository of images which can be easily converted to Singularity

MPI Compile and run MPI programs using self installed MPI implementation within the container

Jupyter Create Jupyter instance which runs custom notebooks Parallelization can be done using IPython Parallel

Singularity

Example

Switching from Docker could be as easy as going from

Docker

```
1        docker run godlovedc/lolcow
```

to

Singularity

```
1        singularity run docker://godlovedc/lolcow
```

In real world it might be more complicated than that...

Pitfalls of Containers for HPC

Abstractions

If abstractions are not trivial it might cause issues for

- Reproducibility
- Performance
- Migration of containers

Unfortunately in HPC abstractions are almost certainly non-trivial

- Specific hardware
- Various libraries (MPI, BLAS etc.)

Pitfalls of Containers for HPC

Portability

HPC performance optimizations can reduce portability due to:

- **CPU architecture** (Model specific optimizations crash on other models)
- **GPU generations** (GPUs have various acceleration capabilities)
- **Interconnects** (for MPI containers correct networks should be used)

Pitfalls of Containers for HPC

Security

- Singularity sacrifices root privileges for security
- Cannot run software which requires root within the container
- Luckily most scientific software doesn’t need root

Pitfalls of Containers for HPC

Maintainability

- HPC software commonly maintained by admins
- Libraries configured and interlinked via tools
  - Spack
  - Easybuild

With containers, you should:

- Maintain containers yourself
- Be aware that dependency containers are still accessible
- Optimize performance if required

Pitfalls of Containers for HPC

Licensing

- Software vendors usually don’t have explicit rules regarding running software within containers
- Can easily violate licences when sharing containers or running them from shared sources

Demo

Notebooks can be found in
https://gitlab-ce.gwdg.de/akhuziy/pcpc-22-containers

DEMO