Containerization

Performance Characterization

Shantenu Jha: Project Manager

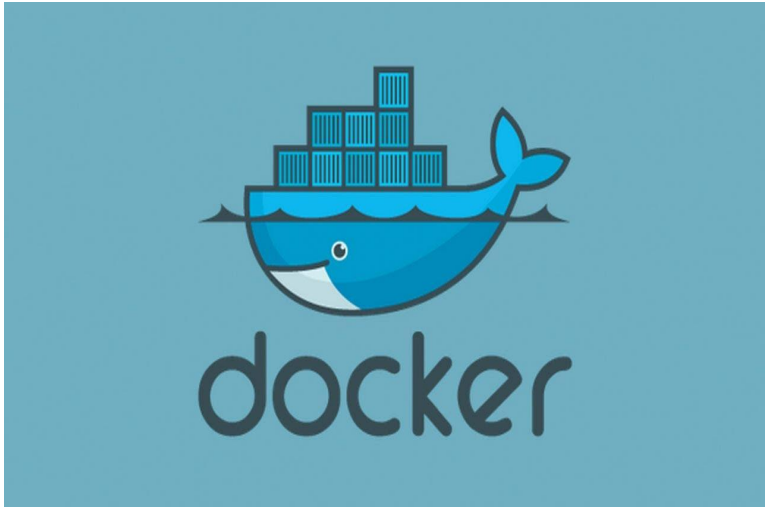
Matteo Turilli: Technical Supervisor

What is a container?

- A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computer environment to another.
- It is lightweight since it shares the machine’s OS system kernel and therefore do not require an OS per application.
- A container image becomes a container at runtime, and said containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly

Docker

- It is the most well known and utilized container platform, designed primarily for network micro-service virtualization.
- Docker facilitates creating, maintaining and distributing container images. Containers are somewhat reproducible, easy to install, well documented and standardized.
- Docker works great for local and private resources, and you can develop and share your work with others through Docker-Hub. However, if you need to scale beyond your local resources, let's say, HPCs, it won't be efficient or even compatible.



Singularity

- Singularity enables users to have full control of their environment. Singularity containers can be used to package entire scientific workflows, software and libraries, and even data. This means that you don’t have to ask your cluster admin to install anything for you, you can put it in a Singularity container and run.

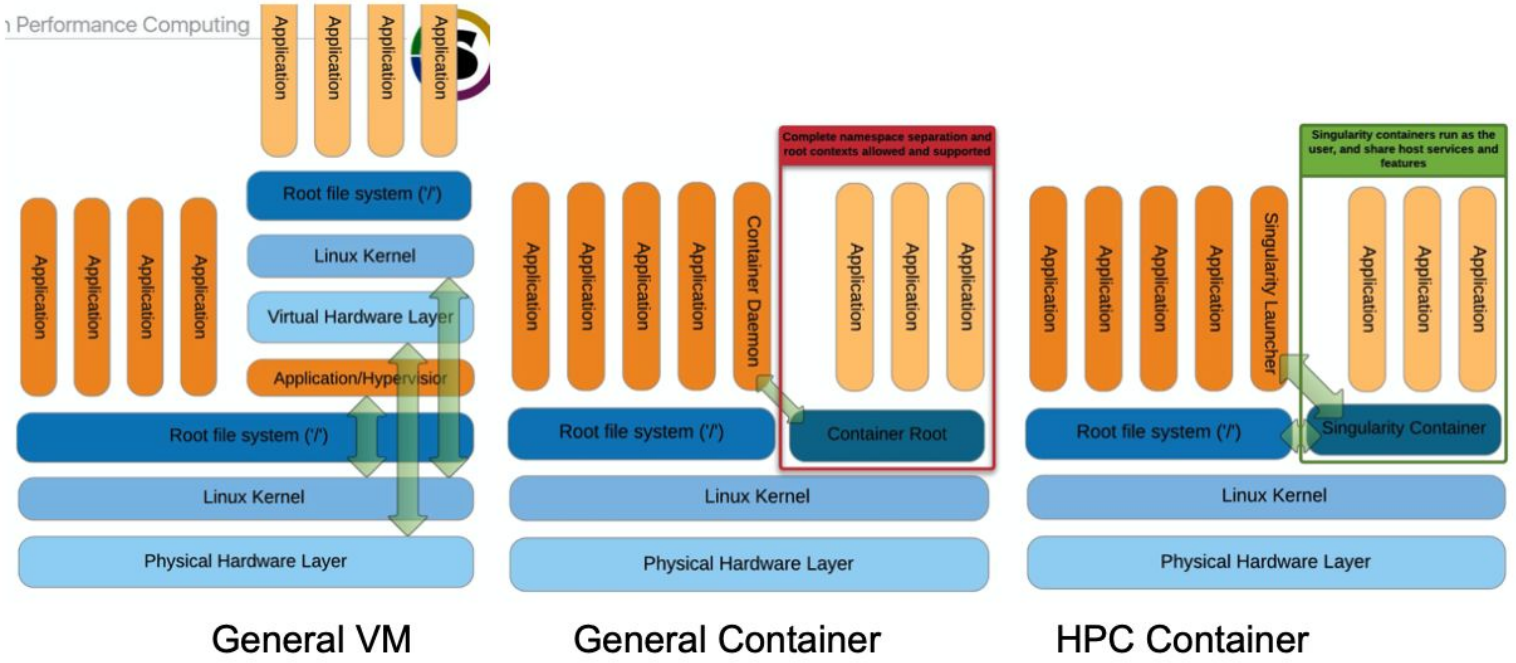
Why Singularity on HPCs?

1. Security: Docker gives superuser privileges. It's hard to give someone limited Docker access. Docker just wasn't designed to keep users out of each other's stuff. Singularity effectively runs as the running user and doesn't result in elevated access.

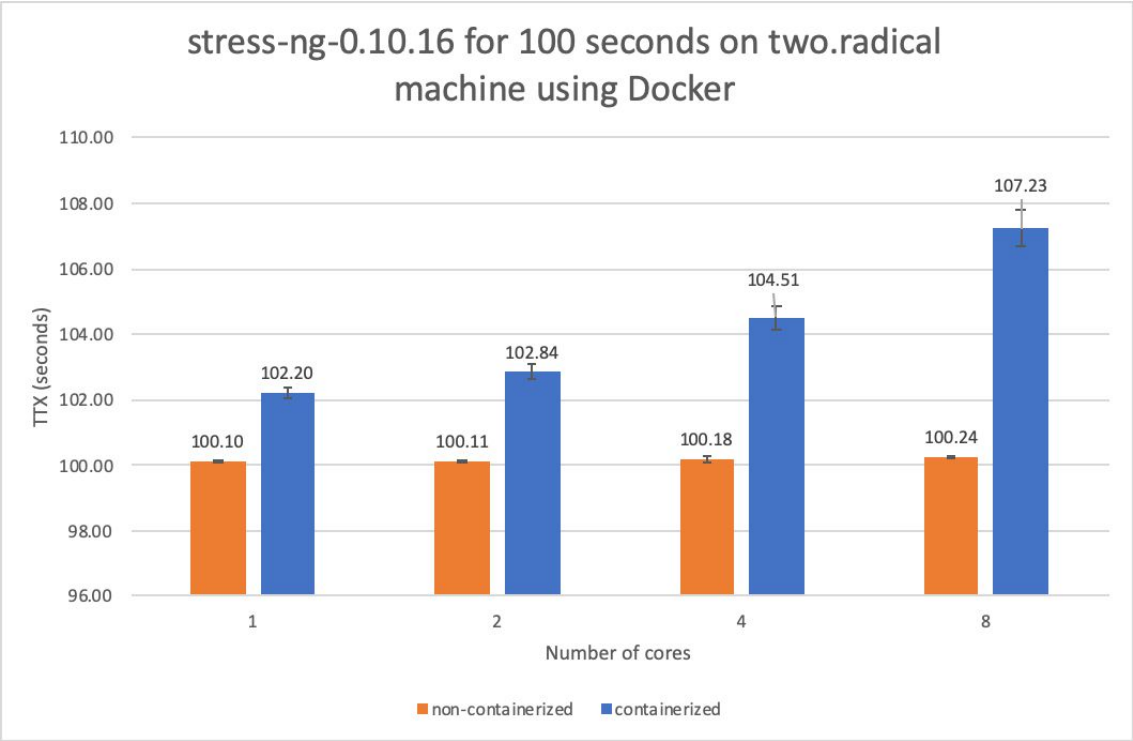
2. Scheduling: Users submit jobs with CPU/memory/time requirements. The Docker command is just an API client that talks to the docker daemon, so the resource requests and actual usages don't match. Singularity runs container processes without a daemon. They just run as child processes.

Conclusion: Docker is just better at running applications on VM or cloud infrastructure. Singularity is better for command line applications and accessing devices like GPUs or MPI hardware.

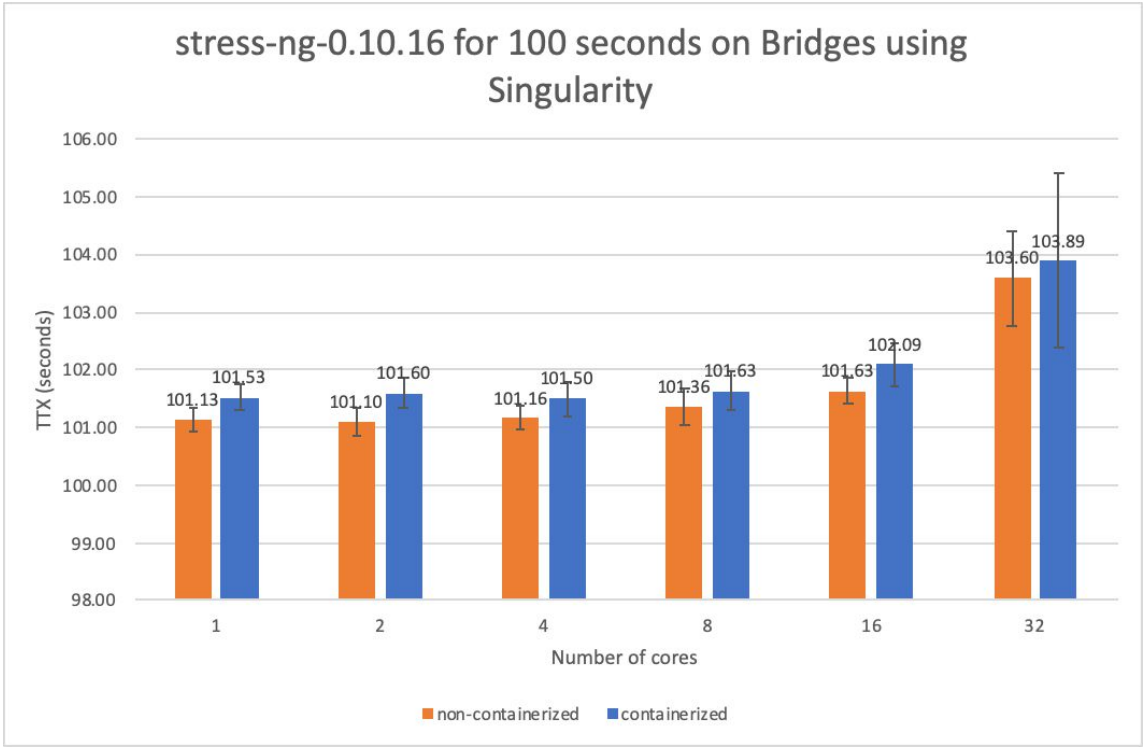
Virtual Machine vs. Docker vs. Singularity



Performance Characterization - Localhost



Performance Characterization - XSEDE Bridges



References