The State of Containers in Scientific Computing

Georg Rath

FOSDEM18/04.02.2018
NERSC

• Primary scientific computing facility of the office of science
• Two supercomputers (Cori, Edison), three clusters
  – Over 800,000 cores
  – Over 50 PB of storage in varying speeds
• Serving more than 6000 scientists
• Astrophysics, Climate & Earth Science, Chemistry, High Energy Physics, Genomics,…
HPC in a nutshell

- A (large) number of compute nodes
- Connected by a High Speed Network
- Accessing data stored in a parallel filesystem
- Run as a shared resource
- Orchestrated by a workload manager
What is the hardest problem in scientific computing?
Installing Software

- Center provided software stack via Environment Modules
  - Lmod, Modules Classic, Modules4
- Error-prone
- Slow*
- Unique
- Not portable
- Leads to user maintained software stacks
  - That depend on system
And then Docker hit...

- Simple
- Portable
- Reproducible
- Leveraging relatively stable Linux APIs
  - Namespaces
  - cgroups
…and we wept.

- High demand by users and admins alike
- Absolutely not built for HPC
- Security nightmare
  - access to Docker is root equivalent*
- A daemon on my compute node?
  - No
What do we want?

- A way to run Docker images on HPC systems
- No fancy stuff
  - Overlay networks
  - Plugins
  - Swarm/Kubernetes
- No daemon
- Secure
- Scalable
- Bonus: works on older kernels
## Great minds think alike

<table>
<thead>
<tr>
<th>Shifter</th>
<th>Charliecloud</th>
<th>Singularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Governance</td>
<td>NERSC</td>
<td>LANL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SyLabs Inc (started at LBNL)</td>
</tr>
<tr>
<td>Mechanism</td>
<td>setuid</td>
<td>usersns</td>
</tr>
<tr>
<td></td>
<td></td>
<td>setuid, usersns</td>
</tr>
<tr>
<td>Image Format</td>
<td>squashfs</td>
<td>Tar File</td>
</tr>
<tr>
<td></td>
<td></td>
<td>squashfs (since 2.4)</td>
</tr>
<tr>
<td>Noteworthy</td>
<td>Focus on HPC</td>
<td>Lightweight</td>
</tr>
<tr>
<td></td>
<td></td>
<td>“Scientific Docker”</td>
</tr>
</tbody>
</table>
What is container, really?

debootstrap stable containerfs/ http://deb.debian.org/debian/
unshare --mount --pid --fork
mount --bind containerfs containerfs
mount --make-private containerfs
cd containerfs

mount -t proc none proc
mount -t sysfs none sys
mount -t tmpfs none tmp
mount -t tmpfs none run

pivot_root . mnt
umount -l mnt
exec bash -i
Access to host hardware/libraries

- Violates containment
- Bind device file into container
- Inject host libraries into the container (eg. libcuda.so)
  - Manually or via libnvidia-container
- Requires ABI compatibility between host/container libs
- Does not work with static linking
- Glibc issues
FROM ubuntu:18.04

# install Tensorflow
RUN apt-get install python3-pip python3-dev
RUN pip3 install tensorflow

COPY ai.py /usr/bin/ai.py

CMD ["/usr/bin/ai.py"]
The need for speed

• Binary build of Tensorflow is not optimized
• Modern processors need vector instructions for performance
• Theoretical Peak Performance Intel Haswell
  – Scalar: ~ 130 GFLOPS
  – AVX: ~ 500 GFLOPS
• Let’s fix this…
An easy fix?

RUN LD_LIBRARY_PATH=${LD_LIBRARY_PATH} \
    bazel build --config=mkl \ 
    --config="opt" \ 
    --copt="-march=haswell" \ 
    --copt="-O3" \ 
    //tensorflow/tools/pip_package:build_pip_package && \
    mkdir ${WHL_DIR} && \
    bazel-bin/tensorflow/tools/pip_package/build_pip_package ${WHL_DIR}

* actual Dockerfile around 80 lines lot more sophisticated
** EasyBuild/Spack highly recommended
Does it pay off?

ResNet-50 on Intel Haswell (CPU only)

- **7x speedup**
- **images/second**
- **binary wheel**
- **built from source**
Portability

- Requires "cross-compiling"
- Different containers with different tags
- Or leverage Docker “fat manifest” containers
  - Introduced with Image Manifest v2.2
  - Specifies architecture and features
  - Not integrated yet
Conclusion

- Containers are a valuable tool for scientific computing
  - User defined software stack
- Containers are not a panacea
  - Portability and performance require work
  - Reproducibility over time will be challenging as well
- Leveraging proven tools in conjunction with containers provides great benefit
Questions?
Thank You