Using Docker containers for photon experiment simulations in HPC environments

Sergey Yakubov, Carsten Fortmann-Grote, Yves Kemp, Frank Schlünzen
Outline

- Introduction
- DESY HPC cluster (Maxwell)
- Docker in HPC cluster environments
- SIMEX HPC simulations with Docker
- Conclusions
Introduction

What is container?

Container manager

Operating System

Infrastructure

Virtual machine

App 1
Bins/Libs

App 2

App 3

Guest OS

Guest OS

Guest OS

App 1

App 2

App 3

Hypervisor

Host Operating System

Infrastructure

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 654220

NOBUGS 2016, Copenhagen, Denmark, 17.10.2016
Sergey Yakubov, DESY
Introduction

Why container?

- Lightweight
- Low overhead
- Micro-services
- Service orchestration
- Software development/testing
- Software deployment
Why container?

- Lightweight
- Low overhead
- Micro-services
- Service orchestration
- Software development/testing
- Software deployment
Introduction

Why Docker?

- Open-source
- 1,500 contributors
- Commercial support
- Docker hub to store images
- Can be used everywhere (well, almost)

Alternatives – LXC, rkt?
The European Cluster of Advanced Laser Light Sources

Maxwell HPC Cluster

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 654220

NOBUGS 2016, Copenhagen, Denmark, 17.10.2016

Sergey Yakubov, DESY
For each job we create an HPC cluster of Docker containers

- Secure (no root access for user)
- High-speed network
- Parallel file system
- Deployed using SLURM
- User friendly
Until February 2016 there was a serious lack of security

- User ID inside a Docker container matched user ID on the host system
  - root access inside Docker = root access to host
  - cannot use 3rd-party containers
  - cannot allow user to execute Docker commands

Since version 1.10 kernel user namespace can be used

- User ID and Group ID are isolated inside a container
  - Experimental kernel parameter in RedHat and Co. (available since version 7.2)

--enable-user-namespace=1
For each job we create an HPC cluster of Docker containers

$ docker run -d <.......> centos_mpi_benchmarks

- Using host network

  --net=host  
  Insecure (does not support user namespaces)!

- Using default bridge network

  Add infiniband devices

  --device=/dev/infiniband/uverbs0  --device=/dev/infiniband/rdma_cm

  Pass IPoIB interface to a container

  pipework ib0 <docker name> <ip address>/24
For each job we create an HPC cluster of Docker containers

```
$ docker run -d <.......> centos_mpi_benchmarks
```

- Sharing a folder in a parallel filesystem

```
-v /home/jdoe/test:/shared
```

- User namespaces should be respected by the filesystem
  - nfs
  - gpfs
  - beegfs
Since we use user namespace

- Shared folder should have read (write) permissions for everyone
  - alternative – stage data
- After job is finished – ownership of the files created in the shared folder must be changed

Alternatively – trusted images can be started without user namespaces

- What is trusted image?
- Docker authorization plugin is used for extra security
**DESY’s repositories**

- **Read-only repository**
  - No user namespaces
  - User cannot upload images
  - `docker exec -u root` - not allowed

- **Open (to DESY users) repository**
  - User namespaces are active
  - User can upload images
  - `docker exec -u root` - allowed

**Dockerhub**

**Third-party repositories (certified)**
Docker for Maxwell - Workflow

Docker in DESY HPC environment

- User submits a job to a resource management (SLURM)

  ```
  #SBATCH --comment="use_docker;max-adm01:5001/centos_mpi_benchmarks;
  /home/yakubov/container_shared:/shared"
  ...
  ```

- SLURM puts the job in a common queue
- As soon as resources are available, SLURM starts a container on each of the allocated nodes (using prolog script)

  ```
  docker run -d \
  -v $DOCKER_HOST_PATH1:$DOCKER_CONTAINER_PATH1 … \
  --name=docker_$SLURM_JOB_ID \
  $DOCKER_IMAGE
  ```
Docker in DESY HPC environment

- And creates a virtual network (SLURM daemon runs as root)

```
/root/bin/pipework ib0 docker_${SLURM_JOB_ID}${mask}.${nnode}/24
```

- User sets-up job steps to be executed (in a script or interactively)

```
docker_run simex.py
docker_mpirun -n 32 simex.py
```

- SLURM removes all containers using epilog script after the job is finished
Examples - MPI Bandwidth and Latency Tests

- Two Maxwell compute nodes, Mellanox Infiniband 56 Gbs (4X FDR)
- Host system vs Docker container
  - ib utilities
    - | Host system | Docker |
      | ib_send_bw  | 44 Gbs | 46.9 Gbs |
      | ib_send_lat | 1.1 μs | 1.07 μs |
  - mpi_benchmarks (source: Lawrence Livermore National Laboratory)
    - | Host system | Docker |
      | mpi_bandwidth | 45.7 Gbs | 44.9 Gbs |
      | mpi_latency   | 1.99 μs  | 1.99 μs  |
Examples – HPCG/HPL Benchmarks

- High-Performance Linpack Benchmark
  [http://www.netlib.org/benchmark/hpl](http://www.netlib.org/benchmark/hpl)

- High Performance Conjugate Gradients
  [http://hpcg-benchmark.org](http://hpcg-benchmark.org)

- Both used by Top500 (officially/unofficially yet)

<table>
<thead>
<tr>
<th>HPCG rank</th>
<th>Cores</th>
<th>Top rank</th>
<th>HPL (PFlops)</th>
<th>HPCG (PFlops)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSCC Tianhe-2</td>
<td>3 120 000</td>
<td>2</td>
<td>33.86</td>
<td>0.58</td>
</tr>
<tr>
<td>RIKEN K computer</td>
<td>705 024</td>
<td>5</td>
<td>10.51</td>
<td>0.46</td>
</tr>
<tr>
<td>DOE Titan</td>
<td>560 640</td>
<td>3</td>
<td>17.59</td>
<td>0.32</td>
</tr>
<tr>
<td>HLRS Cray XC40</td>
<td>185 088</td>
<td>9</td>
<td>5.64</td>
<td>0.14</td>
</tr>
</tbody>
</table>
### Examples – HPCG/HPL Benchmarks

**Benchmark results on Maxwell cluster**

<table>
<thead>
<tr>
<th></th>
<th>Cores</th>
<th>HPL (TFLops)</th>
<th>HPCG (TFLops)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maxwell</td>
<td>64 (2 nodes)</td>
<td>1.56</td>
<td>0.033</td>
</tr>
<tr>
<td>Maxwell + Docker</td>
<td>64 (2 nodes)</td>
<td>1.56</td>
<td>0.033</td>
</tr>
<tr>
<td>Maxwell</td>
<td>368 (15 nodes)</td>
<td>9.0</td>
<td>0.192</td>
</tr>
<tr>
<td>Maxwell + Docker</td>
<td>368 (15 nodes)</td>
<td>9.0</td>
<td>0.192</td>
</tr>
</tbody>
</table>
SIMEX (see presentation from Carsten Fortmann-Grote)

- Deployment is non-trivial
  - Each calculator has its own dependencies and install script
  - Need to install in various environments
  - Users can have admin/non-admin rights
  - Experienced developers/unexperienced users

- Possible solutions
  - Use CMake build system
  - Use binary packages (deb, rpm, ...)
  - Use Docker containers
Deployment using Docker containers

- SIMEX image is on the Docker hub
- Everything is installed inside the image
- To start working with it just type

```
docker run -it simex bash
```

- Or submit a job with python script to SLURM
X-ray wavefront propagation calculator

- Propagation of light through optical elements
- Utilizes SRW (Synchrotron Radiation Workshop) library
- C++ core + python wrappers
- Hybrid OpenMP/MPI parallelization

<table>
<thead>
<tr>
<th>Threads x MPI processes</th>
<th>Number of nodes</th>
<th>Total time</th>
<th>Time/file</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x1</td>
<td>1</td>
<td>11h</td>
<td>1031 s</td>
</tr>
<tr>
<td>40x1</td>
<td>1</td>
<td>65 min</td>
<td>98 s</td>
</tr>
<tr>
<td>4x10</td>
<td>4</td>
<td>7.5 min</td>
<td>45 s</td>
</tr>
<tr>
<td>8x5</td>
<td>8</td>
<td>4.2 min</td>
<td>51 s</td>
</tr>
</tbody>
</table>

Single source file

40 source files
X-ray wavefront propagation calculator

- Propagation of light through optical elements
- Utilizes SRW (Synchrotron Radiation Workshop) library
- C++ core + python wrappers
- Hybrid OpenMP/MPI parallelization

### Performance Results

<table>
<thead>
<tr>
<th>Threads x MPI processes</th>
<th>Number of nodes</th>
<th>Total time</th>
<th>Time/file</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x1</td>
<td>1</td>
<td>11h</td>
<td>1031 s</td>
</tr>
<tr>
<td>4x10</td>
<td>4</td>
<td>7.5 min</td>
<td>45 s</td>
</tr>
<tr>
<td>8x5</td>
<td>8</td>
<td>4.2 min</td>
<td>51 s</td>
</tr>
</tbody>
</table>

160x speed-up

Single source file

40 source files
**Photon diffraction calculator**

- Absorption, emission, and scattering of radiation
- Utilizes SingFEL photon diffractor library
- C++ core + python wrappers
- MPI parallelization

![Graph showing linear speed-up with 200,000 diffraction patterns]
Conclusions

- Running Docker containers on an HPC cluster is possible and
  - does not break system security
  - does not introduce overhead
  - uses general resource scheduling procedures
- Simplifies software development and deployment
  - Can be developed and compiled off-site and deployed instantly on a cluster
- Photon experiment simulations can be efficiently performed on an HPC cluster using “dockerized” SIMEX platform.
Thank you for your attention!