Using Docker container virtualization in DESY HPC environment

Supported by

EUCALL

Sergey Yakubov, Frank Schlünzen, Sven Sternberger, Yves Kemp

HEPiX Spring 2016 Workshop
21.04.2016 Zeuthen

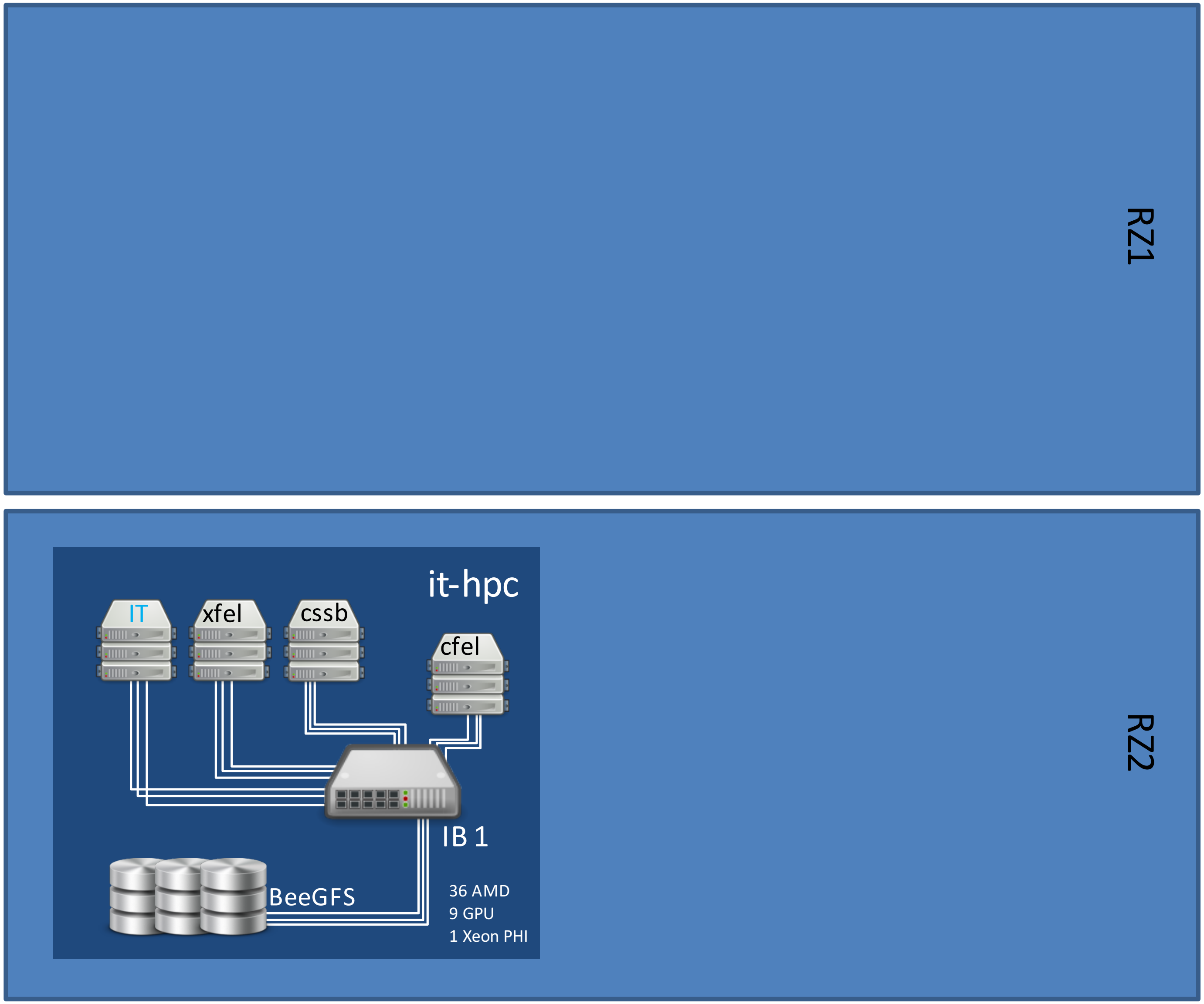
HELMHOLTZ | ASSOCIATION

DESY

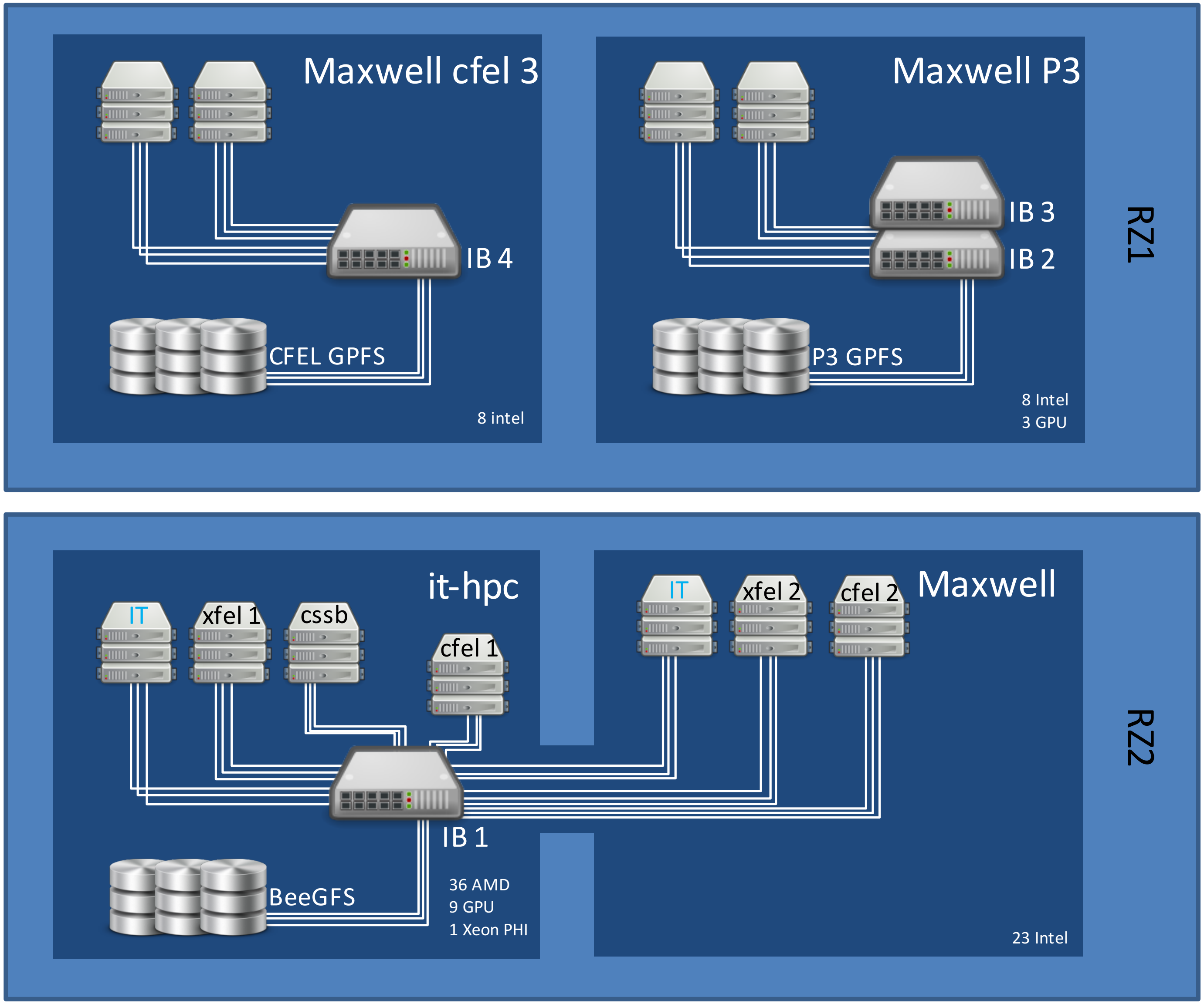
Contents

- DESY HPC cluster (Maxwell)
  - Hardware/Software
  - Resource management system (SLURM)
- Docker
  - Concept
  - Tools
- Docker in HPC cluster environment
  - Security issues
  - Network/IO
  - Workflow (semi) automation
- Examples
- Conclusions/Outlook

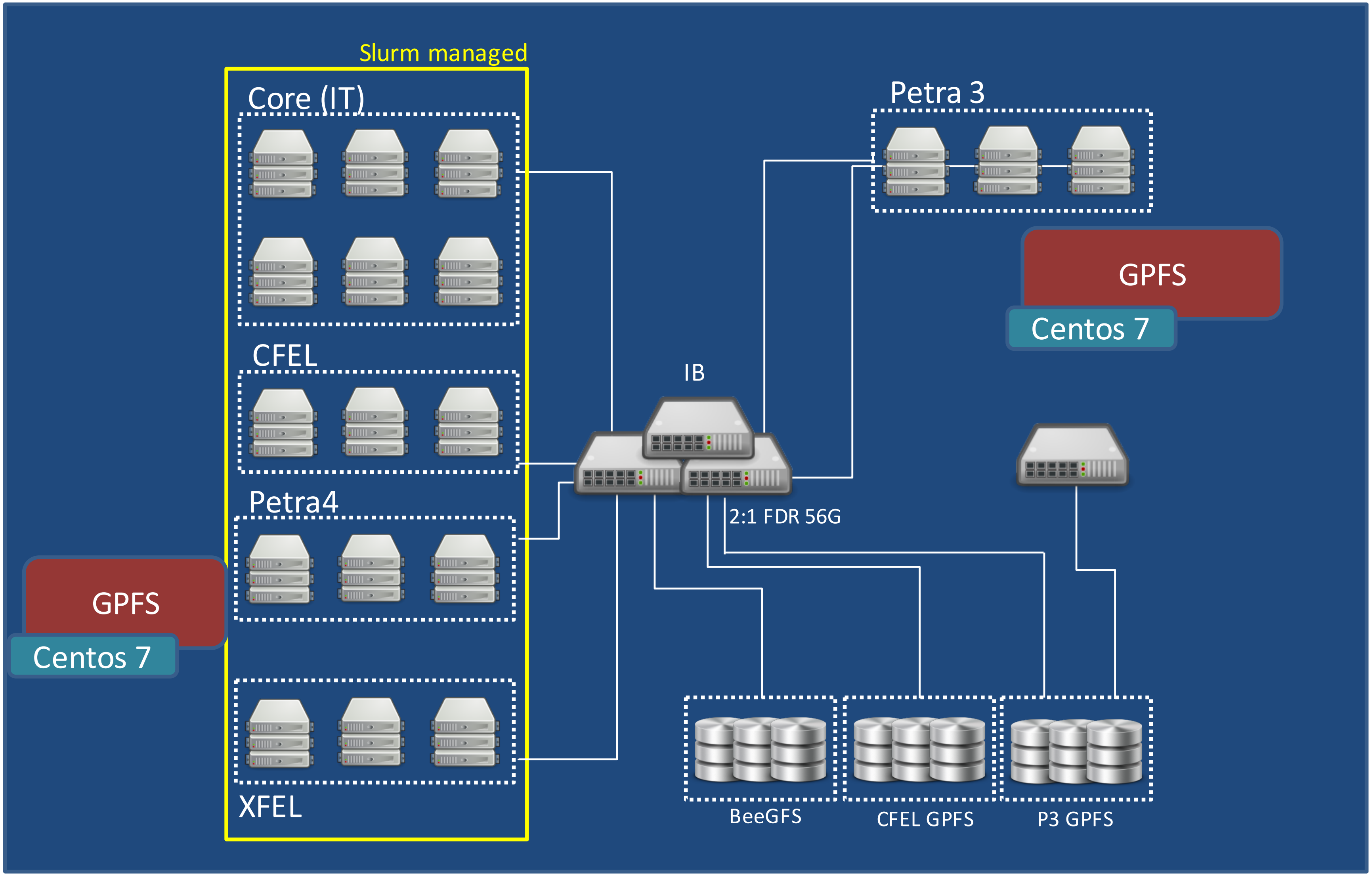
HPC Infrastructure 2015



HPC Infrastructure 02/2016



Maxwell HPC Cluster (04-05/2016)



Maxwell HPC Cluster

> HPC infrastructure

- 89 nodes
- 4592 cores in total (Intel + AMD) , will expand
- 27 TB RAM Total
- 19 nodes with NVIDIA Tesla K20/K40 GPUs
- Fat-tree Infiniband network (blocking factor 2)
- CentOs 7.2 Linux
- Environment Modules

> Resource management - SLURM

- Group dedicated / common partitions
- Tunable restrictions/prioritization/preemption
- PAM prevents non-authorized login to working nodes
- MySQL based accounting

> User point of view - similar to a virtual machine

- Application and all dependencies are installed inside the container
- Can be run on any operating system (Linux, Windows, MacOs)
- Can be run anywhere (laptop, cluster, cloud)

> Implementation is different

- Shares host system kernel
- Isolation via kernel namespaces and cgroups
- Union-capable (layered) file system
- Libcontainer wraps all together

> Result

- Starts faster, much less overhead
- Less isolation

- Docker is an open-source project
  - Rapidly developing since 2013
  - About 1,100 contributors
  - Commercial plans available
- Docker toolbox
  - Docker engine – build and runs containers
  - Docker swarm - clustering of containers
  - Docker machine – sets up Docker Engine (Windows, MacOS, remote clients)
  - Docker registry – provides storage of Docker container images
  - …
- Docker for HPC – not so much developments at the moment
  - SHIFTER from NERSC – under development, pre-release version available

Docker in Maxwell HPC cluster environment

- 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 existing resource management system
  - Does not require too much input from user (automation)

Docker in Maxwell HPC cluster environment - Security

- Until February 2016 there was a serious lack of security
  - User with rights to start a Docker container had basically root access to the host system
  - Only pre-installed and verified containers should be allowed to run on a system
  - Execution of a container should be controlled as well
- 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
```

Docker in Maxwell HPC cluster environment – Network

> 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
```

  - Create virtual IPoIB device

```
pipework ib0 <docker name> <ip address>/24
```

Docker in Maxwell HPC cluster environment – Parallel filesystem

> 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

Docker in Maxwell HPC cluster environment - Workflow

- User submits a job to resource management

```
#SBATCH –ntasks=32
#SBATCH –comment="use_docker;centos_mpi_benchmarks;
/home/jdoe/container_shared;/shared;IB"
```

- 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 $DOCKER_HOST_PATH:$DOCKER_CONTAINER_PATH \
--name=docker_$SLURM_JOB_ID \
--device=/dev/infiniband/uverbs0 --device=/dev/infiniband/rdma_cm  \
$DOCKER_IMAGE
```

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

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

Docker in Maxwell HPC cluster environment - Workflow

- User creates a hostfile, puts it into a working directory

```
slurm_make_hostfile
cp hosts /home/jdoe/container_shared
```

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

```
docker exec -u dockeruser docker_$SLURM_JOB_ID \
mpirun -hostfile /shared/hosts -n 32 hello_world
```

- SLURM removes all containers after job is finished using epilog script (virtual interfaces are removed automatically)

Examples - MPI Bandwidth and Latency Tests

- 2 compute nodes, Mellanox Infiniband 56 Gbs (4X FDR)
- We compare results of Maxwell runs on host system and inside a 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)
- High Performance Conjugate Gradients (http://hpcg-benchmark.org)
- Both used by Top500 (officially/unofficially yet)

HPCG rank	Cores	Top rank	HPL (PFlops)	HPCG (PFlops)
NSCC Tianhe-2	3 120 000	1	33.86	0.58
RIKEN K computer	705 024	4	10.51	0.46
DOE Titan	560 640	2	17.59	0.32
HLRS Cray XC40	185 088	8	5.64	0.14

Examples – HPCG/HPL Benchmarks

- Maxwell HPC cluster (using Intel tuned binaries)

	Cores	HPL (TFLops)	HPCG (TFLops)
Maxwell	64 (2 nodes)	1.56	0.033
Maxwell+Docker	64 (2 nodes)	1.56	0.033
Maxwell	368 (15 nodes)	9.0	0.192
Maxwell+Docker	368 (15 nodes)	9.0	0.192

Conclusions

- Implementation of Docker containers in an HPC cluster infrastructure is possible and
  - does not break system security
  - does not introduce overhead
  - uses general resource scheduling procedures
- Simplifies software development and deployment
  - Software can be developed and compiled off-site and deployed instantly on a cluster
  - No need to install libraries on the cluster
  - Easy to support different versions

Outlook

- Need to “sell“ it to users
  - information
  - support
  - test cases
- Create own repository for Docker images
  - Pre-installed images
  - User images
- Try Docker native networking options (Overlay)
- HPC in a cloud ?

Thank you for your attention!