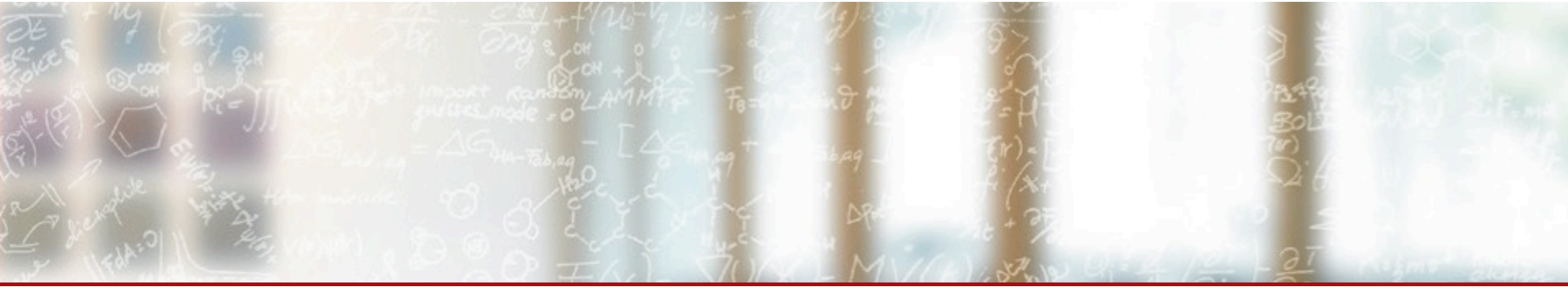
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Versatile HPC Platforms with Containers

HPC-CH

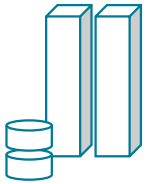
M. Gila, S. Alam, L. Benedicic, F. Cruz, A. Madonna, K. Mariotti - CSCS

October 26, 2017

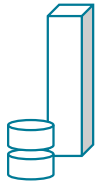
Consolidate HPC resources & platforms

A few years back

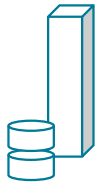
Monte Rosa



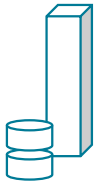
Tödi



Rothorn

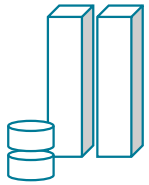


Pilatus

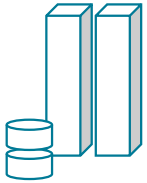


Matterhorn

Fuji

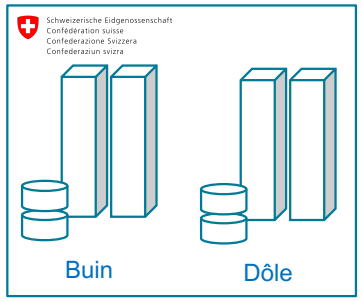


Castor

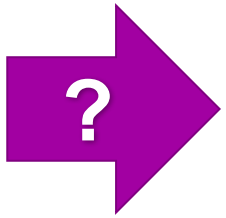


Pollux

Phoenix

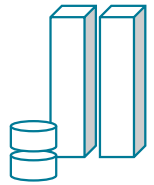


Buin Dôle

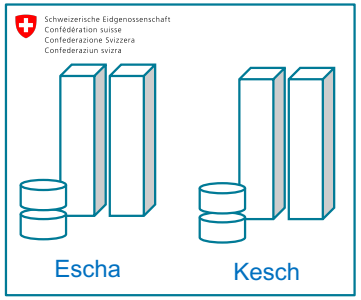


Beyond today

Piz Daint



?



Escha Kesch

How do you make your HPC resources more versatile?

1. Shape your infrastructure

2016→

- 5,320 hybrid nodes (Cray XC50)
  - Nvidia Tesla P100
  - Intel Xeon E5-2690 v3
  - 16 GB HBM2
  - 64 GB DDR4
- 1,431 multi-core nodes (Cray XC40)
  - 2 x Intel Xeon E5-2695 v4
  - 64 and 128 GB DDR4
- Cray Aries dragonfly interconnect
  - ~36 TB/s bisection bandwidth
- Sonexion Lustre file system
  - ~9 PB (Sonexion 3000) & 2.7 PB (Sonexion 1600)
  - External GPFS on selected nodes

2013

Computing

Visualization

Data analysis

\+

2016 →

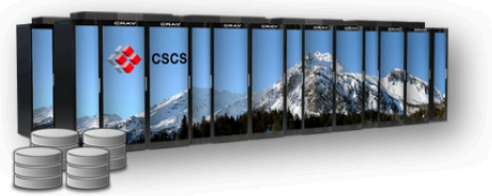
Pre-post processing

Data mover

Data Warp

Machine learning

Deep learning



CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre — User Lab

University of Zurich

MARVEL

PAUL SCHERRER INSTITUT PSI

PRACE

CHIPP Swiss Institute of Particle Physics

2. Embrace new technologies and expand the applications that can run on infrastructure

\+ SHIFTER

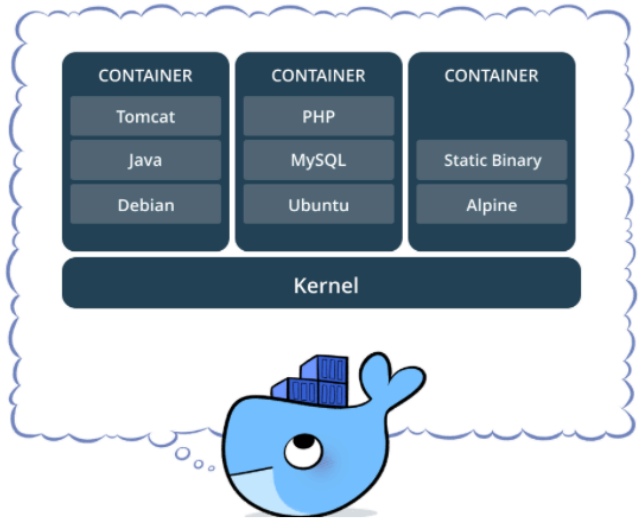
Containers!

What is a container?

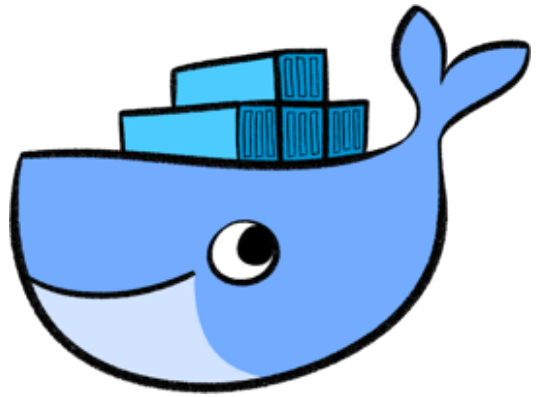
Package software into standardized units for development, shipment and deployment

A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings. Available for both Linux and Windows based apps, containerized software will always run the same, regardless of the environment. Containers isolate software from its surroundings, for example differences between development and staging environments and help reduce conflicts between teams running different software on the same infrastructure.

https://www.docker.com/what-container



Docker



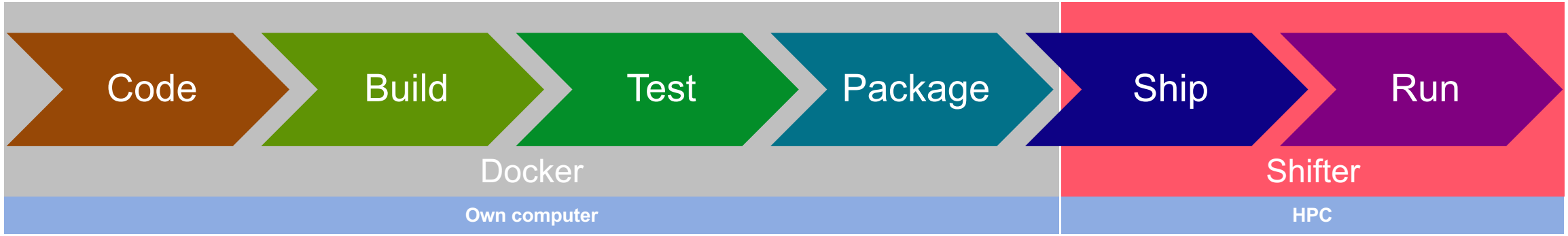
- The most used container technology
- Allows users to run the full workflow on their own computer

- But it has some caveats:
  - No integration with workload managers
  - No user isolation (on shared filesystems)
  - Requires daemons everywhere
  - Not designed for diskless clients

'Relaxed' security model

Shifter

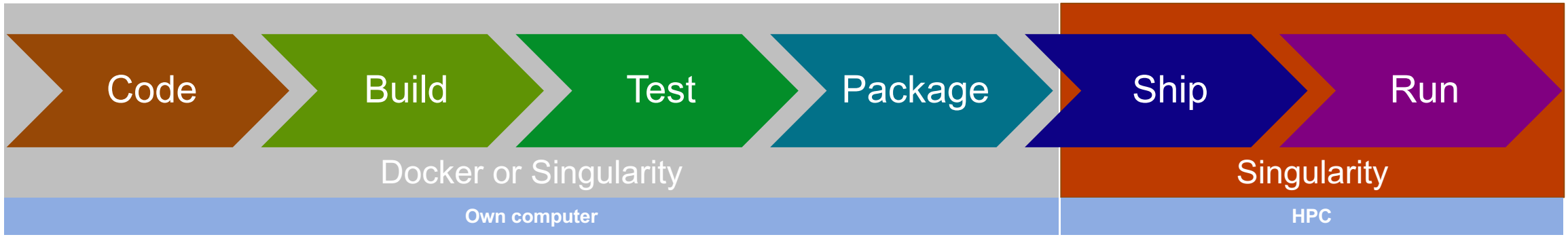
- Open source technology (BSD-like)
- Built for HPC centers, in particular for Cray systems
- Allows users to run the full workflow on their own computer and then ship it to a large supercomputer without changes



- This solves most limitations with Docker
  - Integrates with workload managers
  - User isolation on shared filesystems
  - No daemons
  - Designed for diskless clients

Singularity

- Open source technology (BSD-3)
- Allows users to run the full workflow on their own computer and then ship it to a large supercomputer without changes
- Permits to build the workflow on a supercomputer



- This solves most limitations with Docker
  - Can integrate with workload managers (Slurm)
  - User isolation on shared filesystems
  - No daemons

Containers @ CSCS

- Shifter integrates seamlessly with our Cray Systems
- It is the *natural* choice for us
- Three container types:
  - Third party container (user defined)
  - CSCS-sanctioned containers (anything *ethzcscs/* or *cscs/*)
  - Standard containers (OS, Python, i.e. *centos/* or *python3/*)

- Containers are **hardware-** and **platform-agnostic** by design
  - How do we go about **accessing specialized hardware** like GPUs?
  - What about **MPI**?

- CSCS and NVIDIA co-designed a solution that provides direct access to the GPU
- CSCS extended design to the MPI stack

GPU access with Shifter

- CSCS and NVIDIA co-designed a solution that provides:
  - Direct access to the GPU device characters
  - Automatic discovery of the required libraries at runtime
  - NVIDIA's DGX-1 software stack based on this solution


```
$ nvidia-docker run ethcscs/dockerfiles:cudasamples8.0 ./deviceQuery
./deviceQuery Starting...
CUDA Device Query (Runtime API) version (CUDART static linking)
Detected 1 CUDA Capable device(s)
CUDA Driver Version / Runtime Version 8.0 / 8.0
[...]
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 8.0, CUDA Runtime
Version = 8.0, NumDevs = 1, Device0 = GeForce 940MX
Result = PASS
```

```
$ shifterimg pull ethcscs/dockerfiles:cudasamples8.0
$ salloc -N 1 -C gpu
$ srun shifter --image=ethcscs/dockerfiles:cudasamples8.0 ./deviceQuery

./deviceQuery Starting...
CUDA Device Query (Runtime API) version (CUDART static linking)
Detected 1 CUDA Capable device(s)
CUDA Driver Version / Runtime Version 8.0 / 8.0
[...]
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 8.0, CUDA Runtime
Version = 8.0, NumDevs = 1, Device0 = Tesla P100-PCIE-16GB
Result = PASS
```

MPI with Shifter

- CSCS has extended MPI support, based on the MPICH Application Binary Interface (ABI) compatibility

```
$ srun -n 2 shifter --mpi --image=osu-benchmarks-image ./osu_latency
```

- The MPI library from the container is swapped by the Shifter runtime
- The ABI-compatible MPI from the host system is checked automatically
  - Hardware acceleration is **enabled**

MPI with Shifter - OSU benchmark

```
$ srun -n 2 shifter --mpi --image=osu-benchmarks-image ./osu_latency
```

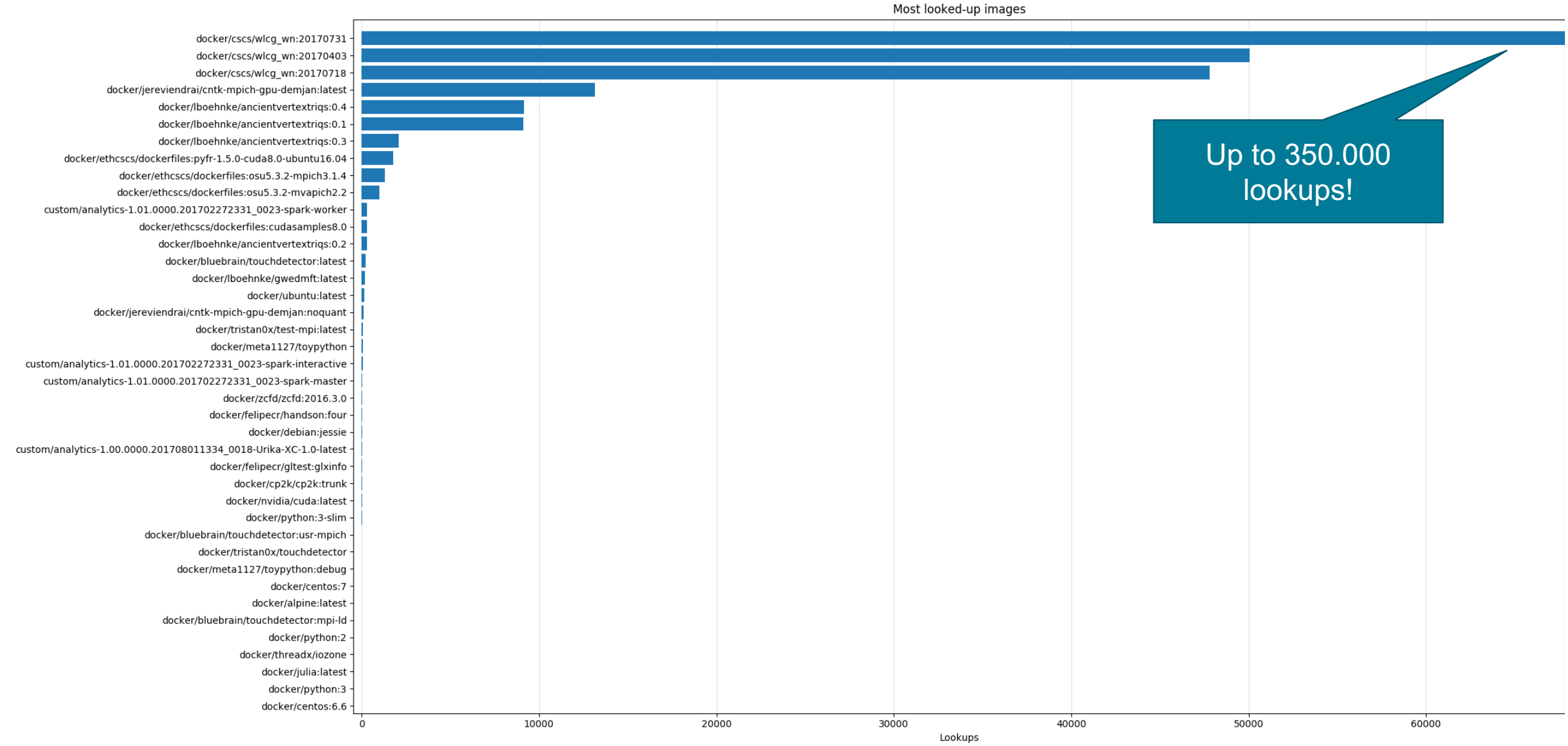
- Host MPI:
  - Cray MPT 7.5.0
  - Cray Aries Interconnect
- Container MPI:
  - MPICH v3.1 (A)
  - MVAPICH2 2.1 (B)
  - Intel MPI Library (C)
- Native performance

		Shifter MPI support *Enabled*			Shifter MPI support *Disabled*		
Size	Native	A	B	C	A	B	C
32	1.1	1.00	1.00	1.00	4.35	6.17	4.41
128	1.1	1.00	1.00	1.00	4.36	6.15	4.51
512	1.1	1.00	1.00	1.00	4.47	6.22	4.56
2K	1.6	1.06	1.00	1.06	4.66	5.03	4.04
8K	4.1	1.00	1.02	1.02	2.17	2.02	1.86
32K	6.5	1.03	1.03	1.03	2.10	2.17	1.91
128K	16.4	1.01	1.01	1.01	2.63	2.84	1.95
512K	56.1	1.00	1.01	1.01	2.23	1.78	1.67
2M	215.7	1.00	1.00	1.00	2.02	1.41	1.37

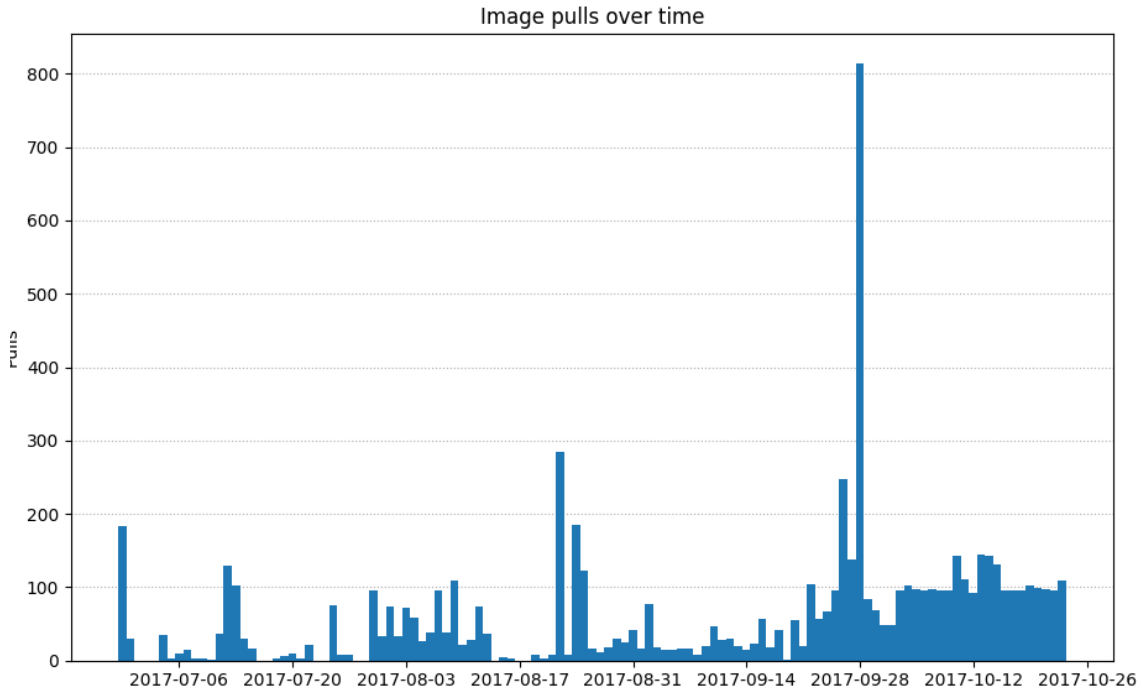
Table 4: Results from `OSU_latency` on Piz Daint: Native runs use Cray MPT 7.5.0 over Cray Aries interconnect; relative performance against native is reported for containers with (A) MPICH 3.1.4, (B) MVAPICH2 2.2, and (C) Intel MPI library using Shifter with MPI support *enabled* and *disabled*.

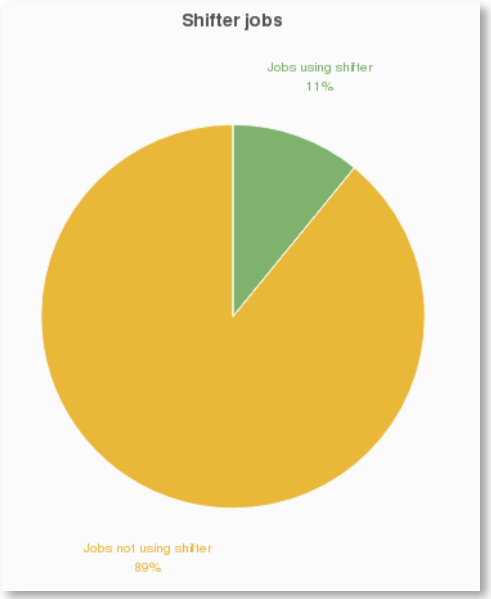
Container image utilization on Piz Daint

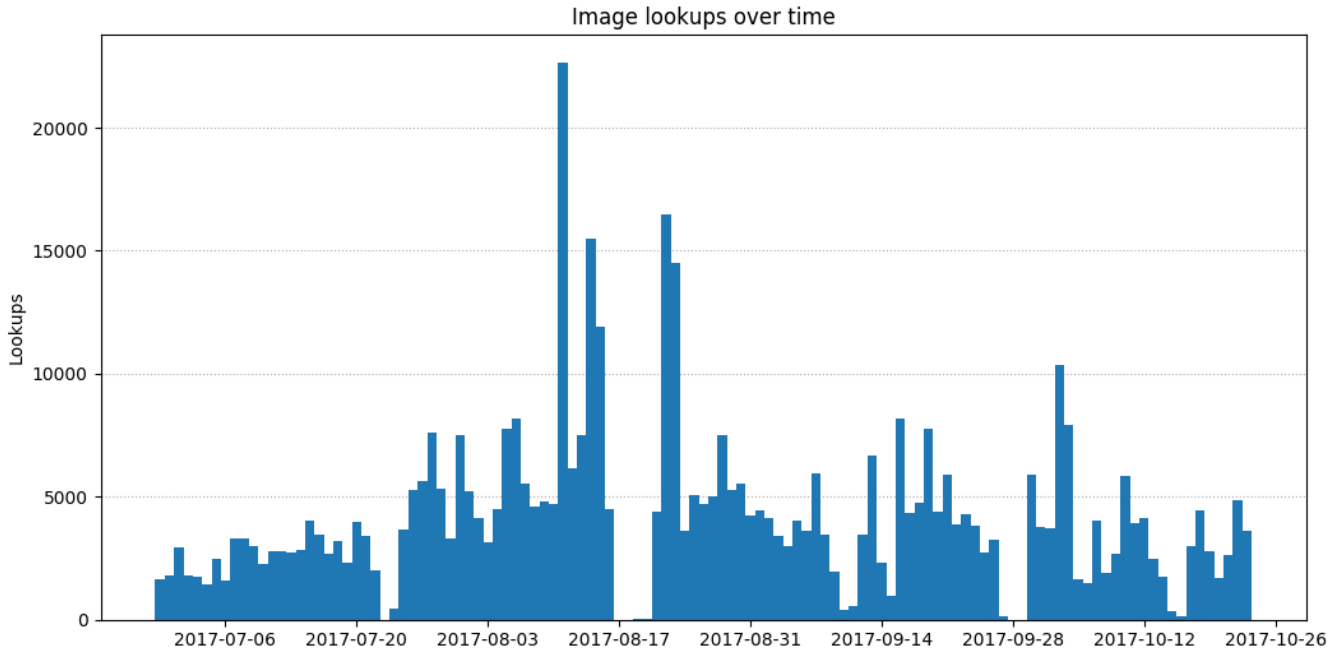
From 2017-06-27 to 2017-10-23



More statistics!







- Constant rate of image pulls → users are utilizing and generating new images
- About 5000 activations/day
- Any given day, about 10% of the jobs run in a container

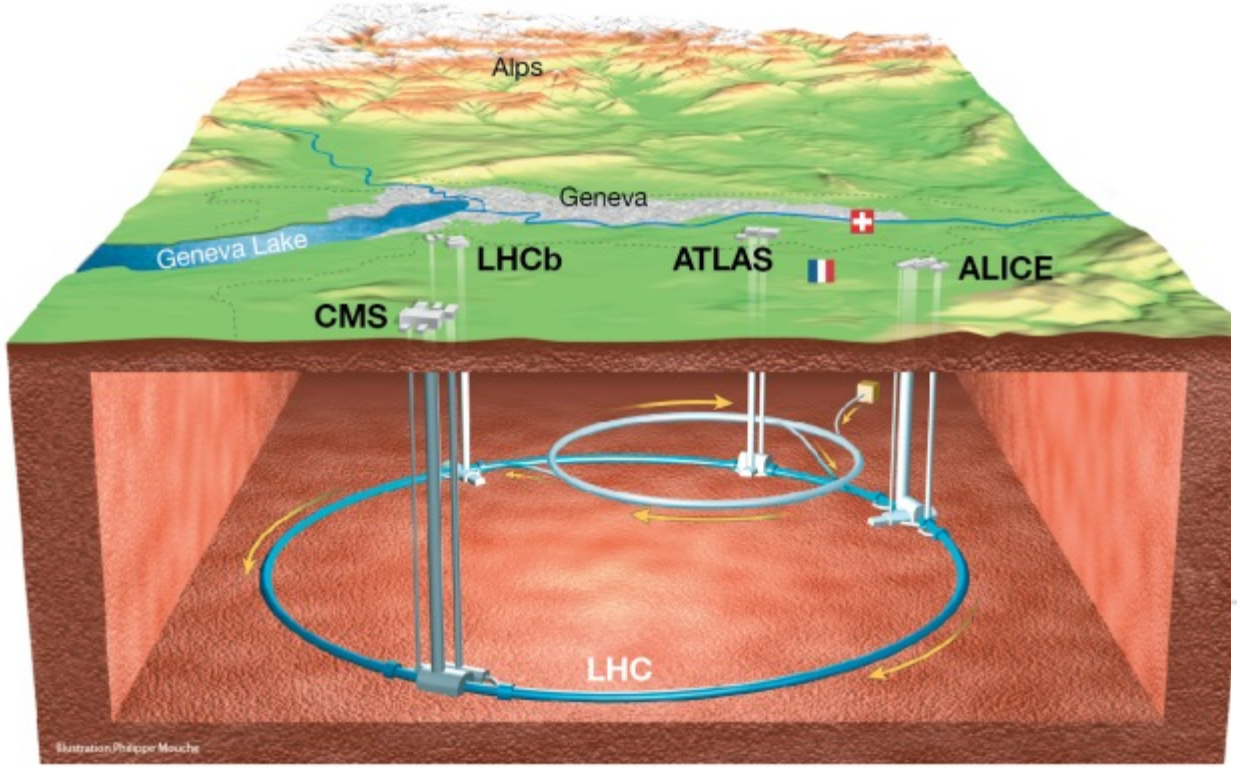
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

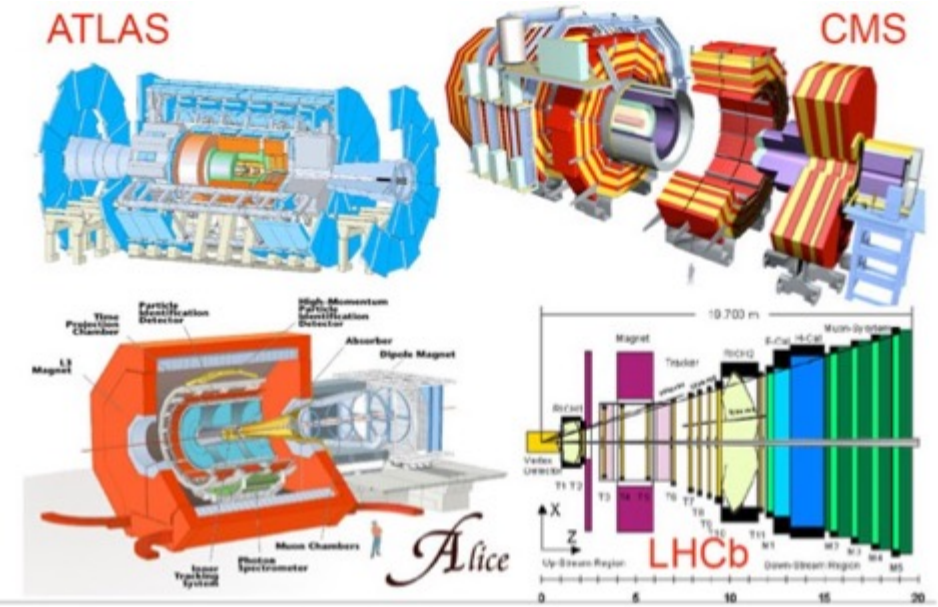
ETH zürich

A practical, not so HPC example: LHConCRAY

This work has been supported by the Swiss National Science Foundation

CERN Large Hadron Collider (LHC) Experiments





The Large Hadron Collider (LHC) with its four detector experiments (ATLAS, CMS, LHCb and ALICE) sits in a 27-km-long *(~17 miles)* circular tunnel, 100m below the ground at the European Organization for Nuclear Research (Cern) on the French-Swiss border, north of Geneva.
(cern.ch)

Worldwide LHC Computing Grid (WLCG)

http://wlcg-public.web.cern.ch/tier-centres

- Tier 0 (CERN and Hungary)
  - safe-keeping of the **raw data** (first copy=$$$$$$$$$$?) first pass reconstruction
  - distribution of **raw data**
  - reconstruction **output** to the **Tier 1s** reprocessing of **data** during LHC down-times
- Tier 1 (13 centers)
  - safe-keeping of a proportional share of raw and reconstructed **data**
  - large-scale reprocessing and safe-keeping of corresponding **output**
  - distribution of **data to Tier 2s**
  - safe-keeping of a share of simulated **data** produced at these **Tier 2s**
- Tier 2 (~ 150 centers)
  - specific analysis tasks
  - proportional share of simulated event production and reconstruction
- Tier 3
  - Local access cluster, no formal agreement with WLCG

LHConCRAY – the problem

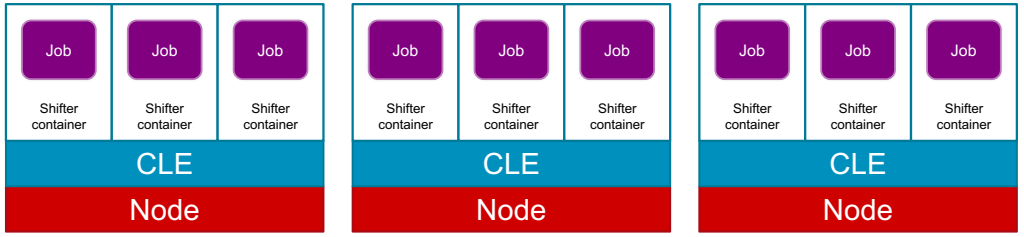
- CSCS runs Phoenix, a dedicated infrastructure holding the Swiss Tier-2 for three experiments: ATLAS, CMS, LHCb

- The main objective is to be able to run Tier-2 workflows on Piz Daint
  - **Without changes** on the original workflows
  - With **equivalent functionality and performance** to the dedicated system (Phoenix)

- There are many differences between Cray XC systems and clusters with several limiting factors (CVMFS, network protocols, etc.), but the first and most important one is that WLCG software stack and middleware is certified for **RHEL-compatible only** operating systems

- How do you run only RHEL-compatible software on Cray Linux Environment? (based on SLES12)

LHConCRAY – the solution



- Containers!

- We created a Docker image based on CentOS 6.9 (*cscs/wlcg_wn*, available in Dockerhub) that has everything that WLCG jobs require from the OS

- Each WLCG job runs within the boundaries of its own container using shifter and Slurm's limits/cgroups

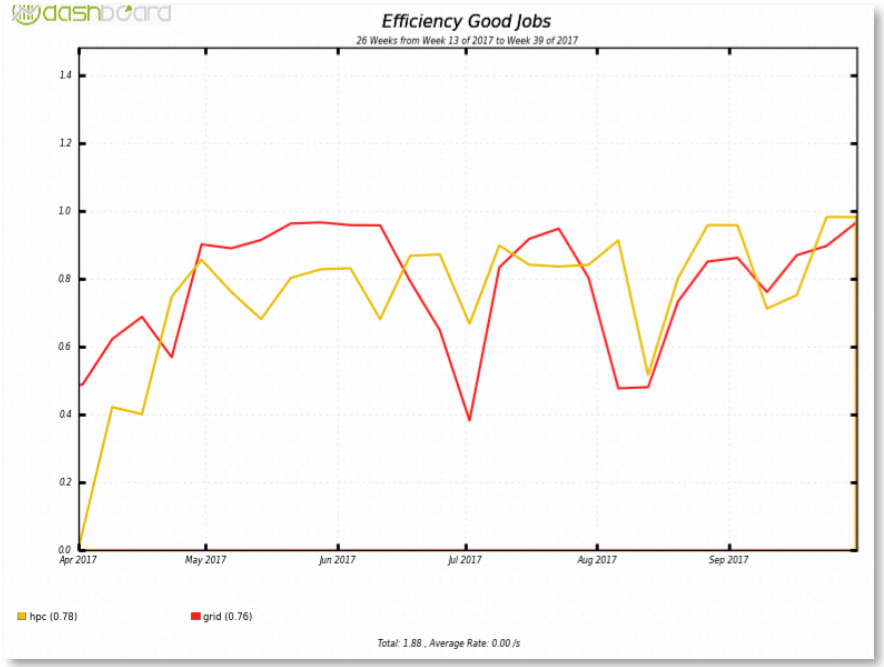
- As a result of the project, today Piz Daint runs **production** WLCG jobs without modifications and with similar performance to those in Phoenix

LHConCRAY – more statistics!

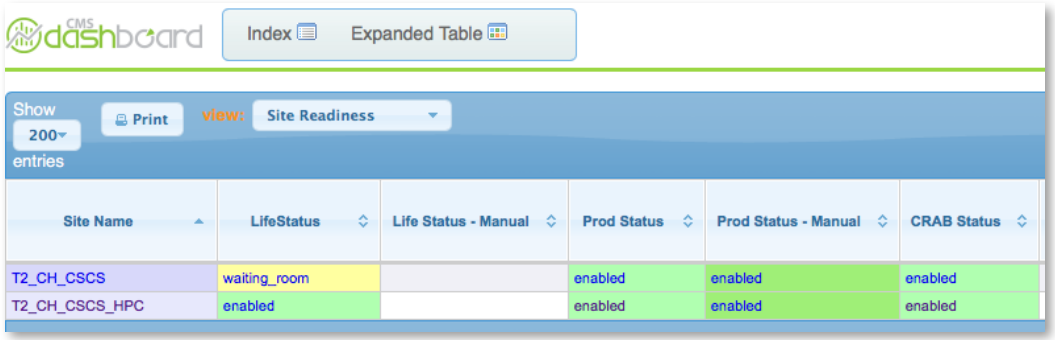
- ATLAS

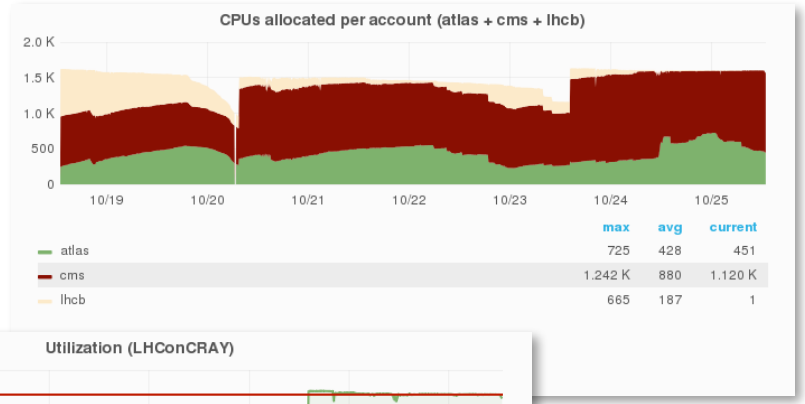
Sites (PanDA resources) (MCP sites are below)

PanDA resource Queue name (where different)	GOC site name	Cloud	Status	Tier
ANALY_CSCS	CSCS-LCG2	DE	online	T2D
ANALY_CSCS-HPC	CSCS-LCG2	DE	online	T2D
CSCS-LCG2 CSCS-LCG2-all-prod-CEs	CSCS-LCG2	DE	online	T2D
CSCS-LCG2-HPC	CSCS-LCG2	DE	online	T2D
CSCS-LCG2-HPC_MCORE	CSCS-LCG2	DE	online	T2D
CSCS-LCG2_MCORE CSCS-LCG2-all-prod-CEs_MCORE	CSCS-LCG2	DE	online	T2D

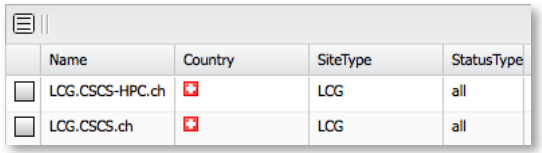


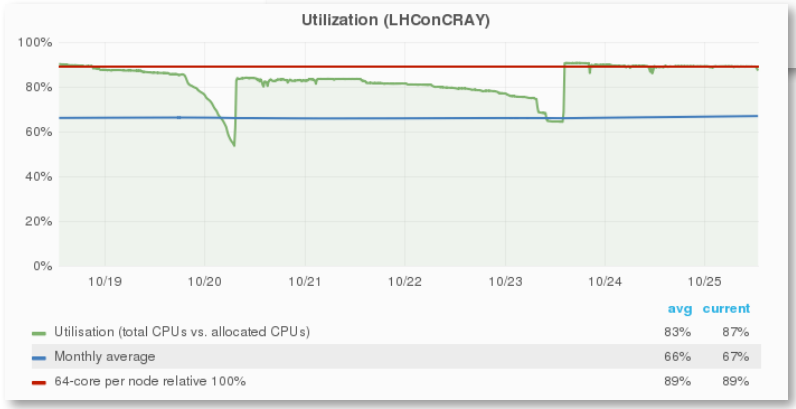
- CMS





- LHCb





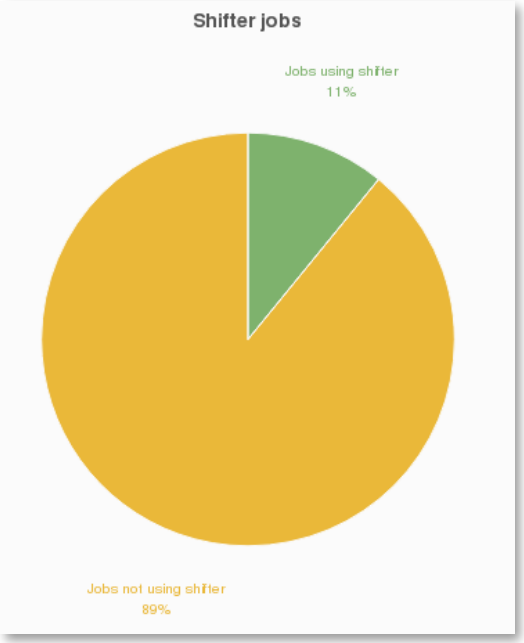
Conclusion

Conclusion

- Containers can **help users** to code, test, deploy and validate their applications on multiple platforms with minimal overhead
  - ✓ GPU
  - ✓ MPI
  - ✓ Almost anything else

- HPC centers can use this technology to make **more versatile and usable platforms**, reaching beyond *classic* HPC applications

But containers are not a replacement for all HPC workflows, they're not the solution to everything!



Some of our users love to use containers

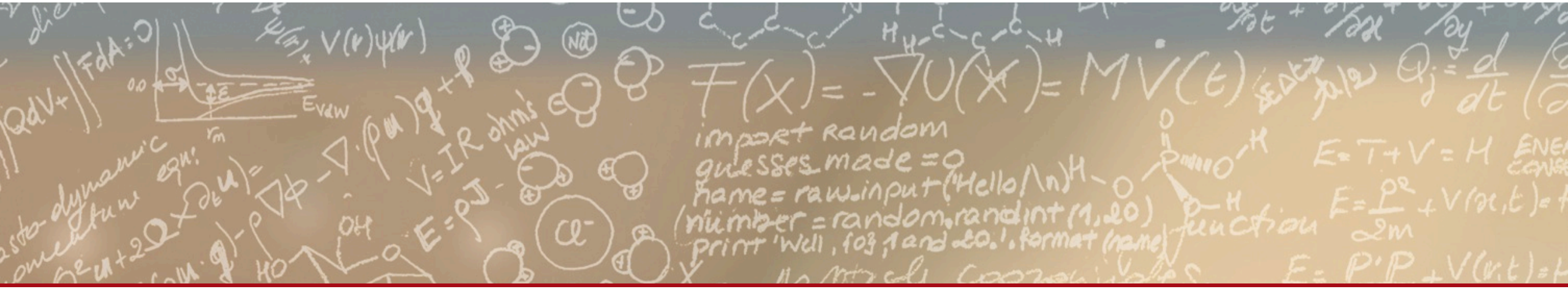
Mon Aug 21 11:55:43 2017 - **Ticket created**

Subject: Thanks for shifter
Date: Mon, 21 Aug 2017 11:55:34 +0200
To: help@cscs.ch
From: [redacted]

Dear CSCS team,

For once I am not writing you about a problem, but to give an honest compliment. Being able to run docker images on daint has really made my life easier. There are still some quirks and room for improvement, but that's minor and overall it is a great improvement.
Here's the (minor) list of suggestion:
- I read about shifter when totally randomly visiting your website. It could have started making my life easier a month ago. I really think such an improvement warrens an announcement.
- It would be great to be able to build the docker images on daint. It is not a big deal, but I am running just a very thin client on my end and the diskspace requirements of building an image forced me to AWS.
- It would be great to have a way to get the images to daint without going through a public repo.
- shifterimg has the '--user' flag for private images but this does not seem to work properly. I did not investigate this deeply, but images that I pull with that option are still visible in the 'shifterimg images' overview from a different account. Are they visible but not runnable?

Thanks,
[redacted]



Thank you for your attention.