

## Cx on HPC-Systems at KIT

**28.04.2022**

René Caspart, Steinbuch Centre for Computing, Karlsruhe Institute of Technology



# Outline

- HPC Systems at KIT
- Cx Services
- Examples using Cx on HPC at KIT

# HPC at KIT

- KIT is a center in the national high performance computing alliance

- Alliance of national academic Tier 2 HPC provider
- NHR@KIT among other aspects focuses on expert level support and software sustainability
- One of the foci is on enabling sustainable software development

- KIT operates two HPC systems

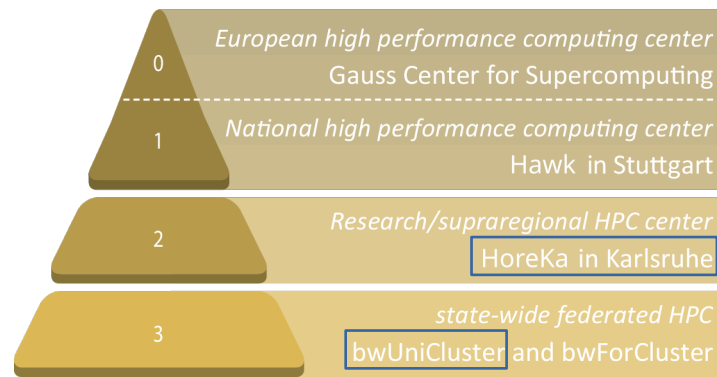
- General purpose Tier-3 system for the state of Baden Württemberg
- Tier-2 NHR-system
  - HAICORE
  - Future Technologies Partition

## ■ Tier 3 HPC System

- Part of the HPC and DIC strategy of the state of Baden-Württemberg
- HPC system for general purpose and teaching
- Open to researchers from all universities in the state of Baden-Württemberg
  - Low entry requirements

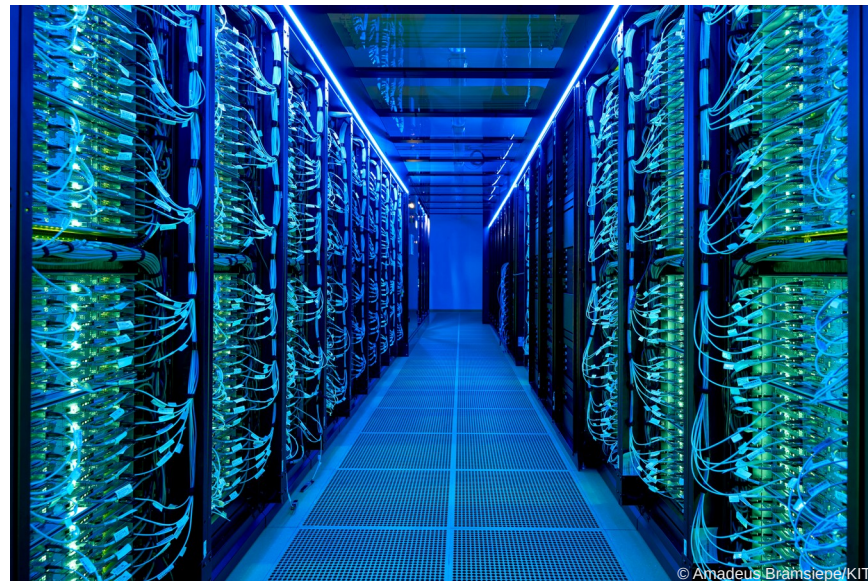
## ■ Standard HPC hardware

- ~32k Intel CPU cores
- ~150 Nvidia V100 GPUs



# Hochleistungsrechner Karlsruhe - HoreKa

- Tier 2 System in the National High Performance Computing alliance
- Modern hybrid system
  - ~60k CPU cores
  - 668 Nvidia A100
  - 200 Gbit/s interconnect
  - 16 PB storage
- Peak #13 in the Green500 and #52 in the TOP500 (June 2021)
- In operation for the scientific community since 1st June 2021



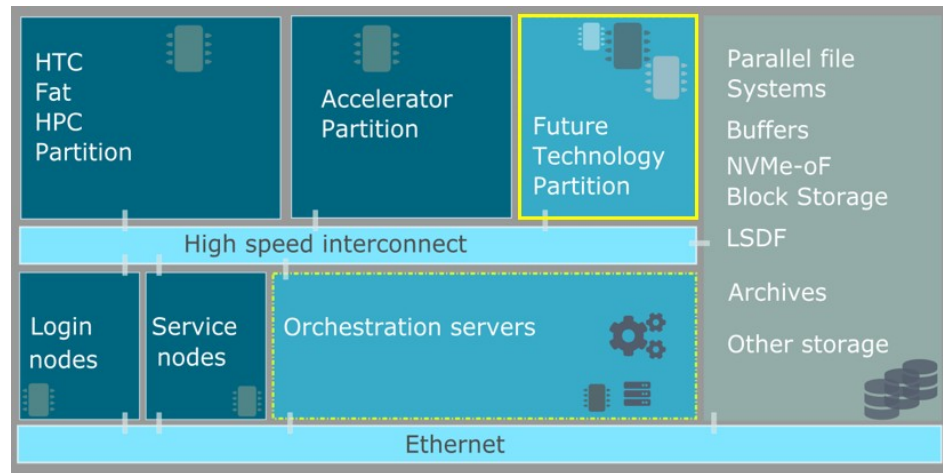
# Providing Additional Value: Future Technologies Partition

## ■ Future Technologies Partition

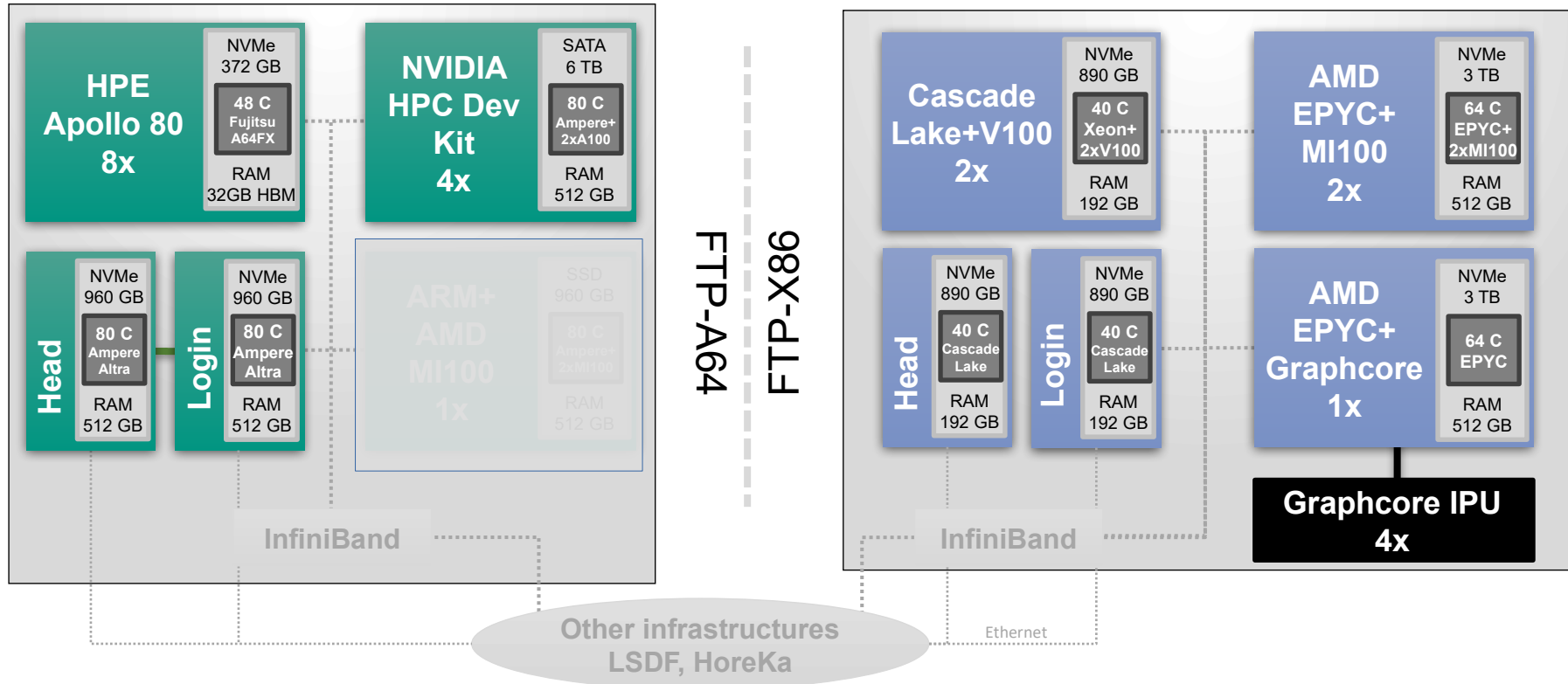
- Providing a testbed for non-standard HPC hardware

## ■ One of the main goals

- Enabling research software engineers to prepare for the future
- Test/benchmark software on future technologies



# The Future Technologies Partition in 2022



# Cx on HPC – Why?

- Cx from public providers
  - Limited resources
  - Only “Standard” hardware
  - Can not test integration with e.g. Slurm, Infiniband, ...
  - No guaranteed performance
    - Important for Performance Evaluation (Continuous Benchmarking)

# Cx on HPC at KIT

- Offer for all HPC users at KIT
    - bwUniCluster 2.0
    - HoreKa
  - Offer for all architectures/systems at KIT
    - Main HPC Clusters
    - Future Technologies Partition
- Wide range of possible architectures, technologies and userbase

# Cx on HPC at KIT

- Usable for any gitlab instance
  - Usable for different usage scenarios
    - Short Cx jobs with fast response (e.g. build tests)
    - Longer Cx jobs with medium response (e.g. larger tests, benchmarking)
    - Predefined Cx schedule (e.g. nightly or weekly builds, benchmarks)
    - Largely varying resource demands
- Wide range of possible usage scenarios

## ■ Identified Challenge:

- Users expect service with „look-and-feel“ as they are used to while getting what they need
  - Easy integration
- Wide range of different parameters (usage scenarios, hardware, ...)

→ Not solvable with a single monolithic service offer

# Cx on HPC at KIT

## ■ Ease of integration in projects and known UI

### ■ Base on gitlab-runner

- Easy to include in any gitlab project
- Known interfaces and configuration in projects

### ■ Container support

- Support via enroot (and singularity)
- Providing custom executors

## ■ Reaction speed and resource demand

### ■ Established different service levels

- Tier 1: Cluster batchsystem → high resource demand and medium/low responsiveness
- Tier 2: Shared Cx node → low resource demand and high responsiveness

#### Specific runners

These runners are specific to this project.

Set up a specific Runner for a project







1. Install GitLab Runner and ensure it's running.
2. Register the runner with this URL:  
<https://scs-gitlab.scc.kit.edu/>

And this registration token:  
72tvKftwsp21xvnB5izv

Reset registration token

Show Runner installation instructions

#### Available specific runners

- #46 (RadTByzL)   Remove runner  
HoreKa Test CI Runner AMD GPU  
gpu-m100 x86
- #45 (wMISDBDz)   Remove runner  
HoreKa Test CI Runner Nvidia GPU  
gpu-a100 x86
- #44 (9w1ZqAij)   Remove runner  
HoreKa Test CI Runner  
cpu x86

# Cx on HPC at KIT – known limitations

## ■ Accounting

- Using the Cx services at KIT requires access to the HPC clusters
  - No self-service, requires project proposals
- Compute time accounted towards project pledges
  - Needs to be considered when applying for compute time

## ■ Accesses

- Cx runners are bound to a specific account
  - Either personal or service account
  - Always directly connected to a defined responsible person

# Cx on HPC at KIT – known limitations

## ■ Integration

- No „drag and drop“ usage
- No shared runners
  - Every user(group) needs to setup their gitlab-runners

Documentation see <https://www.nhr.kit.edu/userdocs/ci/>

Constantly working on improving documentation and training for users

# A brief hands-on Overview

## ■ Cx level-2

- To setup runner (all configuration on gitlab)

```
$ gitlab-runner register
```

start service with

```
$ systemctl --user enable --now gitlab-runner
```

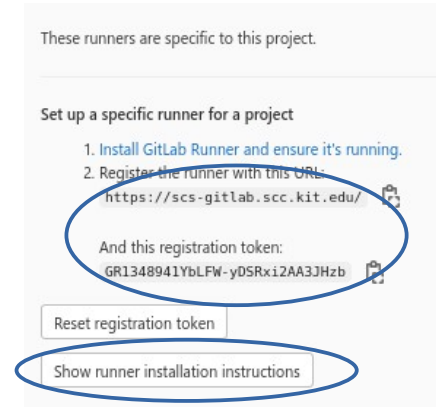
## ■ Cx level-1

- Register gitlab-runner as for level-2

- For example nightly build:

setup a crontab entry for periodically starting your runner as batch job

```
#SCRON -t 00:30:00
#SCRON -p dev_accelerated
#SCRON -A hk-project-scs
0 2 * * * gitlab-runner run --config /hkfs/home/project/hk-project-scs/mb8955/.gitlab-runner/config_a100.toml
```



# Cx on HPC at KIT – One step further

## ■ Runners for different architectures on the FTP

- Only available as level-1 Cx

- Drawback: gitlab-runner per default stores all information in a single config

- Starting the runner starts all configured runners on the current host

## ■ Workaround:

- Different configuration files for architectures/runners

\$ gitlab-runner register -c ~/.gitlab-runner/config\_{arch}.toml

```
#SCRON -t 00:30:00
#SCRON -p amd-milan-mil00
#SCRON -A hk-project-scs
0 2 * * * gitlab-runner run --config /hkfs/home/project/hk-project-scs/mb8955/.gitlab-runner/config_mil00.toml
```

- Alternative: specify all configurations as command line argument

```
0 2 * * * gitlab-runner run-single -u "https://scs-gitlab.scc.kit.edu" -t "RadTByzLB2CLobGq_xt_" --executor shell --wait-timeout 300
```

# Example 1 – NASTJA

## ■ NASTJA

- The NASTJA framework provides an easy way to enable massively parallel simulations for a wide range of multi-physics applications based on stencil algorithms
- Developed at KIT/SCC
- <https://gitlab.com/nastja/nastja>
- Few developers
- Build and tested on CPU and Nvidia GPUs

# Example 1 – NASTJA

## ■ Starting point

- Dedicated Cx runner setup on exclusive machines
  - One provided by us
  - One private machine
- Relatively small resource and time demand for Cx
- Aim: migration to hybrid setup of level 2 Cx@bwUniCluster 2.0 and local runners

## ■ Transition was „easy and painless“

- Gitlab CI configuration (almost) unchanged
- Adding only specific runners and tags

# Example 2 – Dealing with the responsibility in larger Projects

## ■ Issue for larger projects

- Not everyone should be allowed to trigger Cx jobs on the HPC system
- Per default any job triggered runs under the respective user

## ■ Possible solution

- Require dedicated approval step by trusted people for Cx jobs
- E.g. used by Ginkgo and HeAT

# What about Continuous Benchmarking?

- Currently few customers for CB
  - Ginkgo: possible topic for another seminar
  - Setup from us to verify cluster performance e.g. after maintenance
    - Collection of Benchmarks and Mini-apps
    - Simple setup, no graphical illustration or similar
  - Few common aspects so far
- Interesting aspects for the future?
  - Recently enabled energy measurement for jobs on HoreKa (and HAICORE)
  - Benchmarking property in the light of Green IT