SOFTWARE INSTALLATION WITH EASYBUILD ON A MODULAR SUPERCOMPUTING ARCHITECTURE (MSA)

Anke Kreuzer – Jülich Supercomputing Centre (JSC)

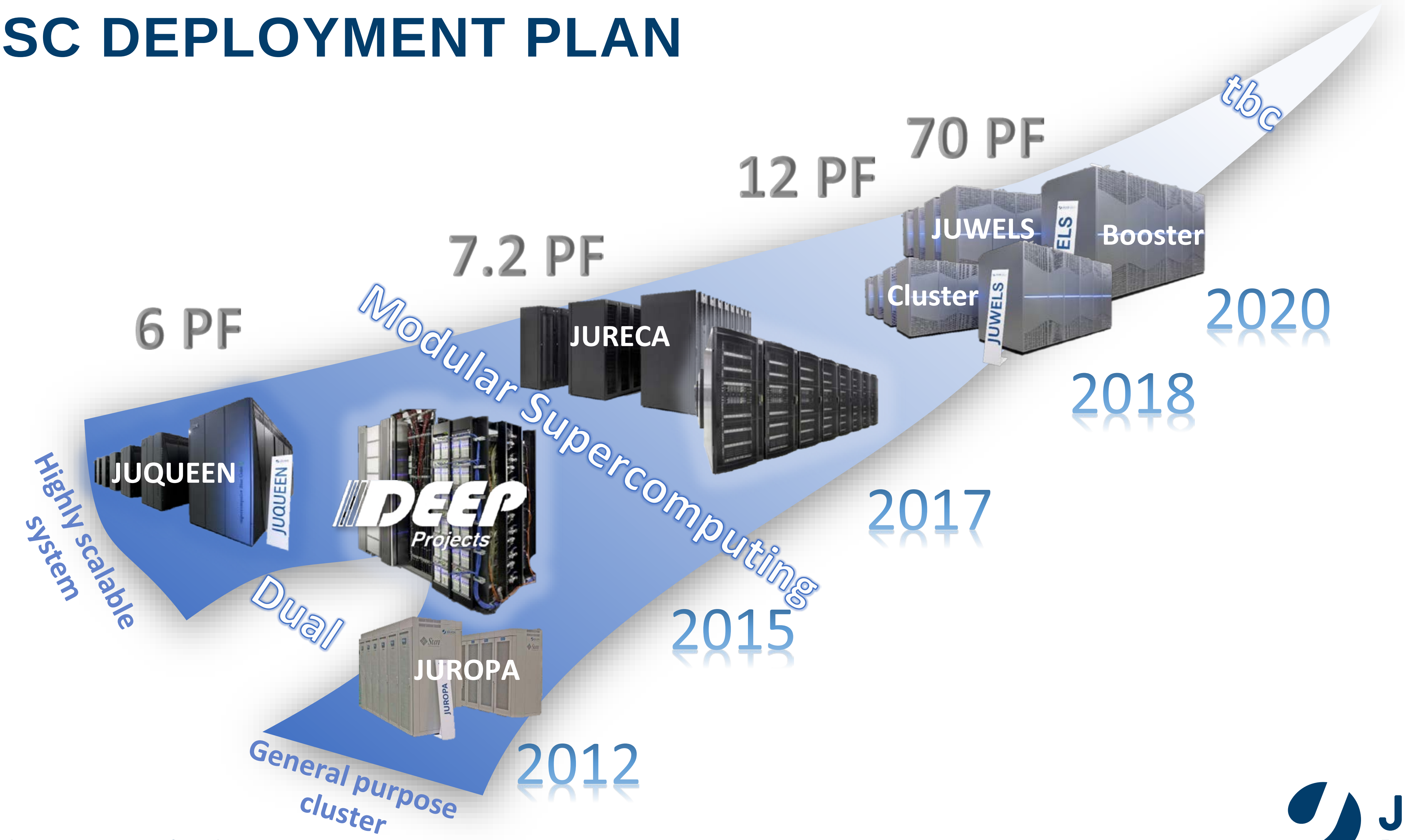
6th EasyBuild User Meeting 2021

OUTLINE

SW INSTALLATION WITH EASYBUILD ON A MSA

- Our systems
- EasyBuild on MSA

JSC DEPLOYMENT PLAN



CURRENT SYSTEMS

System	MSA Module	CPU	GPU
JUWELS	JUWELS Cluster	Intel	NVIDIA
	JUWELS Booster	AMD	NVIDIA
JURECA	JURECA-DC	AMD	NVIDIA
	JURECA Booster	Intel KNL	-
JUSUF		AMD	NVIDIA
HDFML		Intel	NVIDIA
DEEP-EST prototype	DEEP Cluster	Intel	-
	DEEP Booster	Intel	NVIDIA
	DEEP Data Analytics Module	Intel	NVIDIA

OUTLINE

SW INSTALLATION WITH EASYBUILD ON A MSA

- Our systems
- EasyBuild on MSA

EASYBUILD ON MSA

Systems using EasyBuild at JSC:

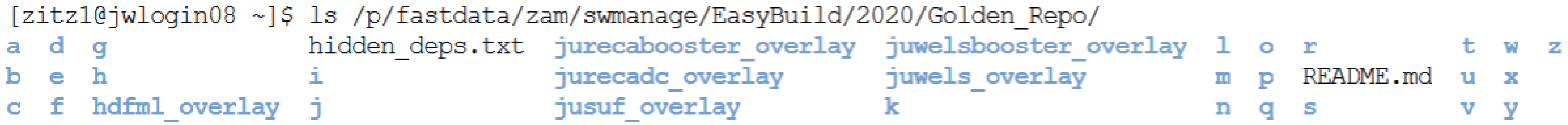
- Two systems with 1 module each (non-MSA)
- Three system with at least 2 modules (MSA)

Impact on SW stack and requirements:

- Different node types mean different requirements:
  - AMD CPU + NVIDIA GPU
  - Intel CPU + NVIDIA GPU
  - Intel Xeon Phi (KNL)
- Node types not only differ from system to system but also from module to module
- xenv tool to load modules from the proper stack in mixed jobs
- Some MSA systems might not have login nodes for all modules but central ones (see JURECA).
  - At JSC we introduced Architecture modules to specify where the software should be installed

EASYBUILD ON MSA – SOFTWARE STACK

- Package base + overlays:

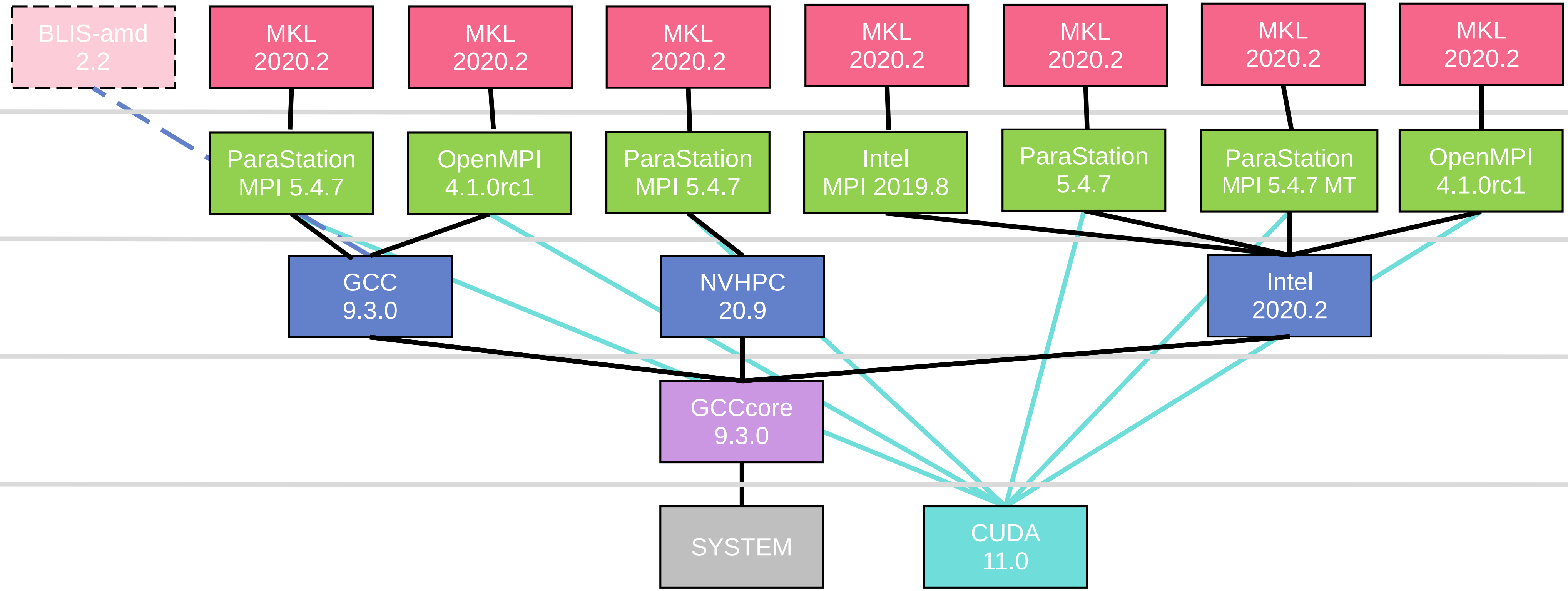


```
[zitz1@jwlogin08 ~]$ ls /p/fastdata/zam/swmanage/EasyBuild/2020/Golden_Repo/
a  d  g              hidden_deps.txt  jurecabooster_overlay  juwelsbooster_overlay  l  o  r          t  w  z
b  e  h              i                jurecadc_overlay       juwels_overlay         m  p  README.md  u  x
c  f  hdfml_overlay  j                jusuf_overlay          k                      n  q  s          v  y
```

→ 689 packages in total (base + overlays)

- Compiler: GCC, iccifort, NVHPC
- MPI: ParaStation MPI, Intel MPI, OpenMPI
- Math: MKL, BLIS (WIP)
- 19 different toolchains

EASYBUILD ON MSA – TOOLCHAIN HIERARCHY



BLIS-amd 2.2

MKL 2020.2 | MKL 2020.2 | MKL 2020.2 | MKL 2020.2 | MKL 2020.2 | MKL 2020.2 | MKL 2020.2

ParaStation MPI 5.4.7 | OpenMPI 4.1.0rc1 | ParaStation MPI 5.4.7 | Intel MPI 2019.8 | ParaStation 5.4.7 | ParaStation MPI 5.4.7 MT | OpenMPI 4.1.0rc1

GCC 9.3.0 | NVHPC 20.9 | Intel 2020.2

GCCcore 9.3.0

SYSTEM | CUDA 11.0

	# packages	Representative packages
Math	131	GROMACS PETSc …
MPI	200	HDF5 Score-P …
Compiler	54	HDF METIS …
Core	232	OpenGL Python …
OS	72	UCX Vtune …

USER VIEW

- Initial user view:
  - Compiler (Intel, GCC, NVHPC)
  - Binary Tools (CUDA, JUBE, Vtune, …)
  - Packages built with GCCcore (CMake, OpenGL, Python, …)
- After loading a compiler:
  - MPI runtimes (ParaStationMPI, OpenMPI, (IntelMPI))
  - Packages built with the chosen compiler (HDF, METIS)
- After loading a MPI runtime:
  - Packages built with the chosen compiler and MPI runtime (HDF5, SIONlib, Score-P,…)
- If a compiler or MPI is loaded on top of the loaded ones Lmod will swap branches and activate/deactivate modules accordingly

USER VIEW

- Hidden modules:
  - Not all packages available for a given combination are visible!
  - Over 150 hidden packages in total (e.g. Java, nvidia-driver, …)
- Bundling extensions
  - Some packages need extensions:
    - Python (), R(), Perl(), etc
    - Each extension as a module would be totally extensive

    →Bundles

STAGE CONCEPT

- Software deployment area for a given timeframe
- Simply a directory
- Default stage upgraded every year
- There is a development stage to test software
  - Test phase for user-based SW installation
- Tested software is added to our repository (and deployed to production)
- Close to seamless transitions between stages during maintenance
- Development and old stages are available but not visible by default

USING HOOKS

- Features of the new hooks:
  - *parse_hook*: manage installations intended for JSC systems
    - Injection of Lmod families (compiler, MPI, toolchain, …)
    - Adding appropriate site contact
  - *pre_ready_hook:* do some checks for bad behaviour
    - Prevent using unsupported toolchains (compilers, MPIs) by default
    - Prevent installing certain things like GCCcore (should only be done by the experts), non-JSC MPIs
  - *end_hook:*
    - If the user is part of the development group and the installation is systemwide, rebuild the system cache

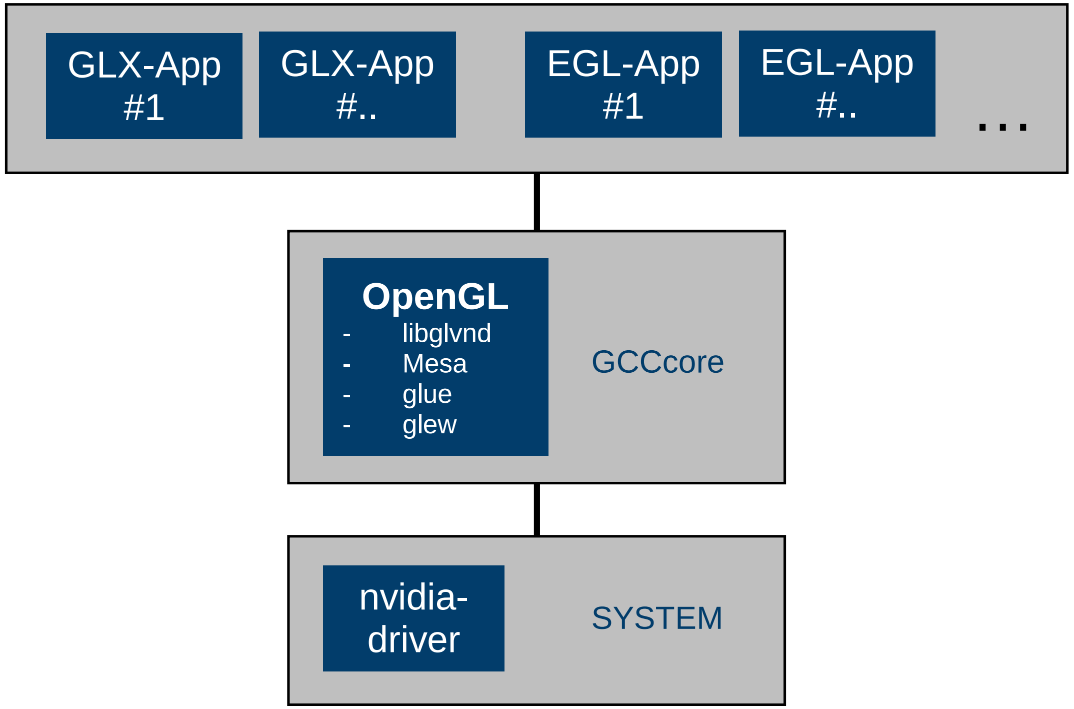
SOFTWARE TEAM

- SW core team:
  - Small team of three people
  - Responsible for core installation (GCCcore, compiler, MPI)
  - Supervises quality standards on easyconfigs before adding them to the Golden repository
    - Check for correct dependencies
    - Proper programming in the easyconfigs (no hardcoded paths, use of EB variables, etc)
- SW team:
  - Several people
  - Each field of applications/packages (math, visualization, I/O, etc.) has one responsible person
  - Allowed to install software in the development stage
  - Can test different compilation options, dependencies, functionality, etc.
  - Anybody in the team can modify any other installation in the development stage

NEW OPENGL MODULE

- Module OpenGL
  - Single dependency for any module in need of OpenGL
  - Includes **`Mesa, glue, glew`**
  - Depends on **`nvidia-driver`**

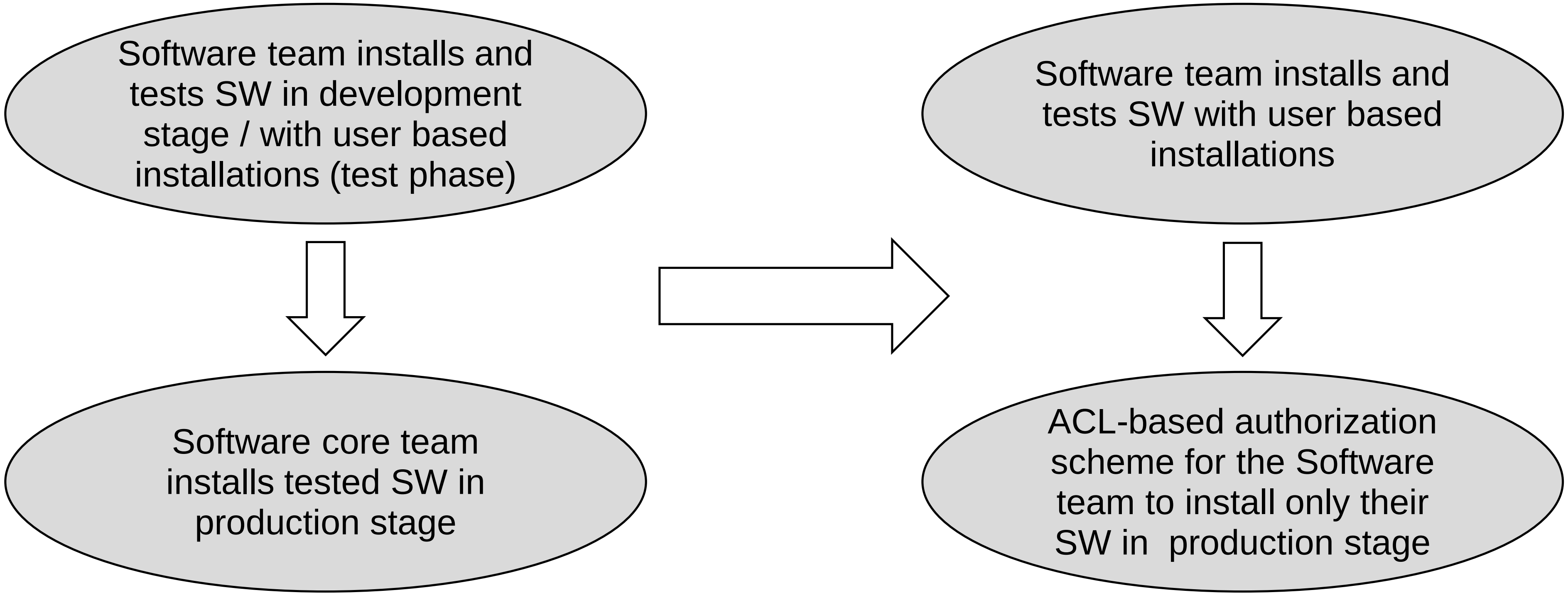
- How GLVND chooses the right driver at runtime
  - GLX: defined by settings of used XScreen
  - EGL: defined by JSON config file with path listed in __EGL_VENDOR_LIBRARY_FILENAMES



Advantage for cluster/multi-cluster installations:
Single modules can serve GPU/non-GPU nodes (with a single dependency to a general OpenGL module) as applications adopt to the best OpenGL driver at runtime.

https://github.com/easybuilders/JSC/blob/2020/Golden_Repo/o/OpenGL/OpenGL-2020-GCCcore-9.3.0.eb

SOFTWARE TEAM ROADMAP



Software team installs and tests SW in development stage / with user based installations (test phase)

Software core team installs tested SW in production stage

How it is now

Software team installs and tests SW with user based installations

ACL-based authorization scheme for the Software team to install only their SW in production stage

How it will be