TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Alexander Grund
Center for Information Services and High Performance Computing (ZIH)

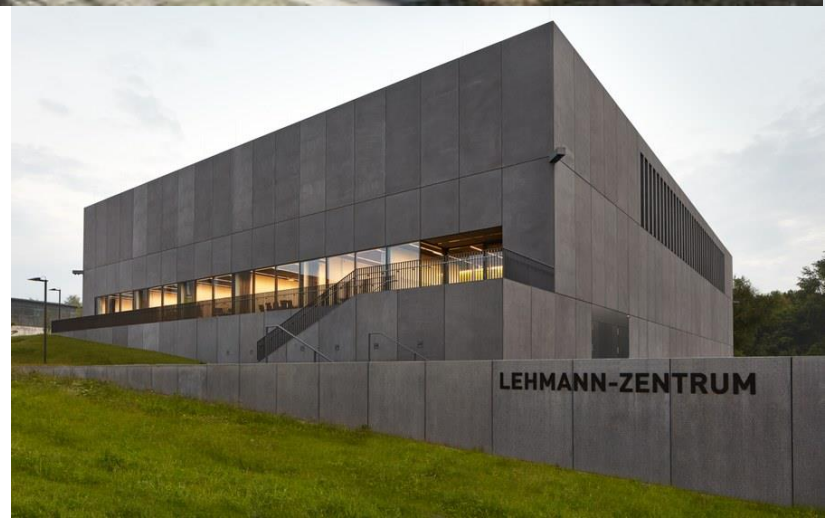
EasyBuild at ZIH – TU Dresden

9th EasyBuild User Meeting, Umeå
24 April 2024

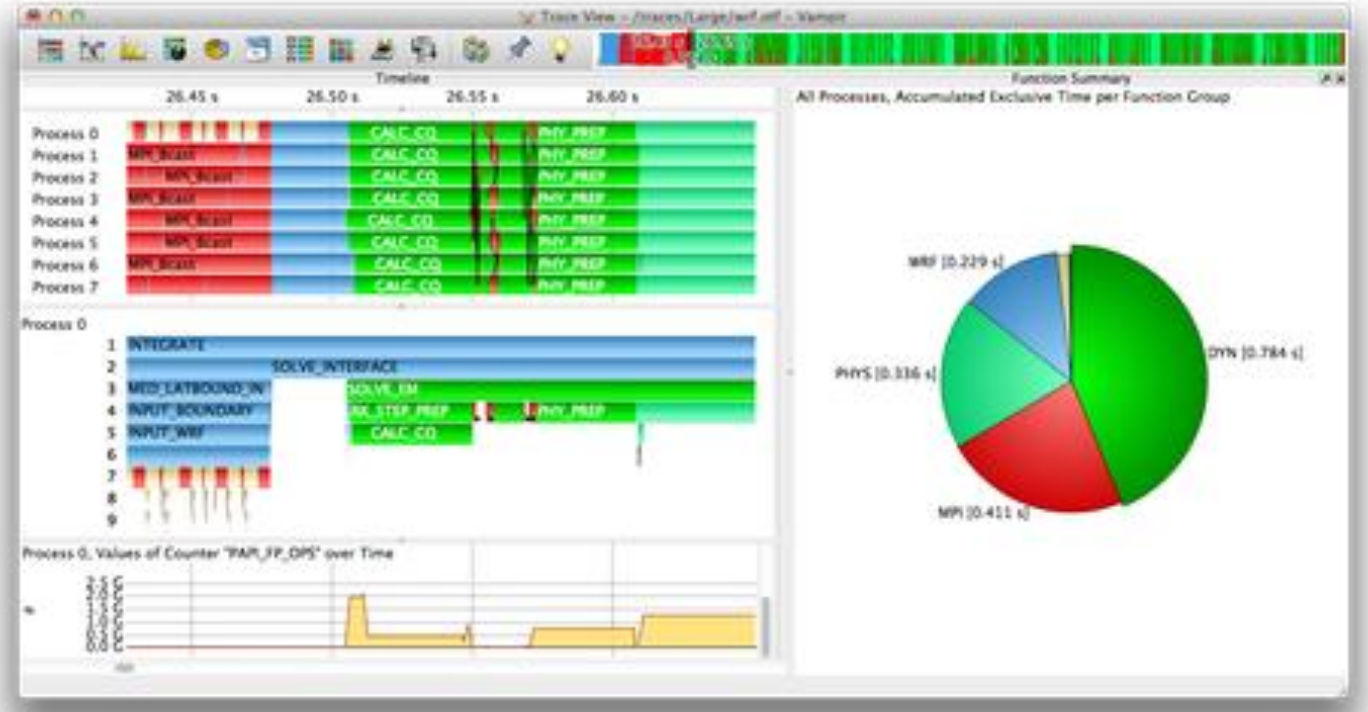
TU (University of Technology) Dresden

— Since 1961
— 30 000 students, 9000 employees
— 17 faculties, 119 disciplines

— Since 2012: “University of Excellence”
— Since 2019: 3 new “clusters of excellence”
  - PoL: Physics of life
  - ct.qmat: complexity and topology in quantum materials
  - CeTI: Center for tactile internet



Center for Information Services and High Performance Computing (ZIH)



— Part of Gauß-Allianz & NHR

— Tasks:
- Operation of central IT infrastructure
- Communication infrastructure
- Support for other departments
- Research & development

— 2022: Restructuring
→ Department of Center for Interdisciplinary Digital Sciences

CIDS ZIH

HPC Cluster

Taurus
— ~ 1600 Haswell nodes, 64 – 256 GB RAM
— 44 GPU nodes (2x K20)
— 64 GPU nodes (4x K80)
— 32 Power9 nodes (6x V100)

Romeo
— 192 nodes 2x AMD EPYC 7702 (128 cores/node)
— 512 GB RAM

Julia
— 32 Intel Cascade Lake 8276 → 896 cores
— 48 TB RAM in unified address space
— 400 TB NVMe memory

Alpha Centauri
— 34 nodes 2x AMD EPYC 7352 (8x A100 GPUs)
— 1 TB RAM

Barnard
— 630 nodes 2x Intel Sapphire Rapids (104 cores/node)
— 512 GB RAM

Software environment

Previously

— LMod & EasyBuild (mixed with hand-built modules)

— Flat module structure

— Hierarchical module structure on 1 architecture

Since 2023

— RHEL 8.7

— Hierarchical module structure

— “release” meta module (23.04, 23.10, 24.04, ...)

User experience

— SLURM
  – 1 partition / architecture
  – Common login nodes for job submission
— Lustre
  – Workspaces
    ▪ Deleted after given timespan
    ▪ Optional reminders (e.g. E-Mail)

— Online documentation
  – Instructions for SLURM, Workspaces, Modules …
  – Hardware overview & recommendations
  – Usage examples for specific software

  – Markdown based
  – Git & CI/CD
    ▪ Pre-commit hooks
    ▪ MR checks (valid style, links, spelling)
    ▪ Automatic 2-step deployment

Module environment

Situation:
- Multiple CPU architectures (Haswell, Cascade Lake, AMD EPYC, ...)
- Multiple GPU architectures (Kepler, Volta, Ampere)

(Old) Approach:

— Shared module folder

— Separate folders for software

— Symlinks of generic folder to architecture specific folder: /sw/installed -> /sw/haswell

- Optimized builds
- Transparent for users
  - E.g. setup paths on login nodes and submit to any partition
- Trouble with some build systems

Upgrading the user experience

Login nodes are generic x86 nodes
- ‼ Loaded modules might not exist on requested partition
- ‼ Building software for specific architecture not easy → Time limit for processes anyway

??? Interactive partitions (1-3 nodes, 8h time limit)
- ‼ Often either blocked or unused
- ‼ Extra work for users, even when just searching for modules
- ‼ Obscure scripts to determine module availability

High load on SLURM controllers

→ Split into clusters
- – 1 partition
- – ~2 login nodes
- – Own SLURM controllers
- √ Own modules (**$LMOD_SYSTEM_NAME**)
- √ Same architecture → Test / Build / Installation possible
- √ Usable for small tasks → No interactive partitions

Module installation

Previously:
- Mostly done by 1 admin
- Set of scripts on the cluster „wrapping“ EasyBuild

New:
- **GitLab:** Scripts (Bash, Python), Configuration, Hooks, EasyConfigs
- Automatic Deployment to cluster
- Selected group with access to repository, including domain experts

EasyConfigs
- Split into folders by target architecture
- Automatic installation in MRs → Spawn SLURM job
- EasyBuild flags from labels
- Pre- & Post-build sanity checks (e.g. Python/Git/LMod versions, $EASYBUILD_IGNORECONFIGFILES)
- Automatic comment in issue/MR, logs as artifacts
- Core script available on cluster → Debug installation issues

Module installation

```
.build_software:
stage: build
script:
   - sg swtest "bash ci-scripts/build-from-ci.sh"
rules:
   - if: $CI_MERGE_REQUEST_TITLE =~ /^Draft:/
     when: never
changes:
   [ecs/$CLUSTER_NAME/$SOFTWARE_TARGET_RELEASE/*.eb]
tags: [$CLUSTER_NAME]
```

```
.build_alpha:
variables:
    CLUSTER_NAME: alpha
   PARAMETERS: $SCHEDULER_PARAMETERS_ALPHA
resource_group: install_rome
extends: .build_software

.build_barnard:
variables:
   CLUSTER_NAME: barnard
   PARAMETERS: $SCHEDULER_PARAMETERS_BARNARD
resource_group: install_rapids
extends: .build_software
```

User modules

Not everything can be installed everywhere by admins

- EasyBuild available as module
- Documented workflow for users to install own modules
- Reuse installed modules → --envvars-user-modules

Conclusion

— EasyBuild & LMod

— Mostly official EasyConfigs
  – Hooks for customization / site specifics

— Hierarchical module structure easier to explain to users

— Separate clusters with own login nodes equal to compute nodes

— Automated installation & deployment via GitLab
  – Involvement of domain experts

— User modules harder with hierarchical module structure
  – “envvars-user-modules” helps but not enough