Managing software on a heterogenous HPC cluster

Alois Schlögl, Stefano Elefante, Andrei Hornoiu, and Stephan Stadlbauer

Institute of Science and Technology (IST) Austria

Introduction: Software is an essential part of any High-Performance Computing (HPC) system, and is often optimized for the available hardware platform. If the cluster is not homogeneous but consists of different hardware, it becomes more challenging to manage the software. This is even more true when the utilization of the cluster should be maximized, and a single scheduler should be used for all hardware.

Challenges: The HPC cluster at IST consists of almost 200 nodes that were purchased over several years, and special hardware and storage systems [1]. There are larger groups of similar nodes (with 16 to 34 nodes each), 21 nodes with 4, 8, or 10 Nvidia GPU's. In total, our nodes have 25 different x86-CPUs (Intel and AMD), and the largest group of 34 nodes does not support the AVX2 instruction set.

About 150 users from 40 research groups from areas like machine learning, biology, physics, chemistry, mathematics etc. have been using the cluster just in the last 12 month. Almost 400 distinct software packages, in over 1000 versions available through the module system Lmod. Python packages from PyPi or Conda, R packages from CRAN, and Bioconductor are not even included in this count. Many of users want to use the latest hardware with the latest software (e.g. machine learning with GPU/cuda), or new hardware (HDR Infiniband, RTX3090), recent version of the operating system is necessary, and the upgrade of the operating system [2] is considered part of the regular maintenance.

Strategies: A variety of tools and strategies were applied, including the use of the module system *Lmod*, using backports repositories [1], use of environment variables that are populated and boot time. In some cases, *Slurm* was configured such that feature/constraint settings are configured in *Slurm*, such that the user gets control on which nodes she wants to run the code. An important question is always where do we need to impose additional workload to the users, and where can we hide the complexity from the users. Specifically, we'll discuss our approach w.r.t. to these items:

- Provide software packages that come in different versions for CPU and GPU nodes.
- How to manage the transition period when upgrading the operating system [1] or the scheduler.
- How to manage different types of Infiniband driver (MLX4 and MLX5), especially w.r.t. of using MPI on top of UCX.
- How to manage different versions of Nvidia/Cuda on the cluster.

Conclusion: The current strategy for software deployment enables us to support different operating systems, different hardware architecture, and will allow us also to integrate new GNU/Linux-based system in our cluster.

References

[1] Kiss, J., Schlögl, A., and Elefante, S., HPC Infrastructure for Cryo-EM at IST Austria Austrian High-Performance Computing meeting AHPC2020, p28 (2020). https://doi.org/10.15479/AT:ISTA:7474

[2] Schlögl, A., Kiss, J., and Elefante, S., Is Debian suitable for running an HPC Cluster? Austrian High-Performance Computing meeting AHPC19, p25 (2019).