Erstellen von Software für das HLRS Cluster

Ziel und Zweck des Media Solution Center ist es HPC-Computing für Unternehmen und nicht-kommerziellen Mitwirkende in der Medienbranche einfacher zugänglich zu machen. HPC hat seinen Ursprung in wissenschaftlichen Rechnen und unterscheidet sich deshalb vom öffentlichen Cloud-Computing, für das eine Vielzahl an Dokumentationen vorhanden ist. Aufgrund dieser Unterschiede ist es für einen typischen Entwickler aus dem Bereich der Medien, der über kein HPC Hintergrundwissen verfügt, oft unklar wie bestimmte Aufgaben angegangen werden können. Dieser Artikel ist für diese Entwickler gedacht und soll helfen die Erstellung und Nutzung von Software auf dem Cluster beim HLRS für einfache Prototypen und Tests zu bewältigen.

Building Software for the HLRS Cluster

The aim and purpose of the Media Solution Center is to make HPC computing more accessible to commercial companies and non-commercial contributors in the media industry. HPC has its origin in scientific computing and therefore is different to public cloud computing for which a lot of documentation exists. Due to these differences, it is often not clear how to approach certain tasks for a typical developer coming from the media sector who has no background in HPC. This article is aimed at those developers and is meant to help approaching the simple task of building and running software on the cluster at HLRS for doing simple prototypes and tests.

1. Software Stack Management

Software often comes with a large number of dependencies, especially in the media sector. For example OpenImageIO (OIIO) has around 13 dependencies including heavy libraries such as Boost, Qt and OpenEXR. Each library comes in many different versions and those must not be mixed when building software further down the dependency chain.

For example consider a tool which depends directly on OIIO and on also depends on another library which also depends on OIIO. Often it can be the case that the version of OIIO the tool depends on is different than the version of OIIO of the depending library. While it still may be possible to build the software with both versions of OIIO in the dependency chain, it will fail and crash during runtime at the latest. Versions of libraries must be kept consistent across the whole dependency hierarchy.

And even if the versions are consistent, the compiler adds more complexity to this. All libraries within

Ansprechpartner: Annekatrin Baumann
baumann@hlrs.de

the dependency hierarchy of a tool must be built with the same compiler/linker tool chain. In general, programs and libraries compiled with older and newer versions of a compiler cannot be linked together. On top of that, all dependencies must link against the same standard C library which interfaces with the system and is given by the linux kernel. So building your tool on some Ubuntu and then running it on a SUSE will not work, as those most likely use different versions of the C runtime library which provides the system calls.

So we see already that building software consistently across the dependency chain can be quite tricky. This becomes even more of a challenge when third party tools from commercial vendors or other production facilities enter the mix. Due to different release cycles, it may well be the case that the local software stack of a studio uses a different version of Boost than for example a tool such as Houdini from SideFX. Software dependencies have to be version consistent across dynamically loaded plugins as well, otherwise they will not work.

Especially in the VFX industry, this situation has caused a lot of frustration in the past. This is why now there is an incentive to standardize the software stack across facilities and vendors with the VFX reference platform (www.vfxplatform.com). It specifies a reference for versions of the most important and common dependencies, such as the C standard library, compiler and some key technologies including Python, Qt and more. It is updated every year and supported by the most important vendors and facilities in the industry. Maybe the HLRS will support the VFX reference platform in the future out of the box.

2. Software stack management at HLRS

In order to discuss software stack management at HLRS, it is important to understand one of the key differences to established public cloud computing services: Homogeneity. With public cloud services like AWS, users can setup the machine type for each node individually. Per node they can choose from a given set of machine configurations. In addition they can boot any arbitrary image on each node running arbitrary operating systems including Windows. Therefore each client can influence hardware and has full control and administrative rights over the operating system on his nodes.

At the HLRS on the other hand, all nodes have the same hardware configuration and the whole system runs on a single operating system which is used and shared by all users. Therefore each client has no influence on the hardware he can use and has very restricted access to and no control over the operating system.

Therefore the typical approach for building software is to build it and all its dependencies on the login nodes of the cluster. Different compilers are supported through virtualization of the user environment. This is done with a tool called Environment Modules which is installed and used on the HLRS operating system.

For more information about the Environment Modules please have a look at:
the HLRS plattform wickie or the documentation of Cray

3. Building Your Software Off-site

However, the on-site building approach is not ideal for the case where source code is not available or cannot be put on the HLRS file system for security reasons. For example in case of the Biene Maya project, a commercial renderer was used for which the source code was not available. In that case the renderer had to be built off-site at the vendor's location and the binaries were moved to the cluster after the fact.

Another reason for building the software stack off-site is convenience. For a client it can be much handier to use local resources, infrastructure and fully administrative rights for prototyping and setting up the software which is supposed to be run on the cluster.

In addition it may well be the case that computing will be distributed across different cluster. This is called Hybrid computing and we believe that in the future, computing will become more and more transparent to the location of compute-clusters the same way power supply used at home is transparent to where it was generated. It should be easy and straightforward to compute a subset of a problem on the HLRS cluster and another subset on a public cloud such as AWS or Google Cloud Platform. However, maintaining that will be difficult if every cluster requires its own software stack.

4. Making Off-site Building More Convenient

When building software off-site for the HLRS cluster, it would be ideal to locally have a system which is as similar to the system which is being found on the login nodes of the cluster. This is a great case for virtualization. To some extent, virtualization allows to reproduce the environment found at the cluster locally but with all the administrative rights of a root user.

While one can use virtualization software like VM VirtualBox directly, it is even easier and simpler to use Vagrant, which is a high-level frontend to those tools. Then, setting up your virtual machine and running the operating system is a matter of a single shell command. You only need to find and download an image of the operating system or make one yourself. At HLRS the operating system is a 64 bit Suse Linux Enterprise Server, Version 11 Patch 3 (SLES 11.3 x64).

What is particular interesting about Vagrant is the fact that it also is able to directly spin up AWS EC2

instances from a given vagrant box image. This makes it easy to have a single software stack and deploy it to both, public cloud services and the HPC cluster at HLRS if the HLRS operating system is used.

3. Conclusion

In this article we briefly discussed software stacks at the HLRS and pointed out that building your software and all its dependencies on-site is the way to go. There are cases where it is necessary to build the software off-site and afterwards move the created binaries to the cluster. For that case we highlighted Vagrant, a high-level frontend which makes it easy to setup the same system which is used at HLRS on a local virtual machine. However, building off-site should only be the last resort.

Stuttgart, 2016, David Körner.

Platform wickie of HLRS

Official Website of Vagrant

VFX reference platform

Ansprechpartner: Annekatrin Baumann
baumann@hlrs.de