Introduction

The ALMA Common Software

All the ALMA Project subsystems are developed with the ALMA Common Software (ACS)[1]. ACS is a distributed object framework using a Component-Container model. It is based on the CORBA standard.

The Standard Test Environment

The Standard Test Environment (STE) is a group of computers which is used to test and run the ACS software. This is a controlled environment where all the machines have the same hardware. All of them run the same software version and similar deployment configurations. This provides a unique platform across all project sites to test and integrate the ALMA software.

Simulation with ACS

There are two ways of doing simulation with the ALMA subsystems[3][4]. The first one uses Real-Time OS (RTOS Linux) and userland tools to simulate the CAN communication to a FIFO queue. The second simulates in non real-time the device's behavior. The only difference for the component is the loaded library at runtime. The ALMA Control Software generation code framework provides by default these features. It generates a basic non-real-time skeleton which is extended by the developer. We want to be able to run on a virtualized STE a full non-real-time simulation and some mixes of virtualized STE servers and real-time computers connected to real hardware.

Virtualizing the ALMA Software

History of virtualization at ALMA

ALMA Computing has used VMs since 2003. The first usage was to provide a standalone ACS environment to develop and test components. During 2008 the Joint IA Observatory (JAO) Computing Group (CG) had no available STE to test and develop. All of them were in production. This led us to research among the available open source Linux hypervisors. Our first test during that time was trying to execute a complete non-real-time simulation with an ACS binary. We successfully tested VMs with the Kernel-based Virtual Machine (KVM), a fork from the well known CPU emulator QEMU. KVM uses most of the QEMU functionality but it also makes usage of native virtualization using the new CPUs extensions available on Intel and AMD CPUs. We could do a full run of holography and we were able to export the data from the observation (AS DM).

Virtualization and we were able to export the data from the observation (AS DM).

Virtualization is a valuable new tool for testing and developing Software of astronomical projects. It multiplies valuable and scarce server resource at a low cost and can be a key element of any computing team. A strategy to define the VM life cycle is always useful to have complete sets of pre-installed machines to speed-up the deployment process. When you want to create a new machine, you just copy a template and then you add the missing pieces of it in less than 30 minutes. We also provide a webservice and tool to deploy existing software builds on your VM from the closest host.

Virtual Machine Policies

We have implemented a life cycle for the VM usage. A user requests a VM specifying if it is a full STE or a developer machine and defining its purpose, lifetime and ACS Software version needed. The sysadmins have to create this VM and look for its deadline. Having this clear policy helped us to prevent the uncontrolled proliferation of VMs called VM sprawl.

Benefits

Among the benefits we have seen of usage of VMs, we have seen increased number of available testing environments for the developers, increased productivity of the Software Engineers by augmenting the number of available development machines on demand. New training environments for Scientist and Operators using non-real-time simulation were created. A new debugging environment for scripts testing is also available before using real hardware. This leads directly to an overall better usage of precious real hardware testing time. Software backward compatibilities issues were also solved thanks to VMs. Under some circumstances it is useful to have an ultra portable version of a STE, e.g. one rack with 3 servers can't fit on the antenna cabin. One laptop with 3 virtualized machines can.

Conclusions

Virtualization is a valuable new tool for testing and developing Software of astronomical projects. It multiplies valuable and scarce server resource at a low cost and can be a key element of any computing team. A strategy to define the VM life cycle must be implemented and followed in order to make a meaningful use of this new tool.

Currently (October 2009), virtualization is in use by a 15 people team. It has been used on a number of tests. Among them:

- First antenna movement: a standalone STE in a laptop was used to monitor an antenna and its electronic components (Front End) while it was moved by the ALMA transporter.
- Correlator tests: a virtual STE was connected to help the test of a new correlator functionality. This was done without touching the real STE.
- ADBM and hardware tests: we connect a virtual STE to a real-time machine and we can test the ADBM and the hardware connected to it.
- We created a set of tools in python based on the setuptools package. This allowed us to easily install it on any Linux distribution. A module named kvmfly was created to provide the required functionality.
- A set of tools and web services were created to manage ACS software versions. Since we are a geographically distributed team, it is a requirement to get from the closest place the version which you want to use.
- Since the STEs have a fixed OS and package software, we created templates of the VMs. These are pre-installed hosts with no ACS software on it. When you boot them, they are ready to receive an ACS version and to run it. Currently we have on set of template per available OS (RedHat 4.4 and RedHat 5.3).
- The drm server must point to your local drm to resolve external ips.
- We de-activated remote reporting tools for VMs.
- Some startup parameters like ACS timeouts must be increased since operation on guests are slower than in real host. This is the case for the idl repository load.
- It is always useful to have complete sets of pre-installed machines to speed-up the deployment process. When you want to create a new machine, you just copy a template and then you add the missing pieces of it in less than 30 minutes. We also provide a webservice and tool to deploy existing software builds on your VM from the closest host.
- Some startup parameters like ACS timeouts must be increased since operation on guests are slower than in real host. This is the case for the idl repository load.
- We have implemented a life cycle for the VM usage. A user requests a VM specifying if it is a full STE or a developer machine and defining its purpose, lifetime and ACS Software version needed. The sysadmins have to create this VM and look for its deadline. Having this clear policy helped us to prevent the uncontrolled proliferation of VMs called VM sprawl.

References


Acknowledgements

This work was done by the Joint ALMA Observatory Computing team and ESO Computing.