Experiences virtualizing the ALMA Common Software

MAURICIO ZAMBRANO[1], DIEGO ARREDONDO[1], MARCELO BARTSCH[1], LIVIO CONDORELLI[2], JORGE IBSEN[2], TZU-CHIANG SHEN[1],STEFANO TUROLLA[2]
[1]Associated Universities, Inc. (AUI), Santiago, Chile
[2]European Southern Observatory (ESO)

ABSTRACT

One of the most exciting subjects in system administration during the last years has been the virtualization of complete systems with all the benefits it implies. Hardware and operative systems are now offering new features that improve the performance of such virtual environments up to a point that nearly matches the real hardware performance. ALMA software is tested among others in simulation on a controlled Standard Tests Environment (STE). STEs are a set of computers and network hardware.
We took an Open source hypervisor and made a set of tools to build, manage and deploy complete sets of virtual STEs, which are used for testing the ALMA Common Software. For many years, ALMA has provided pre-installed virtual machines (VMs) to developers.
We took this approach one step further, allowing to run ALMA software in simulation on a fully virtualized STE on a single server. The work we present here shows the goals, scopes and policies for virtualization and the benefits gained from their usage since its adoption in late 2008.

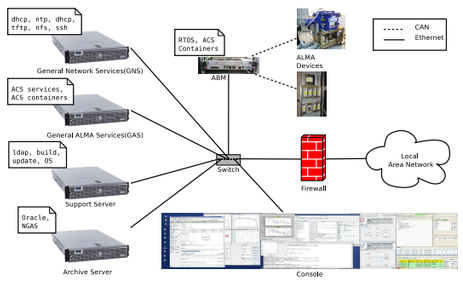
Introduction

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 were 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. As of today, a STE is composed by 4 main servers , where each machine provides its own specific functionality as shown on the picture. We also have console machines used to operate the system and many industrial class computers, the Antenna Bus Master (ABM) which are used to run the code used to control ALMA hardware devices. These are real-time computers with a CAN bus which is used to send the commands[2].



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 Alma Observatory (JAO) Computing Group (CG) hadn't any 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 project QEMU. KVM use 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 (ASDM).
Some decisions we made in our implementation:

- VM's disk would be files on top of ext3 file system. This is not the best performance option but it allows us to have more space on the VMs than physic space on the host thanks to the usage of the QCOW2 (QEMU Copy On Write) file format. QCOW2 only uses space when a VM is really using it. A better I/O performance can be obtained when using raw disk or LVM partitions. QCOW2 also offers some features like memory status saving.
- The server hosting the VMs (the host), has to have a bridge instead of traditional Ethernet in order to allow a VM to be on the network or behind a firewall.
- We also emulated a full firewall system using the well known m0n0wall.
- 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 kvmPy 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).

Host and Guest configuration

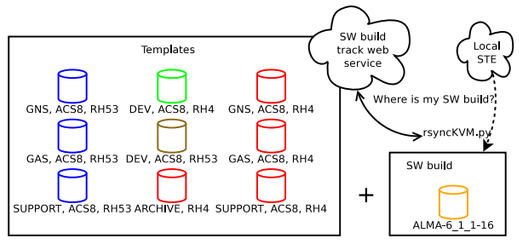
The hosts are the machines that will run the virtual machines. There are usually servers with multiple cores, 8 in our case, a big amount of RAM to share with the hosts, 16GB for us, big local disks and one or more Ethernet cards. The CPUs should have the virtualization extensions to enable the full capabilities of the hypervisor. In order to provide more than 4GB of RAM to the guests, a 64 bit OS must be installed on the hosts. The network configuration should be handled by bridges.

The guest, the virtual machine, has to be configured with the same amount of disks than the real server. The installation from the media is flexible enough to adapt itself to a smaller disk size on the fly. Tools are provided to generate easily VM's disks, to install the base OS and to run and to start groups of Guests. After installing a VM a small number of changes have to be performed:

- The dns server must point to your local dns 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.

Host templates and software deployment

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.



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.
- ABM and hardware tests: we connect a virtual STE to a real-time machine and we can test the ABM and the hardware connected to it.
- Builds test and build farm: we can compile ACS releases on a single VM although this is 40 to 50 percent slower than in real machines. On the other hand, on a single host we can have up to 5 VMs compiling in parallel. This allow us to test in simulation multiple ACS branches before going to real hardware.
- Development: used to develop multiple components in simulation among them, Ethernet Device, WeatherStation and Control Monitoring.
- Exporting to other VMs disk format: thanks to its flexibility we can export our KVM-made VMs to other formats like vmware desktop.

This work was developed with KVM but we are curently evaluating other virtualization technologies as vmware or Xen.

Acknowledgements

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

References

[1] Raffi, G. et al., "The ALMA Common Software (ACS) as a basis for a distributed software development" in [*Proceedings of Astronomical Data Analysis Software and Systems XI*], (2001).
[2] Farris, A. et al., "The ALMA Telescope Control System" in [*Proceedings of ICALEPCS 2005*], (2005).
[3] Hiriart, R. et al., "Integration and Testing in the ALMA Control Subsystem" in [*These Proceedings*], (2008).
[4] Matias Mora, et al. "Hardware device simulation framework in the ALMA Control subsystem" in [*ADASS*], (2008).

Atacama Large Millimeter/submillimeter Array
http://www.almaobservatory.org/

Contact e-mail: mzambran@alma.cl / For more information about Joint ALMA Office Computing Group please visit our web site http://almasw.hq.eso.org/almasw/bin/view/JAO/WebHome