Experiences designing and building a PRAGMA Cloud Scheduler

[Short Paper]

Shava Smallen
San Diego Supercomputer Center
University of California San Diego
ssmallen@sdsc.edu

Nadya Williams
San Diego Supercomputer Center
University of California San Diego
nadya@sdsc.edu

Philip Papadopoulos
San Diego Supercomputer Center
University of California San Diego
phil@sdsc.edu

ABSTRACT

The Pacific Rim Application and Grid Middleware Assembly (PRAGMA) is a community of research scientists and institutions from around the Pacific Rim that work together to enable scientific expeditions in areas of biodiversity distribution and lake ecology. Over the past four years, the technology focus for PRAGMA partners has shifted to cloud and software defined networking as enabling technologies. During PRAGMA 27 meeting, the Resources Working Group discussed rebooting the persistent PRAGMA Testbed as a way for sites to contribute and use shared resources, leveraging technologies such as PRAGMA Boot, Personal Cloud Controller (PCC), and virtual network overlays (i.e. IPOP and ViNe). To make PRAGMA resource sharing easier, a lightweight scheduler was proposed and discussed as a way to enable access to the resources and to manage resources reservations. This short paper discusses the design process and initial prototype of a simple cloud scheduler for PRAGMA.

Keywords

Cloud, Scheduling, Resource Sharing, Virtualization

1. INTRODUCTION

PRAGMA researches actively collaborate to enable scientific expeditions in areas of computational chemistry, telescience, biodiversity, and lake ecology [**?**]. Founded in 2002, PRAGMA originally sought to advance the use of grid technologies in applications among a community of investigators working with leading institution sites around the Pacific Rim [**?**]. This included the deployment of a shared PRAGMA Grid testbed where participating sites could contribute and use resources as needed to develop and test new middleware and to conduct scientific experiments. Testbed sites needed to install at a minimum Globus [**?**] and a local HPC job scheduler as well as other optional software such as MPICH-G2 [**?**] and Ninf-G [**?**]. In November 2006, nineteen sites in thirteen countries were part of the testbed. By 2009, the testbed had grown to twenty-seven sites in fifteen countries with a total of 1008 CPUs, more than 1.3 terabytes of memory, and over 24.7 terabytes of online storage. However, as of 2011, the number of sites started to decline and maintaining the numerous and complex middleware and scientific applications at each local site required significant people effort and expertise. Since PRAGMA participants often have varying levels of of funding, staff, and expertise, PRAGMA started to shift away from grid towards cloud technologies to simplify the infrastructure and to lower the amount of effort any site would need to participate in the testbed.

The first phase of the PRAGMA cloud testbed started with three sites and focused around the creation of application-specific virtual machines and making them portable to different cloud hosting environments like Rocks [**?**], OpenNebula [**?**], and Eucalyptus [**?**]. One early example was an AutoDock [**?**] virtual cluster created for the Avian Flu Grid research team in 2011 [**?**]. By 2012, the cloud testbed had ten sites with a total of 367 CPUs, 2.5 terabytes of memory, and 657 terabytes of online storage. In 2013 PRAGMA shifted its focus to the creation and management of virtual clusters to make it easier for users to assemble a multi-node virtual environment for running their scientific experiments. The following technologies have been explored to facilitate the operation and use of virtual clusters on the PRAGMA cloud testbed.

Pragma_boot: rather than requiring a single cloud system at all sites, PRAGMA encourages the sites to deploy any virtualization technology that best fits their expertise. However, enabling a virtual cluster to be ported to different cloud hosting environment is a complex problem and requires tooling to retain the cluster relationship between head and worker nodes as well as software configurations. In 2013, three pilot sites (UCSD, AIST, NCHC) developed a script for an automated virtual cluster porting and demonstrated the same virtual cluster image being booted in three different cloud hosting environments, including Rocks/Xen, OpenNebula/KVM, and Amazon EC2. Based on those experiences, the porting script was redesigned and reimplemented as *pragma_boot* toolkit with "drivers" to support virtual cluster migration to virtual environments enabled on

Rocks and OpenNebula clusters.

Personal Cloud Controller: to provide users with an easy-to-use interface for managing the life cycle of their virtual clusters, a Personal Cloud Controller (PCC) was started in 2014. This lightweight tool was designed to manage startup, status monitoring, and shutdown of a virtual cluster and was built on top of pragma_boot and a well-known resource and workload management system HTCondor [**?**]. PCC also provided an option to create a multi-site virtual cluster leveraging an open-source virtual network overlay software IPOP [**?**] for creating its private network.

Software-Defined Networking: PRAGMA began investigating software-defined networking technologies such as OpenFlow [**?**] in 2013 as a way to create a private network between multi-site virtual clusters and to protect access to sensitive datasets [**?**]. PRAGMA then created the Experimental Network Testbed (PRAGMA-ENT) as a breakable, international testbed for use by PRAGMA researchers and collaborators. In 2014, The PRAGMA-ENT team worked to create an international reliable direct point-to-point data Layer-2 connection with network engineers and developed AutoVFlow [**?**] that allows to provision private virtual network slices for each application, user, and/or project [**?**].

In late 2014 during the PRAGMA27 workshop, a lightweight scheduler was proposed to coordinate virtual hosts and clusters running on the different cloud deployments via leverage the above described technologies. The next section discusses the general requirements of the cloud scheduler and Section 3 discusses the design options that were considered and why a simple calendar solution was selected. A summary of calendar systems that were examined is provided in Section 4. Section 5 discusses our early implementation with one of the calendar systems called Booked and Section 6 concludes with our planned future work.

2. SCHEDULER REQUIREMENTS

PRAGMA had the following three key requirements for a cloud testbed scheduler to enable multiple users access to the resources at a given time:

Low participation overhead: One key requirement for the cloud testbed scheduler was that it had to be lightweight and required minimal effort and minimal expertise for a site to add their cloud deployment to the list of resources available for scheduling. Similarly, the scheduler would need to work across multiple cloud deployments (e.g., Rocks, OpenNebula, and OpenStack [**?**]) so that sites do not have to learn and install specific cloud deployment tools in addition to their currently used.

NOTE THE FOLLOWING IS A PART OF AN IMPLEMENTATION NOT A REQUIREMENT, MOVE TO DIFFERENT SECTION THAT DESCRIBES IMPLEMENTATION: Sites just need to run a SSH service and install the pragma_boot package, which is a small Python package and can be installed via RPM and can be configured in less than an hour.

Easy to use: Currently, to execute a virtual cluster on the PRAGMA cloud testbed, users needs to contact each site individually to gain access to their cloud deployment, upload their images, and launch their virtual cluster either using pragma_boot if it is available or manually via a sequence of defined steps. The design of the PRAGMA cloud scheduler should first simplify this process by ensuring that pragma_boot gets deployed more broadly and port it to more cloud environments as needed. The scheduler should also provide a simple Web interface for users to see the available resources and sites, construct their virtual cluster, and manage their images.

Scale to tens of users: While the PRAGMA community was composed of nineteen active institution sites in 2014, the initial number of expected users for the cloud testbed is in the tens of users, not hundreds nor thousands. Therefore, we can prioritize simplicity over scalability and give higher priority to the requirements of low participation overhead and ease of use.

3. SCHEDULER DESIGN

In our design discussions, we first looked at the existing solutions that could be adapted for PRAGMA's scheduling needs:

Existing HPC schedulers: Open source batch schedulers like Slurm [**?**], TORQUE [**?**], and HTCondor are commonly used in HPC to manage distributed shared resources. While these batch schedulers are widely used, they either do not work across multiple sites (TORQUE) or are too heavy weight for what PRAGMA needs, requiring custom configuration and multiple daemons at each site (Slurm and HTCondor). Furthermore, to be adopted by PRAGMA users who are scientists in many different domains, we wanted a low barrier to entry like a simple Web based interface for scheduling, rather than the command-line tools and queue abstractions that these batch schedulers provide.

Grid'5000: NEEDS TO BE WRITTEN. Add short description of Grid'5000. They use OAR which is a batch scheduler similar to above and the calendar gui called GridPrems is one of three options for interacting with Grid'5000. A key difference is that testbed resources are dedicated rather than like on PRAGMA where they could be shared; I believe also single admin model. Interesting set of tools. We can say we liked their calendar approach the best.

DHCP Leases: NEEDS TO BE WRITTEN.

GENI/PlanetLab: NEEDS TO BE WRITTEN. ORCA

Out of these approaches, we liked the Grid'5000 calendar reservation interface, GridPrems, for scheduling the best. The calendar reservation scheduling is an interface that all users are familiar with in their everyday lives and would be well suited to the different PRAGMA users who have varying levels of cloud expertise.

4. CALENDAR SYSTEMS

In addition to GridPrems interface, we decided to look at other calendar reservation systems.

Discuss selection of Booked.

5. CLOUD SCHEDULER PILOT

Discuss creation of Cloud Scheduler Pilot.

6. CONCLUSIONS AND FUTURE WORK

Some interesting conclusion.

7. ACKNOWLEDGMENTS

The authors would like to thank Jose Fortes and Mauricio Tsugawa for their help during our initial design discussions and for providing resources at University of Florida during development.

8. REFERENCES

[1] M. Bowman, S. K. Debray, and L. L. Peterson. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.*, 15(5):795–825, November 1993.

[2] J. Braams. Babel, a multilingual style-option system for use with latex's standard document styles. *TUGboat*, 12(2):291–301, June 1991.

[3] M. Clark. Post congress tristesse. In *TeX90 Conference Proceedings*, pages 84–89. TeX Users Group, March 1991.

[4] M. Herlihy. A methodology for implementing highly concurrent data objects. *ACM Trans. Program. Lang. Syst.*, 15(5):745–770, November 1993.

[5] L. Lamport. *LaTeX User's Guide and Document Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[6] S. Salas and E. Hille. *Calculus: One and Several Variable*. John Wiley and Sons, New York, 1978.