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 individuals and sites from around the Pacific Rim that actively collaborate to enable scientific expeditions in areas like biodiversity 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, 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 overlay networks. 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 resource reservations. This short paper discusses the design process and initial prototype of a simple cloud scheduler for PRAGMA.

Keywords

Cloud, Scheduling, Resource Sharing

1. INTRODUCTION

The Pacific Rim Application and Grid Middleware Assembly (PRAGMA) is a community of individuals and sites from around the Pacific Rim that actively collaborate to enable scientific expeditions in areas like computational chemistry, telescience, biodiversity, and lake ecology [11]. 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 [12]. 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 conduct scientific experiments. Testbed sites needed to install at a minimum Globus [1] and a local HPC job scheduler as well as other optional software such as MPICH-G2 [4] and Ninf-G [18]. By November 2006, nineteen site in thirteen countries were part of the testbed, with a total of 662 CPUs, nearly 1 terabyte of memory, and 7.3 terabytes of online storage [13]. In 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 by 2011, the number of sites started to decline and maintaining the numerous and complex middleware and scientific applications at a 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 lower the amount of effort an 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 [10], OpenNebula [7], and Eucalyptus [8]. One early example was an AutoDoc [2] virtual machine created for the Avian Flu research team in 2011 [14]. 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 addition to the previously managed cloud tools, To make it easier for users to assemble a multi-node virtual environment for running their scientific experiments, PRAGMA shifted its focus to the creation and management of virtual clusters in 2013. The following technologies have been explored to facilitate both the operation and use of virtual clusters on the PRAGMA cloud testbed.

pragma_boot: Rather than requiring a single cloud system, PRAGMA sites are free to deploy any cloud tool that best fits their expertise. However, enabling a virtual cluster to be ported to different cloud systems 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 an automated virtual cluster porting script 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* and "drivers" were implemented for both Rocks and OpenNebula.

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) tool 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 management tool called HTCondor [5]. PCC also provided an option to create a multi-site virtual cluster leveraging a network overlay tool called IPOP [3] for the private network.

Software-defined Networking: PRAGMA began investigating software-defined networking technologies like OpenFlow [6] in 2013 as a way to create a private network between multi-site virtual clusters and to protect access to sensitive datasets [15]. 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 [19] to provision private virtual network slices for each application, user, and/or project [16].

In late 2014 during the PRAGMA27 workshop, a lightweight scheduler was proposed to coordinate running on the different cloud deployments and leverage the above 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 considered 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 time:

Low participation overhead: One key requirement for the cloud testbed scheduler was that it be lightweight and require minimal effort and minimal expertise for a site to add their cloud deployment to the list of schedulable resources. Similarly the scheduler would need to work across multiple cloud deployment tools (e.g., Rocks, OpenNebula, and OpenStack [9]) so that sites do not have to learn and install one specific cloud deployment tool. 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: To execute a virtual cluster on the PRAGMA cloud testbed currently, users needs to contact each site individually to gain access to their cloud deployment, upload their images, and if pragma_boot is not available, perform all the steps manually to launch their virtual cluster. The design of the PRAGMA cloud scheduler should first simplify that process by ensuring pragma_boot gets deployed more broadly and ported it to more cloud systems as needed. The scheduler should also provide a simple Web interface for users to see the available resources at 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 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 easy of use.

3. SCHEDULER DESIGN

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

Existing HPC schedulers: Open source batch schedulers like Slurm [20], TORQUE [17], and HTCondor are commonly used in HPC to manage space 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, PRAGMA users change over time and 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] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11:115–128, 1996.

[2] D. S. Goodsell and A. J. Olson. Automated docking of substrates to proteins by simulated annealing. *Proteins: Structure, Function, and Bioinformatics*, 8(3):195–202, 1990.

[3] P. S. Juste, D. Wolinsky, P. Oscar Boykin, M. J. Covington, and R. J. Figueiredo. Socialvpn: Enabling wide-area collaboration with integrated social and overlay networks. *Comput. Netw.*, 54(12):1926–1938, Aug. 2010.

[4] N. T. Karonis, B. Toonen, and I. Foster. Mpich-g2: A grid-enabled implementation of the message passing interface. *Journal of Parallel and Distributed Computing*, 63(5):551–563, May 2003.

[5] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor - a hunter of idle workstations. In *ICDCS*, pages 104–111, 1988.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[7] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente. Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. *Computer*, 45(12):65–72, 2012.

[8] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pages 124–131, Washington, DC, USA, 2009. IEEE Computer Society.

[9] OpenStack Open Source Cloud Computing Software Website. `http://www.openstack.org`, 2015.

[10] P. M. Papadopoulos, M. J. Katz, and G. Bruno. Npaci rocks clusters: tools for easily deploying and maintaining manageable high-performance linux clusters. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 10–11. Springer, 2001.

[11] The Pacific Rim Application and Grid Middleware Assembly Website. `http://www.pragma-grid.net`, 2015.

[12] PRAGMA Collaboration Overview 2003-2004, November 2004.

[13] Collaboration Overview: Annual Report 2005-2006, November 2006.

[14] Collaboration Overview: Annual Report 2010-2011, November 2011.

[15] Collaboration Overview: Annual Report 2012-2013, November 2013.

[16] Collaboration Overview: Annual Report 2013-2014, November 2014.

[17] G. Staples. Torque resource manager. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 8. ACM, 2006.

[18] Y. Tanaka, H. Nakada, S. Sekiguchi, T. Suzumura, and S. Matsuoka. Ninf-g: A reference implementation of rpc-based programming middleware for grid computing. *Journal of Grid computing*, 1(1):41–51, 2003.

[19] H. Yamanaka, E. Kawai, S. Ishii, and S. Shimojo. Autovflow: Autonomous virtualization for wide-area openflow networks. In *Software Defined Networks (EWSDN), 2014 Third European Workshop on*, pages 67–72. IEEE, 2014.

[20] A. B. Yoo, M. A. Jette, and M. Grondona. Slurm: Simple linux utility for resource management. In *Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer, 2003.