

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 X 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 [10]. 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 [11]. 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 [16]. 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 [12]. 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 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 [9], OpenNebula [7], and Eucalyptus [8]. One early example was an AutoDoc [2] virtual machine created for the Avian Flu research team in 2011 [13]. 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. 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 ex-

periences, 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 [14]. 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 [17] to provision private virtual network slices for each application, user, and/or project [15].

In late 2014 during the PRAGMA27 workshop, a lightweight scheduler was proposed to coordinate access to 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

Sites willing to participate will have to install a PRAGMA Package - a very small set of components. Users can request access to resources similar to requesting a room (i.e., a room reservation system) PCC would be used to automatically start and monitor user's VMs slash VC via *pragma_boot* up at sites Only support 10s of users

Repeat longer version of text in abstract with appropriate citations.

Discuss requirements and technologies like *pragma boot*, etc.

3. SCHEDULER DESIGN

Discuss process of narrowing down scheduler design from meeting ideas (e.g., batch scheduler, Grid 5000, GENI, Google doc, DHCP leases).

4. CALENDARING SYSTEMS

Discuss selection of Booked.

5. CLOUD SCHEDULER PILOT

Discuss creation of Cloud Scheduler Pilot.

6. CONCLUSIONS AND FUTURE WORK

Some interesting conclusion.

7. 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. Grzegorzczak, 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] 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.
- [10] The Pacific Rim Application and Grid Middleware Assembly Website. <http://www.pragma-grid.net>, 2013.
- [11] PRAGMA Collaboration Overview 2003-2004, November 2004.
- [12] Collaboration Overview: Annual Report 2005-2006, November 2006.
- [13] Collaboration Overview: Annual Report 2010-2011, November 2011.
- [14] Collaboration Overview: Annual Report 2012-2013, November 2013.
- [15] Collaboration Overview: Annual Report 2013-2014, November 2014.
- [16] 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.

- [17] 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.