Setup and management of a small national computational facility: *What we've learned the first 10 years.*

George Tsouloupas
HPC Facility
The Cyprus Institute
Nicosia, Cyprus
g.tsouloupas@cyi.ac.cy

Thekla Loizou
HPC Facility
The Cyprus Institute
Nicosia, Cyprus
t.loizou@cyi.ac.cy

Panayiotis Vorkas
HPC Facility
The Cyprus Institute
Nicosia, Cyprus
p.vorkas@cyi.ac.cy

***Abstract*—The Cyprus Institute HPC Facility (HPCF) turns 10 years old this October. The Facility was established with the mandate to provide a computational resource open to all research scientists in the Eastern Mediterranean. This short paper is a brief overview of the main takeaways from starting this facility from scratch and then evolving it from a single HPC system to a diverse multi-system environment over it's 10-year history. It briefly touches on rationale behind technical, policy and other decisions and the main lessons learned by the facility staff.**

I. HISTORY

The High Performance Computing Facility (HPCF) was set up in 2010 with the vision of serving the High Performance Computing (HPC) needs of a (small) country. At the time the Cyprus Institute was just 5 years old and had one (1) HPC sysadmin. Other than the 1m (euro) grant to buy a computer, budgets were extremely tight. The data-center was built while we waited for procurement of the system to conclude. Once the system was installed and after a long and tedious acceptance and verification procedure the production call open in early 2012. At the same time the staff was increased to two sysadmins and one user support engineer.

II. LESSON #1: STAND ON THE SHOULDERS OF GIANTS AND GET YOUR POLICIES STRAIGHT FROM THE GET-GO.

From the inception of the Facility back in 2010 and being awarded the grant to set up the HPC facility in the country by the Cyprus Research Promotion Foundation we relied heavily on the experiences of others on setting up rock-solid policies that embodied decades of experience at the National Center for Supercomputing Applications (NCSA) of the University of Illinois and the Juelich Supercomputing Center (JSC) at Juelich. From the get-go we had advanced policies on a range of issues ranging from peer-review processes for production access to scheduling policies. We inherited years of experience and, in terms of policies we had to change very little over 10 years. Jira[1] (ticketing system), Confluence[2] (wiki), Nagios[3] (functionality monitoring -- now moved to Icinga[4]), Ganglia[5] (performance monitoring -- now moved to Prometheus[6] and Grafana[7]), were put in place after guidance and in many cases technical help from the teams at the NCSA[8] and JSC[9].

III. LESSON #2: AUTOMATION AND A DO-IT-YOURSELF ATTITUDE.

A. *Infrastructure Automation*

Initial system deployment was delivered using XCAT [10] and a set of custom scripts by IBM engineers. After an upgrade or two, the scripts fell apart. We needed a new approach for automation and we picked one through an evaluation process. We tested Puppet, Chef and Ansible [11]. We picked Ansible for two reasons: 1) it was simple to get started, and 2) it was based on python which we were starting to standardize on (we were already using python for larger scripts and on EasyBuild [12]).

Ansible allowed us to treat our infrastructure as code and host all of our configuration. After a bare-bones Operating System installation using XCAT, virtually all system configuration is done through Ansible. We currently have 5 HPC clusters of various sizes and that can be redeployed from scratch in a matter of hours. The Facility staff now operates under a "manual configuration prohibited" directive. While it took several months to get a fully automatic deployment of the first system, the speed of subsequent redeploys, upgrades and node rebuilds have been worth it.

B. *Software Automation*

When we first started operation of Cy-Tera back in 2011, the initial software on our clusters was manually installed and was provided to the users using Environment modules. However, software installation on HPC systems is a very demanding and time-consuming process for system administrators. We were lucky enough at that time to be introduced to EasyBuild, a software build and installation framework that allows you to manage (scientific) software on HPC systems in an efficient way. EasyBuild builds and installs software fully automatically, it is easily configurable, it supports automatic dependency resolution and has a growing community. EasyBuild was initially released on 6th of April 2012 and our facility was one of the first EasyBuild users. Throughout the years we have contributed back to EasyBuild plenty of software packages, with the main focus on Bioinformatics packages. An important milestone for EasyBuild was the support for CUDA installation, code to which our team was one of the main authors [13]. We have also organized two EasyBuild hackathons in Nicosia in

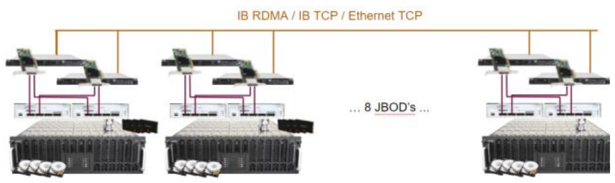

Figure 1: Custom Storage System based on BeeGFS and ZFS.

2013[14], to introduce local and regional system administrators and users to EasyBuild.

C. *Storage*

Operating under a very restrictive budget and a growing need for HJPC storage we operated using IBM GPFS and hardware from DDN. Our cost per Terabyte of usable storage was in the order of €400/TB (usable). The alternative was either a LUSTRE-based storage or a new development out of the Fraunhoffer Institute, BeeGFS (then-called fhGFS). LUSTRE had some incompatibilities with our hardware so we opted for BeeGFS and have not regretted it. We have operated several BeeGFS instances, the largest of which is currently 4PB and will be upgraded to 6PB in Dec 2020 (see Figure 1). Care needs to be taken to allow for balancing the storage targets and good knowledge of BeeGFS and ZFS is a must. Otherwise the system is very stable and performant. Considering that we procure all components off-the-shelf (servers, JBOD's and disks) we were able to drive costs to below €100 per usable TB (saving hundreds of thousands in the process). On the Openstack side we again source all components individually and maintain a growing CEPH[18] deployment.

IV. Lesson #3: Diversify, advanced computing is not just HPC

It is easy to fall into the trap of "elitism" when it comes to allocation time on an hpc system, especially when peer-review is involved. We avoided this pitfall in two main ways: 1) allow for generous and streamlined preparatory access (a "no application is too small" mentality where virtually all applicatns we granted at least some amount of time, even if it was very little), and 2) make different types of systems available that can serve as diverse a set of applications as possible.

A. *ON-PREMISE CLOUD*

Early on we saw a need for users, especially internal users to run software not meant for HPC, it came in the form of data ingestion web applications, data repositories and data presentation and analysis (mostly data generated on our HPC). Usually these tools came in the form of web-based tools. (see Figure 1).

We started using Opennebula[15] while we started to grasp the basics but we quickly decided to move to OpenStack[16] before our user-base grew. The move was mostly due to FOMO (Fear Of Missing Out) rather than real needs. Openstack, a quite complicated but promising platform, took a big toll on a really small team. Our initial OpenStack deployments were done using Mirantis Fuel. The installation

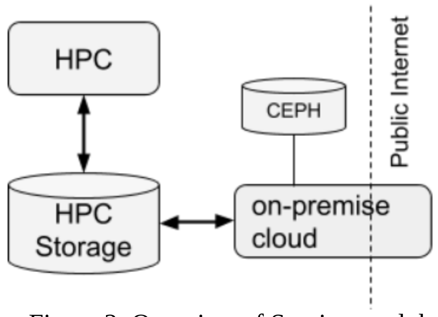

Figure 2: Overview of Services and the

was straight-forward but allowed for little customization. This proved to be a showstopper since we were trying to use repurposed nodes from older HPC clusters. After looking at several projects we finally went with Openstack-Ansible [17]. We have already been familiar with using Ansible as our configuration management tool, so being able to deploy and configure OpenStack using Ansible playbooks and roles was the ideal solution for us. Openstack-Ansible (OSA) is an active and well maintained project, with releases that follow the OpenStack releases.

We always try to maintain a test environment for OpenStack so as to perform any changes or upgrades to our test environment before applying them to our production environment.

Acknowledgment

The authors and sysadmins of the HPCF would like to thank all of the staff at the NCSA of the University of Illinois, especially Brett Bode, for their continued support.

References

[1] Jira - https://www.atlassian.com/software/jira
[2] Confluence - https://www.atlassian.com/software/confluence
[3] Nagios - https://www.nagios.org
[4] Icinga- https://icinga.com
[5] Ganglia - http://ganglia.sourceforge.net
[6] Prometheus - https://prometheus.io
[7] Grafana - https://grafana.com
[8] NSCA - http://www.ncsa.illinois.edu
[9] JSC - https://www.fz-juelich.de
[10] Xcat - https://xcat.org
[11] Ansible - https://ansible.com
[12] Easybuild - https://easybuild.readthedocs.io/en/latest
[13] Easybuild Cuda code - Cudahttps://github.com/easybuilders/easybuild-easyblocks/blob/develop/easybuild/easyblocks/c/cuda.py
[14] 3rd & 4th Easybuild Hackathon - https://github.com/boegel/easybuild-wiki/blob/master/3rd-easybuild-hackathon.md and https://github.com/boegel/easybuild-wiki/blob/master/4th-EasyBuild-hackathon---meeting-minutes-day-1.md
[15] Opennebula- https://opennebula.io
[16] Openstack - https://www.openstack.org
[17] Openstack-Ansible - https://docs.openstack.org/openstack-ansible/latest
[18] Ceph - https://ceph.io
[19] Ubuntu - https://ubuntu.com