The integration of Workload Management Systems for the ProtoDUNE Software and Computing cluster

CERN Summer Student Report

Summer Student: Oriana-Maria ONICIUC
EP-NU, CERN,
Faculty of Computer Science,
Alexandru Ioan Cuza University of Iaşi (UAIC)

Main supervisor: Dr. Nektarios BENEKOS
EP-NU, CERN

August 25, 2017

Abstract

The protoDUNE experimental program is designed to test and validate the technologies for DUNE. All of the many elements in the chain of data acquisition, storage, distribution and processing are critically important to derive physics results from the data. To achieve these, a software stack has been chosen to implement automatic propagation of configurations across all the nodes in the NP cluster. This report presents the architecture of the system and the operations through which the cluster features can be scaled.

I. Introduction

The universe is permeated with neutrinos. First discovered in 1956, neutrinos have some mysterious characteristics. They are nearly massless, neutral particles that interact so rarely with other matter. These tiny particles could be key to a deeper understanding of our universe.

I.1. Neutrino Platform and ProtoDUNE

Working in the neutrino sector is a worldwide priority promising physics beyond the Standard Model. The Deep Underground Neutrino Experiment, DUNE, is a proposed international neutrino experiment. The protoDUNE program is designed to test and validate the technologies and design that will be applied to the construction of the DUNE. The CERN Neutrino Platform's R&D facility is located in a test-beam hall (EHN1 hall) on CERN's Prévessin site. There are two apparatuses to be used in beam test: single-phase (NP04) and dual-phase (NP02) LArTPC detectors.

I.2. Workload Management Systems

Workload management is the process of distributing the different workloads that a system handles. The workload manager of the system fulfills these expectations by controlling access of the workload to system resources through prioritization or other algorithms specified by the system administrator. It also ensures that workloads are distributed properly among available execution units or servers.

The data rate and volume will be substantial and we need an architecture that can be scalable and handle a big computing power. The main goal is the implementation of a pilot-based mechanism for computational payload deployment. We worked in configuring and designing the architecture for the cluster of computers that are located in the EHN1 hall, for their future use in the experiment.

II. Neutrino Platform Computing Infrastructure

II.1. Hardware Resources

The Neutrino Platform (NP) cluster consists of about 300 nodes. These nodes are DELL Poweredge 1950 machines each with 16 cores. A part of these machines have a 2 TB hard disk that makes them suitable also for storing data for different jobs. Featuring Intel Xeon processors, the servers offer the power and performance that we will need in Data Quality Monitoring (DQM) and Prompt Processing.

The software build process is CPU intensive, and the build tools in use allow parallelization to many cores. A build machine with at least 16 cores is desirable. The number of cores that can be profitably used by a build machine depends on the I/O capabilities to the disks storing applications. Since the build node should not be used to run programs, even to test built code, then the code must reside on shared disks.

II.2. Designing system topology (Physical Architecture)

The NP cluster consisted, at the beginning, just from bare metal machines and the switches which were connected to the CERN GPN. There is an evident need for an automated configuration management system for our cluster. To manage the machines hosted in EHN1, some virtual machine (VM) are hosted on CERN OpenStack Cloud infrastructure. The whole system has to be integrated in the CERN Puppet Orchestration.

Conceptual diagram of the raw data flow in protoDUNE is presented in the next figure. It shows the general logic of data flow, and does not include specific assumptions about the system. This is motivated by the experience and architecture of the LHC experiments and it is proposed in one of the initial Technical Design Reports.

II.2.1. Puppet Architecture

According to the proposed topology of the system we created the following architecture. The main challenge we encountered was that the system will use physical and virtual machines, but

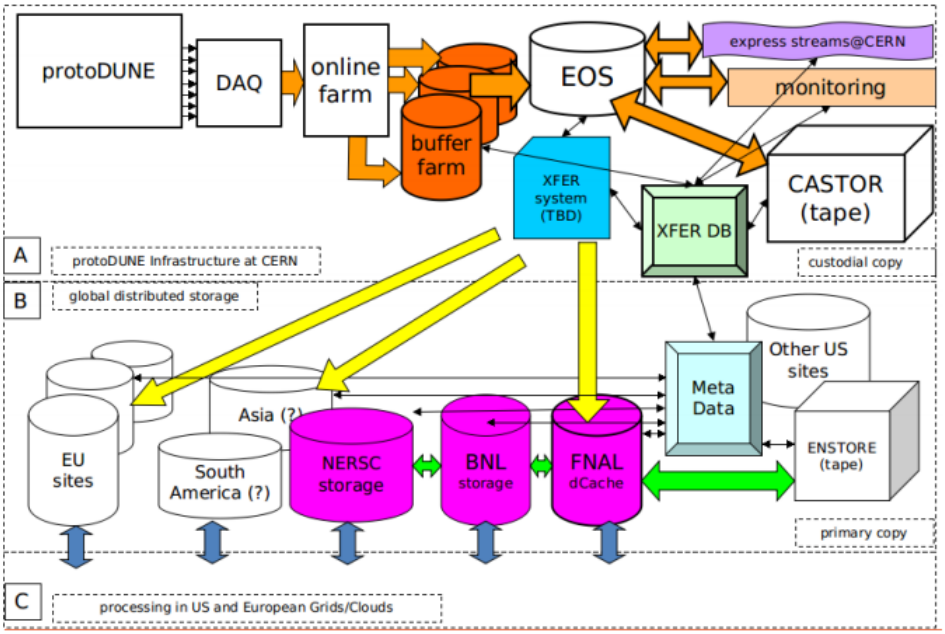


Figure 1: Conceptual diagram of the flow of raw data in protoDUNE

a good management of the host groups and environments can solve this problem. For the main host group we created a new environment, lbd_branch, and there we tested all the features for the cluster.

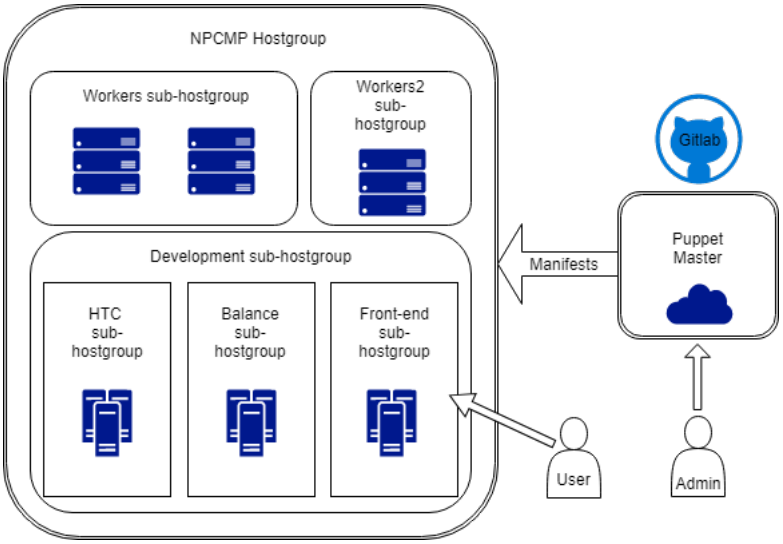


Figure 2: System architecture

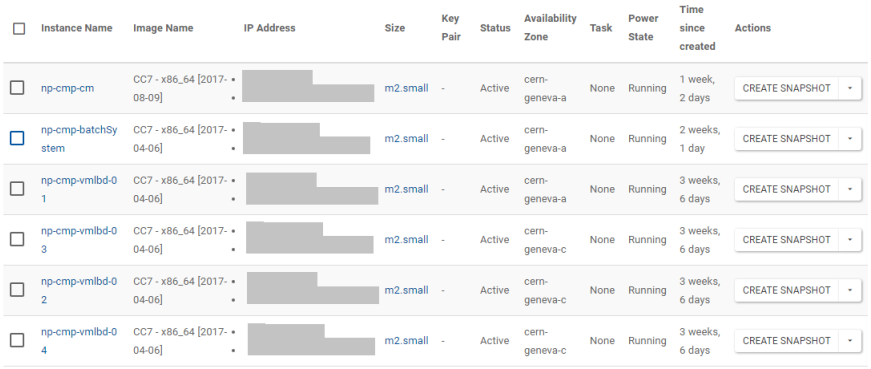
All nodes in the **npcmp host group** are all part of the same service, they have common configuration and are managed by the same group of people (np-cmp-admin e-group). The whole npcmp host group is managed by the **Puppet Master**.

Each host group and sub-host group is managed through the Puppet **manifest**. A puppet run occurs each hour in all the nodes. The manifests are stored using the Git version control system.

The **workers sub-host group** contains all the physical machines that are part of the cluster and that will run all compute-intensive jobs. In a future HTCondor infrastructure, these machines will be the workers of the system. We use the **workers2 sub-host group** for testing new modules for our system.

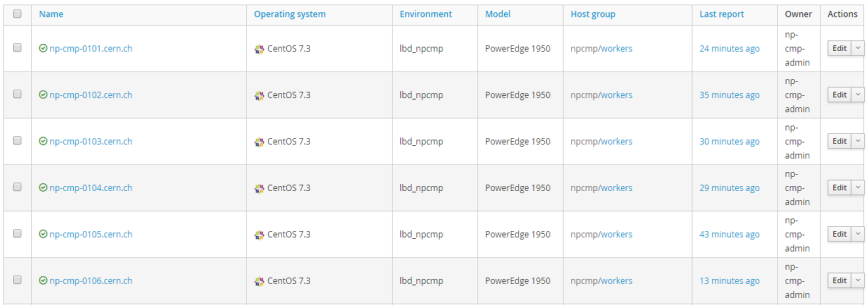
The **development sub-host group** is split in more sub-host groups. The **front-end sub-host group** is the interface between the users of the system. It can be extended to an interactive interface in which the user will constantly monitor his jobs. The **balance sub-host group** is used for the load balancing under the alias *npcmp.cern.ch*. The last **sub-host group, HTC** will be used as a HTCondor Master - negotiator for the jobs submitted in the cluster.

The following interfaces show us how the machines are working and which is their status. These two tools are available on the admin side, for the np-cmp-admins e-group.



Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
np-cmp-cm	CC7 - x86_64 [2017-08-09]		m2.small	-	Active	cern-geneva-a	None	Running	1 week, 2 days	CREATE SNAPSHOT
np-cmp-batchSystem	CC7 - x86_64 [2017-04-06]		m2.small	-	Active	cern-geneva-a	None	Running	2 weeks, 1 day	CREATE SNAPSHOT
np-cmp-vmlbd-01	CC7 - x86_64 [2017-04-06]		m2.small	-	Active	cern-geneva-a	None	Running	3 weeks, 6 days	CREATE SNAPSHOT
np-cmp-vmlbd-03	CC7 - x86_64 [2017-04-06]		m2.small	-	Active	cern-geneva-c	None	Running	3 weeks, 6 days	CREATE SNAPSHOT
np-cmp-vmlbd-02	CC7 - x86_64 [2017-04-06]		m2.small	-	Active	cern-geneva-c	None	Running	3 weeks, 6 days	CREATE SNAPSHOT
np-cmp-vmlbd-04	CC7 - x86_64 [2017-04-06]		m2.small	-	Active	cern-geneva-c	None	Running	3 weeks, 6 days	CREATE SNAPSHOT

Figure 3: Virtual machines on OpenStack project EP npcmp



Name	Operating system	Environment	Model	Host group	Last report	Owner	Actions
np-cmp-0101.cern.ch	CentOS 7.3	lbd_npcmp	PowerEdge 1950	npcmp/workers	24 minutes ago	np-cmp-admin	Edit
np-cmp-0102.cern.ch	CentOS 7.3	lbd_npcmp	PowerEdge 1950	npcmp/workers	35 minutes ago	np-cmp-admin	Edit
np-cmp-0103.cern.ch	CentOS 7.3	lbd_npcmp	PowerEdge 1950	npcmp/workers	30 minutes ago	np-cmp-admin	Edit
np-cmp-0104.cern.ch	CentOS 7.3	lbd_npcmp	PowerEdge 1950	npcmp/workers	29 minutes ago	np-cmp-admin	Edit
np-cmp-0105.cern.ch	CentOS 7.3	lbd_npcmp	PowerEdge 1950	npcmp/workers	43 minutes ago	np-cmp-admin	Edit
np-cmp-0106.cern.ch	CentOS 7.3	lbd_npcmp	PowerEdge 1950	npcmp/workers	13 minutes ago	np-cmp-admin	Edit

Figure 4: Physical machines in host group

II.2.2. Other Software and Computing Services

The code management (Puppet manifests, Hiera variables etc.) is done through Gitlab (Git repository: https://gitlab.cern.ch/ai/it-puppet-hostgroup-npcmp). For visualization and reporting the CERN IT department uses Foreman. Monitoring is done through Kibana and ELK suite. There is also a set of reusable Puppet modules (Nagios, Ganglia etc.) maintained

by several groups in IT which can be used for extending the functionality that the NP cluster provides.

Name	Hosts	Hosts including sub-groups
npcmp	0	260
npcmp/development	0	5
npcmp/development/balance	4	4
npcmp/development/frontend	0	0
npcmp/development/htc	1	1
npcmp/workers	251	251
npcmp/workers2	4	4

Figure 5: Foreman hostgroups architecture

III. System Administration

There are plenty of factors to be considered when assessing the potential for system scalability. In this chapter we will analyze different ways in which we can improve our system and make it functional for new changes in the structure.

III.1. Adding new nodes

This task can be completed in two manners: for physical nodes, or for virtual machines. The next images will show us how can we proceed in order to accomplish the adding of new machines to our cluster.

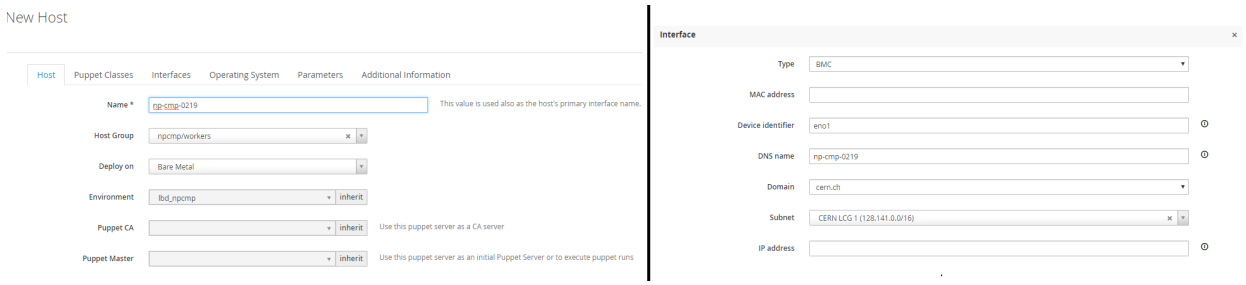


Figure 6: Virtual machines on OpenStack project EP npcmp

To add a physical machine in Foreman, we set the desired name, host group, interfaces (IP, MAC, FQDN), operating system. Foreman resolves a kickstart template for the physical host triggering an installation by first, preparing the host for installation and rebooting the box to begin installation process. After setting the desired parameters we can use a CLI to perform a similar task by running ai-installhost command on aiadm. Physical machines must be

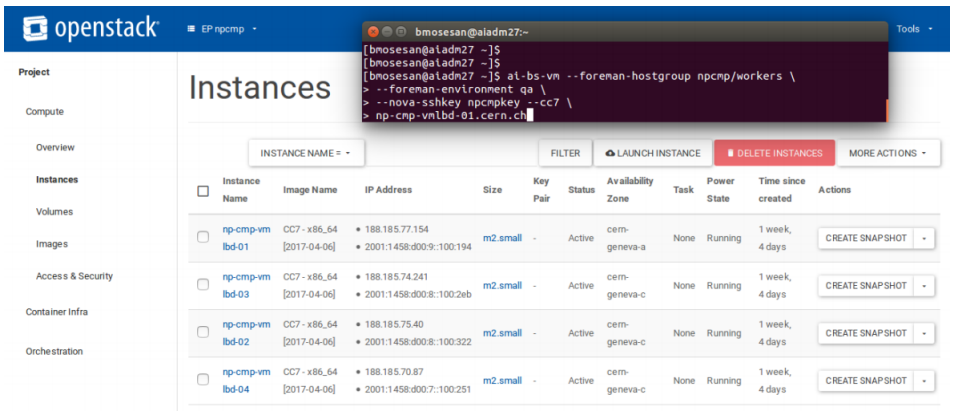


Figure 7: Physical machines in host group

rebooted over the network in order to start the installation of the new OS and Puppet manifest.

Adding a new virtual machine can be done in the command line following the steps provided in the CERN Configuration Management System User Guide.

III.2. Updating Information

The process of editing the information about an node that is already in the cluster can be done through the Foreman interface and the same form as the one for adding nodes will be displayed. The primary interface type cannot be modified, but an new interface can be added. We can use the same procedure for both, physical machines and virtual machines.

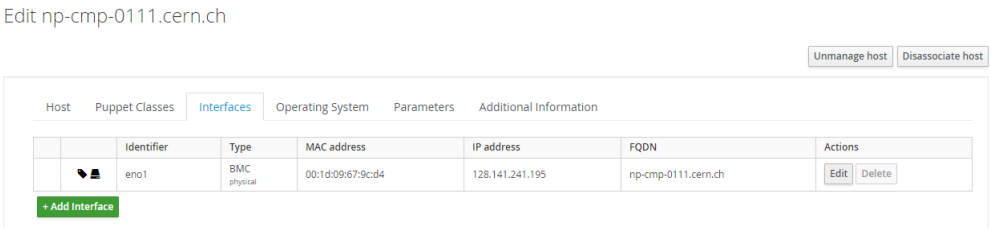


Figure 8: Update hosts details

III.3. Creating new sub-host group

New sub-host groups can be added in any sub-host groups existent. To configure the functionality of the new sub-host group we need to create a new manifest (according to the Puppet standards). After making this manifest we add to the staging area, commit and push to Git. These new codes become visible to Puppet master by less than a minute.

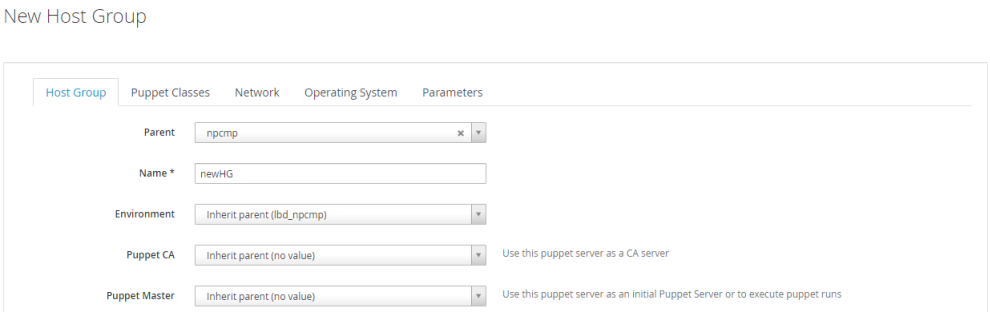


Figure 9: Physical machines in host group

III.4. Integrating new modules

Puppet does away with the worry of dependencies as it can install packages on behalf of the user. Modules are configurable via Hiera variables that are set at host group level. Puppet has a set of reusable modules maintained by several IT groups that can be configured on our nodes. AFS and SSSD were configured using existing modules so our VMs have access to AFS and SSSD and has an interactive shell. It is worth noting that all the relevant monitoring bits for the new running services are brought in by Puppet automatically. Examples of packages to be installed using Puppet are Ganglia, Nagios, EOSclient etc.

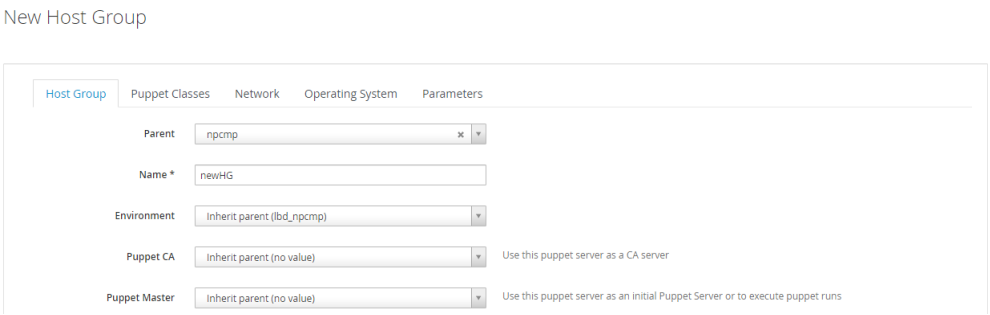


Figure 10: Physical machines in host group

IV. Conclusions and Future Work

The NP cluster consisted, at the beginning, just from bare metal machines. The configuration management system of the cluster was done based on Puppet and Foreman. The design and development of the system architecture are in progress. Part of the work or tasks to be done in the future for this cluster is to setup a notification system to alert users and/or admins

of the Neutrino system in real time information about the nodes and jobs. The implementation of new features requires adding new modules and integrating the new components in the existing project.

Acknowledgements

I would like to express my special thanks and gratitude to dr. Nektarios Benekos for supervising me on my work for the ProtoDUNE experiment at CERN. I would like to thank my professors from the Faculty of Computer Science, Alexandru Ioan Cuza University of Iaşi, for guiding and mentoring me through the past years. Finally, my parents, family and friends who supported me all the way.

Bibliography

[1] Neutrino Computing Cluster at CERN. https://twiki.cern.ch/twiki/bin/view/cenf/neutrinoclustercern.

[2] N. Benekos, M. Potekhin, and B. Viren. A design of the protodune/np04 prompt processing system, 2017.

[3] The DUNE Collaboration. *The Single-Phase ProtoDUNE Technical Design Report*, chapter 5 Software and Computing. 2017.

[4] The ProtoDUNE Dual Phase Collaboration. Yearly progress report on wa105/ protodune dual-phase (2017), 2017.

[5] The ProtoDUNE-SP Collaboration. Report on protodune-sp, np-04, 2016.

[6] Puppet Features. https://www.netways.de/en/products/puppet/features/.

[7] CERN Configuration Management System User Guide. http://configdocs.web.cern.ch/configdocs/.

[8] CERN OpenStack Private Cloud Guide. http://clouddocs.web.cern.ch/clouddocs/.

[9] Tom Junk and Andrew Norman. Protodune-single phase software and computing project plan, 2017.

[10] Workload Management. https://www.techopedia.com/definition/30462/workload-management.

[11] Neutrinos. http://www.fnal.gov/pub/science/particle-physics/experiments/neutrinos.html.

[12] CERN Neutrino Platform. http://home.cern/about/experiments/cern-neutrino-platform.

[13] M. Potekhin, B. Viren, S. Fuess, O. Gutsche, R. Illingworth, M. Mengel, and A. Norman. Design of the data management system for the protodune experiment, 2017.

[14] CERN Prototype. https://dune.bnl.gov/wiki/cern-prototype.

[15] Computing with HTCondor. https://research.cs.wisc.edu/htcondor/index.html.