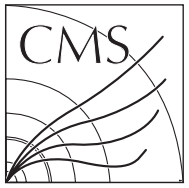
Available on CMS information server

CMS CR -2011/348



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

06 December 2011 (v2, 23 March 2012)

Integrating Amazon EC2 with the CMS Production Framework

Andrew Malone Melo for the CMS Collaboration

Abstract

As cloud middleware and cloud providers have become more robust, various experiments with experience in Grid submission have begun to investigate the possibility of taking previously Grid-Enabled applications and making them compatible with Cloud Computing. Successful implementation will allow for dynamic scaling of the available hardware resources, providing access to peak-load handling capabilities and possibly resulting in lower costs to the experiment. Here we discuss current work within the CMS collaboration at the LHC to both perform computation on EC2, both for production and analysis use-cases. We also discuss break-even points between dedicated and cloud resources using real-world costs derived from a CMS site.

Presented at *ACAT 2011: 14th International Workshop on Advanced Computing and Analysis Techniques in Physics Research*

Integrating Amazon EC2 with the CMS Production Framework

Andrew Melo*, Paul Sheldon

Vanderbilt University, 6301 Stevenson Center, Nashville, TN, 37235, USA

E-mail: andrew.m.melo@vanderbilt.edu

Abstract. As cloud middleware and cloud providers have become more robust, various experiments with experience in Grid submission have begun to investigate the possibility of taking previously Grid-Enabled applications and making them compatible with Cloud Computing. Successful implementation will allow for dynamic scaling of the available hardware resources, providing access to peak-load handling capabilities and possibly resulting in lower costs to the experiment. Here we discuss current work within the CMS collaboration at the LHC to both perform computation on EC2, both for production and analysis use-cases. We also discuss break-even points between dedicated and cloud resources using real-world costs derived from a CMS site.

1. Introduction

Large international scientific collaborations face significant computing challenges; multi-Petabyte data sets that require petaflops of computing as well as globally distributed users and computing resources. To meet those computing challenges, collaborations have invested heavily in dedicated computing resources. Such an approach requires the expense and careful management of floor space, power, and cooling, the use of grid technologies and high performance networks, and the implementation of highly developed strategies for data movement and placement.

The Compact Muon Solenoid[1] (CMS) experiment at the Large Hadron Collider[2] (LHC), for example, processes several petabytes of data yearly, using approximately 45,000 cores purchased and maintained by the experiment and spread globally across the grid. Core usage is highly managed to insure that the highest priority experimental goals are met. There is some fluctuation in demand: the experiment doesn't log data year-round and user demand has significant variation depending on the schedule of conferences. The experiment makes opportunistic use of shared resources such as those offered by various academic computing centers/collaborations. However, fixed operational costs and asset depreciation reduce the benefits and cost-effectiveness of increased hardware capacity when such hardware is not being used regularly.

Recently, a proliferation of "computing as a utility" or "cloud computing" services have made available to users the ability to purchase CPU time at an hourly rate, with little to no financial commitment. Service providers are often established internet businesses, such as Google and Amazon, that have built extremely large compute server facilities, initaially for their own purposes. Amazon's Elastic Compute Cloud[3] (EC2) is a commercial on-demand compute service where a user can rent

* Corresponding author.

virtual machines by the hour and merely pay the hourly rate plus bandwidth charges. The costs of housing, powering, cooling, maintaining, and managing this hardware is borne by the provider. A growing number of varied users are sourcing their short-term computing needs to EC2 and similar services, when it would otherwise be difficult, expensive and time-consuming to roll out more capacity for what may be a transient need[4].

2. Cloud Computing Services and CMS

Cloud computing resources free the user from the burden of housing, servicing, and managing expensive and sometimes unreliable equipment. They also allow the user to purchase only those compute cycles and bits of storage needed to fulfill their specific goals or mission. In principle these can be purchased just-in-time as needed. Economies of scale may lead to a lower support cost per unit of hardware, and these lower costs may be passed on to the user.

There are also potential disadvantages to cloud computing services. The provider must balance availability with inactivity – if it over buys hardware then it has increased its cost without recovery, and these costs must be passed on to the consumer. If it under-buys hardware, then an important user advantage, being able to get access to resources when needed, is compromised. Necessary profit margins required by a provider also increase cost of service. Economies of scale may not dramatically lower support costs, since large scale infrastructure increases management and maintenance complexity, as does the need to support a user base with broad requirements and usage patterns. Supporting infrastructure that has only one "user" (like a dedicated CMS site) is likely to be significantly easier to manage and less expensive.

We have investigated using EC2 to supplement or replace the compute capacity of CMS by renting out virtual machines and integrating them with a virtual site consisting of solely cloud resources.

3. Methodology

We sought to provide tools and the framework for end users to integrate cloud resources to their workflows as seamlessly as possible. To do this, we created a virtual Compute Element (CE) and registered it with the Open Science Grid (OSG) and CMS. On the CE, we installed and configured a number of services required to accept jobs from the grid and submit them to the cloud. Standard GRAM, Condor, and Gridftp services were installed and client scripts were created to automate starting and stopping virtual machines (VMs) on EC2. Once the user creates their VMs, the VMs are programmed to bootstrap themselves and connect back to the Condor master located on the CE, allowing them to join the dynamic pool of VMs available to the CE. Then the user can submit jobs using the standard CMS analysis tools (such as CRAB) to the virtual site where the jobs would be scheduled to run within EC2.

To lower the amount of maintenance required, we use the ready-built virtual machine provided by CERN (CernVM[5]). Maintained by CERN, it has a virtual, distributed filesystem that stores the CMS software (CMSSW) installs. This allows us to decouple the release schedule for our VM slave from the release schedule for CMSSW. An additional advantage of CernVM is that it comes with all the necessary CMS-specific prerequisites installed, some of which aren't provided in base Redhat linux install.

An important issue with this architecture is that by default, any user can submit jobs to the virtual site and have their jobs run another users' VMs. In a shared-use environment, such as an academic cluster, there is an expectation of a sharing of resources. Under this architecture, users (or groups of users) are paying for their CPU time directly from Amazon and have an expectation to be able to utilize the resources they've paid for.

Our solution to this issue is a combination of Condor logic and webpages that let the owner of the VM configure an Access Control List (ACL) of allowed users to run on their resources. Each job received from the grid by GRAM has its Distinguished Name (DN) attached to it. The DN is the identity of the user, verified cryptographically with a grid certificate signed by a certificate service (for

example, DOEGrids or CERN Certification Authority). By using some scripts and custom Condor logic on the slaves, we are able to restrict jobs to only run on specific nodes, preventing use of others' resources.

Originally, we had intended to create a virtual Storage Element (SE) using the same ACL structure. However, in July 2011, Amazon changed their bandwidth pricing and made inbound transfers free of charge. Simultaneously, CMS rolled out xrootd to each of their larger sites. Xrootd is a remote file access protocol that lets applications access ROOT files over the Wide Area Network (WAN). Nearly 100% of CMS' datasets are available remotely through xrootd obviating the need to stage-in input datasets. A lot of work has been done in CMS to optimize this remote I/O, and it is reported that even transatlantic I/O (from Lincoln, NE, USA to CERN) only suffers a 10% performance hit. Given these advances, we decided to leave the SE as a possible future enhancement to this framework.

Once caveat within CMS is that the various frameworks (such as CRAB for analysis or WMAgent for production) perform checks to verify that data is actually at a site's SE before jobs can be sent to the CE. Since we've opted to not attach storage to our virtual SE, this site mapping would fail on the virtual CE. Fortunately, patches[6] exist to relax this restriction, allowing jobs to run on CEs, even if the data isn't there. The xrootd service is intended as a fallback; since WAN network links are relatively expensive, it's not efficient for the experiment to provide the functionality by default. For our testing, we patched our analysis tool to ignore data locality, enabling our jobs to be scheduled even if the requested data isn't stored locally.

To test our architecture, we ran jobs which would take data produced from the experiment, scan each event for certain characteristics, and write out a subset of each event to a different file if it matched. Each job was run on both EC2 and Vanderbilt and transferred their outputs to Fermilab. To isolate the performance of the xrootd infrastructure, we ran the jobs at Vanderbilt twice: once reading data locally and once reading data over the WAN from a different site using xrootd. At Amazon, the nodes were the "large" instance with 7.5GB of memory and 2 virtual cores. At Vanderbilt, the nodes were typically 8 core Xeon processors clocked at 2.27GHz with 32GB of memory.

4. Results

Once everything was configured, a number of EC2 instances were started and dummy CMSSW jobs were executed to pre-warm the CernVMFS caches. Then, analysis jobs were executed at both EC2 and Vanderbilt. Conveniently, this approach left the client-side interface unchanged (except for the patches we applied), so we were able to use our standard performance measurements to compare the two sites.

A direct comparison between running at EC2 and running with dedicated resources is difficult to replicate due to load on external resources such as storage elements and network links. Additionally, even on idealized resources, different workloads will demonstrate different performance due to differing read access patterns and amount of CPU time required. With those caveats, we observed a processing rate (in seconds/1000 events) of 288 at EC2, 265 at Vanderbilt using local storage, and 310 at Vanderbilt using remote storage. This rate is calculated from the walltime of the jobs and doesn't include time to stage the outputs back to Fermilab. The stage-out times were roughly comparable in all cases, possibly due to the small size of the outputs and the large overhead in setting up connections using the SRM protocol.

5. Costs

An important consideration when evaluating cloud resources compared to dedicated resources is the costs involved. Since the pricing model is variable based on utilization, there is a natural break-even point when it would be more cost-effective to purchase dedicated hardware and vice-versa. Amazon has recently added more pricing structures ranging from a zero upfront cost/high hourly cost model to a high upfront cost/low hourly cost model. Clients can choose to purchase reserved instances for a period of one or three years which grants a lower hourly rate for those resources[7]. The Vanderbilt site provides dedicated resources at a cost of $213.60 per core per year. This amortizes the purchase price

of the hardware over three years, and includes all power, cooling, facilities, network bandwidth, and the costs of support, maintenance, and administration. The hardware includes the compute node and an appropriate share of the costs for internal cluster networking, racks, cluster system services, and shared disk space.

By comparing the efficiencies of the two sites and applying the current prices, we can make a rough comparison between the cost-effectiveness of the two models. At $213.60 per core per year, Vanderbilt can perform this analysis at an approximate cost of $1.79 per 1 million events. Similarly, using on-demand instances, one can perform the same analysis using EC2 at an approximate cost of $13.60 per million events. However, with dedicated resources a year commitment is required and if they are utilized less than 14.5% throughout that year, it would be more cost-effective to simply purchase resources with EC2. This is a break-even point between the two approaches and is roughly the point where a user no longer would want to purchase dedicated hardware to handle bursts of demand.

A user could save some money by purchasing reserved instances, but only at the price tier with the lowest up-front costs. The "medium" and "heavy" utilization reserved instances have up-front yearly and triennial costs which already exceed the cost of dedicated hardware even before additional hourly usage fees are incurred.

Perhaps counter-intuitively, reserved instances make the break-even point worse. The break-even point between dedicated hardware and on-demand EC2 resources is at 1,200 hours/year, but the same break-even point between dedicated hardware and Amazon's light reserved instances is less than 300/hours year. Currently, this means that for the same amount of money, one could either purchase dedicated hardware for a year, use 1,200 hours of on-demand EC2 time, or less than 300 hours of reserved instance time. With reserved instances, the break-even point for EC2 drops to 3.5% for the one-year option and 5.5% for the three-year option.

6. Conclusions

Scientific computing requires an enormous amount of compute resources. These resources have traditionally been provided by the scientific collaborations themselves, requiring a significant commitment in equipment, manpower and facilities. Additionally, many scientific collaborations have bursty resource requirements, often related to the operation of their experiment and the schedule of scientific conferences. This leads to either resources being under-utilized or incurred delays due to a lack of resources when demand increases.

Cloud computing, or computing as a service, has potential to solve these problems. By allowing users and organizations to purchase compute time on-demand with little to no commitment, it is possible to meet rapidly changing demand without the financial commitment of purchasing hardware that will remain largely idle and unused.

We used Amazon's EC2 service to perform physics analyses with data from the CMS experiment. By registering a grid site and configuring a compute element, we were able to almost seamlessly integrate EC2 with the CMS' analysis tools. We also created an ACL system to ensure that each virtual machine would only be used by authorized users. Through our tests, we show these resources are technically feasible but are not cost-effective unless the resources are going to be used less than 14.5% per year. Any greater than that and dedicated hardware begins to become the more cost-effective option.

7. Acknowledgements

The authors would like to think Burt Holzman for his helpful discussions about Condor and grid software. We would also like to thank Brian Bockleman for his assistance with xrootd.

[1] http://cms.web.cern.ch/cms/

[2] http://public.web.cern.ch/public/en/LHC/LHC-en.html

[3] http://aws.amazon.com/
[4] http://aws.amazon.com/solutions/case-studies/
[5] http://cernvm.cern.ch/
[6] https://twiki.cern.ch/twiki/bin/view/Main/HdfsXrootdUsage
[7] http://aws.amazon.com/ec2/#pricing