Dynamic Extensions of Batch Systems with Cloud Resources

2011 J. Phys.: Conf. Ser. 331 062034

(http://iopscience.iop.org/1742-6596/331/6/062034)

Journal of Physics: Conference Series **331** (2011) 062034 doi:10.1088/1742-6596/331/6/062034

Dynamic Extensions of Batch Systems with Cloud Resources

T Hauth[1], G Quast[1], M Kunze[2], V Büge[1], A Scheurer[1] and C Baun[2]

[1] Karlsruhe Institute of Technology (KIT), Institut für Experimentelle Kernphysik (EKP), Campus Süd, Postfach 69 80, 76128 Karlsruhe, Germany

[2] Karlsruhe Institute of Technology (KIT), Steinbuch Centre for Computing (SCC), Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany

E-mail: hauth@ekp.uni-karlsruhe.de

Abstract. Compute clusters use Portable Batch Systems (PBS) to distribute workload among individual cluster machines. To extend standard batch systems to Cloud infrastructures, a new service monitors the number of queued jobs and keeps track of the price of available resources. This meta-scheduler dynamically adapts the number of Cloud worker nodes according to the requirement profile. Two different worker node topologies are presented and tested on the Amazon EC2 Cloud service.

1. Introduction

High Performance Computing infrastructures at research centers and universities are traditionally maintained by different user groups or in a centralized way by the computing department. The latter intrinsically profits from the existing infrastructure and expert knowledge combined with load balancing between different groups. Assuming a traditional cluster setup, the price for this is a common operating system infrastructure for all users. Here virtualization of operating systems is a promising technique to overcome this drawback as it allows deploying the needed number of fully customized worker nodes on a short time scale.

Although Cloud Computing is well accepted for the deployment of service machines, it is also an ideal technique for a dynamic allocation of batch system worker nodes. In the following, an extension of standard batch systems to allocate Cloud resources dynamically in the context of High Energy Physics computing is presented, supporting both the Ubuntu Enterprise Cloud [1] using Eucalyptus [2] as back end and Amazon Elastic Computing Cloud (EC2) [3].

1.1. Portable Batch System (PBS)

Traditionally, cluster based computing infrastructures employ a PBS to distribute computing jobs among cluster machines called worker nodes. Users submit their jobs to batch queues holding them until resources are available and the jobs can be handed over to worker nodes. Typical installations use a static list of worker nodes within the local data center. This list rarely changes and nodes are only removed for maintenance or due to hardware failures.

A popular open-source implementation of a PBS is the TORQUE resource manager by Cluster Resources [4] which is used and maintained by the scientists of the EKP [5] in a local computing cluster.

1.2. Cloud Computing

The term Cloud Computing covers a large variety of applications. This paper refers to the Infrastructure-as-a-Service (IaaS) paradigm. Cloud providers offer the possibility to run pre-packed or custom virtual machine images on physical machines within their data center. In contrast to cluster based computing infrastructures, it is quite common to boot up many Cloud instances for a short amount of time and to discard them, as soon as they are not needed any more.

During the last years, various Cloud providers and implementations entered the service market. One of the first to offer Cloud services was Amazon with EC2. They offer different machine configurations, so called machine types, on a pay-per-hour basis. The software used by Amazon to provide and manage their Cloud services is proprietary and not available to the public.

The company Eucalyptus Systems provides a solution to fill this gap with an open-source Cloud Computing infrastructure software, called Eucalyptus, which implements the same API as Amazon's EC2 does.

The Cloud Computing research group at the Steinbuch Centre for Computing [6] at the Karlsruhe Institute of Technology runs a private Cloud based on Eucalyptus. At its current stage of expansion, the private Cloud can run up to 40 single-core virtual machines.

2. Extending Local Infrastructures with Cloud Resources

The EKP institute is taking an active role in processing and analyzing data sampled by the Large Hadron Collider's (LHC) Compact Muon Solenoid (CMS) physics experiment at CERN [7]. As part of this research project, scientists at EKP have to utilize as many computing resources as needed to get the analysis jobs done in time. Up to now, a cluster PBS handles this load. Due to the static nature of this installation, it is not possible to add more computing resources if needed. Especially during peek demands, the utilization of Cloud resources to extend the local computing capacity is desired. Traditional cluster setups do not offer this kind of dynamic extendibility.

Moreover, the operating system and application setup of the local research groups are specifically tuned to meet the needs of the CMS experiment. In addition, a second cluster shared among nine different institutions could be used, but the cluster setup does not cover the specific operating system and software needs of CMS. All users of the cluster have to run their applications on the same software stack. Related research of the EKP on batch virtualization[8] provides a solution to overcome this drawback on traditional clusters.

Cloud Computing allows to customize the software setup down to the operating system and multiple virtual machine images can be maintained and the one best suited for a specific computation job can be booted. Moreover, since many Cloud sites by different companies are available, the Cloud provider can be changed in an easy fashioned way or more than one provider can be used simultaneously. This allows to leverage the different Cloud provider's payment models and react on pricing changes.

3. ROCED: A Cloud Meta-Scheduler

In order to profit from the advantages discussed above, a prototype meta-scheduler called ROCED (**R**esponsive **o**n-demand **C**loud **E**nabled **D**eployment) was developed and features a modular design to provide a flexible way to assess different concepts and technologies.

3.1. Design Idea

The ROCED scheduler processes various input parameters and decides about the utilization of the connected Cloud resources. The following input parameters are defined:

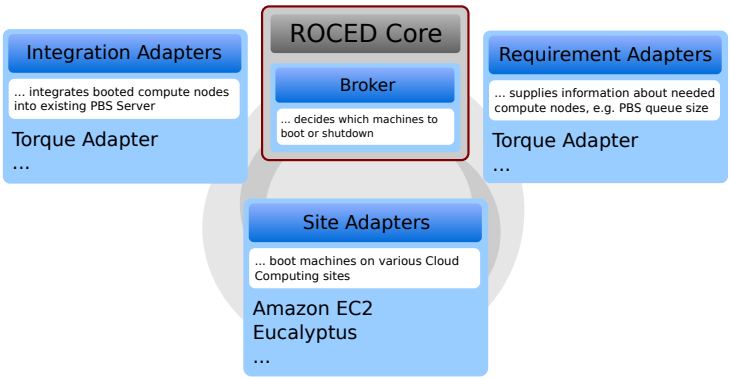

Figure 1. ROCED's modular design allows for the replacement of any component: PBS (Integration and Requirement Adapters), Cloud back end (Site Adapters).

- Current pricing of Cloud resources
- Current workload on the batch system
- Priority of certain processing jobs
- Number of overall worker nodes
- Manual decision of the administrator to increase or decrease the computing power of the system

Depending on pre-defined rules and the current requirement profile, ROCED will decide which Cloud sites to use and how many instances to start inside the Cloud. In the same step, it will shut down instances which are no longer needed in order to free Cloud resources. Since many Cloud providers are following the principle pay-as-you-go on an hourly basis, this allows to keep the costs as low as possible.

Once the Cloud worker nodes are started and running, ROCED integrates them into the batch server's worker nodes listing. Therefore an OpenVPN [9] tunnel is established between the Cloud instance and the local network. Afterwards, the new worker nodes are registered at the batch server and jobs are being transferred for processing.

If ROCED decides that a specific worker node is no longer needed within the batch system, this node is excluded from the list of active nodes. As soon as the node finished its remaining jobs, it is removed from the PBS and the Cloud instance is stopped.

This guided way of node registration and deregistration ensures that the operations of the PBS server are not disturbed by adding and removing worker nodes.

3.2. Implementation

ROCED is implemented using Python 2.6. All major components of the system are managed by so-called Adapters which supply a concrete implementation of various abstract functionalities (Figure 1).

So far, ROCED supports the Amazon EC2 and Eucalyptus Cloud interfaces and the TORQUE resource manager. Other batch servers and Cloud providers can be seamlessly integrated into the application by programming new Adapters. The Broker component, which decides about the Cloud usage, will then take the new resources into consideration.

3.3. Cloud Worker Node Topology

Two different types of topologies to attach Cloud worker nodes to a local batch server were investigated and can be set up in a dynamic way using ROCED.

3.3.1. Topology 1: Route jobs to a specific Cloud PBS. To support a large amount of worker nodes it is preferred to boot a second, dedicated PBS server within the Cloud data center. Once ROCED has booted new Cloud worker nodes, they will be added to the Cloud PBS. This nodes can use the internal network infrastructure of the data center for the communication with the PBS server. The internal network traffic is free of charge for most Cloud providers. Jobs submitted to the local PBS are routed to the Cloud PBS. The TORQUE installation was configured to provide a separate queue `cloud` which forwards all jobs to the Cloud PBS.

This approach takes more effort to set up, but the benefit is, that the PBS communication will not leave the Cloud data center.

3.3.2. Topology 2: Attach Cloud worker nodes to local PBS directly. As soon as the Cloud worker node is started and the OpenVPN tunnel is established between the node and the local network, the node name and the VPN network IP address are added to the local PBS machine. All node status reports and job information are sent between node and PBS server via the VPN. This allows an easy addition of new Cloud worker nodes to the local PBS since all Cloud specifics are abstracted by the VPN tunnel.

As the number of Cloud worker nodes increases, the communication between nodes and server will require higher networking bandwidth. This can exhaust the bandwidth of the VPN tunnel which will degrade the performance of other networking services. In conclusion, this topology is feasible for small to medium sized installations.

4. Setting Up Topology 1: Routing Local Jobs to Cloud PBS

4.1. Test Setup

A Linux distribution that suits the needs of particle physics applications is Scientific Linux [10]. In order to generate a Scientific Linux 5.4 x86-64 machine image with the size of 4 GB and to upload it to the Amazon Cloud, the supplied Amazon toolkit was used. The only additional software installed was the TORQUE worker node executable for 64-bit architecture (TORQUE Version 2.4.6). This machine image will act as the worker node.

The machine providing the TORQUE server was run within the Amazon EC2 Cloud site. For that purpose, an Ubuntu 9.10 AMD64 image and TORQUE 2.4.7 was installed from source. A batch queue was created and configured so each worker node will calculate one batch job at a time.

A queue named `cloud` was created on the local TORQUE server and configured to route all incoming jobs to the Cloud PBS.

The ROCED meta-scheduler itself ran on an Ubuntu 9.10 64-bit desktop machine within the local network and monitored the batch queue of the TORQUE server in the Cloud. The maximum number of Cloud instances ROCED was allowed to start on EC2 was set to 19, so including the Cloud PBS machine, 20 virtual machines were running at the most.

4.2. Test Run

To stress test ROCED and the PBS server-to-node communication within the Cloud environment, 1000 jobs were submitted, each running for a period of 10 seconds. Such short job run time was chosen so the batch server has to transfer jobs to its nodes very often and thus increasing the overall load on the batch system.

As seen in Figure 2, as soon as the jobs appear in the batch queue, ROCED immediately starts the maximum of 19 possible Cloud instances. It takes 130 seconds until the first machine is booted and integrated into the PBS. After 300 seconds all 19 machines are booted and integrated into the PBS. As soon as the batch queue is depleted, ROCED starts to remove worker nodes from the PBS and terminates the Cloud instances.

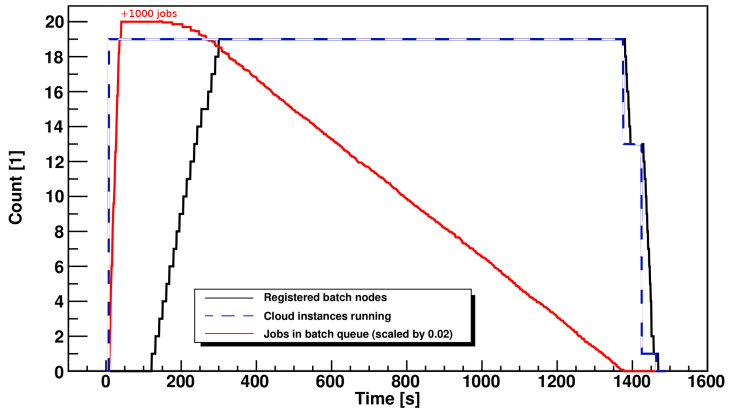

Figure 2. 1000 batch jobs added to the batch queue. Note that `Jobs in batch queue` is scaled by 0.02.

In conclusion, it is shown that a large number of jobs can be transferred to the Cloud PBS, ROCED is able to dynamically add the needed worker nodes and the PBS can leverage the new computing resources immediately.

5. Setting Up Topology 2: Attaching Cloud Worker Nodes to Local PBS

The next example presented is the possibility to extend a local PBS, which already has a number of static worker nodes attached. As soon as the local batch queue holds more than a specified amount of jobs, the so called Cloud threshold, new Cloud worker nodes are allocated. This can multiply the batch system's computing capacity. The system configuration is equal to the setup in the first example, except for the TORQUE and ROCED setup.

5.1. Test Setup

One worker node residing in the local network is statically attached to the PBS system. New Cloud worker nodes are connected via a VPN tunnel to the local network and added to the worker node list of the local PBS server. All test jobs are submitted to the queue of the local batch server and each worker node is configured to process two jobs at a time.

ROCED monitors the size of the queue on the local PBS and is configured to start Cloud worker nodes as soon as the queue size exceeds 8 jobs (Cloud threshold = 8). ROCED was configured to start 4 Cloud worker node instances at maximum.

5.2. Test Run

To test the adaptiveness of ROCED's Cloud usage, a varying number of jobs were sent the to PBS where each job runs for 200 seconds. (Figure 3).

As the first 8 jobs are submitted to the batch server, 2 are instantly handed to the static worker node. The other 6 jobs are left in the queue. If 8 more jobs are added, the Cloud threshold of 8 is exceeded and ROCED starts and integrates 3 out of 4 Cloud worker nodes required to empty the queue. If more jobs appear in the queue, ROCED starts all available 4 Cloud worker nodes thus increasing the computing power of the local PBS by a factor of five.

In conclusion, it was shown that ROCED is able to increase the computing power of a local PBS outfitted with static worker nodes in a dynamic and requirement-based way.

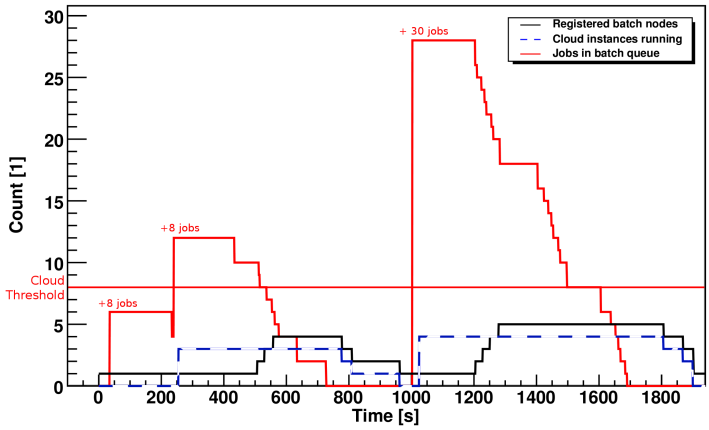

Figure 3. A varying number of jobs is added to the batch queue. ROCED will only start new Cloud worker nodes if the batch queue holds more than 8 jobs (Cloud threshold).

6. Conclusion, Outlook and Future Work

By using a meta-scheduler, it is possible to extend the computing resources of a standard batch system using Cloud Computing resources of different providers. The PBS system was made Cloud-aware without disturbing regular operations.

Two Cloud worker node topologies were introduced and the advantages and drawbacks were evaluated. Attaching Cloud worker nodes to a local PBS server is easy to set up, but results in an increased traffic between the Cloud data center and the local network. Routing jobs to a cloud based PBS is more complex to set up, but has the advantage that some of the PBS network traffic is limited to the Cloud data center. A test installation using ROCED demonstrated the feasibility of both approaches incorporating TORQUE and the Amazon EC2 service.

Next steps are the extension of the EKP's production cluster with ROCED and Cloud resources in order to handle peak loads on the system - so called Cloud bursting.

Acknowledgements

We want to thank all involved EKP and SCC members of KIT Campus North and South for their support and help during the development and implementation of ROCED. In addition we thank the "Bundesministerium für Bildung und Forschung" (BMBF) as well as the Helmholtz Alliance "Physics at the Terascale" for their financial support

References

[1] Canonical `http://www.ubuntu.com/cloud`
[2] Eucalyptus `http://open.eucalyptus.com`
[3] Amazon EC2 `http://aws.amazon.com/ec2/`
[4] Cluster Resources `http://www.clusterresources.com`
[5] EKP `http://www-ekp.physik.uni-karlsruhe.de`
[6] SCC `http://www.scc.kit.edu`
[7] CERN `http://www.cern.ch`
[8] Büge V, Hessling H, Kemp Y, Kunze M, Oberst O, Quast G, Scheurer A and Synge O 2010 *Journal of Physics: Conference Series* **219** 052010 URL `http://stacks.iop.org/1742-6596/219/i=5/a=052010`
[9] OpenVPN `http://openvpn.net`
[10] Scientific Linux `https://www.scientificlinux.org`