Dynamic partitioning as a way to exploit new computing paradigms: the cloud use case.

To cite this article: Vincenzo Ciaschini et al 2015 J. Phys.: Conf. Ser. 664 022014

View the article online for updates and enhancements.

You may also like
- Intellectual analysis and estimation of gross greenhouse gas emissions
  Mikhail Gorodnichev, Yuri Trofimenko, Timur Potapchenko et al.
- Site in a box: Improving the Tier 3 experience
  J M Dost, E M Fajardo, T R Jones et al.
- LHCOPN and LHCONE: Status and Future Evolution
  E Martelli and S Stancu
Dynamic partitioning as a way to exploit new computing paradigms: the cloud use case.

Vincenzo Ciaschini¹, Stefano Dal Pra¹, Luca dell’Agnello¹
¹INFN-CNAF, viale Berti-Pichat 6/2, 40127 Bologna, Italy
E-mail: stefano.dalpra@cnaf.infn.it

Abstract. The WLCG community and many groups in the HEP community have based their computing strategy on the Grid paradigm, which proved successful and still ensures its goals. However, Grid technology has not spread much over other communities; in the commercial world, the cloud paradigm is the emerging way to provide computing services. WLCG experiments aim to achieve integration of their existing current computing model with cloud deployments and take advantage of the so-called opportunistic resources (including HPC facilities) which are usually not Grid compliant. One missing feature in the most common cloud frameworks, is the concept of job scheduler, which plays a key role in a traditional computing centre, by enabling a fairshare based access at the resources to the experiments in a scenario where demand greatly outstrips availability. At CNAF we are investigating the possibility to access the Tier-1 computing resources as an OpenStack based cloud service. The system, exploiting the dynamic partitioning mechanism already being used to enable Multicore computing, allowed us to avoid a static splitting of the computing resources in the Tier-1 farm, while permitting a share friendly approach. The hosts in a dynamically partitioned farm may be moved to or from the partition, according to suitable policies for request and release of computing resources. Nodes being requested in the partition switch their role and become available to play a different one. In the cloud use case hosts may switch from acting as Worker Node in the Batch system farm to cloud compute node member, made available to tenants. In this paper we describe the dynamic partitioning concept, its implementation and integration with our current batch system, LSF.

1. Introduction
The Tier–1 farm at INFN-CNAF manages computing resources, currently amounting to ~15000 slots, dedicated to LHC HEP experiments and other scientific communities. These are accessible by Virtual Organizations (VOs) as traditional Grid resources, through CREAM Computing Elements and managed by the LSF batch system [1]. Direct job submission to LSF is also possible to local users, which are usually members of smaller experiments or groups. The normal workload at the site is saturation, meaning that every available computing resource is busy all the time, and Grid or local jobs are pending in the wait for their turn to run. As an increasing demand for cloud resources has emerged, we have studied the possibility to dynamically move computing resources from Grid to Cloud and vice-versa.

2. Motivation
There are a number of good reasons for a computing centre such as the INFN Tier–1 to desire a reversible and flexible way to dynamically vary the quota of Cloud and Grid resources:
A VO may want to dedicate a certain amount of computing power to a “cloud computing campaign”, then claim back the resources to Grid usage.

A VO, or the site itself, may want to perform a “smooth migration” from Grid to Cloud, moving resources a few at a time.

A group may want to use both Grid and Cloud resources at different shares over time.
3. Analysis and requirements

We begin considering the scenario of a computing farm whose nodes are partitioned per-role between Grid and Cloud resources, and focusing our attention on the process of expanding one partition by moving computing resources from the other one. Then we consider finer details needed to provide a fairly complete description of the dynamic partitioning model.

3.1. From Grid to Cloud

Assuming that new Cloud computing resources are to be provided at the site and assigned to a tenant (analogous of usergroup in batch system terminology, or VO in case of HEP communities) it is necessary to take off a number of Worker Nodes (WN) from the control of the LSF batch system and then make them available as Compute Nodes (CN) under the control of a Cloud Controller (CC).

It is assumed that each WN selected for role conversion has running jobs dispatched by LSF, and a consequent drain time will be needed. During this transition phase, no new activity can begin at the node.

After the drain time finishes, the node can be notified as available to a cloud controller, and assigned to one tenant. From then on that hypervisor would be an active CN in the Cloud partition of the computing farm.

3.2. From Cloud to Grid

If cloud resources are not needed anymore by a tenant, a number of Compute Nodes can be reclaimed back at work in the Grid partition.

Each CN selected for role conversion is expected to host Virtual Machines in it, dedicated to some kind of HEP computation. A drain time will be needed, during which the Cloud Controller must not instantiate new VMs on the CN. Furthermore, running VM should be requested to finish their work until a definite deadline. This would enable a graceful job termination at the VM. Live migration of running VMs to other hypervisors can also take place, provided that free hosting resources are available.

After the drain time finishes, active VMs in the CN can be destroyed and the node can be notified as available to LSF. From then on that node would be an active WN in the Grid farm.

3.3. Adapting fairshare logic

One of the most important tasks of a batch system is that of arbitrating access to scarce computing resources among competing usergroups. The commonly adopted method to provide this capability is the so called fairshare policy. At dispatch time, the batch system selects for execution pending jobs from the groups having higher dynamic priority, so that to grant a configured ratio of runtime and CPU time over a sliding time window covering recent history. Every experiment using computing resources has agreed on a pledge of resources that the site grants to make available.

This fact has an important side-effect. When resources are to be moved from Grid to Cloud and assigned to a tenant, the pledge in LSF for the corresponding usergroup must be lowered consistently. For example, if a VO requires the site to provide an amount of HS06 resources for Cloud computation, an equivalent number of WN is to be put off the control of the LSF master, activated as CN and assigned to its vo tenant. In LSF, the share of the u_vo usergroup must be reduced by the right amount. Not only this is needed to keep the overall quota of that VO constant, it is also needed to prevent any other usergroup to be negatively impacted by a smaller and less powerful Grid farm. An alternative solution can be considered: when a node moves to Cloud, it is artificially kept busy in LSF, by forcing there empty jobs, executing sleep and submitted on behalf of the same user that will use the resource at cloud side. This approach has
the advantage of not requiring adjustments of the fairshare quotas in LSF: every active cloud resource will have a corresponding “fake job” for the same user at Grid side.

3.4. Further details
Commuting the role of a node, WN ↔ CN, should not be an operation requiring reinstallation or reboots. These can be in fact avoided by having both the needed components and daemons up and running at the node; namely, the LSF daemons and the Openstack (currently, Juno) ones. All we need should be a simple way to trigger the role switch of a node and enabling or disabling its availability for Cloud or Grid activity.

One obvious implication is that no more than one set of daemons at a time can be enabled, since the machine should not play two roles at a time. Furthermore, during draintime phases, both the roles are disabled. Theoretically, this is not strictly needed, and it would be desirable having nodes performing a gradual transition and remaining in a hybrid state, where both cloud and grid jobs can coexist while the machine is changing role from WN to CN or in the opposite direction. This goal is however considered as a “next level” optimization and not currently pursued.

The draintime phase of CNs switching to WN differs from those of WNs by one important detail: there is no native direct way to know that a VM has finished its tasks nor it can be expected to automatically disappear after finishing them. In this case, a reasonable timeout must be defined, after which the VMs are destroyed. To prevent this to cause computational losses, the Machine / Job Features Task Force \cite{4} has defined a protocol that permits the VM to have access to a variety of information, included a $\text{shutdowntime}$ which the cloud user can consider.

3.5. Driving the partition
Having implemented the $\text{director}$, a mechanism to take care of the role switching of the nodes, different strategies can be adopted to manage it, i.e. to decide when to trigger the migration from one side of the partition to the other and how many nodes have to be involved:

**Manual** The most simple and immediate one, is controlled by an administrator, which specifies the number of nodes, ownergroup and direction of the migration, upon request from the experiment.

**User driven** The need for more Cloud resources can be driven by the tenant, by issuing a rate of request for VM instantiations. The number of requests per unit of time that the Cloud Controller cannot satisfy because of lack of available Compute Nodes can be used to trigger migration of some WN to CN. Stopping the resource request flow, could indicate that the cloud resources can be freed. Although this strategy could work, it must be agreed with the user.

4. Implementation

4.1. Role switching

**WN $\rightarrow$ CN** To disable the WN role, one straightforward method would be that of closing it with the $\text{badmin hclose}$ command, however a more versatile strategy has been adopted, based on the definition in LSF of custom External Load Index Managers (ELIM) and External S UBmission (ESUB) modifier scripts. The elim script is run by LSF at each node in the cluster and reports at regular times the value of a custom $\text{dcloud}$ flag, whose value indicates the partition to which the node belongs. The esub script is run at job submission time on the submitting host and simply alter its submission parameters by adding the requirement of running on a node having $\text{dcloud}==0$. This mechanism is a variant of the one implemented to provision multi–core resources with LSF and more deeply detailed in \cite{3}. 

\cite{4}

21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015) IOP Publishing
Journal of Physics: Conference Series 664 (2015) 022014 doi:10.1088/1742-6596/664/2/022014
Conversely, when a node migrates from the cloud partition to lsf, the CN must be disabled to prevent instantiations of new VMs. This is done using APIs from the nova compute module of Openstack. Then, every running VM at the node must be terminated after a convenient timeout, before the conversion can happen. The node is disabled (but not removed) as a Compute Node so that it remains visible to the batch system and known to the infrastructure, but will not be used to instantiate new Virtual Machines.

4.2. Overall system and director
Figure 1 describes the system and details its working. While each sub-farm is easily recognizable in its basic components, the new service, the Partition Director, is the additional service that will move hosts between the two partitions.

Figure 2 is the description of the algorithm implemented by the Partition Director, described as a state machine above system detailing exactly how and when the transitions take place.

- At $T = 0$, all nodes are $c_i \in G = \{c_1, \ldots, c_N\}$
- When $k$ Compute Nodes are requested, they are moved to Drain from $G$ to $D_G = \{c_1, \ldots, c_k\}$ by the director.
- When the drain finishes, it is moved from $D_G$ to $C$ and becomes available as a Compute Node.
- When a Compute Node $c_i \in C$ must work again as a WN, it is moved to $D_C$ and begins a drain time. The duration can be specified through the shutdowntime parameter from the machinejob features.
- When a Compute Node $c_i \in D_C$ expires its shutdowntime, Existing VMs are destroyed and the node moves to $G$.
- The elim script on each node $w_i$ updates its dcloud status:

$$
dcloud(w_i) = \begin{cases} 
1 & \text{if } c_i \in D_G \cup C \\
0 & \text{if } c_i \in G \cup D_C 
\end{cases}
$$

5. Current status and future work
The dynamic partitioning is currently being developed according to this model on a small testbed. Decision has been taken to adopt the most recent available versions, both for LSF and OpenStack (kilo). This in turn implies the adoption of RedHat7 or derived distributions for Compute Nodes and Cloud Controllers. This has a drawback since there is no official available
middleware for Worker Nodes in WLCG for such distributions. On the other hand, better support is provided in the Openstack API set to implement the director component.

As an improvement to the model described here, the possibility of nodes working in a hybrid status can be investigated. A hybrid node would be able to host both batch jobs and cloud VMs at the same time. On the LSF side this could be achieved by adding one more External Load Index, representing the current number of running jobs at the node, and updating the logic of ESUB and ELIM. At the cloud side, however, the ability to dynamically change the amount of available resources on a CN seems to be currently missing. This means that hybrid state could only be possible during the migration of a node from Cloud to Grid: every time a VM terminates, the corresponding slots can be declared available to the batch system.

6. Conclusions
The dynamic partitioning model enables the coexistence at a site of a batch-system based farm with a Cloud one. Computing resources are shared in mutual exclusion from a common set of physical nodes.

The user can decide how many resources dedicate to Grid computing and how many to cloud, and a way to let the user itself drive the request for more Grid or Cloud resources has been proposed.

References
[1] LSF Admin guide [http://www-304.ibm.com/support/customercare/sas/ib/plcomp/platformlsf.html]
[2] Openstack api-reference [http://goo.gl/3ZZTJ1]
[3] S.Dal Pra “Efficient provisioning for multicore applications with LSF”, CHEP-2015, [http://goo.gl/7hRVUf]
[4] Machine / Job Features Task Force [https://twiki.cern.ch/twiki/bin/view/LCG/MachineJobFeatures]