# Dynamic partitioning as a way to exploit new computing paradigms: the cloud usecase.

Vincenzo Ciaschini, Stefano Dal Pra, Luca dell'Agnello

INFN–CNAF, {vincenzo.ciaschini, stefano.dalpra, luca.dellagnello}@cnaf.infn.it

## Problem, usecase, motivation

- The whole INFN-T1 farm ($\sim$ 15000 cores) is currently accessible as a "traditional" Grid resource (CREAM Computing Element, LSF Batch System)
- Problem: We would like to be able to dedicate hardware resources to Cloud Computing for HEP purposes in a flexible and reversible manner.
- Use cases:
  - A VO may want to dedicate a certain amount of computing power to a "cloud computing campaign", then move back the resources to Grid.
  - A VO may want to perform a "smooth migration" from Grid to cloud, moving resources a few at a time.
  - A team may need interactive usage of computing resources.

## Analysis

Providing resources for both Grid and Cloud computing requires to:

- Remove a number of WNs from the control of the LSF batch system.
- Enable them as Compute Nodes (CN) under the control of a Cloud Controller.
- Assign them to one ore more tenants.
- Possibly convert additional WNs to Compute Node, or reclaim some of them back in the Grid farm.

## Shares

Shares in the Grid farm must be adjusted, so that:

- Any experiment moving $k$ WN from Grid to Cloud, should have its share in LSF reduced accordingly.
- Any experiment not using cloud resources, should not be affected by the reduced power of the Grid farm.

## Wall–clock Time

An overall Wallclock–Time must be accounted, by adding two components:

- Grid–side, the Wall–clock time is accounted per–job, as usual.
- Cloud–side, the Wall–clock time is accounted per–node

## Exploiting a solution: dynamic partitioning

A dynamic partitioning mechanism has been deployed at INFN-T1 for the provisioning of multi–core resources. The same technique can be adapted to achieve a Cloud partition.

- The Cloud partition can grow or shrink on a per–need basis (Elasticity).
- On each node, both LSF and Openstack daemons are active. Only one or the other mode can be enabled at a time.
- A Draining phase is needed before moving from a partition to the other
- When a WN is assigned to the Cloud partition, LSF stops dispatching jobs to it (*Draining*). Then it becomes available to the Cloud Controller.

## The implementation

- elim script. It runs on the WN and defines the value of the `dcloud` flag.
- esub script. It is executed at the submission host for each submitted job, enforcing a request for nodes having a resource `dcloud!=1`.
- director script. implements the logic of the partitioning model. It runs at regular times on a master node and selects which WNs or CNs are to be moved from the partition they belong to.

## The Partition Director

- Implemented as finite state machine
- LSF side:
  - manages the status of the `dcloud` flag on the nodes. This is achieved by customizing `esub,elim` scripts and enable/disable job dispatching.
- Cloud side:
  - enable/disable scheduling to the CNs (ref. to Openstack, Juno; this is done using api call to `nova-compute`).
  - destroy existing VM on the CN after a timeout ($\sim$24h). This can be achieved thanks to the work done by the WLCG MachineJobFeatures TaskForce.

## The Dynamic Partitioning model

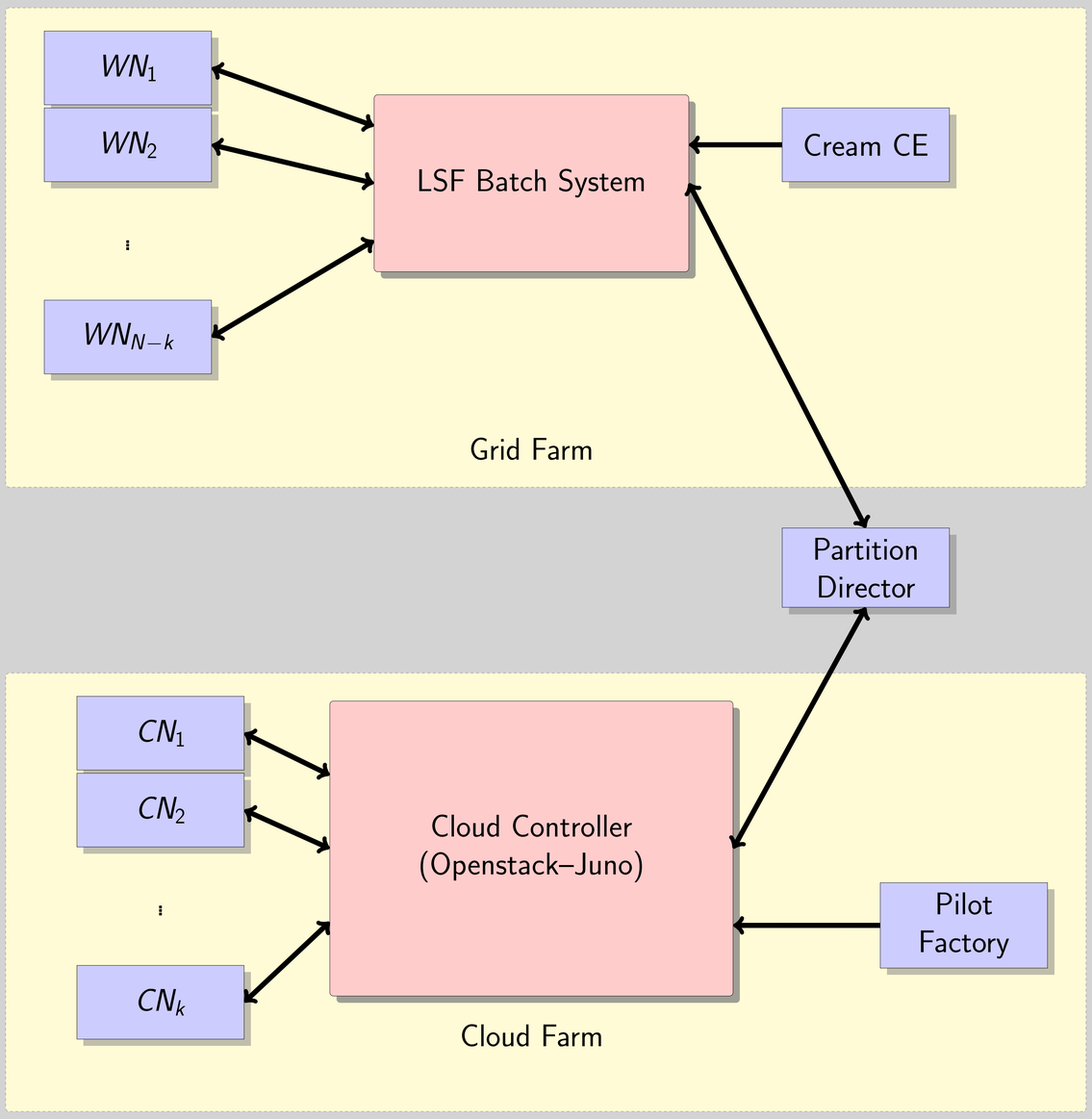


Figure: The partition director switches a node between Worker Node or Compute Node roles.

## Dynamic of the dcloud partition

- 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:

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

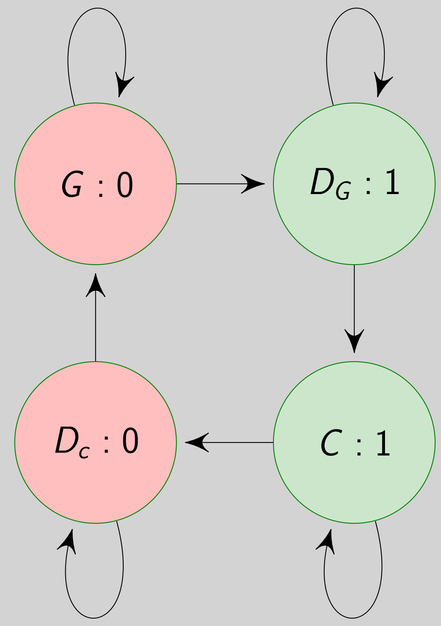


Figure: The Status Transition Map

## Conclusions

- Dynamic partitioning permits cohexistence of Grid and Cloud applications.
- Transition from Cloud–mode to Grid–mode requires to deal with existing VMs after a draining time. User's applications should be aware *machinejob* aware.

## References


- Openstack api–reference: http://goo.gl/3ZZTJl
- S.Dal Pra "Efficient provisioning for multicore applications with LSF ", CHEP–2015, http://goo.gl/7hRVUf
- S. Dal Pra "Job Packing: optimized configuration for job scheduling", HEPiX Spring 2013 Workshop, http://goo.gl/3ZZTJl
- LSF Admin guide http://goo.gl/tZ0fmj