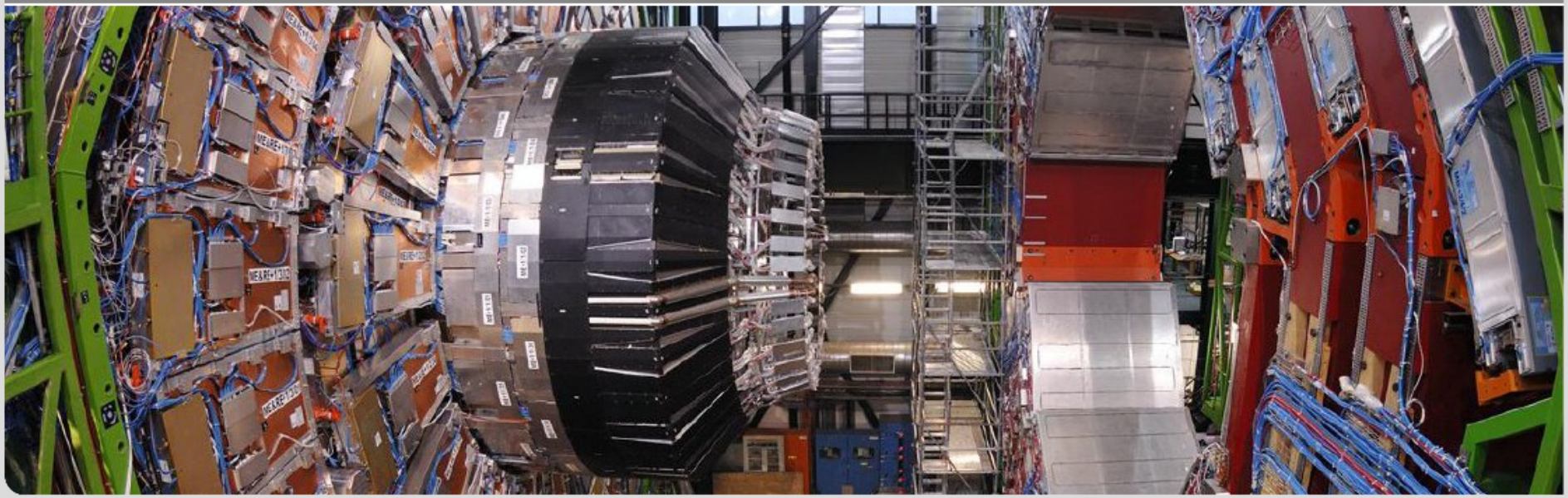
Karlsruhe Institute of Technology

Dynamic Extensions of local Batch Systems with Cloud Resources

C. Baun, V. Büge, **T. Hauth**, M. Kunze, G. Quast, A. Scheurer

2nd Workshop on adapting applications and computing services to multi-core and virtualization, CERN

INSTITUT FÜR EXPERIMENTELLE KERNPHYSIK (EKP) · FAKULTÄT FÜR PHYSIK



Content

- Recap Cloud Computing
- Cloud Computing and Batch Job Processing
- Introducing COCCID: A Framework for Cloud utilization
- Using Amazon EC2 Resources with COCCID
- Running Physics Jobs

Recap Cloud Computing

- Different Cloud Providers offer the service to run virtual machines within their data centers (Cloud Sites)
- Actual hardware and networking infrastructure is abstracted
- Starting and stopping of virtual machines is “cheap” and can be done via a web service API
- Custom machine images can be uploaded
- Most business models are pay-per-hour

One public Cloud example:

Amazon EC2
Offers computing and storage resources on a pay-per-hour model.

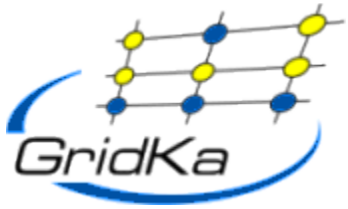
One private Cloud example:

Eucalyptus
KIT maintains a private Cloud testbed based on Eucalyptus. This 40 virtual machines can be used by KIT members.

Current Computing Infrastructure at EKP

The EKP institute is taking an active role in processing and analyzing data sampled by the CMS experiment at the LHC. Using all available computing resources is key to get fast and statistically reliable results.

- Private Cluster with 50 cores, 100 TB storage running Scientific Linux 5 managed by EKP members
- Shared Cluster with 1600 cores, 400 TB storage running SuSE
- Grid resources at KIT (GridKa)
  - `T1_DE_KIT` : Tier 1 within the WLCG
  - `T3_DE_KIT` : Tier 3 within the WLCG



Extending our Infrastructure

- Lot of non-Scientific-Linux Cluster resources available, but not suited for our specific software needs
- Virtualization technologies allow us to use non-Scientific-Linux Clusters, e.g. SuSE Cluster

Cloud Computing and Batch Job Processing

Cloud Computing Services are today mostly used for purposes like web hosting, backup or redundant infrastructure.

Typical Clusters use **batch systems** (PBS, SGE, ...) to distribute the workload to statically attached machines, so called Worker Nodes.

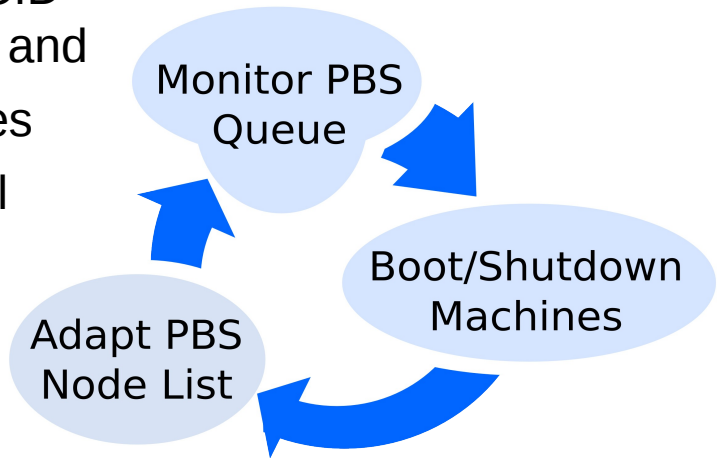
We want to

- Extend our local batch systems resources to Cloud Sites
- Use our own machine images to run our jobs in
- Support multiple Cloud Sites in a transparent way
- Automatically select the cheapest Cloud Site at that time
- Handle peak loads by automatically adding new machines
- Shutdown Cloud machines as soon as they are not needed any more in order to save money

Extending local PBS with Cloud resources

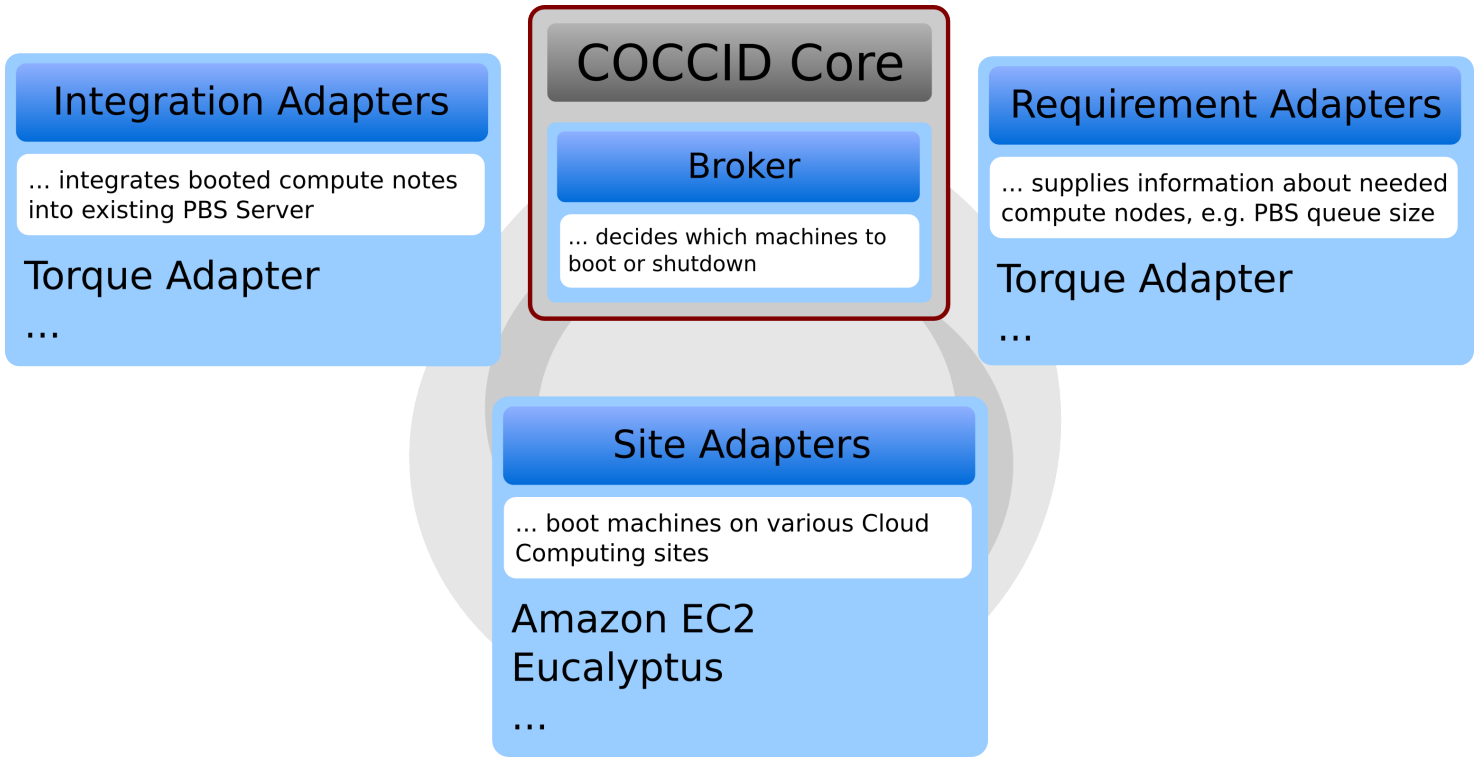
Introducing a Meta-Scheduler which manages the Cloud utilization:
Charming on-demand Cloud Computing Infrastructure Deployment (COCCID)

- COCCID takes various input parameters
  - Number of jobs in batch queue
  - Cloud Site costs
  - Number of Cloud machines running (per Cloud Site and overall)
  - Administrators manual decision to add or remove a certain amount of machines
- Using this input and pre-configured rules, COCCID decides how many Cloud machines are needed and starts or stops them on the according Cloud Sites
- This evaluation is performed in a regular interval to adapt the infrastructure to changes in the requirement profile



COCCID – a Brief Introduction

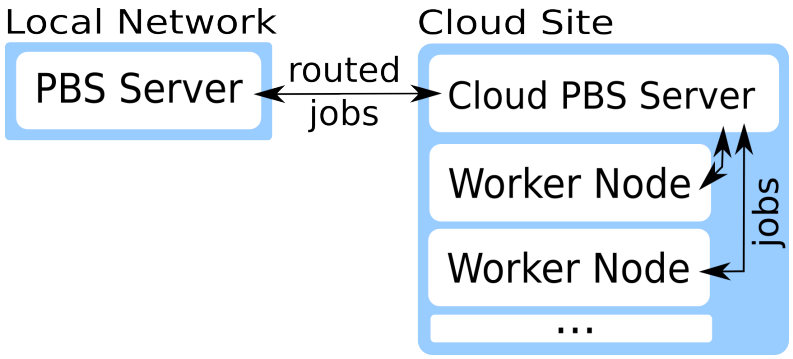
- Written in Python 2.6
- Intended as a test-bed for Cloud related research and evaluation
- Features a modular design: any adapter can be swapped to test new use-cases and scenarios or support more Cloud Sites and PBSes



Worker Node Topology

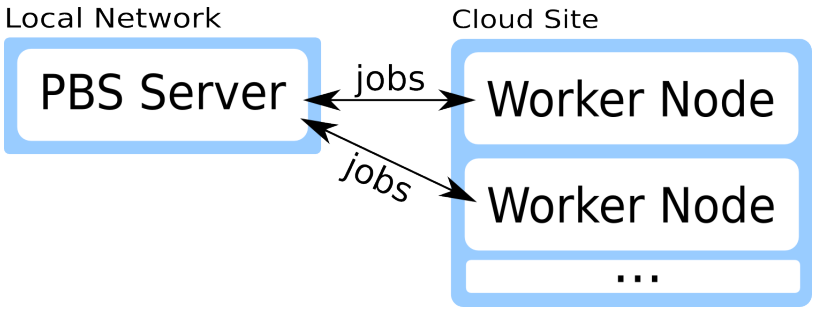
We investigated 2 different topologies when attaching Cloud Worker Nodes to a PBS Infrastructure. Both concepts work but have their specific features.

Top1: *Boot a dedicated PBS Server in the Cloud and attach Worker Nodes there. Jobs are routed from the local PBS to the Cloud PBS*



pro scales better for many Worker Nodes
con specific queue on local PBS needed which routes jobs to the cloud PBS

Top2: *Attach Cloud Worker Nodes to a local PBS directly*



pro easy setup, transparent to PBS Server
con link to Cloud can become a bottleneck for a large number of Worker Nodes due to Server-Node communications

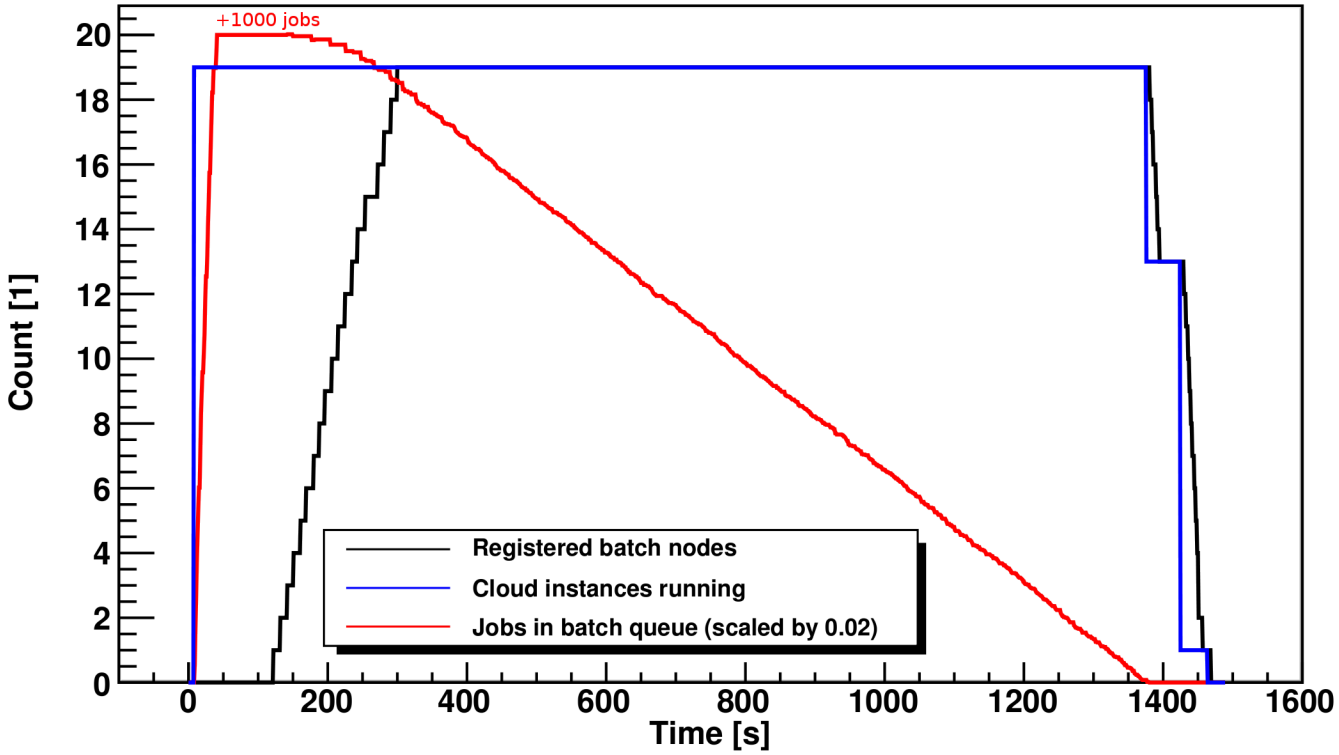
Topology 2 works best for small and medium sized installations due to its easy setup and transparent behavior towards the PBS Server.

Prototype Studies on Amazon EC2

- All virtual machines running on the same hardware configuration (m1.large):
  - 7.5 GB RAM, 2x 64 bit cores with 2 EC2 Compute Units each (1 EC2 Compute Unit ~ 1.0-1.2 GHz 2007 Opteron), 850 GB disk
- Torque Server 2.4.7 running on one Cloud machine (Ubuntu 9.10 amd64)
- Worker Nodes running Scientific Linux 5.4 64bit
- Amazon EC2 account limit: 20 machines max running at the same time
- COCCID has been configured to dynamically add up to 19 machines to the PBS, depending on the number of jobs in the batch queue

Running 1000 jobs

To stress test COCCID and the PBS Server-Node communication within a cloud environment, we submitted 1000 jobs, each running 10 seconds (“sleep 10s“).



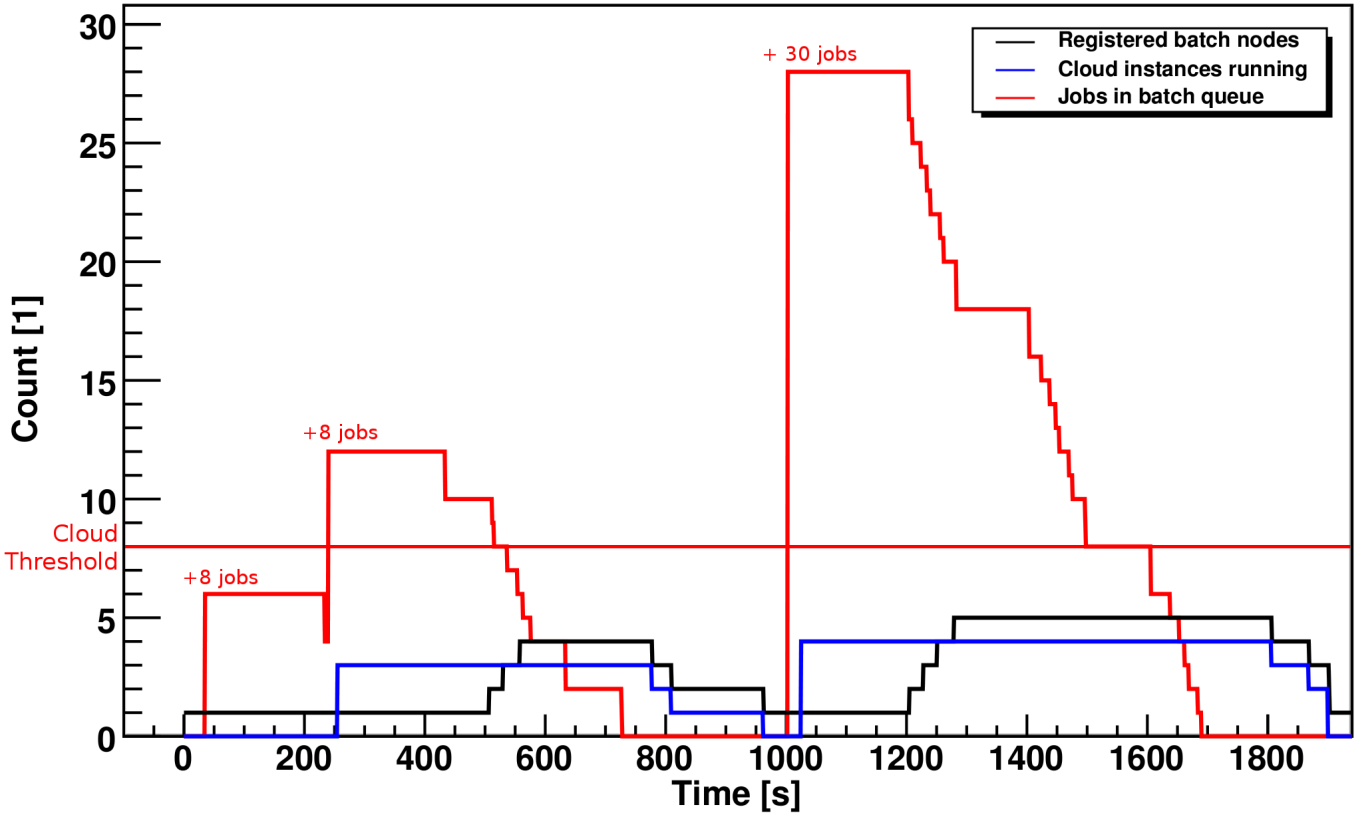
The setup is able to dynamically add the needed Worker Nodes and the PBS can distribute the jobs among them.

Extending local PBS with on-demand Cloud Nodes

- Local PBS with 1 statically attached Worker Node (2 CPUs)
- Cloud Nodes are started as soon as more than 8 jobs are queued. This is called the Cloud Threshold
- 4 Cloud Nodes are started at maximum
- As soon as the queue size falls below the Cloud Threshold, Cloud machines are shutdown, thus freeing Cloud resources
- Cloud machines featuring 2 CPUs, running Scientific Linux 5.4 64bit on Amazon EC2

Extending local PBS with Cloud Nodes

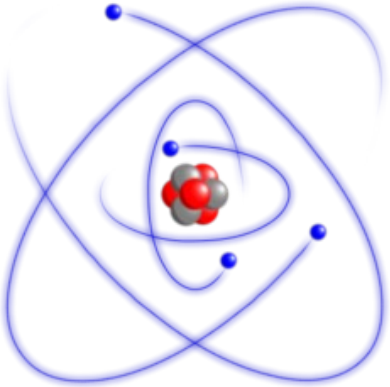
Depending on queue size, a maximum 4 machines are started, each job running 200 seconds



COCCID is able to rapidly increase the computing power of a local PBS by adding Cloud Worker Nodes and adapt to sudden changes in the requirement profile.

Running Physics Jobs

- Scientific Linux 5.4 64bit with CMS experiment software is running on Amazon EC2 and Eucalyptus Cloud Sites
- Monte-Carlo-Production Jobs have been successfully computed
- Job input files have been staged from local storage using a NFS-mount via an OpenVPN Tunnel
- .root output files have been copied back to local storage



Problems

- VPN won't scale well for more Worker Nodes and I/O intensive jobs
- Access to WLCG data from Public Clouds is slow and large data transfers into Cloud Sites is costly

Possible Solution: Private Clouds located within the data centers of WLCG Tier 3 sites can allow customized Worker Nodes and fast access to experiment data and storage at the same time.

Conclusion

- Using Cloud Computing, we are no longer restricted to Scientific Linux Clusters
- We introduced COCCID which can extend local Computing resources to multiple Cloud Providers
- COCCID is able to adapt the available Worker Nodes depending on various input parameters
- We showed the feasibility of two different PBS topologies

Outlook

- Use COCCID on the institute's production cluster to handle peak loads using Cloud resources (“Cloud Bursting”)
- Evaluate I/O intensive use-cases besides Monte-Carlo-Production

Karlsruhe Institute of Technology

Thanks for your attention !

Questions / Comments ?

hauth@ekp.uni-karlsruhe.de

References

- Institut für experimentelle Kernphysik / KIT - http://www-ekp.physik.uni-karlsruhe.de/
- Steinbuch Centre for Computing – Cloud Research Group - http://www.scc.kit.edu/personen/rgcc.php
- Amazon Web Services - http://aws.amazon.com/
- Eucalyptus - http://open.eucalyptus.com/
- Compact Muon Solenoid (CMS) - http://cms.web.cern.ch/
- GridKa - http://grid.fzk.de/
- Worldwide LHC Computing Grid (WLCG) - http://lcg.web.cern.ch/lcg/
- Torque - http://www.clusterresources.com/products/torque-resource-manager.php