How to control bottlenecks using BQS resources

Julien Devémy – CC-IN2P3
julien.devemy@in2p3.fr

About BQS

- BQS is the Batch System used and developed at CC-IN2P3
- BQS both manages a 1500 processors farm and a 64 processors parallel cluster
- BQS is attached to the LCG/EGEE Grid via a locally developed JobManager
- And BQS uses *resources* to control and restrict access to several services

What is a BQS resource

- A BQS resource is **a kind of semaphore**. It's composed of:
  - An Identifier
  - A Current value
  - A Maximum value
  - A list of Administrators (generally *Group Managers*)
  - A list of Authorized Users (if the resource is restricted)
  - A status: *Active* or *Drained*

1 BQS resource per Service to control

Example of a BQS resource

- Identifier: *hpss_babar*
- Current Value: *126* (Currently 126 usages of the service)
- Maximum Value: *500* (Maximum 500 usages of the service)
- Administrators: *devemy, suzanne*
- Authorized users: **.babar*
  (Any user of BaBar can require this resource)
- Status: *Active*
  (Jobs requiring this resource are **not blocked in queue**)

How it works (1)

- For each submitted job, the user indicates the BQS resources *required* by his job:

  Ex: `qsub ... -l hpss,sps_atlas=2,oracle`

indicate that the job reserves 2 simultaneous usages of the resource `sps_atlas`

- If the resource is *restricted* and the user is *not authorized*, the user **cannot submit the job**

How it works (2)

- Before spawning the job to a WorkerNode, BQS checks if all the needed resources are available:
  (*Active* and *not at the Max value*)
  - If it's **not OK**, the job **stays queued**
  - If it's **OK**, the job **becomes running** (incrementing the current values of the required resources)
- **While it's running**, a job can *free* its BQS resource
- When the job ends, its associated resources are *freed* (decremented)

Benefits (1)

- These resources permit to:
  - *avoid bottlenecks:*

    This system avoids the execution of **too many jobs using a given service**
  - *optimize batch farm usage:*

    The machines are **no more filled with jobs waiting for a service** (so doing nothing)
  - *run the farm in degraded mode:*

    If **a service is DOWN**, the corresponding resource is **drained**, so no jobs requiring this resource are spawned but *other jobs continue their own life*

Benefits (2)

- BQS resources also permit **to delegate a part of management** to experiments:

Possibilities for *Resources Administrators* (generally *Group Managers*) to:

- manage *authorized users*
- change resource *Maximum value*
- *drain* the resource

(For resources that are public only *BQS Administrators* are *Resources Administrators)*

Limitations

- Only works for batch jobs
  - Other type of access (interactive...) to services are not taken into account
- Users cooperation needed (!!!)
  - A job can access a service *without indicating the corresponding resource*: **No monitoring is done**
- Concept not supported yet by the GRID
  - The Resource Broker doesn't provide this type of information

BQS resources at CC-IN2P3

- **~100 BQS resources** (most are private resources i.e. managed by *Group Managers*)
- Example:
  - hpss_babar... (Mass storage System access for BaBar)
  - oracle, mysql...
  - Whatever users want !
- **More than 80%** of jobs ask for BQS resources

The future of BQS resources

- Hierarchical resources:
  - If a user indicate a “low level” resource (like a special HPSS resource), the job will have *automatically dependents resources* (global HPSS resource)
  - → Make resources management *simpler*
- Resources *automatically* added for a group, an user or a VO
- Control of *the piracy usage* of a service i.e. a job using a service **without** asking for the corresponding resource

Conclusion

- Technically, the *BQS resource* is not a revolution, it's just a kind of *Semaphore* for jobs, but

Despite of its simplicity, it's a very useful and powerful tool to efficiently manage jobs and external services

Questions ?