Whole-nodes / multi-core job scheduling

Davide Salomoni

INFN CNAF

Foreword

- Whole-node / multi-core scheduling is a request coming from experiments. See e.g. the December 2011 GDB, Ian Fisk, http://indico.cern.ch/getFile.py/access?contribId=8&sessionId=3&resId=0&materialId=slides&confId=106651
- Ian Bird (email to Torre and myself), Nov 29, 2011:
  – At the Management Board today there was a question on one aspect of the work of your TEG. This was related to ideas and strategies for whole node scheduling. With the aggressive parameters of the machine in 2012 which will push pileup much higher, the issue of memory consumption in CMS reconstruction jobs becomes problematic again (I guess there are similar effects in ATLAS too). Sharing memory on worker nodes helps this significantly. We will need to understand how to configure sites to support whole node scheduling.
  – I'll add that we also need do define what is needed on the experiment side to support this.

Points discussed in the email thread

- Multi-core scheduling should be implemented regardless of early/late binding.
- Several sites serve multiple VOs (not only LHC) and support multiple “middleware”.
  - Any solution should not require sites to deploy configurations that are “unique” (and difficult to maintain) to WLCG
- Sites (I would say experiments as well) prefer to share resources rather than partition them statically.
  - A simple solution is to dedicate a separate “queue” (an entry point) to whole nodes, and let users access this queue.
  - But if we want sites to try and exploit resource sharing, we should let some details about the multi-core request reach the site, i.e. the LRMS.
- Virtualization technologies could be used here (see also the Cloud and Virtualization sections – Ulrich) and should be transparent to users – they may offer flexibility to sites (caveat: there may be significant performance penalties, esp. with I/O intensive tasks.
  - This is an area where knowing whether a job is going to be I/O intensive will probably help a resource provider (e.g. let only CPU-intensive jobs run on to virtualized nodes)
  - See the I/O vs. CPU bound jobs section – Owen

Multi-core resource requirements

- Already implemented in Grid middleware, notably in the CREAM CE (http://wiki.italiangrid.it/twiki/bin/view/CREAM/UserGuide#3_1_Submission_on_multi_core_res)
  - At least for LSF and PBS – what about SGE? (SLURM?)
  - Used in production for MPI jobs today; for whole-nodes / multi-core scheduling, # of nodes = 1
  - Uses JDL to let users specify what they want
    - This mechanism supports requests for both whole nodes and multi-cores
    - With limits: OK wrt # of cores, what about RAM size? Local disk? (important?)
    - How to configure sites wrt Advance Reservation mechanisms? (CREAM does not currently issue AR requests)
- A variant of this is to define common "tags" (similarly to what we have e.g. for Storage classes) defining a set of requirements
  - The goal would be to simplify resource requirements; however, semantic of tags must be properly defined across sites, possibly leading to complications
  - Not a big issue for CREAM
  - Opinions?
- How is multi-core scheduling handled by other CEs? (e.g. Condor)
- How hard would it be for experiment frameworks (if they require multi-core scheduling, of course) to use this mechanism?
- Testing / deployment plan?
  - Timeframe to implement CE changes if any required? (CREAM, Condor, others?)
  - Sites willing to test?
  - Experiments willing to test?