EGEE User Forum

Contribution ID: **89** Type: **Oral contribution**

Scheduling Interactive Jobs

Thursday 2 March 2006 14:00 (30 minutes)

1.Introduction

In the 70s, the transition from batch systems to interactive computing has been the enabling tool for the widespread diffusion of advances in IC technology. Grids are facing the same challenge. The exponential coefficients in network performance enable the virtualization and pooling of processors and storage; large-scale user involvement might require seamless integration of the grid power into everyday use.

In this paper,interaction is a short name for all situations of display-action loop, ranging from a code-test-debug process in plain ascii, to computational steering through virtual/augmented reality interfaces, as well as portal access to grid resources, or complex and partially local workflows. At various levels, EGEE HEP and biomedical communities provide examples of the requirements of a turnaround time at the human scale. Section 2 will provide experimental evidence on this fact.

Virtual machines provide a powerful new layer of abstraction in distributed computing environments. The freedom of scheduling and even migrating an entire OS and associated computations considerably eases the coexistence of deadline bound short jobs and long running batch jobs. The EGEE execution model is not based on such virtual machines, thus the scheduling issues must be addressed through the standard middleware components, broker and local schedulers. Section 3 and 4 will demonstrate that QoS and fast turnaround time are indeed feasible within these constraints.

1. EGEE usage The current use of EGEE makes a strong case for a specific support for short jobs. Through the analysis of the LB log of a broker, we can propose quantitative data to support this affirmation. The broker logged is grid09.lal.in2p3.fr, running successive versions of LCG; the trace covers one year (October 2004 to October 2005), with 66 distinct users and more than 90000 successful jobs, all production. This trace provides both the job intrinsic execution time t (evaluated as the timestamp of event 10/LRMS minus the timestamp of event 8/LRMS), and the makespan m, that is the time from submission to completion (evaluated as the timestamp of event 10/LogMonitor minus the timestamp of event 17/UI). The intrinsic execution time might be overestimated if the sites where the job is run accept concurrent execution.

The striking fact is the very large number of extremely short jobs. We call Short Deadline Jobs (SDJ) those where t < 10 minutes, and Medium Jobs (MJ) those with t between ten minutes and one hour. SDJ account for more than 90% of the total number of jobs, and consume nearly 20 of the total execution time, in the same range as jobs with t less than one hour (17%).
Next, we considering the overhead o =(m-t)/t. As usual, the overhead decreases with execution time, but for SDJ, the overhead is often many orders of magnitude superior to t. For MJ, the overhead is of the same order of magnitude as t.
Thus, the EGEE service for SDJ is seriously insufficient.
One could argue that bundling many SDJ into one MJ could lower the overhead.
However, interactivity will not be reached, because results will also come in a

bundle: for graphical interactivity, the result must obviously be pipelined with visualization; in the test-debug-correct cycle, there might be not very many jobs to run.

With respect to grid management, an interactivity situation translates into a QoS requirement: just as video rendering or music playing requires special scheduling on a personal computer, or video streaming requires network differentiated services, servicing SDJ requires a specific grid guarantee, namely a small bound on the makespan, which is usually known as a deadline in the framework of QoS. The overhead has two components: first the queuing time, and second the cost of traversal of the middleware protocol stack. The first issue is related to the grid scheduling policy, while the second is related to grid scheduling implementation.

1. A Scheduling Policy for SDJ

Deadline scheduling usually relies on the concept of breaking the allocation of resources into quanta, of time for a processor, or through packet slots for network routing. For job scheduling, the problem is a priori much more difficult, because jobs are not partitionable: except for checkpointable jobs, a job that has started running cannot be suspended and restarted later. Condor has pioneered migration-based environments, which provide such a feature transparently, but deploying constrained suspension in EGEE would be much too invasive, with respect to existing middleware. Thus, SDJ should not be queued at all, which seems to be incompatible with the most basic mechanism of grid scheduling policies.

The EGEE scheduling policy is largely decentralized: all queues are located on the sites, and the actual time scheduling is enacted by the local schedulers. Most often, these schedulers do not allow time-sharing (except for monitoring). The key for servicing SDJ is to allow controlled time-sharing, which transparently leverages the kernel multiplexing to jobs, through a combination of processor virtualization and slot permanent reservation. The SDJ scheduling system has two components.
- A local component, composed of dedicated single-entry queues and a configuration of the local scheduler. Technical details for can be found at http://egee-na4.ct.infn.it/wiki/index.php/ShortJobs. It ensures the followig properties: the delay incurred by batch jobs is at most doubled; the resource usage is not degraded, eg by idling processors; and finally the policies governing resource sharing (VOs, EGEE and non EGEE users,...) are not impacted.
- A global component, composed of job typing and mapping policy at the broker level. While it is easy to ensure that SDJ are directed to resources accepting SDJ, LCG and gLite do not provide the means to prevent non-SDJ jobs from using the SDJ queues, and this requires a minor modification of the broker code.

It must be noticed that no explicit user reservation is required: seamless integration also means that explicit advance reservation is no more applicable than it would be for accessing a personal computer or a video-on-demand service.

In the most frequent case, SDJ will run with under the best effort Linux scheduling policy (SCHED_OTHER); however, if hard real-time constraints must be met, this scheme is fully compatible with preemption (SCHED_FIFO or SCHED_RR policies). In any case, the limits on resource usage(e.g. as enforced by Maui) implement access control; thus the job might be rejected. The WMS notifies rejection to the application, which could decide on the most adequate reaction, for instance submission as a normal job or switching to local computation.

1. User-level scheduling Recent reports (gLite WMS Test) show impressively low middleware penalty, in the order of a few seconds, which should be available in gLite3.0. It also hints that the broker is not too heavily impacted by many simultaneous access. However, for ultra-small jobs, with execution time of the same order (XXSDJ), even this penalty is too high. Moreover, the notification time remains in the order of minutes. In the gPTM3D project, we have shown that an additional layer of user-level scheduling provides a solution which is fully compatible with EGEE organization of sharing. The scheduling and execution agents are quite different from those in Dirac: they do not constitute a permanent overlay, but are launched just as any LCG/gLite job, namely an SDJ job; moreover, they work in connected mode, more like glogin-based applications. Besides this particular case, an open issue is the internal SDJ scheduling. Consider for instance a portal, where many users ask for a continuous stream of execution of SDJ (whether XXSDJ or regular SDJ). The portal could dynamically launch

such scheduling/worker agents and delegate to them the implementation of the so-called (period, slice) model used in soft real-time scheduling.

Primary author: GERMAIN-RENAUD, Cecile (LRI and LAL)

Co-authors: LOOMIS, Charles (LAL); TEXIER, Romain (LRI)

Presenter: GERMAIN-RENAUD, Cecile (LRI and LAL)

Session Classification: 2c: Special type of jobs (MPI, SDJ, interactive jobs, ...) - Information systems

Track Classification: Special type of jobs (MPI, SDJ, interactive jobs, ...) and information systems