Grid Scheduling for Interactive Analysis

Cecile Germain-Renaud, Charles Loomis, Romain Texier, Angel Osorio

▶ **To cite this version:**

Cecile Germain-Renaud, Charles Loomis, Romain Texier, Angel Osorio. Grid Scheduling for Interactive Analysis. HealthGrid 2006, Jun 2006, Valencia/Spain, Spain. pp.25-33. inria-00117491

HAL Id: inria-00117491
https://inria.hal.science/inria-00117491

Submitted on 1 Dec 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Book Title
Book Editors
IOS Press, 2003

Grid Scheduling for Interactive Analysis

Cécile Germain-Renaud [a,1], Romain Texier [a] Angel Osorio [b] and Charles Loomis [c]
[a] *Laboratoire de Recherche en Informatique*
[b] *Laboratoire d'informatique et de mécanique pour les sciences de l'ingénieur*
[c] *Laboratoire de l'Accélérateur Linéaire*

Abstract. Grids are facing the challenge of moving from batch systems to interactive computing. In the 70s, standalone computer systems have met this challenge, and this was the starting point of pervasive computing. Meeting this challenge will allow grids to be the infrastructure for ambient intelligence and ubiquitous computing. This paper shows that EGEE, the largest world grid, does not yet provide the services required for interactive computing, but that it is amenable to this evolution through relatively modest middleware evolution. A case study on medical image analysis exemplifies the particular needs of ultra-short jobs.

Keywords. Medical Image Analysis Grid Middleware Scheduling

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 [7] enable the virtualization and pooling of processors and storage. In the field of biomedical application, widespread diffusion of grid technology might require seamless integration of the grid power into everyday use.

In the more specific area of medical image processing, algorithms often involve a visual evaluation or exploration of the results. In some cases (e.g. rigid registration of multimodal images of the same patient), algorithms are sufficiently automatic to be executed remotely without interaction and the results sent for visualization to the user. In other cases, such as inter-subject registration, it may be necessary to use the anatomical knowledge of the user to better define the expected result (anatomical correspondences between cortical areas in the brain are loosely defined). In such a case, the interaction may be limited to an alternation of independent distant computations and user correction requests, but a soft real-time interaction would be much more interesting. A last class of image processing algorithm, like pre-operative planning, deeply involves the user and requires at least soft real-time to be really useful.

However, the need for fast turnaround time on the grid is not limited to medical image analysis, but encompasses all situations of display-action loop, ranging from a test and debug process on the exploration of databases , to computational steering through virtual/augmented reality interfaces, as well as portal access to grid resources, or complex

[1]Correspondence to: Cécile Germain-Renaud, LRI Université Paris-Sud. ; E-mail: cecile.germain@lri.fr

and partially local workflows. A critical system requirement is thus the need to move Grids from exclusive batch-oriented processing to general purpose processing, including interactive tasks. Section 2 of this paper will provide experimental evidence about the reality of the need, from the analysis of the activity of a segment of the EGEE grid heavily used by its biomedical community, the biomed VO.

The next question is then a strategy to support interactive jobs on a grid. Virtual machines provide a powerful new layer of abstraction in distributed computing environments [5,2]. 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. However, a production grid is a prerequisite for biomedical and clinical potential users. One of the goals of the AGIR project [3] is to interact with production grids in order to define and implement the new grid services required by medical image analysis, with the EGEE grid as an important target. Section 3 of this paper presents some advances towards this goal, in the area of grid scheduling. 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. We demonstrate that QoS and fast turnaround time can be supported by a production grid.

2. 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.

Fig. 1 shows the histogram of intrinsic execution times. 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 consume nearly 20% of the total execution time, in the same range as jobs with t less than one hour (17%).

Fig. 2 plots the *overhead ratio* o_r as a function of the execution time t. The overhead ratio o_r is formally defined as $(m - t)/t$. Its interpretation is the ratio of the *overhead* $o = m - t$ (which is the time spent "in the system") to the actual execution time t. The components of the overhead are twofold:

Queuing time , which depends on the jobs submitted by other grid users. It is a *scheduling policy* issue.

Middleware penalty : these are the various delays incurred along a job lifecycle because of the job management system, which is the cost of traversal of the middleware protocol stack. Here, the issue is the efficiency of the middleware *implementation.*

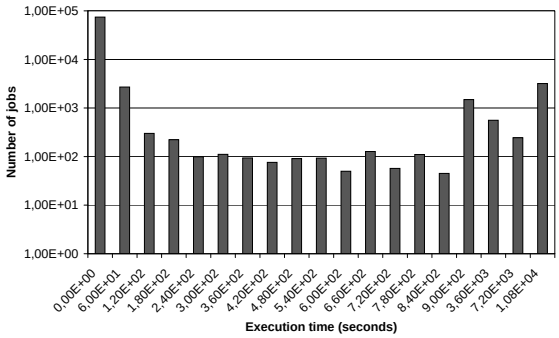

Figure 1. Distribution of execution times

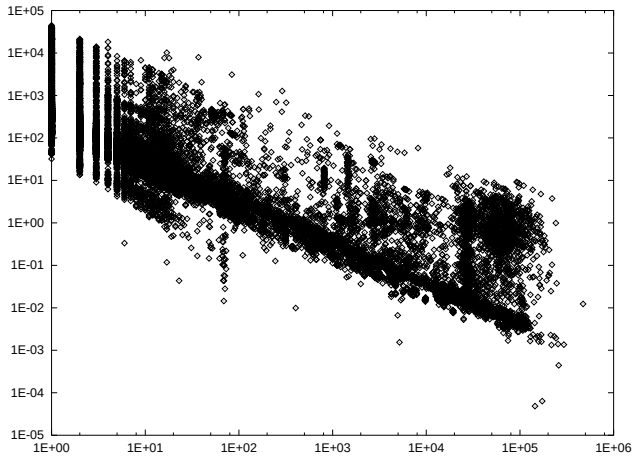

Figure 2. The overhead ratio as a function of execution time - Execution time in seconds, overhead ratio dimensionless (see text for explanation)

The two components are orthogonal: even with a perfect middleware, if, for instance, the jobs were served on a first-come-first served basis, a job will be queued (and thus have to wait) until all its predecessors have been served (note that the EGEE basic scheduling scheme is more complicated). Thus, limiting the delays created by these two components must be addressed separatly, as shown in the next section.

However, the first information provided by fig 2 is that, for SDJ, the overhead is often many **orders of magnitude** superior to t. This is absolutely dissuasive for grid-enabling SDJ. 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.

3. Scheduling for interactivity

3.1. 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 [6] 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 queues and a configuration of the local scheduler. Technical details for MAUI can be found at [11]. It ensures that:
  - Immediate execution of SDJ if resource are available.
  - The delay incurred by batch jobs has a fixed multiplicative bound.
  - The policy is work-conserving, implying that the resource usage is not degraded, eg by idling processors.
  - 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 EGEE Workload Management System.

For the local component, the first question is to prove correctness. Extensive experiments have been conducted on the EGEE cluster at LAL. Fig. 3 (a) shows a case where three kind of jobs are allowed to run concurrently: batch, SDJ, and dteam. On a dual-processor, only two of each kind actually runs, which ensures bounded delay. Fig 3(b) gives the overall site view; the fraction intended limitation of SDJ-dedicated resources (10 running jobs maximum) is achieved.

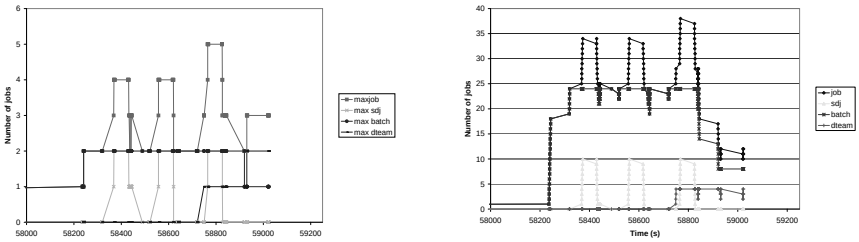

Figure 3. Local scheduling (a) on one machine (dual-processor) (b) on a site

For the global component, the long-term technical solution would be a modification of the Glue Schema. This schema is the information model currently used by EGEE, Open Science Grid, and many other grid projects. In this schema, the target of a job is a Computing Elements (CE), which is mainly a site queue so far. Thus, a new CE attribute (eg QueueAttribute) should be created with the following functions: publish that this queue accepts SDJ jobs and only them. However, the operational use of the Glue schema as a common ground for interoperability between international grids makes its evolution a long process (even if it can be expected to be satisfied in the medium term, because the requirement for this category of attribute meets other ones of the same type, for instance for MPI jobs). Thus, a short term solution has been set up: on one hand, a boolean attribute in the JDL (SDJ) is created; on the other hand, CE dedicated to SDJ must have a name suffixed by ".sdj"; the user interface will translate the boolean attribute towards a regular expression is the JDL requirement `RegExp("*sdj$",other.GlueCEUniqueID)`; finally, the WMS will interpret the lack of this requirement as a prescription not to direct a job to the sdj-suffixed CE. These features will be integrated in gLite 3.2.

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.

3.2. User-level scheduling

Considering the grid middleware penalty for submission, scheduling and mapping of jobs, it cannot be reasonably hoped to reach the order of second, which would be needed for ultra-small jobs, such those considered in the next section. With the most recent and tuned EGEE middleware (gLite 3.0), the middleware penalty remains in the order of minutes. In the gPTM3D project [4], we have shown that an additional layer of user-level scheduling provides a solution which is fully compatible with EGEE organization of sharing.

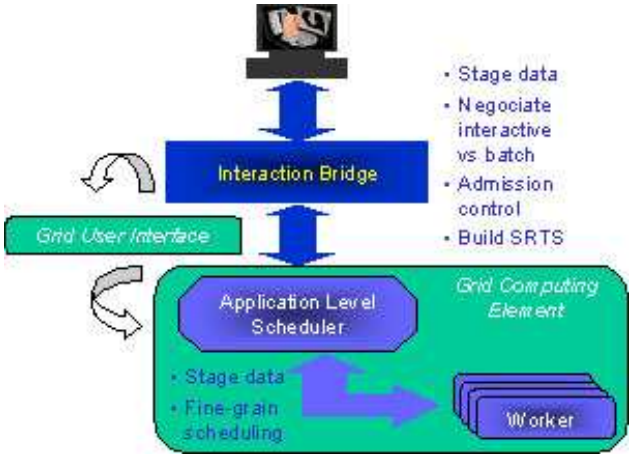

Figure 4. gPTM3D architecture

4. gPTM3D

4.1. Interactive Volume Reconstruction

PTM3D [9] is a fully featured DICOM images analyzer developed at LIMSI. PTM3D transfers, archives and visualizes DICOM-encoded data; besides moving independently along the usual three axes, the user is able to view the cross-section of the DICOM image along an arbitrary plane and to move it. PTM3D provides computer-aided generation of three-dimensional representations from CT, MRI, PET-scan, or echography 3D data. A reconstructed volume (organ, tumor) is displayed inside the 3D view. The reconstruction also provides the volume measurement, required for therapeutic decisions. The system currently runs on standard PC computers and it is used on line in radiology centres. Clinical motivation for grid-enabled volume reconstruction is described in [4].

The first step in grid-enabling PTM3D (gPTM3D) is to speedup compute-intensive tasks, such as the volume reconstruction of the whole body used in percutaneous nephrolithotomy planning. The volume reconstruction module has been coupled with EGEE with the following results:

- the overall response time is compatible with user requirements (less than 2 minutes), while the sequential time on a 3GHz, 2MB memory PC is typically 20 minutes.
- the local interaction scheme (stop, restart, improve the segmentation) remains strictly unmodified.

This first step has implemented fine grain parallelism and data-flow execution on top on a large scale and file-oriented grid system. The architecture based on Application Level Scheduler/Worker agents shown in fig 4 is fully functional on EGEE. The Interaction Bridge (IB) acts as a proxy in-between the PTM3D workstation, which is not EGEE-enabled, and the EGEE world. When opening an interactive session, the PTM3D workstation connects to the IB; in turn, the IB launches a scheduler and a set of workers on an EGEE node, through fully standard requests to an EGEE User Interface; a stream is established between the scheduler and the PTM3D front-end through the IB. When

the actual volume reconstruction is required, the scheduler receives contours; the Scheduler/Worker agents follow a pull model, each worker computing one slice of the reconstructed volume at a time, and sending it back to the scheduler, which forwards them to IB from where they finally reach the front-end. The next step will be to implement a scheme where the IB and the scheduler cooperate to respectively define and enforce a soft real-time schedule.

User-level scheduling has been proposed in many other contexts, and a case for it has been made in the AppleS [1] project. In a production grid framework, the Dirac [10] project has proposed a permanent grid overlay where scheduling agents pull work from a central dispatching component. Our work differs from Dirac in two respects: first, the scheduling and execution agents are launched just as any EGEE job, and are thus subject to all regulations related to sharing: typically, they are SDJ, thus will be aborted if they exceed the limits of this type of jobs.Moreover, they work in connected mode, more like glogin-based applications [8].

4.2. Grid-enabling Image Exploration

In the previous section, the grid was used only as a provider of computing power, while the data were located on the front-end. Sharing data is a well-known need for algorithmic research, but this is true for clinical research as well. We have started the process of extending PTM3D toward accepting remote data access. The integration of gPTM3D with the Medical Data Management scheme presented in another paper is the final goal. However, at the present time, we consider a most restrictive scheme, which uses the internal format of PTM3D images, where the slices are bundled in a 3D file (bdi and bdg formats). In this context, the main issues are the access latency. The ongoing work targets adaptation to the user activity, mainly trough interactive selection of the image resolution and the region of interest.

5. An architecture for grid interactivity

The scheme described in the previous sections virtualizes the resources at the coarse grain of batch versus short deadline jobs. An open issue is scheduling across SDJ. Consider for instance a portal, where many users ask for a continuous stream of execution of SDJ. This situation can be modelled with the so-called (period, slice) model used in soft real-time scheduling, where a fraction (slice) of each period of time should be allocated to each user in order to keep happy. To be coherent with a software architecture based on VOs, global regulation of SDJ should be left to the implementation of sharing policies (ultimately implemented by site schedulers). However, it is the responsibility of the provider of a particular service to arbitrate between its users. The Interaction Bridge described in the previous section is the adequate location for this arbitration. Figure 5 describes the resulting architecture.

Acknowledgements

This work was partially funded by ACI Masses de Données AGIR. We thank Fabrizio Pacini of EGEE JRA1 for his help with the Glue schema specification.

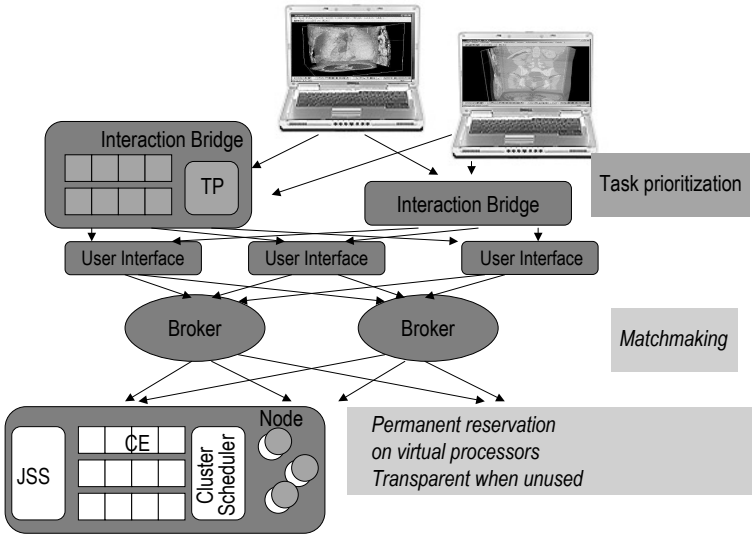

Figure 5. A two-level scheduling architecture

References

[1] H. Casanova, G. Obertelli, F. Berman, and R. Wolski. The AppLeS parameter sweep template: user-level middleware for the grid. In *Procs 2000 ACM/IEEE conference on Supercomputing (CDROM)*, 2000.

[2] A. Bavier et al. Operating Systems Support for Planetary-Scale Network Services. In *Procs. 1st Symp. on Networked System Design and Implementation (NSDI Š04)*, 2004.

[3] C. Germain, V. Breton, P. Clarysse, Y. Gaudeau, T. Glatard, E. Jeannot, Y. Legré, C. Loomis, J. Montagnat, J-M Moureaux, A. Osorio, and X. Pennec et R. Texier. Grid-enabling medical image analysis. *Journal of Clinical Monitoring and Computing*, 19(4-5):339–349, 2005. Extended version of the BioGrid 2005 paper.

[4] C. Germain, R. Texier, and A. Osorio. Exploration of Medical Images on the Grid. *Methods of Information in Medecine*, 44(2):227–232, 2005.

[5] B. Lin and P. A. Dinda. VSched: Mixing Batch And Interactive Virtual Machines Using Periodic Real-time Scheduling. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, 2005.

[6] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor : A hunter of idle workstations. In *8th International Conference on Distributed Computing Systems*, pages 104–111. IEEE Computer Society Press, 1988.

[7] L. G. Roberts. Beyond Moore's law: Internet growth trends. *Computer*, 3(1):117–119, 2000.

[8] H. Rosmanith and D. Kranzlmuller. glogin - A Multifunctional, Interactive Tunnel into the Grid. In *Procs 5th IEEE/ACM Int. Workshop on Grid Computing (GRID'04)*, 2004.

[9] V. Servois, A. Osorio, and J. Atif et al. A new pc based software for prostatic 3d segmentation and volume measurement. application to permanent prostate brachytherapy (ppb) evaluation using ct and mr images fusion. InfoRAD 2002 - RSNA'02, 2002.

[10] A. Tsaregorodtsev, V. Garonne, and I. Stokes-Rees. DIRAC: A Scalable Lightweight Architecture for High Throughput Computing. In *Procs 5th IEEE/ACM Int. Workshop on Grid*

Computing (GRID'04), 2004.
[11] SDJ WG wiki site. http://egee-na4.ct.infn.it/wiki/index.php/ShortJobs.