Enabling Grids for GATE Monte-Carlo Radiation Therapy Simulations with the GATE-Lab

Sorina Camarasu-Pop[1], Tristan Glatard[1], Hugues Benoit-Cattin[1] and David Sarrut[2]
[1]*Université de Lyon, CREATIS; CNRS UMR5220; Inserm U1044; INSA-Lyon; Université Lyon 1*
[2]*Université de Lyon, CREATIS; CNRS UMR5220; Inserm U1044; INSA-Lyon; Université Lyon 1; Centre Léon Bérard*
France

1. Introduction

Among radiation therapy simulation methods, Monte-Carlo approaches are known to be the most accurate but they are heavy to use because of their computing time. Nowadays they can be accelerated with the help of the ever-increasing computing power and distributed resources mutualised in clusters, clouds or grids (Montagnat et al. (2005)).
Grid infrastructures are used both for experimental and production purposes. They have been designed to support data and computing requirements for a large spectrum of applications from various scientific domains. Nevertheless they are not yet at a "plug and play" phase that would allow applications to be easily and efficiently deployed on the existing infrastructure. Efforts are required to achieve reliable and efficient execution on the grid for new applications and to provide user-friendly execution environments to end-users.
This chapter presents a solution for reliable, user-friendly and fast execution of GATE (Jan et al. (2004)) on a grid. Developed within the OpenGate[1] international collaboration, GATE is a Monte-Carlo based open-source software for nuclear medicine simulations, especially for TEP and SPECT imaging, and for radiation therapy applications. The solution proposed here enables transparent grid execution from a user-friendly interface. The application is parallelized automatically and a dynamic partitioning can also be used for further reducing the execution time and improving the robustness to job failures (Camarasu-Pop et al. (2010)). This chapter will discuss in more detail the implemented solution by describing the user interface, the system architecture, as well as the dynamic optimization strategy. Usage and performance results illustrating the system adoption will then be presented. To conclude with, it will discuss the lessons learned in building the system.

2. Related work

Different parallelization methods for Monte-Carlo simulations have been proposed for execution on distributed environments. This section will present related work on

[1] http://opengatecollaboration.healthgrid.org/

parallelization methods, on challenges raised by distributed environments (e.g. production grids), as well as on end user interfaces allowing to run such parallelized applications on grids.

2.1 Parallelization methods

The simulation in a particle tracking Monte-Carlo system consists in the successive stochastic tracking through matter of a large number of individual particles. Each particle has an initial set of properties (type, location, direction, energy, etc) and its interaction with matter is determined according to realistic interaction probabilities and angular distributions. Accurate results require the simulation of a large number of particles and physical interactions can also produce other particles that must be tracked. Therefore, typical radiation therapy simulations can take several days or even weeks to complete on a single computer. However, they can be easily parallelized on distributed systems. Instead of sequentially simulating a large number (up to several billions) of particles, smaller groups (bags) of particles can be simulated independently and eventually merged. This process is valid only if the sub-simulations are statistically independent, which in GATE is guaranteed by using a special random number generator as explained by Reuillon et al. (2008).

Among existing parallelization methods, the simplest and most commonly used is particle parallelism. In this case the geometry information is replicated on each processor and particles are distributed between available processors as described in Maigne et al. (2004). The authors present first results obtained by running GATE in parallel on multiple processors of the DataGrid[2] project. In this example the number of particles simulated on each of the N processors represents a fraction P/N of the total of P particles. This static distribution of particles often underexploits resources when the simulation is executed on heterogenous platforms like computer grids. Indeed, as particles are evenly distributed among tasks, tasks running on fast resources complete before other ones and these resources may remain idle if the scheduler cannot assign them other tasks (e.g. at the end of the simulation). Moreover, it may happen that a task is allocated to a slow resource towards the end of the simulation, thus slowing down the completion of the whole application (Cirne et al. (2007)).

A possible solution to this problem is the dynamic distribution and/or reassignment of particles to available processors during runtime. Dynamic partitioning is proposed in Galyuk et al. (2002) and in Procassini et al. (2005) for spatial parallelism. Spatial parallelism involves splitting the geometry into domains and then assigning a specific domain to one processor. This method is usually needed when the problem geometry has a significant size so that one processor does not have enough memory to store all particles/zones. Spatial parallelism may introduce load imbalance between processors, as spatial domains will require different amounts of computational work. In Procassini et al. (2005), a dynamic load balancing algorithm distributes the available processors to the spatial domains. Communications are generated between processors to transmit changes from the last state. In Galyuk et al. (2002), the parallelization is done using an MPI implementation and is based on a semaphore principle under distributed memory conditions. These two implementations are therefore cluster oriented and are not adapted for grid usage where communications between processors are very costly.

Camarasu-Pop et al. (2010) propose a dynamic partitioning algorithm for GATE radiotherapy simulations using pilot jobs. Statistically independent simulations are launched on available resources and they keep on running until the desired number of particles is reached.

[2] http://eu-datagrid.web.cern.ch

The simulation is no longer split into sub-simulations; instead each computing resource contributes to the whole simulation until it is completed. The pilot-job master periodically sums up the number of simulated particles and sends stop signals to tasks when needed. Due to the randomness involved in Monte-Carlo simulations, this is possible with no communication between tasks.

2.2 Applications execution on grids

Significant work has been put in parallelizing Monte-Carlo algorithms and running them on distributed systems. Nevertheless, they are rarely accessible to physicians and researchers because of the recurrent errors and complexity introduced by grid infrastructures. Grid middleware has to be coupled with other tools to ensure quality of service (QoS) (Tan et al. (2010)), to facilitate application porting (Camarasu-Pop et al. (2008)) and to offer high-level execution interfaces (Olabarriaga et al. (2010)).

Failures are recurrent on large grid infrastructures like the EGI grid, where the success rate is usualy of 80 to 85% (Jacq et al. (2008)). When splitting one Monte-Carlo simulation into sub-tasks, it is important that all of them complete successfully in order to retrieve the final result. Therefore, failed tasks must be resubmitted, further slowing down the application completion. Frameworks such as pilot jobs have been introduced during the last years (Ahn et al. (2008); Bagnasco et al. (2008); Kacsuk et al. (2008); Maeno (2008); Moscicki (2003b); Sfiligoi (2008); Tsaregorodtsev et al. (2008)) and are now extensively used in order to improve QoS and cope with the recurrent errors and high latencies caused by the grid heterogeneity.

Application porting and reusability have also been facilitated with the help of workflow technology, as discussed in Deelman et al. (2005); Glatard et al. (2008); Kacsuk & Sipos (2005); Maheshwari et al. (2009); Oinn et al. (2004). Engines now allow the execution of workflow applications on various grid middleware in a generic way.

On top of these tools, high-level execution interfaces are essential for end users with no specific grid knowledge. This problem has been acknowledged by different communities who already developed a certain number of grid portals such as Genius[3] or GridSphere[4]. A few communities have customized these tools for their needs or even developed their own specific portals. This is also the case for some of the Monte-Carlo applications running on clusters or grids as detailed hereafter.

2.3 End user interfaces/portals for Monte-Carlo applications

Early portals for Monte-Carlo applications were designed to provide physicists convenient access to grid tools and services, as presented by Engh et al. (2003) and Compostella et al. (2007). These portals provided functionalities like job submission and monitoring, as well as results browsing. Nevertheless, they were setup at a time when grid reliability was poor and end users still had to handle significant debugging processes.

Recent portals integrate increasingly powerful functionalities and are often targeting specialized communities.

The E-IMRT platform described in José Carlos Mouriño Gallego (2007); Pena et al. (2009) offers radiotherapists a set of algorithms to optimize and validate radiotherapy treatments. It has three main components, namely characterization of linear accelerators, radiotherapy treatment planning optimization and verification and data repository. The IMRT (Intensity-Modulated Radiation Therapy) treatment verification is based on a Monte-Carlo

[3] https://genius.ct.infn.it/

[4] http://www.gridsphere.org

method and as such is an excellent candidate to be executed on a grid. These services are accessible through an user-friendly web page, where the implementation of the verification and optimization algorithms, as well as the complexity of the distributed infrastructure on which they run, are hidden to the user. The E-IMRT platform became one of the 25 Grid Business experiments (BEs) from the BEinGRID (Business Experiments in Grid) project. BEinEIMRT (M.G. Bugeiro (2009)) provides on-demand e-Health computational services (like tumor detection and radiotherapy planning) to Health organisations like clinics and hospitals. All services are provided by the Centro de Supercomputación de Galicia (CESGA), which is the customary provider. When CESGA is under peak demand, external resources are added for a limited time period[5]. This process is transparent for the end user, reducing execution time and increasing the QoS.
The HOPE (HOspital Platform for E-health) platform[6] was designed to enable GATE simulations in a grid environment. Its user-friendly interface was built taking into account feedback from healthcare professionals. Like the E-IMRT platform, it targeted especially physicians and medical physicists.
The PARTNER (PARticle Training Network for European Radiotherapy) project aims at reinforcing research and training professionals in the rapidly emerging field of hadron therapy. The project started at the end of 2008 for a period of 4 years. It has reviewed[7] existing ICT (Information and Communications Technology) based medical collaborative infrastructures and by the end of the project it will build a grid testbed allowing the training of a new generation of researchers.
Researchers' requirements are different from those of the clinicians and medical physicists who are targeted by the E-IMRT and HOPE platforms. These platforms were mainly designed to perform the same (limited) set of (standard/validated) applications to several data: for example computing IMRT treatment plans on different patient data. In contrast, researchers need to be able to easily test simulations with different sets of parameters, to perform computing intensive simulations that lead to large phase-spaces or to study the design of new imaging devices, such the ones that will be developed to monitor the dose deposit in hadrontherapy situations thanks to the outgoing particles produced by nuclear interactions.

3. The GATE-Lab

The GATE-Lab targets mainly researchers, both in academic and R&D industrial environments. Indeed, GATE is designed for a wide range of purposes in the fields of PET, SPECT or CT imaging, and radiation therapy, including hadrontherapy. The end user in our case is not the medical physicist sending a treatment plan to be computed, but the researcher who provides a file of macros describing a simulation. The research community behind the GATE simulator needs to execute various GATE simulations easily and rapidly without worrying about the computing resources provided by the European Grid Infrastructure. This section will describe more into detail the GATE-Lab, its architecture and advanced features. To begin with, it will shortly present the underlying grid infrastructure.

[5] http://www.ogf.org/documents/GFD.167.pdf
[6] http://eu-acgt.org/news/newsletters/summer-2009/single-article/archive/2009/july/article/hope-hospital-platform-for-e-health.html
[7] https://espace.cern.ch/partnersite/workspace/faust/Shared%20Documents/FaustinRoman.WP22.D1.pdf

3.1 Grid environment

The European Grid Infrastructure (EGI[8]) is currently the largest production grid worldwide providing more than 100,000 CPUs and several Petabytes of storage. This distributed computing infrastructure was built by projects (DataGrid, EGEE-I, -II and -III) spanning from 2002 to 2010 and is now supported in collaboration with National Grid Initiatives (NGIs). EGI is used on a daily basis by thousands of scientists organized in over 200 Virtual Organizations. It uses the gLite middleware (Laure et al. (2006)), which provides high level services for scheduling and running computational jobs, as well as for data and grid infrastructure management.

A User Interface (UI) is the initial point of access to the grid from which the user can be authenticated and authorized to use the grid resources. From the UI the user can submit and cancel jobs, query their status and retrieve their output. These tasks are taken into account by a Workload Management System (WMS), which queues the user requests and dispatches them to the different computing centres available. The gateway to each computing centre is one or more Computing Element (CE) that will distribute the workload over the Worker Nodes (WN) i.e. the computing units available at this center. Files are stored on Storage Elements (SEs) and registered in the File Catalogs, which permit to locate files (or replicas) distributed on the grid. Data Management services are responsible for locating, replicating and accessing the data transparently.

3.2 Architecture

The GATE-Lab has been integrated in the porting and execution platform supporting grid applications in use at the Creatis laboratory. As illustrated in Fig. 1, the platform complies to a three-tier architecture composed of (i) the client which consists in the VBrowser (Olabarriaga et al. (2006)) offering a GUI for grid data management (ii) a lab server managing task submission, monitoring and error handling and (iii) the grid itself, externally administrated and accessible through gLite.

The user interacts with the GATE-Lab client, which is a VBrowser plugin designed specifically for GATE simulations and using the VBrowser as a GUI for grid data management. The GATE-Lab performs automatic parameter checking, input files bundling and uploading, simulation submission and history management. The lab server hosts the DIANE pilot-job framework (Moscicki (2003a)) and the MOTEUR workflow engine (Glatard et al. (2008)). MOTEUR generates grid tasks from the GATE workflow description and submits them to a DIANE master using a generic task manager. According to the number of waiting tasks, the agent controller submits pilot jobs to the grid Workload Management System (WMS). Once running, DIANE pilots download and execute tasks on the grid worker nodes (WN), periodically uploading standard output and standard error to the master.

3.2.1 The client

The GATE-Lab main tab allows for preparing a new simulation and launching it on the grid. The end user is asked for a name for the new simulation and for its main macro file. Starting from this main macro file, the client checks the simulation parameters, looks for all local inputs files, bundles them into an archive and uploads it to the grid. The end user is also asked for an estimation of the total CPU time of the simulation. Depending on this estimation, the simulation is split into a different number of tasks, so that the parallelization and therefore the

[8] https://www.egi.eu

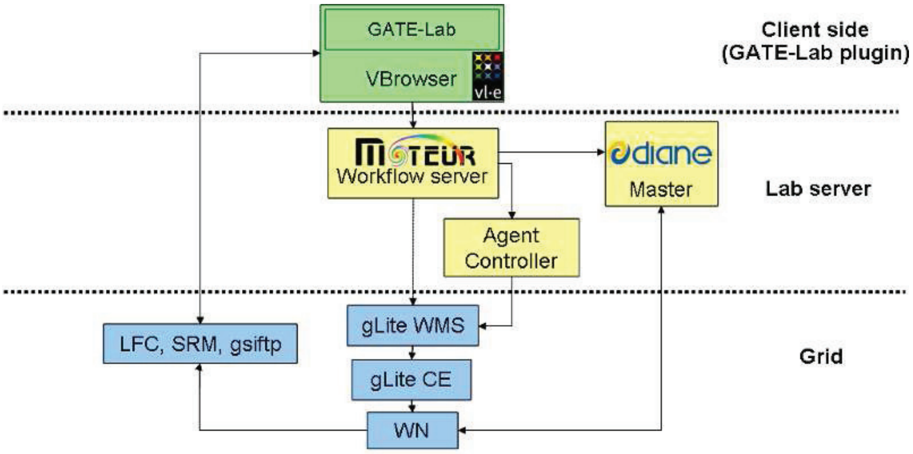
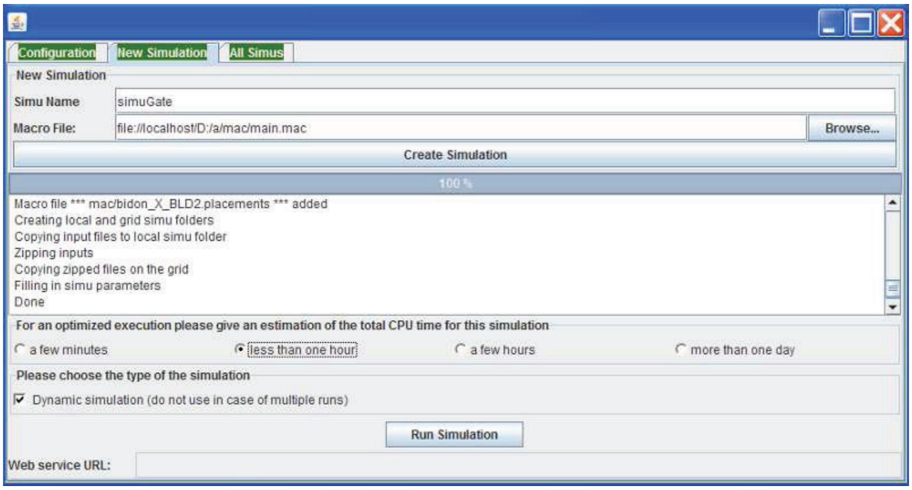


Fig. 1. Grid execution environment. On the client side, VBrowser and the GATE-Lab plugin offer GUI for data management and launching GATE simulations on the grid. The lab server hosts the MOTEUR workflow engine and DIANE pilot-job framework. MOTEUR enacts application workflow and submits tasks to the DIANE master offering pilot-job execution. The agent controller submits pilot jobs to the grid.

speedup should be improved. At this point, a Şone-clickŤ GATE simulation is made possible for the user who does not have to be aware of grid internals as shown in Fig 2.



New Simulation

Simu Name: simuGate

Macro File: file://localhost/D:/a/mac/main.mac

```
Macro file *** mac/bidon_X_BLD2.placements *** added
Creating local and grid simu folders
Copying input files to local simu folder
Zipping inputs
Copying zipped files on the grid
Filling in simu parameters
Done
```

For an optimized execution please give an estimation of the total CPU time for this simulation

a few minutes [✓ less than one hour] a few hours more than one day

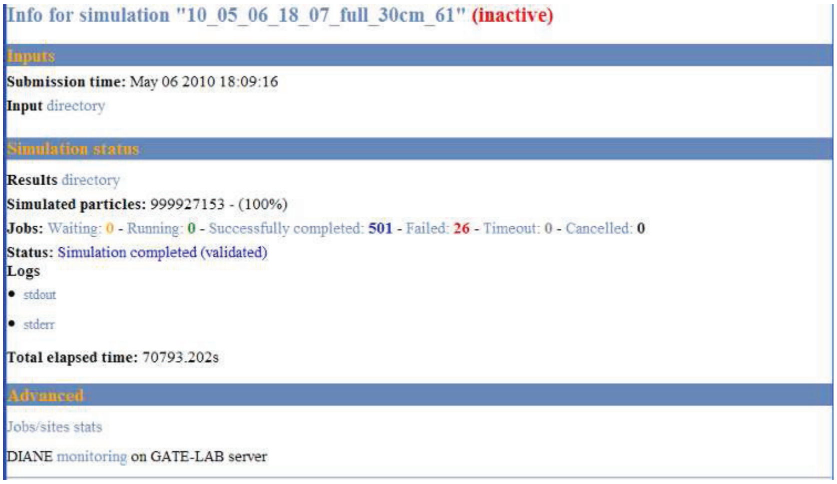
Please choose the type of the simulation

[✓] Dynamic simulation (do not use in case of multiple runs)

Web service URL:

Fig. 2. GATE-Lab - create new simulation. Starting from the main macro file, the client checks the simulation parameters, looks for all local inputs files, bundles them into an archive and uploads it to the grid. Running GATE simulations on the grid is possible with a few clicks.

The number of parallel tasks to submit to the grid is automatically determined based on the end user estimation of the total CPU time needed for the complete simulation. The total CPU time depends on the number of events, but also on their type. Therefore, it is difficult to



Info for simulation "10_05_06_18_07_full_30cm_61" (inactive)

Inputs

Submission time: May 06 2010 18:09:16
Input directory

Simulation status

Results directory
Simulated particles: 999927153 - (100%)
Jobs: Waiting: 0 - Running: 0 - Successfully completed: 501 - Failed: 26 - Timeout: 0 - Cancelled: 0
Status: Simulation completed (validated)
Logs

- stdout
- stderr

Total elapsed time: 70793.202s

Advanced

Jobs/sites stats
DIANE monitoring on GATE-LAB server

Fig. 3. GATE-Lab - simulation monitoring. Simulation status is monitored in real time and outputs are accessible with an URL.

automatically forecast and the user who submits the simulation is the best person to estimate it. Nevertheless, he or she does not need to have the grid knowledge allowing to choose the right number of tasks for the application parallelization. The GATE-Lab integrates a mapping between the time estimation given by the end user and the number of tasks to submit. At the moment, the mapping corresponds to a minimum of 5 jobs for short simulations (a few minutes) and goes to a maximum of 500 jobs for simulations lasting more than a day.
The GATE-Lab implements two parallelization methods. The dynamic method gives the best results and is used by default. Nevertheless it cannot be used with all types of GATE simulations like for example simulations with multiple runs. In the Geant4 toolkit (Allison et al. (2006)) on which Gate is based, a "run" is the largest unit of simulation and consists of a sequence of "events" (particles history). Simulations dealing with time varying geometry perform successive runs. Therefore the end user has the possibility to use the standard static splitting method by simply deactivating the corresponding box (see Fig 2). More detail on these splitting methods will be given in section 3.3.1.
The GATE-Lab client has two more tabs: one for simulation history and the other for configuration purposes. The simulation history tab gives the list of links to all simulations launched by the user. The end user can thus easily retrieve his previous simulations, browse their web page or delete them. The configuration tab is important for allowing the user to choose certain parameters like for example the GATE release. Indeed, this was a strong requirement of the research community who needs to test new releases or switch to older ones for tests.
Once the GATE simulation is launched, its status is monitored and the end user can follow it in real time. The monitoring interface is presented in Fig 3. It is based on a PHP script available to the end user either within the VBrowser or from a simple web browser.
When the total number of events has been reached, a merging job is automatically launched. It merges all outputs generated in parallel and produces a final result easily accessible to the end-user through the VBrowser.

3.2.2 The Lab server

The lab server hosts the DIANE pilot-jobs framework and the MOTEUR workflow engine. MOTEUR generates grid tasks from the GATE workflow description and submits them to a DIANE master using a generic task manager. Pilot-job frameworks provide late-task binding to resources. Thus, tasks are no longer pushed to computing resources but generic pilots are submitted. Once running, pilots connect back to a central pool, fetching tasks when available and dying otherwise. Pilots are submitted on the grid with standard gLite command-line tools. This architecture is presented in Figure 4.
The GATE executable is deployed "on the fly" by the pilot-jobs. In order to do so, a GATE "release" is prepared (once and for all) for each new GATE version and stored on grid SEs. The release contains the GATE executable (compiled on a compatible architecture) and the necessary shared libraries. For each GATE task, pilots download the GATE release as well as all input files previously stored on grid SEs. To limit file transfers, a cache system has been implemented in pilots. Pilots cache the release and the inputs on the WN in case their next task requires the same files.
The agent controller monitors the number of pilots and decides of their submission according to the number of tasks created by MOTEUR. This mechanism, together with the indication on the total CPU time given by the end user, allows to modulate the number of resources asked/used depending on the size of the simulation.

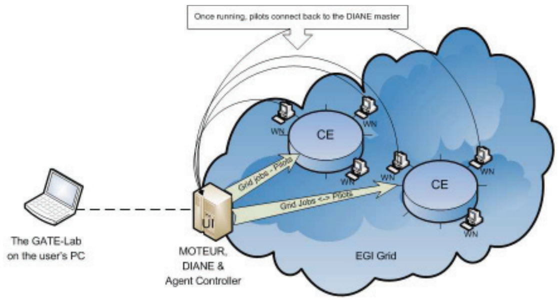


Fig. 4. DIANE pilot-jobs framework integrated into the Creatis grid execution environment.

Error handling is implemented by MOTEUR by resubmitting tasks up to a maximal number of times when they fail. Faulty pilots (i.e. pilots running a task that fails) are also removed and resubmitted if necessary (if no other 'free' pilots are available).
Authentication is based on X509 grid certificates. Short lived (maximum of 24h) grid proxies are generated with the VBrowser based on the user certificate. For GATE simulations longer than 24h, users have to renew their proxy. To ensure basic security, user credentials are delegated to the MOTEUR server, which starts a new DIANE master for every user.

3.3 Advanced features

The GATE-Lab is more than a friendly user interface for running GATE simulation on the grid. It integrates a set of advanced features which enhance its performance.

3.3.1Dynamic parallelization

As presented in Section 2.1, static parallelization methods tend to lead to poor scheduling mainly because of the heterogeneity of the grid resources. The GATE-Lab offers two parallelization options that have proved to be well adapted to the grid usage (Camarasu-Pop

et al. (2010)). Both methods use pilot jobs, which bring considerable advantages in terms of latency reduction and fault-tolerance on the grid (Germain Renaud et al. (2008)).
The first method is an intermediate solution based on a dynamic distribution (and reassignment) of statically partitioned tasks. In this case, a simulation is statically divided into a large number of tasks (of a convenient granularity) that are dynamically distributed to available resources. This method gives reasonably good results but underexploits the available resources, especially towards the end of the simulation.
Therefore a new dynamic task partitioning strategy was proposed and implemented to balance the number of particles that each resource has to simulate depending on its performance. The proposed dynamic task partitioning consists in a Şdo-whileŤ algorithm with no initial splitting. Each computing resource contributes to the whole simulation until it is completed. As summarized in section 4 and detailed in Camarasu-Pop et al. (2010), the dynamic parallelization brings significant performance improvement, the makespan being on average twice smaller than with the previous method. Indeed a significant amount of time is saved on the completion of the last tasks. However this method cannot be used for all GATE simulations at the moment. GATE simulations with multiple runs for example need to respect additional criteria when parallelized. For these simulations the first method has to be used. Therefore, both methods are available in the GATE-Lab.

3.3.2 Stop and merge

GATE simulations can be very long and end-users may want to stop them beforehand if they consider that the number of already simulated particles is large enough and that the needed uncertainty level has been reached. Moreover, as previously mentioned, it is the last part of the simulation that is usually the slowest one. Therefore a "stop and merge" feature has been added to the GATE-Lab. It allows end users to stop the simulation at will and trigger the merging phase. This stop feature is possible due to an extension of the GATE code which is also used for the dynamic parallelisation. The GATE code was adjusted to handle stop signals during simulation. This add-on allows the application to pause regularly and check if the stop signal has been received. If this is the case, GATE saves its results and stops. Once all parallel results are saved and uploaded on the grid storage elements, the merging phase is launched to produce the final result.
This "stop and merge" feature raises a new challenge. Between the moment when the end-user sends the stop signal and the moment when the partial results are uploaded on the storage elements of the grid, job failures can happen. This will result in having less simulated particles than the user believed to have when he decided to stop the simulation. In a regular situation, failed jobs are resubmitted in order to ensure 100% complete results. In the situation of a stop and merge demand, the end-user is probably not willing to wait for job resubmission and wants the final result right away if the number of lost particles is not too important. Currently jobs that fail after the stop signal are not resubmitted. Nevertheless, an interaction with the end-user will have to be considered. The user will be presented the real status of the simulation and will have to make a second decision based on this new, more accurate information.

3.3.3 Incremental merge

Merging the partial results achieved in parallel is essential for the completion of the GATE simulation. This is a critical and delicate task because of its data intensive character. Indeed, partial results are stored on geographically distributed (and sometimes unreliable) grid

storage elements. The merging phase has the difficult task of downloading all partial results (currently up to 500 tarballs of a few tens of MB each) on one machine. Transfers are often long and may sometimes fail, thus endangering the success of the whole simulation as can be seen in the Results section.
Therefore, to improve the overall success rate and the performance of the merging phase, an incremental merge was implemented and is currently evaluated. In general, the chances to successfully retrieve a partial result are maximal shortly after it has been uploaded. The incremental merge does not wait for all results to be ready. It is launched along with the simulation as soon as a given number N (for example 5) of partial results have been produced. It merges these results into an intermediate result that will be in turn merged with the next partial or intermediate results. Thus, partial results are downloaded soon after they are produced, limiting the risk that they become unavailable. Moreover, a file that is temporarily unavailable has more chances to be retrieved by one of the successive instances of the incremental merge.

4. Results

The system has been running since 2009 but has been more intensely used from April 1st to September 17th 2010 by two radiotherapy researchers, exploiting the resources supporting the biomed Virtual Organization of the EGEE grid. These gather 200+ CEs spread over 50 countries and are continuously shared by some 100 users. A detailed snapshot of the infrastructure usage is available on accounting portals[9]. A total of 197 simulations were submitted among which 89 were completely successful, 42 were "half-successful" (i.e. only some jobs completed) and 66 completely failed (no job completed). Tables 1 and 2 summarize performance results.
The elapsed time is the time perceived by the user, i.e., the duration between the launching of the simulation and the completion of the last task. The CPU time is a cumulative value on all the tasks of the simulation. It is an indication of the time that the simulation would have required on a single machine representative of the average grid node. The speed-up factor is a ratio between the CPU time and the elapsed time. The data/computing ratio is computed as the cumulative data transfer time (download + upload) divided by the cumulative CPU time. It gives an indication on the efficiency of the grid deployment. The error rate is computed as the ratio between the number of failed tasks (total tasks - successful tasks) and the total number of tasks.
Among the successful simulations (Table 1), a total of 8.4 CPU years were consumed and 27,798 grid tasks were submitted. Sixteen (16) simulations were slowed down by the grid execution. These were small test 6-job simulations, which suggests that a better handling of test simulations (e.g. by local execution) would be useful. The speed-up of the remaining simulations ranges from 3.12 to 146 with an overall value of 47. Although this value seems of little significance given the amount of available resources on the grid, it has to be noticed that this performance was obtained in production, on a shared platform. It is a sustainable level of performance that can be delivered to a group of users on a daily basis. As expected, there is some positive correlation between the speed-up and the number of submitted jobs, which suggests that the execution time choices made by the users through the GATE-Lab interface make sense.

[9] http://www3.egee.cesga.es/

	Elapsed	CPU	Speed -up	Data/ Computing	Total tasks	Error rate
min	397s	26s	0	0	6	0
max	3.9 days	174 days	146	16.65	815	0.61
average	0.73 days	34 days	-	-	312	-
all simulations	65 days	8.4 years	47	0.01	27798	0.09

Table 1. Performance figures of successful simulations (45% of all GATE-Lab simulations).

	Elapsed	CPU	Speed -up	Data/ Computing	Total tasks	Error rate
min	261s	26s	0	0	2	0
max	2 days	200 days	369	9.07	977	1
average	0.69 days	54 days	-	-	373	-
all simulations	31 days	6.5 years	77	0.01	16417	0.33

Table 2. Performance figures of half-successful simulations (some completed jobs, 21% of all GATE-Lab simulations).

The overall data/computing ratio is 1%, which is a very good value on wide scale platforms such as the EGI. Again it shows that the number of jobs was properly chosen. High data/computing values are observed for 6-job simulations though and few outliers are also seen for 502- or 256-job simulations. These probably come from temporary network issues or overloads on the storage system.
The average job error ratio is 9%, which is a good value on the EGI. Most of the errors come from data transfer issues. Other sources of errors include pilot-master communication problems (lost pilots) and application issues coming from the configuration of the worker nodes.
Half-successful simulations (Table 2) did not complete due to some repeated failures in simulation or merge tasks. Some of these simulations could still be exploited thanks to manual recovery, in particular in case of merge failures. As expected, the overall error ratio (33%) is far greater than for successful simulations. The speed-up is also higher than for successful simulations due to the lack of merge task.
Failed simulations represented 26 days of elapsed time and a total of 2074 grid tasks. These failures are due either to critical problems of the infrastructure (GATE-Lab or grid) or to user mistakes. In the future, mechanisms must be envisaged to detect these issues earlier in the simulation.
Figure 6 shows an example of task flow obtained with static respectively dynamic partitioning of a GATE simulation. The dynamic partitioning obviously leads to a better scheduling than the static. Thanks to the lack of resubmission and better resource exploitation, no slowdown is observed towards the end of the simulation for the dynamic approach. Figure 5 shows the simulation completion along time for the dynamic (red curves) and static (black curves) parallelization modes. During the first phase of the simulation (up to 400,000 particles), the dynamic parallelization shows a better throughput (number of simulated particles per seconds) but the overall time gain remains modest. More importantly, the dynamic parallelization dramatically improves the performance of the simulation during its last phase (from 400,000 simulated particles to 450,000). This highlights the benefit obtained using our load-balancing method.

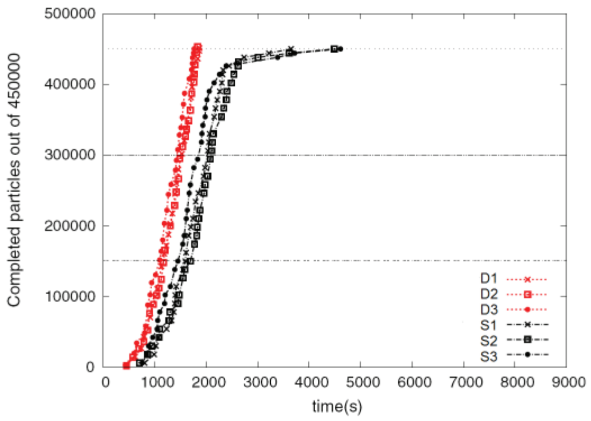


Fig. 5. GATE simulation completion using the static VS the dynamic parallelization approach. When using the static method the last tasks are considerably slowing down the simulation (figure extracted from Camarasu-Pop et al. (2010)).

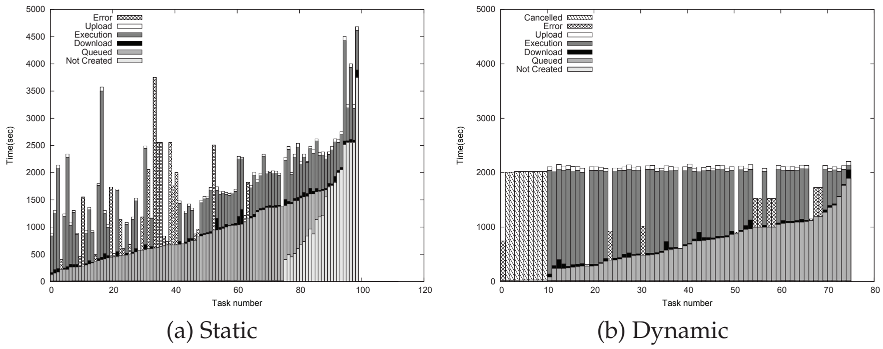


Fig. 6. Example of task flow obtained with (a) Static & (b) Dynamic partitioning of a GATE simulation on EGI with 75 submitted pilots. Available resources are all exploited until the end of the simulation. Errors are compensated without any pilot resubmissions. The dynamic scheduling obviously outperforms the one obtained with static partitioning (figure extracted from Camarasu-Pop et al. (2010)).

5. Discussion / Lessons learned

The GATE-Lab has been created to serve the GATE (radiotherapy) research community. The main challenge of the project was to make a sustainable service from a research code running on a distributed environment. The difficulty of this task resided not only in the grid complexity and its rendering transparent to the research community, but also in integrating in the system an evolving research code.

Concerning the execution of GATE on grid nodes, we learned that it was important to have "application administrators" among GATE researchers. They are at the interface between grid and GATE experts ensuring thus a good communication between the two communities. As an opensource software developed by the international OpenGATE collaboration, GATE is always evolving and new releases are available periodically. In order to run on the grid,

these new releases must be compiled and packaged properly for the grid environment. GATE application administrators are in charge of preparing these releases with the help of the tools provided by grid experts.
Regarding the grid environment, we grew aware of the importance of using pilot jobs. Pilot jobs bring dramatic improvement w.r.t. default gLite, both in terms of reliability and performance. gLite standard submission never manages to complete 100% of the simulation, the success rate being of 75% on average. Workflows are also very useful for application porting and reusability. A middleware-independent workflow description of the application simplifies migration to other execution frameworks. In our case, we can easily extend GATE submission to platforms where pilots are not an option or migrate to other pilot frameworks.
In spite of all the efforts put into the optimization and automation of the GATE-Lab, user support is needed regularly. Most of the time support is needed for the merging phase and other unexpected error handling like for example temporary failures of grid services.
Last but not least, the merging phase represents a real challenge because of its data intensive character and its performance needs to be improved. From our experience we know that this merging activity and related problems are not specific to GATE; they are common to many applications parallelized on the grid.

6. Conclusion

This chapter presented a solution for reliable, user-friendly and fast execution of GATE (an open-source software for nuclear medicine and radiation therapy simulations) on the grid by using the GATE-Lab. The GATE-Lab targets researchers wishing to easily execute GATE with different sets of parameters which do not necessarily correspond to the treatment plans used in clinic.
The GATE-Lab plugin is build upon the porting and execution platform supporting grid applications in use at the Creatis laboratory. The plugin has been specifically designed to answer researchers' requirements. The current release is the result of a mature project started two years ago and which evolved according to the researchers' feedback. Advanced features like the stop on merge and the incremental merge have been added at the researchers' request. During the last 5 months the system provided an average speed-up of almost 50. Although not outstanding, this is a good level of performance since it was delivered on a shared infrastructure, when several other users from our lab and others were also running heavy computations. Most of the simulations were successful but it was noticed that the support for test and prototyping simulations was not very efficient and that user mistakes leading to numerous task failures should be better detected.
The dynamic parallelization is a powerful feature that has been proposed and implemented for GATE. It is a general Monte-Carlo load-balancing method that could greatly speed-up other kinds of Monte-Carlo simulations.
The GATE-Lab is already intensively used by the (non-clinical) radiation therapy researchers at Creatis. As future work we plan to make it available for other research teams worldwide. Within the hGate project we also plan to extend the GATE-Lab to other computing platforms (e.g. local clusters).

7. Acknowledgments

This work is co-funded by the French national research agency (ANR), hGATE project under contract number ANR-09-COSI-004-01. It also falls into the scope of the scientific topics of the

French National Grid Institute (IdG). The authors would like to thank the site administrators of the European Grid Initiative and the GGUS support for their work and Fabrice Bellet for his help with application compiling and customization for the grid. We also would like to thank Piter T. de Boer and Spiros Koulouzis (NIKHEF) for the support with the VBrowser, Jakub T. Mosciki (CERN) for the help with DIANE and the Modalis team (CNRS, I3S) for the MOTEUR developments.

8. References

Ahn, S., Namgyu, K., Seehoon, L., Soonwook, H., Dukyun, N., Koblitz, B., Breton, V. & Sangyong, H. (2008). Improvement of Task Retrieval Performance Using AMGA in a Large-Scale Virtual Screening, *NCM'08*, pp. 456–463.

Allison, J., Amako, K., Apostolakis, J., Araujo, H., Dubois, P., Asai, M., Barrand, G., Capra, R., Chauvie, S., Chytracek, R., Cirrone, G., Cooperman, G., Cosmo, G., Cuttone, G., Daquino, G., Donszelmann, M., Dressel, M., Folger, G., Foppiano, F., Generowicz, J., Grichine, V., Guatelli, S., Gumplinger, P., Heikkinen, A., Hrivnacova, I., Howard, A., Incerti, S., Ivanchenko, V., Johnson, T., Jones, F., Koi, T., Kokoulin, R., Kossov, M., Kurashige, H., Lara, V., Larsson, S., Lei, F., Link, O., Longo, F., Maire, M., Mantero, A., Mascialino, B., Mclaren, I., Lorenzo, P., Minamimoto, K., Murakami, K., Nieminen, P., Pandola, L., Parlati, S., Peralta, L., Perl, J., Pfeiffer, A., Pia, M., Ribon, A., Rodrigues, P., Russo, G., Sadilov, S., Santin, G., Sasaki, T., Smith, D., Starkov, N., Tanaka, S., Tcherniaev, E., Tome, B., Trindade, A., Truscott, P., Urban, L., Verderi, M., Walkden, A., Wellisch, J., Williams, D., Wright, D. & Yoshida, H. (2006). Geant4 Developments and Applications, *IEEE Transactions on Nuclear Science* 53: 270–278.

Bagnasco, S., Betev, L., Buncic, P., Carminati, F., Cirstoiu, C., Grigoras, C., Hayrapetyan, A., Harutyunyan, A., Peters, A. J. & Saiz, P. (2008). Alien: Alice environment on the grid, *Journal of Physics: Conference Series* 119(6).

Camarasu-Pop, S., Benoit-Cattin, H., Guigues, L., Clarysse, P., Bernard, O. & Friboulet, D. (2008). Towards a virtual radiological platform based on a grid infrastructure, *Medical imaging on grids: achievements and perspectives (MICCAI Grid Workshop)*, New York, USA, pp. 85–95.

Camarasu-Pop, S., Glatard, T., Mościcki, J., Benoit-Cattin, H. & Sarrut, D. (2010). Dynamic partitioning of GATE Monte-Carlo simulations on EGEE, *Journal of Grid Computing* .

Cirne, W., Brasileiro, F., Paranhos, D., Goes, L. & Voorsluys, W. (2007). On the efficacy, efficiency and emergent behavior of task replication in large distributed systems, *Parallel Computing* 33: 213–234.

Compostella, G., Lucchesi, D., Griso, S. P. & Sfiligoi, I. (2007). CDF Monte Carlo Production on LCG Grid via LcgCAF Portal, *E-SCIENCE '07: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing*, IEEE Computer Society, Washington, DC, USA, pp. 11–16.

Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G. B., Good, J., Laity, A., Jacob, J. C. & Katz, D. S. (2005). Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems, *Scientific Programming Journal* 13(3): 219–237.

Engh, D., Smallen, S., Gieraltowski, J., Fang, L., Gardner, R., Gannon, D. & Bramley, R. (2003). GRAPPA: Grid access portal for physics applications, *CoRR* cs.DC/0306133.

Galyuk, Y. P., Memnonov, V., Zhuravleva, S. E. & Zolotarev, V. I. (2002). Grid technology with dynamic load balancing for Monte Carlo simulations, *PARA '02: Proceedings of the 6th*

International Conference on Applied Parallel Computing Advanced Scientific Computing, Springer-Verlag, London, UK, pp. 515–520.

Germain Renaud, C., Loomis, C., Moscicki, J. & Texier, R. (2008). Scheduling for Responsive Grids, *Journal of Grid Computing* 6: 15–27.

Glatard, T., Montagnat, J., Lingrand, D. & Pennec, X. (2008). Flexible and efficient workflow deployement of data-intensive applications on grids with MOTEUR, *International Journal of High Performance Computing Applications (IJHPCA)* 22(3): 347–360.

Jacq, N., Salzemann, J., Jacq, F., Legré, Y., Medernach, E., Montagnat, J., Maass, A., Reichstadt, M., Schwichtenberg, H., Sridhar, M., Kasam, V., Zimmermann, M., Hofmann, M. & Breton, V. (2008). Grid enabled virtual screening against malaria, *Journal of Grid Computing* 6: 29–43.

Jan, S., Santin, G., Strul, D., Staelens, S., AssiÉ, K., Autret, D., Avner, S., Barbier, R., Bardiès, M., Bloomfield, P. M., Brasse, D., Breton, V., Bruyndonckx, P., Buvat, I., Chatziioannou, A. F., Choi, Y., Chung, Y. H., Comtat, C., Donnarieix, D., Ferrer, L., Glick, S. J., Groiselle, C. J., Guez, D., Honore, P. F., Kerhoas-Cavata, S., Kirov, A. S., Kohli, V., Koole, M., Krieguer, M., van der Laan, D. J., Lamare, F., Largeron, G., Lartizien, C., Lazaro, D., Maas, M. C., Maigne, L., Mayet, F., Melot, F., Merheb, C., Pennacchio, E., Perez, J., Pietrzyk, U., Rannou, F. R., Rey, M., Schaart, D. R., Schmidtlein, C. R., Simon, L., Song, T. Y., Vieira, J. M., Visvikis, D., de Walle, R. V., Wieërs, E. & Morel, C. (2004). GATE: a simulation toolkit for PET and SPECT., *Phys Med Biol* 49(19): 4543–4561.

José Carlos Mouriño Gallego, Andrés Gómez C. F. S. F. J. G. C. D. A. R. S. J. P. G. F. G. R. D. G. C. M. P. C. (2007). *1st Iberian Grid Infrastructure Conference Proceedings (IBERGRID)*.

Kacsuk, P., Farkas, Z. & Fedak, G. (2008). Towards making BOINC and EGEE interoperable, *4th eScience Conference*, Indianapolis, pp. 478–484.

Kacsuk, P. & Sipos, G. (2005). Multi-Grid, Multi-User Workflows in the P-GRADE Grid Portal, *Journal of Grid Computing (JGC)* 3(3-4): 221 – 238.

Laure, E., Fisher, S., Frohner, A., Grandi, C., Kunszt, P., Krenek, A., Mulmo, O., Pacini, F., Prelz, F., White, J., Barroso, M., Buncic, P., Byrom, R., Cornwall, L., Craig, M., Meglio, A. D., Djaoui, A., Giacomini, F., Hahkala, J., Hemmer, F., Hicks, S., Edlund, A., Maraschini, A., Middleton, R., Sgaravatto, M., Steenbakkers, M., Walk, J. & Wilson, A. (2006). Programming the Grid with gLite, *Computational Methods in Science and Technology* 12(1): 33–45.

Maeno, T. (2008). Panda: distributed production and distributed analysis system for atlas, *Journal of Physics: Conference Series* 119(6): 062036 (4pp).

Maheshwari, K., Missier, P., Goble, C. & Montagnat, J. (2009). Medical Image Processing Workflow Support on the EGEE Grid with Taverna, *Intl Symposium on Computer Based Medical Systems(CBMS'09)*, IEEE.

Maigne, L., Hill, D., Calvat, P., Breton, V., Lazaro, D., Reuillon, R., Legré, Y. & Donnarieix, D. (2004). Parallelization of Monte-Carlo simulations and submission to a grid environment, *Parallel Processing Letters HealthGRID 2004*, Vol. 14, Clermont-Ferrand France, pp. 177–196.

M.G. Bugeiro, J.C. MouriÃśo, A. G. C. V. E. H. I. L. y. D. R. (2009). Integration of slas with gridway in beineimrt project, *3rd Iberian Grid Infrastructure Conference*.

Montagnat, J., Breton, V. & Magnin, I. (2005). Partitionning medical image databases for content-based queries on a grid, *Methods of Information in Medicine (MIM)* 44(2): 154–160.

Moscicki, J. T. (2003a). Diane - distributed analysis environment for grid-enabled simulation and analysis of physics data, *Nuclear Science Symposium Conference Record, 2003 IEEE*, Vol. 3, pp. 1617–1620 Vol.3.

Moscicki, J. T. (2003b). Distributed analysis environment for HEP and interdisciplinary applications, *Nuclear Instruments and Methods in Physics Research A* 502: 426âĂŞ429.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M. R., Wipat, A. & Li, P. (2004). Taverna: A tool for the composition and enactment of bioinformatics workflows, *Bioinformatics journal* 17(20): 3045–3054.

Olabarriaga, S., de Boer, P. T., Maheshwari, K., Belloum, A., Snel, J., Nederveen, A. & Bouwhuis, M. (eds) (2006). *Virtual Lab for fMRI: Bridging the Usability Gap*, IEEE, Amsterdam.

Olabarriaga, S., Glatard, T. & de Boer, P. T. (2010). A virtual laboratory for medical image analysis, *IEEE Transactions on Information Technology In Biomedicine (TITB)* .

Pena, J., González-Castaño, D. M., Gomez, F., Gago-Arias, A., González-Castaño, F. J., Rodríguez-Silva, D. A., González, D., Pombar, M., Sánchez, M., Portas, B. C., Gómez, A. & Mouriño, C. (2009). E-IMRT: a web platform for the verification and optimization of radiation treatment plans., *in* R. Magjarevic, J. H. Nagel, O. Dössel & W. C. Schlegel (eds), *World Congress on Medical Physics and Biomedical Engineering, September 7 - 12, 2009, Munich, Germany*, Vol. 25/1 of *IFMBE Proceedings*, Springer Berlin Heidelberg, pp. 511–514.

Procassini, R., O'Brien, M. & Taylor, J. (2005). Load Balancing of Parallel Monte Carlo Transport Calculations, *Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications*, Palais des Papes, Avignon, Fra.

Reuillon, R., Hill, D., Gouinaud, C., El Bitar, Z., Breton, V. & Buvat, I. (2008). Monte Carlo Simulation With The GATE Software Using Grid Computing, *Proceedings of NOTERE 2008 8ème Conférence Internationale sur les NOuvelles TEchnologies de la REpartition, NOTERE 2008*, Lyon France.

Sfiligoi, I. (2008). glideinWMS - A generic pilot-based workload management system, *Journal of Physics: Conference Series* 119(6): 062044 (9pp).

Tan, W.-J., Ching, C. T. M., Camarasu-Pop, S., Calvat, P. & Glatard, T. (2010). Two experiments with application-level quality of service on the egee grid, *GMAC '10: Proceeding of the 2nd workshop on Grids meets autonomic computing*, ACM, New York, NY, USA, pp. 11–20.

Tsaregorodtsev, A., Bargiotti, M., Brook, N., Ramo, A. C., Castellani, G., Charpentier, P., Cioffi, C., Closier, J., Diaz, R. G., Kuznetsov, G., Li, Y. Y., Nandakumar, R., Paterson, S., Santinelli, R., Smith, A. C., Miguelez, M. S. & Jimenez, S. G. (2008). Dirac: a community grid solution, *Journal of Physics: Conference Series* 119(6): 062048 (12pp).