High-performance computing for Monte Carlo radiotherapy calculations

BY P. DOWNES\(^1\), G. YAIKHOM\(^1\), J. P. GIDDY\(^1\), D. W. WALKER\(^1,\)*, E. SPEZI\(^2\) AND D. G. LEWIS\(^2\)

1 School of Computer Science, Cardiff University, Roath, Cardiff CF24 3AA, UK
2 Department of Medical Physics, Velindre Cancer Centre, Whitchurch, Cardiff CF14 2TL, UK

We report on the RTGrid project, which investigates approaches for using high-performance computing infrastructures, such as the grid, in order to reduce the turnaround time of Monte Carlo (MC) simulation-based radiotherapy treatment planning. The main aim of this project is to render accurate dose calculations using MC simulations clinically feasible. To this end, we have successfully implemented and deployed the RTGrid distributed simulation framework for MC dose calculations. In this paper, we present the main experimental findings.

**Keywords:** radiotherapy; Monte Carlo; simulation; high-performance computing; National Grid Service; Condor

1. Introduction

The aim of radiotherapy treatment planning is to determine how a prescribed dosage of X-rays is delivered to the tumour while reducing harmful effects on the surrounding healthy tissues and organs. This is done using software that strikes a balance between the calculation time and the accuracy of the dosage produced by the calculation. Owing to its high accuracy, radiotherapy treatment planning using Monte Carlo (MC) simulations is preferred to traditional approaches. However, MC approaches are computation intensive, and typically take days to calculate dosage for a single patient. Their full clinical usage is therefore hampered by the long calculation times, in addition to the significant computational resources that are required to make this approach practical for routine treatment planning.

In the RTGrid project, we are investigating the use of distributed computing resources in order to reduce the run-time of MC simulations to no more than a few hours, with the aim of making this approach clinically deployable. Of particular interest is the radiotherapy treatment of anatomically inhomogeneous areas of the body such as the lungs, head and neck, for which the current planning systems are not sufficiently accurate, and where the use of MC

* Author for correspondence (david.w.walker@cs.cardiff.ac.uk).

One contribution of 16 to a Theme Issue ‘Crossing boundaries: computational science, e-Science and global e-Infrastructure I. Selected papers from the UK e-Science All Hands Meeting 2008’.
techniques is likely to have a significant impact on the determination of more accurate dosage. Furthermore, this will also increase our understanding of dose–response relationships (Mohan 1997).

Radiotherapy dose calculation methods have evolved significantly over three decades, from the introduction of computerized planning using ‘beam libraries’ of data measured in a water medium, through the use of convolution/superposition methods based on pre-computed elemental dose deposition ‘kernels’ (Ahnesjö & Aspradakis 1999), to the early stages of deployment of event-by-event MC simulation of radiation transport (Verhaegen & Seuntjens 2003). Dose calculations using MC simulations have two main advantages over alternative techniques. (i) They allow accurate characterization of the radiation source and its interactions everywhere within the complex anatomy of the patient (Arnfield et al. 2000). Hence, the dose in and around non-water-equivalent regions (such as the lungs, head and neck) can be calculated more accurately; therefore, increasing the reliability with which dose–response relationships are determined. And (ii) increasing complexity of the problem to be modelled does not significantly affect the complexity of the MC model itself (Fishman 1996).

Radiotherapy treatment planning systems are generally developed by commercial vendors and are licensed for clinical use by the relevant national or international regulatory authorities. The next generation of such systems is likely to incorporate some form of MC simulation of radiation transport, but with certain variance reduction techniques (Kawrakow & Fippel 2000) including approximations made in the dose calculation models to speed up the process. An alternative approach would be to use more exact simulations, together with parallel and distributed computing methods to reduce calculation times. Whatever the dose calculation method involved in the clinical treatment plan, it is a requirement on the part of the clinical scientist(s) responsible for the production and quality control of those plans to ensure that each is verified using independent dose calculations (Mayles et al. 1999). Given the increasing complexity of treatment techniques and the variety of calculation algorithms becoming available, it is most important to be able to rely on the most accurate dose calculation platforms for plan verification.

2. The RTGrid simulation framework

In a hospital, clinical scientists responsible for treatment planning using MC simulations may not have the required expertise to effectively use distributed computing resources. Hence, in order to make any MC simulation framework clinically usable, the framework must furnish a simple graphical user interface that provides secure access to the underlying software and hardware components of the framework. In the RTGrid system, we achieve this by using a service-oriented architecture where the presentation layer is decoupled from the complicated details of the execution layer. Since the details of this architecture and the associated database have been described elsewhere (Yaikhom et al. 2008), we will not reproduce these details here.

(a) The experiment manager

When an experiment is started within the RTGrid simulation framework, it activates a background process in the RTGrid system, known as the experiment
The main task of this process is to monitor changes in the RTGrid database that are related to service requests received from the users, and to carry out the necessary actions in response to these changes. When a new experiment is started, for instance, the experiment master spawns a new child process, known as the experiment manager. This experiment manager controls every aspect of the new experiment, and is eventually terminated when the experiment has concluded.

Every new experiment in the RTGrid system is derived from a simulation template, known as the profile. Profiles contain simulation data, and several executable scripts that must be run at specific phases of the simulation. These phases are described using the Job Submission Description Language (JSDL) (Anjomshoaa et al. 2005). JSDL is a good choice for our purposes as it provides the attributes we require and is being increasingly supported in batch queuing systems, meaning less translation is required. To run a given simulation experiment, the experiment manager first runs the pre.sh script specified in the profile that was used to derive the experiment. This pre-processing script populates the experiment directory with the required input files. Where input files must differ between jobs, each of the files is given a different filename, which also includes a job identifier. Within the JSDL job description, it is possible to specify place holders for job identifiers. If such place holders are found in the JSDL document, the experiment manager replaces these with the correct job identifiers. Finally, the experiment manager uses the architecture and the operating system parameters specified in the JSDL document to determine the type of resource that has been requested for running the experiment. When a matching resource is found, the experiment manager submits all of the jobs to this resource. It then monitors the status of each of these jobs and updates the database accordingly to reflect any changes. Once all of the jobs have completed, it concludes the simulation by running the post.sh script, thus collating all of the output files generated by the various jobs. Finally, the job is marked complete.

(b) The RTGrid resource broker

To submit jobs to a distributed resource, the experiment manager must first load a job management module that provides the necessary interfaces for carrying out communications with the resource. Additionally, this module also performs the necessary actions to set up the correct environment on the remote node, and to run the commands specified in the profile scripts. The RTGrid system currently supports the following three resource brokers: (i) Condor (Litzkow et al. 1988), (ii) Globus (Foster 2005), and (iii) GridWay (Huedo et al. 2005). Furthermore, we have also implemented an in-house broker: the RTGrid resource broker (RTB).

The key feature of RTB is that it allows the RTGrid system to submit jobs using specific details supplied by the owner of the experiment. With the third-party brokers, we were faced with the problem of not being able to submit jobs using credentials of the experiment owners. For instance, with a Condor pool, all of the jobs must be submitted using credentials that belong to the owner of the RTGrid system. Similarly, GridWay does not allow job submissions using separate credentials that belong to each of the different users. On the other hand, although Globus allows user-specific credentials, submissions are restricted to a single resource. In other words, if we are to use these resource brokers, all of
the jobs must be submitted using credentials that belong to the maintainer of the RTGrid system. Since this is not ideal for security reasons, we resolve this complication through RTB.

RTB resolves another issue concerning resource brokers where a centralized scheduling mechanism chooses the best resources by analysing real-time resource information. Our experience with Nimrod/G (Abramson et al. 2000), a grid-enabled parametric job farming system, has shown that this information is often incorrect, and always out of date, as lightly loaded resources are continuously interrupted by multiple resource brokers. Furthermore, in a well-maintained grid, we would expect all of the resources to be actively loaded with some job. In such a situation, resource brokers that use such a scheduling mechanism are reduced to ineffective random allocation. Contrary to this, however, what works well in such situations is the Condor glide-in approach (Thain et al. 2005). A glide-in is a self-installing Condor execution service that may be submitted to a remote resource. When a glide-in is initialized, it joins the specified Condor pool and then becomes available for accepting new jobs. Since Condor schedules a job to a resource only when the glide-in at that resource has been initialized, jobs are never delayed because a bad scheduling decision was made in choosing a particular resource. However, Condor does not provide a framework for submitting glide-ins on demand, as installing glide-ins requires several network ports to be opened on the remote resource; which leads to the problem of glide-ins remaining active without running any job, thereby excluding runnable jobs.

RTB provides a solution to this problem by implementing an approach where glide-ins and jobs are dispatched using a single secure network port that is opened on all of the remote resources. When RTB finds a collection of pending jobs, these jobs are not allocated immediately to the various resources available. Instead, RTB submits a number of agents that provide a batch queue on each of the resources that are deemed suitable. If an agent is initialized successfully, it contacts RTB over the secure connection, which uses the secure network port. In response to this, RTB allocates one of the jobs from the collection. If an agent submitted to a remote resource fails to initialize, RTB will not allocate any job to that resource, thus reducing the inconvenience to other users of that resource.

RTB is written in Erlang (Armstrong 1997) and is built around the Yaws web server (http://yaws.hyber.org/). We chose the Yaws server as it has been shown to scale well with many connections (http://www.sics.se/~joe/apachevsyaws.html). RTB consists of four Erlang agent processes: (i) information process, (ii) starter process, (iii) status process, and (iv) cancel process. The information process monitors resource information, such as architecture and usage policy. The starter process monitors the number of jobs submitted by each user. When the number of pending jobs for a user is more than the number of agents on a resource, the starter process starts a new agent using the Globus ToolKit. The process looks for any restrictions in the information available for a cluster (such as the GLUCEPOLICYMAXTOTALJOBS attribute of the GLUE schema, http://glueschema.forge.cnaf.infn.it/) and does not start a new agent if there are more than a few (typically four) agents submitted but not yet initialized. This ensures that busy clusters get fewer agents and do not appear to be swamped by a single user. Again, experience with Nimrod indicates that, while automated tools submit jobs to a cluster assuming the local batch scheduler will interleave users’ jobs fairly, there is often an unwritten policy not to swamp...
the batch scheduler, and local users watching the queue protest vigorously to the system administrator about an apparently selfish user submitting a large number of jobs.

The status process monitors submitted jobs. In addition to carrying out basic housekeeping tasks (record maintenance and clean-up of the Globus job), it also detects when an agent exits without reporting the job status. Normally, an agent should send a status report with success, or failure, details before exiting. If an agent exits without sending such a status report, the status process replaces this job in the pending queue so that it is rescheduled to a new agent (unless the job was cancelled). The cancel process checks for jobs that have been cancelled by the user, and uses Globus mechanisms to terminate the agent running the job. When the agent terminates, the status process detects this and removes the job.

3. Experimentation and discussion

MC simulations are computationally expensive, as they simulate particles and their interactions. In this section, we will discuss experimental results concerning the use of distributed computational resources for carrying out MC simulation-based radiotherapy treatment planning through the RTGrid system.

For carrying out the simulations, we used EGSNRC (Kawrakow 2000) and BEAMnrc (Rogers et al. 1995). BEAMnrc is a set of source and geometry routines specifically designed to facilitate the modelling of medical linear accelerators (or linacs), whereas EGSNRC handles the radiation transport of the particles being simulated. For an in-depth discussion on the various simulation packages, see Rogers (2006).

To run the simulations, we used distributed resources available at the Cardiff University Condor Pool and the UK e-Science National Grid Service (NGS) (http://www.ngs.ac.uk/access.html). The Cardiff Condor Pool (Osborne & Hardisty 2006) contains between 250 and 1200 Windows machines at any given time. All the machines run Windows XP SP2 and are upgraded on a 3-year cycle. The processors on the machines are mostly Pentium-4s, and the amount of memory varies between machines. Two hundred of the Condor machines are dedicated, whereas the remaining are transient as they belong to employees within the university network. The version of Condor running on the pool machines is predominantly 6.8.6, with some machines running versions 6.8.8 and 6.6.11. The grid resources used in this work were the NGS core sites at Leeds, Manchester, Oxford and Rutherford Appleton Laboratory. The nodes on these sites have AMD Opteron processors with at least 2 GB of memory per core, and they run the Linux operating system.

(a) Timing study

For the timing study, we chose a straightforward simulation experiment that produced consistent results. The main outcome of such a simulation was a three-dimensional matrix containing the amount of energy (radiation dosage) deposited in each voxel. Since the simulation was based on MC techniques, the experiment also produced a three-dimensional matrix of the statistical uncertainty associated with the radiation dosage. We chose the number of
particles to be simulated so that there would be less than 2 per cent uncertainty
in the resulting dose matrix. Consequently, the number of particles simulated
also determined the execution times of these simulations.

The timing study involved repeatedly submitting an experiment to the same
resource, until we were sure of the reproducibility of the timing results. In order
to study the overall execution time and throughput, the experiments were
divided into sets of 20, 40, 80, 120 and 160 jobs. Each of the jobs in an experiment
was created by uniformly distributing all of the particles to be simulated across
all of the jobs that must be created. This led to nearly equivalent execution times
(on the same machine) for each of the jobs within an experiment, and kept the
overall computation required for each experiment almost constant. Furthermore,
all of the experiments were run sequentially, one after another, so that they
would not compete with each other for resources. This timing study analysed the
effects of heterogeneity to simulation time, and, in the case of the Cardiff Condor
Pool, the effects of using non-dedicated resources. This study also analysed the
speed-up due to the use of the RTGrid system, as compared with using a
single workstation.

The overall speed-up was calculated using the following equation:

\[
\text{speedup} = \frac{T_{\text{single}}}{T_{\text{pre}} + \max(T_{\text{queue}} + T_{\text{execute}}) + T_{\text{post}}},
\]

where \(T_{\text{single}}\) is the single processor execute time; \(T_{\text{pre}}\) is the pre-processing time;
\(T_{\text{queue}}\) is the queueing time; \(T_{\text{execute}}\) is the execute time; and \(T_{\text{post}}\) is the post-
processing time. On a Windows workstation with a single Pentium-4 processor,
our benchmark simulation experiment took 82.2 hours to run, which we take as
the value of \(T_{\text{single}}\). The value of \(\max(T_{\text{queue}} + T_{\text{execute}})\) is the time taken from
the submission of the first job to the completion of the final job. In fact, the
denominator of equation (3.1) gives the total simulation time.

The efficiency was calculated using the following equation:

\[
\text{efficiency} = \frac{\text{speedup}}{\text{NumProcs}},
\]

where speed-up is the value calculated in equation (3.1), and NumProcs gives the
number of jobs in the experiment (each job is assigned to a different resource).

The results of the timing study conducted on the Cardiff Condor Pool and
the NGS are shown in tables 1 and 2, respectively. We have broken down
the execution time according to the three main phases of the simulation: (i) the
pre-processing, (ii) the execution, and (iii) the post-processing. Within parentheses, we give the percentage of the execution time of each phase against the total simulation time. Each of the simulation experiments was repeated at least five times, thus cancelling out the effects of unpredictable load variation on the resources due to other users. From these timing results, we took the minimum execution times for each of the phases, since these were the ones least affected by load variations.

In the above tables, we can see that the execution times for pre-processing and post-processing phases increased as the number of jobs in the experiment increased. On the other hand, it is evident that, as the number of jobs in an experiment increased, the overall run-time of the simulation decreased, therefore increasing the speed-up. It is also clear that, compared with the pre- and post-processing stages, the actual simulation-related calculations contributed the most to the overall execution time of the experiment; in all of the experiments, the execution phase contributed at least 97 per cent of the overall simulation time. In table 2, note the efficiency of the experiment with 20 jobs, which had a value of 1.05. We believe that this was partly owing to the more powerful processing power available through the NGS. Furthermore, one must note the fact that, due to the non-dedicated nature of the nodes, Condor pools must cope with performance-degrading events such as pre-emption, suspension and eviction. Based on these analyses, we believe that the NGS fares better than Condor pools as a resource pool for treatment planning using MC simulations.

It should be noted that clusters based on the WINDOWS operating system have some limitations when used for high-performance computing. Some of these issues include large overheads due to the large size of the operating system kernel, and general security weaknesses concerning the sharing of resources. Furthermore, due to the lack of robust checkpointing facilities on this platform, valuable computation time may be lost when it is used with Condor. Despite these weaknesses, however, clusters based on the WINDOWS operating system have the advantage of ubiquity—that is, there are usually several WINDOWS workstations sitting idle in companies and universities, which make using these resources cost-effective.

Concerning the use of NGS resources, however, it is important to note that the simulation times of some of the experiments on the NGS were greatly affected by an irreproducible file transfer problem, which randomly truncated files during transfer. In such cases, the experiment manager would retry submitting the jobs...
for three more times before marking them as failed. Our investigation of the problem suggested that the file transfer error was connected to the Globus GASS file transfer code. Unfortunately, however, due to the unpredictable occurrence of this error, we have been unable to debug the code to solve this issue. Hence, we carried out our experiments through retries when such an error was detected.

To further alleviate the file transfer problem, the RTB agent was configured to send output files up to three times using two different methods: first, using a standard Globus–URL–copy transfer from the execution node, then uploading the file to the RTB using the secure communication link between RTB and the agent, and finally, attempting to carry out a Globus–URL–copy transfer from the execution nodes. In the second method, the transfer can cope with premature expiration of the grid proxy certificates on the remote node. Returning to table 2, we can see that the speed-up and efficiency for the experiments with 80, 120 and 160 jobs were severely degraded by the burden of carrying out retries until such a time as the file transfers concluded without any problem. This can be understood if we consider the amount of time that was spent transferring files between the RTGrid system and the remote resources. For a typical simulation, it is necessary for the RTB to transfer data and executable files that easily reach 11 MB. Furthermore, each of the jobs produce output files that could easily reach 25 MB. Of course, these file transfer times have been included in the calculation of the simulation times.

(b) Treatment plan simulations

In order to analyse the advantages of using distributed computational resources, we also carried out treatment planning experiments. For this, we used two types of treatment plans. The first treatment plan was a four-field conformal treatment plan for a tumour on the chest wall of a patient. This plan required four radiation fields that had to be delivered from four different orientations around the patient. The second treatment plan was an intensity-modulated radiation therapy (IMRT) treatment plan. This plan was slightly more complex than the four-field treatment plan, because it used 10 fields of radiation, and each field was broken down into a number of segments so that each of these segments had a different radiation intensity. Owing to this intensity modulation, IMRT plans are highly conformal.

The execute and queue times for the four-field treatment plan simulations on the Cardiff Condor Pool and the NGS are shown in figures 1 and 2, respectively. Here, we define the execute time as the time spent on executing a job to completion. This accounts only for the actual CPU hours spent on the calculations. The queue time is defined as the amount of time a job had to wait after submission to the remote resource, before being executed. With non-dedicated resources, such as the Cardiff Condor Pool, the queue time includes the time spent restarting jobs that were prematurely evicted before completion. In figure 1, we can see these irregularities where the queue time is exceptionally long. Note, however, that the irregularities in the queue times on the NGS (as shown in figure 2) were primarily due to the file transfer problem that we discussed previously. In the two graphs, we can see that the execute times were consistent across all of the jobs. This is because, since some of the fields require more calculations than others, the number of particles was divided unevenly between all of the jobs.
The four-field conformal treatment plan took 159.45 CPU hours on our reference single-processor workstation. On the other hand, the same plan took 18.08 hours on the Cardiff Condor Pool. This gives us an effective speed-up of 8.82. Furthermore, the same plan took 7.17 hours on the NGS, thus giving us an effective speed-up of 22.24. Similarly, the IMRT plan took 3898.4 CPU hours on our reference single-processor workstation. However, the same plan took 262.32 hours on the Cardiff Condor Pool and 166.87 hours on the NGS. Thus, using the Cardiff Condor Pool and the NGS yielded respective speed-ups of 14.80 and 23.25. Based on these findings, we believe that using distributed computational resources through the RTGrid system can significantly reduce the dose calculation times of radiotherapy treatment planning using MC simulations.
4. Conclusions

In this paper, we discussed the RTGrid distributed simulation framework for carrying out MC simulations for radiotherapy treatment planning. We carried out various experiments using this framework, where the simulations were sent to non-dedicated resources on the Cardiff Condor pool and to dedicated resources on the NGS. We found that we could significantly reduce the dose calculation times by using these resources. However, we also learned that one of the main disadvantages of non-dedicated resources is wasted time due to the pre-emption, suspension and eviction of jobs. With regard to the NGS, performance was degraded by an irreproducible file transfer problem, which meant that experiments had to undergo several retries whenever the transferred files were found to be corrupt. Finally, we found that the overall calculation times were predominantly determined by the manner in which the simulation was split into independent jobs. Since several physical factors in the MC model affect this splitting, we must investigate these factors further in order to ascertain the best approaches. On an operational level, it is crucial that simulations using patient data take every precaution to ensure patient confidentiality.

We are indebted to the Engineering and Physical Sciences Research Council (EPSRC) for funding this research (grant no. EP/C538641/1), and to the NGS and the Cardiff Condor Pool for providing the computational resources. We would also like to thank the anonymous reviewers for their valuable comments and suggestions.

References


Phil. Trans. R. Soc. A (2009)


Phil. Trans. R. Soc. A (2009)