OptorSim v2.0
Installation and User Guide

David G. Cameron¹, Ruben Carvajal-Schiaffino², Jamie Ferguson¹, A. Paul Millar¹, Caitriana Nicholson¹, Kurt Stockinger³, Floriano Zini²

¹ University of Glasgow, Glasgow, G12 8QQ, Scotland
² ITC-irst, Via Sommarive 18, 38050 Povo (Trento), Italy
³ CERN, European Organization for Nuclear Research, 1211 Geneva, Switzerland

November 2, 2004

Abstract

OptorSim is a Data Grid simulator, written in Java, which was initially developed by Work Package 2 of the European Data Grid project. The goal of OptorSim is to allow experimentation with and evaluation of various replica optimisation strategies. Using a Grid configuration and a replica optimiser algorithm as input, OptorSim runs a number of Grid jobs on the simulated Grid. It also allows a user to visualise the performance of the algorithm. This document describes the installation and use of OptorSim.
Contents

1 Introduction 3

2 Installation and Running 4
   2.1 Software Dependencies .............................. 4
   2.2 Installing OpToroSim ............................... 4
   2.3 Using OpToroSim ................................. 5

3 Configuration Files 5
   3.1 The Grid Configuration File ....................... 5
   3.2 The Job Configuration File ....................... 7
   3.3 Simulation Parameters .......................... 8
       3.3.1 General Parameters ....................... 8
       3.3.2 Auction Parameters ...................... 11
       3.3.3 Background Network Traffic Parameters .. 11
       3.3.4 GUI Parameters .......................... 12
       3.3.5 Statistics Parameters ................... 12
       3.3.6 Time Model ............................... 12
   3.4 Simulating Background Network Traffic .......... 13

4 Using the Graphical User Interface 14
   4.1 Starting the GUI ................................. 14
   4.2 Components of the GUI .......................... 14
   4.3 Pausing, Restarting and Stopping the GUI .......... 17
   4.4 Saving Data .................................... 17

5 Outputs from OpToroSim 17

6 Future Work 19
1 Introduction

In a DataGrid a user typically submits a job to the Grid. Each job must access a set of files, possibly located at multiple Grid sites, and use computational resources to perform operations on the data contained within the files. The Grid must make scheduling and replication decisions for each job based on the current state of the Grid resources (location of data, network load, workload and features of computing nodes), in order to minimise the resource impact of all the jobs and allow them to run as fast as possible.

File replication is an important technique for improving the use of resources and the overall efficiency of the Grid. Any file on the Grid may have several copies, or replicas, in different locations on the Grid. The EDG Replica Optimisation Service (ROS) [3] was designed to optimise the creation and deletion of these replicas on Grid sites, thus helping to maximise the job throughput. At the heart of the ROS is a replica optimisation algorithm and OptorSim was initially developed to simulate and evaluate the behaviour of algorithms which could be incorporated into the ROS.

To access files, two types of reference may be used: a logical file name (LFN) and a physical file name (PFN). An LFN is an abstract reference to a file, which is independent both of where the file is stored and of how many replicas exist. A PFN refers to a specific replica of some LFN, located at some particular site. Each LFN will have one PFN for each replica on the Grid.

A job will usually request LFNs for data access. An LFN is resolved to get the list of PFNs, and optimisation then proceeds by finding which PFN can be accessed in the shortest time. Thus, the first optimisation criterion is to minimise access latencies of files.

Dynamic replication of files, i.e. the creation of new replicas and removal of old replicas as time progresses, is another important part of the optimisation process. The aim of dynamic replication is to position files in such a way that the network resources used to transfer files by future jobs will be reduced as much as possible. When optimised, the file replicas on the Grid will be distributed in such a way that the average job time for all jobs will be minimised. OptorSim has been developed to test both file access optimisation and dynamic replication strategies. The specific optimisation algorithms that have been tested are discussed in [4, 5].

OptorSim simulates a Grid to which jobs are submitted and which consists of a number of sites, each of which may provide computational and data-storage resources for submitted jobs. Each site consists of zero or more Computing Elements and zero or more Storage Elements. Computing Elements run jobs, which use the data in files stored on Storage Elements, and a single Resource Broker controls the scheduling of jobs. Sites without Storage or Computing Elements act as network nodes or routers.

For more details about the architecture of the model and the code implementation, we refer to [4], the JavaDoc available with the code distribution and the OptorSim website [2].
2 Installation and Running

These instructions describe how to install OptorSim. A tarball of the package can be downloaded from the OptorSim website:

http://cern.ch/edg-up2/optimization/optorsim.html

2.1 Software Dependencies

- Java(TM) 2 SDK: Version 1.4.0 or greater
- ant: Version 1.5.0 or greater (to re-build the code after any changes or to run functional tests)

2.2 Installing OptorSim

1. Start by unpacking the code from the tar ball:
   - On Unix systems using an archiving utility such as tar, e.g.
     $ tar xvfz optorsim-2.0.tar.gz
   - On MS Windows by using an archiving package such as WinZip (do not open the file directly from the browser but save it to disk first and then open it).

2. Go down into the optorsim directory:
   $ cd optorsim-2.0/

3. Set the user path to contain the bin/ directory which contains scripts for using OptorSim, e.g. for Bash shells
   $ export PATH=$PATH:$PWD/bin
   for csh/tcsh:
   $ set path = ($path $PWD/bin)

   OptorSim is now ready to run! For UNIX users, there is a suite of functional tests available which test the main components of the simulation to check everything is working properly. These can be run using ant and are best run on an unloaded computer:
   $ ant func-test
   If the source code is modified, it can be recompiled using ant:
   $ ant

   The JavaDoc can also be compiled with:
   $ ant doc

   The resulting html pages are in the doc/api directory.
2.3 Using OptorSim

- The main executable is called OptorSim and exists in two forms:
  - `OptorSim.sh` (bash shell)
  - `OptorSim.bat` (MS Windows)

Both executables are equivalent. In general, all OptorSim executable scripts can be found in the `bin` directory and this should be added to the user’s path, as explained in Section 2.2.

- When running OptorSim the user must have write permission to the current directory so OptorSim can write output files.

- OptorSim is run from the command line and takes either zero or one arguments. The optional argument is the parameters file (explained in the next section), and if no file is specified the default parameters file located at `examples/parameters.conf` is used.

Usage:

```
$ OptorSim.sh/bat [parameters file]
```

3 Configuration Files

There are several configuration files used to control various inputs to OptorSim. The Grid Configuration File describes the Grid topology and the content of each site; that is, the resources available and the network connections to other sites. The Job Configuration File contains information on the simulated files, jobs and the site policies for each site (the list of files each site will accept). The Simulation Parameters File contains various simulation parameters which the user can modify.

If the user wishes to simulate background network traffic, a Bandwidth Configuration File is needed, along with a number of data files to describe the simulated traffic (see Section 3.4).

Sample configuration files for OptorSim can be found in the `examples/` directory. When using the supplied examples, the grid and job configuration files must match up, for example:

```
simple_grid.conf and simple_job.conf.
```

3.1 The Grid Configuration File

This file describes the status of the resources of each site and the layout of the simulated Grid. The example configuration file shown in Figure 1 describes the Grid in Figure 2.

Each row in the configuration file is the information for one site:
0 1 100 000. 000. 100. 000. 000. 000. 000. 000. 000. 000.
0 1 100 000. 000. 100. 000. 000. 000. 000. 000. 000. 000.
5 1 100 100. 100. 000. 100. 100. 000. 000. 000. 000. 000.
0 1 100 000. 000. 100. 000. 000. 000. 000. 000. 100. 000. 000.
0 1 100 000. 000. 100. 000. 000. 000. 000. 000. 000. 000. 000.
0 1 500 000. 000. 000. 000. 000. 000. 100. 000. 000.
0 1 100 000. 000. 000. 100. 000. 000. 000. 100. 000. 000.
2 1 100 000. 000. 000. 000. 000. 100. 100. 000. 100. 100.
0 1 300 000. 000. 000. 000. 000. 000. 000. 000. 100. 000. 000.
0 1 300 000. 000. 000. 000. 000. 000. 000. 000. 100. 000. 000.

Figure 1: A simple example Grid configuration file.

```
0 1 100
5 1 100
0 1 100
0 1 100
0 1 500
0 1 100
2 1 100
0 1 300
0 1 300
```

![Grid Diagram](image)

Figure 2: A simple example Grid.

- The first column shows the number of worker nodes in the Computing Element (CE) at the site (if one is present).
- The second column is the number of Storage Elements (SEs) at the site.
- The third column is the size in MB of the SE, if one is present.
- The rest of the table is a site vs. site matrix giving the maximum bandwidth (i.e. link capacity) between each site in Mb/s. The matrix is diagonally symmetric. Entries along the diagonal are ignored since network bandwidth within a site is assumed to be infinite.

In the above example, the third row in the grid configuration file corresponds to site 2 in picture 2. This sites has a CE with 5 worker nodes and a SE with a size of 100 MB. It is linked to sites 0, 1, 3 and 4 and each bandwidth is 100 Mb/s.

---

1We assume there is a maximum of one CE per site.
2We assume this number is one or zero.
3.2 The Job Configuration File

This file contains information about the files present on the Grid and the jobs submitted during the simulation. An example is shown in Figure 3.

\begin{filetable}
File1 1000 1
File2 1000 2
File3 1000 3
File4 1000 4
File5 1000 5
File6 1000 101
File7 1000 102
File8 1000 103
File9 1000 104
\end

\begin{jobtable}
job1 File1 File2 File3 File4 File5
job2 File6 File7 File8 File9
\end

\begin{cescheduletable}
2 job1 job2
7 job1 job2
\end

\begin{jobselectionprobability}
job1 0.8
job2 0.2
\end

Figure 3: A simple example job configuration file.

- The first part (between \texttt{begin\{filetable\}} and the first \texttt{end}) is a list of file names, their sizes in MB and their unique numerical identifiers.
Each file name is a *Logical File Name* and is unique. The file IDs are used by a particular optimisation algorithm, namely the Economic Model optimisation algorithm[4, 5, 7].

- Each row in the second part of the job configuration file (between `begin{jobtable}` and the second `end`) contains a job name and the list of files required for the job.

- Each row in the third part of the job configuration file (between `begin{c sched ule table}` and the third `end`) gives the scheduling policy for each CE i.e., which jobs each CE is willing to run. In this example both the CEs at site 2 and 7 will run job1 and job2.

- The fourth part of the job configuration file, which is between `begin{jobselection probability}` and the fourth `end`, determines the frequency with which each job will be submitted to the Grid. In the example above job1 has a 80% probability of being submitted and job2 has a 20% chance.

**Note:** It is very important that the CEs that are assigned jobs match up with the CEs declared in the grid configuration file.

### 3.3 Simulation Parameters

There are many simulation parameters, which can be set by the user in a parameters file. The default parameters file is found in

```plaintext
examples/parameters.conf
```

and this is the one used if no option is specified at the command line when OptorSim is started. Following is an explanation of each parameter; in the parameters file, spaces in the parameter names should be replaced by dots.

#### 3.3.1 General Parameters

- **grid configuration file** - The configuration file to describe the Grid topology.

- **job configuration file** - The configuration file to describe the jobs.

- **bandwidth configuration file** - The configuration file to describe background network traffic (see Section 3.4).

---

3This algorithm uses a statistical model that evaluates which files are most likely to be requested by a CE in the future, based on the IDs of previously accessed files on this CE. File IDs are used to express file similarity: the closer two file IDs are, the higher is the expected relation between the contents of the corresponding files. Since files used in the same job ought to be more similar than files requested by different jobs, the IDs for files in the same job are numerically very close, while files used in different jobs have their IDs far apart.
**number jobs** - The number of jobs submitted during the simulation run.

**users** - Determines the pattern in which Grid users submit jobs to the Resource Broker. Options:

1. **Simple**: submit jobs at regular intervals until all jobs have been submitted. The interval is set by the *job delay* parameter (below).
2. **Random**: submit jobs at intervals which are uniformly random between zero and twice the *job delay*.
3. **CMS DC04**: based on job submission patterns from the CMS Data Challenge 2004 [1]. Submit jobs according to a Gaussian distribution centred around 3pm, with the total number submitted each day just under 700.

**scheduler** - Determines the scheduling strategy of the Resource Broker. Options:

1. **Random**: jobs are scheduled randomly to any Computing Element that will run the job.
2. **Queue Length**: schedules to the Computing Element with the shortest queue of waiting jobs. If two CEs have the same shortest queue length one of them is chosen at random.
3. **File access cost**: schedules to the Computing Element from which the cost to access all the files required for the job (in terms of network latencies) is the smallest. If two CEs have the same smallest access cost one of them is chosen at random.
4. **File access cost + job queue access cost**: scheduling is done using a combination of the access cost for the files and the access cost for all the jobs in the queue at each Computing Element.

**optimiser** - The optimisation algorithm to be used (see [4, 5] for more detail). Options:

1. **SimpleOptimiser**: no replication, files are accessed remotely;
2. **DeleteOldestFileOptimiser**: always replicates, deleting oldest file if there is no enough space for replication;
3. **DeleteLeastAccessedFileOptimiser**: always replicates, deleting least frequently accessed file if there is not enough space for replication;
4. **EcoModelOptimiser Binomial prediction function**: replicates when eco-model says yes, possibly deleting least valuable file according to Binomial prediction function;
5. **EcoModelOptimiser Zipf-like prediction function**: replicates when eco-model says yes, possibly deleting least valuable file according to Zipf-like prediction function.
dt - For optimisers (3), (4), and (5), the time period in milliseconds over which the past file access history is considered. When using the economic model no files will be deleted until this time has passed from the beginning of the simulation, to allow good access history statistics to build up.

access pattern generator - Determines the order in which files are accessed within a job (see [4, 5] for more details). Options:

1. **SequentialAccessGenerator**: Files are accessed in the order stated in the job configuration file.
2. **RandomAccessGenerator**: Files are accessed using a uniform random distribution;
3. **RandomWalkUnitaryAccessGenerator**: Files are accessed using a unitary random walk, starting from a file chosen using a uniform random distribution.
4. **RandomWalkGaussianAccessGenerator**: Files are accessed using a Gaussian random walk, starting from a file chosen using a uniform random distribution.
5. **RandomZipfAccessGenerator**: Files are accessed using a Zipf-like distribution\(^4\).

shape - The shape parameter $\alpha$ for the Zipf-like access pattern.

job set fraction - The fraction of files specified for the job that are accessed by a running job. For example, if a job is specified to require 10 files in the job configuration file:

- if job set fraction is set to 2.5 then the job will require 25 files every time it runs;
- if job set fraction is set to 0.5 the the job will require 5 files every time it runs.

initial file distribution - Determines where the master copies of the files\(^5\) are placed at the start of the simulation. This can either be a set of numbers separated by commas or random to distribute them randomly throughout the Grid. In both cases the files are distributed using a uniform probability distribution. For example, if initial file distribution = 0,2,3, then each master copy will be stored in one of the sites 0, 2, or 3, randomly chosen using a uniform distribution.

fill all sites - If this is set to yes, all the SEs in the Grid will be filled randomly with replicas of the master files. For each SE, files are picked

\(^4\)In a Zipf-like distribution the popularity of the files accessed by a job is given by $P(f_i) = 1/i^\alpha$, with $0 \leq \alpha < 1$, where $P(f_i)$ is the popularity of the $i$-th most popular file and $\alpha$ is close to 1.

\(^5\)Master files cannot be deleted by the replica optimiser algorithms.
at random until it is full (making sure there are no duplications). If the SE is larger than the size of all the files, all files are added to the SE.

**job delay** - The basic time interval (in milliseconds) between jobs being submitted to the Grid by the Users during simulation. The actual submission interval depends on the type of user chosen (above).

**random seed** - Determines whether the seed used by various methods within OptorSim where random numbers are required is fixed or random. If this is set to **yes**, it will be random; if **no**, it will be fixed. For example, if it is **yes**, a different set of jobs will run each time the simulation is run. If it is **no**, the same jobs will run each time.

**max queue size** - The maximum number of jobs the CE will hold in its queue before it refuses to accept any more.

**file process time** - The time in milliseconds for a CE to process each file. This is divided by the number of worker nodes in the CE i.e., all worker nodes in a CE are supposed to execute the same job.

### 3.3.2 Auction Parameters

The Economic Model can use an auction protocol as a way to select the best replica for a given file request or to select the best source file for replicating. More details can be found in [5].

**auction flag** - Set to **yes** to use the auction protocol.

**hop count** - The maximum distance (in number of sites) covered by auction messages.

**timeout** - The time (in milliseconds) the auction will wait for replies to its bid message before it chooses the winning bid.

**timeout reduction factor** - The factor, in the range [0,1] by which the auction timeout is reduced for nested auctions. This is so the nested auctions have time to complete and return a bid to the parent auction before the parent’s auction timeout expires.

**auction log** - If this parameter is set to **yes**, then the simulation outputs auction information to the file **auction.log**. This can slow the simulation down a little bit.

### 3.3.3 Background Network Traffic Parameters

OptorSim allows the simulation of contending network traffic which can vary over time. This is explained in more detail in Section 3.4.
background bandwidth - Set to yes to use background network traffic, no to switch it off.

data directory - The directory in which the background bandwidth data files are stored.

default background - The file to use when no other data are available.

time of day - The time of day used as starting point. This should be in hours, with minutes after the decimal point e.g. 22.5 for 22:30, and must be on the hour or half-hour.

3.3.4 GUI Parameters

OptSim has a Graphical User Interface (GUI) that allows the user to control the simulation and visualise simulation outputs\(^6\). This can be chosen using the parameters file:

gui - Set to yes to activate the GUI.

3.3.5 Statistics Parameters

OptSim keeps track of a number of statistics as the simulation runs (see Section 5) which are stored hierarchically: those relevant to the whole Grid, to individual sites and to CEs/SEs within sites. The level of statistics output can be modified using the parameters file.

statistics - There are three choices of statistics output:

1. None: Turn off statistics output.
2. Simple: Only print top-level (whole Grid) statistics.
3. Full: Print full statistics for all elements of the Grid.

The default is 3.

3.3.6 Time Model

There are two time models implemented in OptSim, one time-based and one event-driven, and OptSim can be run in either mode with the same end results. In the time-based model, the simulation proceeds in real time. In the event-driven model, whenever all the CE and RB threads are inactive, the simulation time is advanced to the point when the next thread should be activated. The use of the event-driven model speeds up the running of the simulation considerably, whereas the time-based model may be desirable for demonstration or other purposes.

\(^6\)Instructions for using the Graphical User Interface (GUI) are given in detail in section 4.
time advance - Set to yes to use the event-driven time model or no to use the time-based model.

The event-driven model is the default.

3.4 Simulating Background Network Traffic

Inclusion of background network traffic requires a configuration file and one or more data files. The configuration file is a site-by-site matrix giving, for each pair of sites, the name of the data file containing the relevant bandwidth information and also the time difference between the reference time zone (see below) and the source site. The source sites are in rows, with the destination sites in columns, and each entry is of the form <filename>,<time zone>. If a data file is not available for a particular link, a default may be specified in the parameters file and represented in the bandwidth configuration file by a hyphen.

An example is shown in Figure 4. For the simple Grid in Figure 2, suppose we have data files 2_to_7.dat, 5_to_9.dat, 3_to_4.dat and 8_to_7.dat available. Suppose also that sites 0, 1, 2, 3 and 4 are in a time zone 8 hours behind sites 5, 6, 7, 8 and 9 and that the reference time is that of site 2. Then the third line, for instance, shows the network bandwidth information for links between site 2 and the remaining 9 sites. The available network bandwidth for site 2 to site 7 is retrieved from the data file 2_to_7.dat and the time zone difference is 8 hours. The network bandwidth from site 2 to the remaining sites is based on the default value.

```
-0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0 -0.0
```

Figure 4: A simple example background bandwidth configuration file.

The time zone information is required due to the data file format, which consists of 48 lines (one per half-hour of the day) of the form:

<time>,<mean available bandwidth>,<standard deviation>

As an example, part of a real data file is shown in Figure 5. Time starts at 0000 hours and is incremented in half-hour steps, so each half-hour period has an associated mean bandwidth and standard deviation for the
link, which are expressed as fractions of the maximum bandwidth for that link. Time should be expressed in hours (0, 0.5, ..., 23, 23.5). Local time must be accounted for, so a reference time zone is chosen against which all others are calibrated.

Files for some links in the EDG, CMS and GridPP testbeds are supplied with OptorSim in the appropriate subdirectories of the examples/bw_data directory.

4 Using the Graphical User Interface

4.1 Starting the GUI

If the parameter gui is set to yes in the configuration file, the simulation can be monitored visually through the Graphical User Interface (GUI). In this case, when OptorSim is started from the command line, the GUI is displayed as shown in Figure 6. The simulation is then started by choosing Start from the Simulation menu.

4.2 Components of the GUI

The GUI window, shown in Figure 7 while the simulation is running, consists of three panes: a hierarchy pane, a statistics pane, and an information pane. These can be hidden or redisplayed using the Display menu.

Hierarchy pane. The hierarchy pane (Figure 8, left) gives a hierarchical view of the Grid components, down to the SE/CE level. Clicking on any of the components will bring up the relevant statistics for that component in the statistics pane.

Statistics pane. The statistics pane (Figure 8, right) contains a number of tabs with graphs of accumulated statistics relevant to the selected node, plus one with a tabular summary of the current status. These are refreshed with a frequency which is defined by the user; this is
Figure 6: Screenshot of GUI at startup.

Figure 7: Screenshot of GUI while running.
done by choosing Change sample rate from the Statistics menu. The default is a rate of one-fiftieth the number of jobs per second. For histograms, the number of bins can be changed by choosing the appropriate option from the Statistics menu.

**Information pane.** The Information pane contains three tabs - Parameters, Terminal and Logical View - which are shown in Figure 9. The Parameters tab shows the current settings of the simulation parameters; the Terminal tab contains the standard output from the simulation; and the Logical View tab shows all current file transfers between sites. Clicking on a node in the Logical View will bring up the relevant statistics for that site in the statistics pane.

Figure 9: The Information pane tabs: Parameters (left), Terminal (centre) and Logical View (right).
4.3 Pausing, Restarting and Stopping the GUI

The GUI may be paused, restarted and stopped from the Simulation menu. Please note that in the current version, it is not possible to start a new simulation after one has stopped. It is necessary to exit and re-run the GUI from the command line.

4.4 Saving Data

When the simulation has finished running, the user may save the results by choosing Produce Summary from the File menu. This results in the creation of a directory called allGraphs@sim_completed, within the current working directory, in which are stored all the statistics graphs in jpeg format. These are linked from the file Summary@sim_completed.html (which is also created in the current working directory), along with the summary tables for each Grid component.

Such a summary may be produced at any time during the simulation if the simulation is paused, as the time taken to collate the data would result in new readings being missed if the simulation was running. In this case, the graphs would be put in a directory called allGraphs@<date>, where <date> is the date, in simulation time, at which the summary was taken. Similarly, the html file is called Summary@<date>.html.

Individual graphs may be saved at any time by selecting Save Graph / Summary Table from the File menu while the desired graph is displayed in the statistics pane, or by right-clicking the mouse in the statistics pane then choosing Save Graph. The graph is stored in the current directory in the form <graph name>@<date>.jpeg where <graph name> gives the name of the relevant grid component with the type of graph, and <date> is the date, in simulation time, at which the graph was saved.

A copy of the parameters used may also be saved at any time, as a jpeg file, by choosing Save Parameters Table from the File menu. The file will appear in the current working directory as parameters.jpeg.

5 Outputs from OptorSim

There are various ways in which OptorSim provides output to the user so that they can evaluate the results they require from it. These include both information printed to the screen and that displayed by the GUI (see Section 4). Some metrics to measure the effectiveness of different optimisation strategies are discussed in [6]

A variety of statistics describing the state of the Grid and information on all the jobs run can be output to the screen at the end of the simulation (see Section 3.3.5). Firstly a summary is given of all the resources present on each site and which files each Storage Element contains.

Statistical information is then given according to the option specified in the parameters file (Section 3.3.5). If full statistics output is chosen, this is
given in a tree-like format where each branch from the main trunk, which is the whole Grid or “Grid Container”, is a site on the Grid. From each site is a branch to each CE and SE it contains, with their statistics. If top-level statistics output is chosen, only those for the whole Grid are displayed. The following information is given for the different components of the Grid:

**The whole Grid**

- **remoteReads**: The total number of remote file accesses that took place.
- **localReads**: The total number of local file accesses that took place.
- **replications**: The total number of file replications that took place.
- **ENU**: The effective network usage:

$$r_{ENU} = \frac{N_{\text{remote file accesses}} + N_{\text{file replications}}}{N_{\text{remote file accesses}} + N_{\text{local file accesses}}}$$

This is effectively the ratio of files transferred to files requested, so a low value indicates that the optimisation strategy used is better at putting files in the right places.

- **ceUsage**: The percentage of time that Computing Elements have been active (the average of the ceUsage for each site).
- **totalJobTime**: The combined total time in seconds of all the jobs run.

**Grid sites**

- **remoteReads**: The number of remote file accesses that took place at this site.
- **localReads**: The number of local file accesses that took place at this site.
- **ceUsage**: The percentage of time that the CE at this site has been active.
- **fileAccesses**: The number of times a file on this site has been accessed by another site, i.e. the number of file transfers originating from this site.
- **routedFiles**: The number of file transfers that were routed via this site.
- **totalJobTime**: The total time in seconds taken to run all the jobs submitted to this site.

**Storage Elements**

- **capacity**: The capacity in MB of the Storage Element.
- **usage**: The amount of space in MB used by files at this Storage Element.
Computing Elements

- **jobTimes**: The running time in milliseconds for each job run on this Computing Element is listed.
- **jobTimesWithQueue**: The time in milliseconds for each job run on this Computing Element, including their queueing time, is listed.
- **remoteReads**: The number of times this Computing Element read a file from a remote site.
- **numberOfJobs**: The number of jobs run by this Computing Element.
- **workerNodes**: The number of worker nodes in this Computing Element.
- **localReads**: The number of times this Computing Element read a file from a Storage Element on the same site.
- **usage**: The percentage of the time this Computing Element was running jobs i.e. not idle.
- **jobFiles**: For each job run by this Computing Element, the files required are listed in the order they were accessed.
- **totalJobTime**: The total time in seconds taken to run all the jobs on this Computing Element.

Note that for all the components, totalJobTime is defined as the sum of all the individual job times, including their queueing times.

When the GUI is used (see Section 4), a number of graphs of these quantities as a function of time may be viewed, as well as summaries of the instantaneous state of the components.

6 Future Work

There are many directions in which the development of OptorSim could proceed. Some of the possibilities for future work are:

- The addition of job budgets to the economic models.
- Modelling different kinds of jobs, e.g. Monte-Carlo jobs, and having new files added to the Grid as the simulation runs.
- Better modelling of Computing Elements, such as worker nodes which can run jobs in parallel.
- Modelling different types of Storage Element, such as disk and tape.
- Modelling a dynamic Grid where not all resources, such as Computing and Storage Elements, are always available.
• Making parameters modifiable from the GUI, so that the simulation can be restarted with different parameters without exiting from the GUI.

• Enabling the addition/removal of sites on the Grid and changing of the resources available using the GUI.

References


