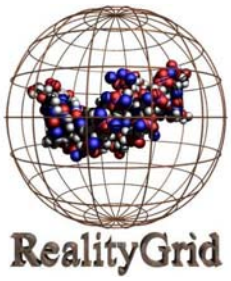
RealityGrid Launcher User Manual



Andrew Porter.

SVE Group, Manchester Computing, University of Manchester.

14th July 2004.

Change Log

Author	Comments	Date
Andrew Porter	Initial draft	14th July 2004

Acknowledgements

The RealityGrid Launcher is mainly the work of Mark Riding, with some further enhancements by Andrew Porter. Many of the launching scripts it uses were provided by Robin Pinning.

Introduction

The RealityGrid Launcher is intended to be used to launch jobs (both for computation and for visualization) on remote machines, migrate running jobs and restart jobs from previously-created checkpoints. This document aims to give new users an overview of how to use the launcher, and to explain the functionality which it offers.

Starting the Launcher

The configuration file

On start-up, the launcher reads the configuration file "default.conf," the structure of which is shown in Figure 1 below. Before running the launcher for the first time this configuration file will typically have to be edited. In the "GSHs" section, specify the Grid Service Handle (GSH) of the top-level registry with which services will be registered. In the "containers" section specify the address of one or more OGSI::Lite containers by giving the address of the hosting machine and the port on which the container is listening. In the "targets" section specify the address of one or more machines that have been configured to be used as compute resources (*i.e.* RealityGrid directory structure in place, software installed and working globus or SSH access). If you wish to include your local machine in this list then include an entry with name="localhost." Similarly, in the "vizTargets" section, specify the address of one or more machines that are available and configured to run visualization jobs. In the "globus" section, specify the location of the globus installation on your machine (a full globus installation is not necessary – the CoG kit is sufficient).

Finally, in the "applications" section give details of each application that you wish to launch. Note that these must be compiled and installed on the target machines specified in the "targets" and "vizTargets" sections. The "name" attribute of the application must correspond with a launching script, "<name>_launch.sh", in the "scripts" directory of the launcher. *e.g.* if name="namd" then "namd_launch.sh" must be present in the "scripts" directory. The "inputs" attribute specifies the number of input IOTypes (*i.e.* input data channels controlled by the RealityGrid steering library) that the application has. Typically this is zero for a simulation and one for a visualization (note that applications with more than one input are not currently supported).

The "hasInputFile" attribute specifies whether the job requires an input file, the "restartable" attribute specifies whether jobs of this type may be restarted from an existing checkpoint and the "isViz" attribute specifies whether the job is a visualization (and thus exports a window back to the user's machine).

```
<?xml version="1.0"?>
<launcher>

 <GSHs>
  <topLevelRegistry
value="http://some.machine.address:50000/Session/ServiceGroupRegistration/service?77
77777777"/>
 </GSHs>

 <containers>
  <container port="50000">container.machine1.address</container>
  <container port="50000">container.machine2.address</container>
 </containers>

 <targets>
  <machine name="compute.machine1.address" os="irix"
           jobmanager="compute.machine1.address/jobmanager-pbs"
           queue="none" />
  <machine name="compute.machine2.address" os="aix"
           jobmanager="frontend.address/jobmanager-fork"
           queue="escience" />
  <machine name="localhost" os="linux"
           jobmanager="fork"
           queue="none" />
 </targets>

 <vizTargets>
  <machine name="viz.machine1.address" os="irix"
           jobmanager="viz.machine1.address/jobmanager-pbs"
           queue="none" />
  <machine name="viz.machine2.address" os="linux"
           jobmanager="viz.machine2.address/jobmanager-pbs"
           queue="none" />
 </vizTargets>

 <globus location="/usr/local/globus"/>

 <applications>
  <application name="lbe3d" inputs="0" hasInputFile="yes" restartable="yes"
               isViz="no"/>
  <application name="lbe3dviz" inputs="1" hasInputFile="no" restartable="no"
               isViz="yes"/>
  <application name="namd" inputs="0" hasInputFile="yes" restartable="yes"
               isViz="no"/>
 </applications>

</launcher>
```

Figure 1: The structure of the configuration file.

Environment set-up

The launcher is able to fire up the RealityGrid, Qt steering client for a job that it has just created. For this to work correctly you must have the steering client installed in ~/RealityGrid/reg_qt_steerer/steerer.

Remote job launching is performed using the globus toolkit which must therefore be installed on the client machine. The simplest solution to this is to install the globus CoG kit. Alternatively, version 2.4.1 of the full toolkit is required. Which of these options to use is controlled by setting the ReG_LAUNCH environment variable to either ‘cog’ or ‘globus’ (case sensitive). The globus toolkit uses X.509 certificates for authentication and therefore the user requires a certificate that is accepted on the machines listed in the configuration file. Before attempting to launch a job, the user must create a valid proxy certificate on their machine by executing grid-proxy-init.

It is recommended that a user test basic globus functionality before attempting to use the launcher. For instance, to check that you can authenticate to a machine do:

```
globusrun –a –r machine.address.here
```

(assuming the globus bin directory is in you path).

Launching

Run the launcher by executing the command './reg_qt_launcher' at the command prompt. You will be presented with the main window:

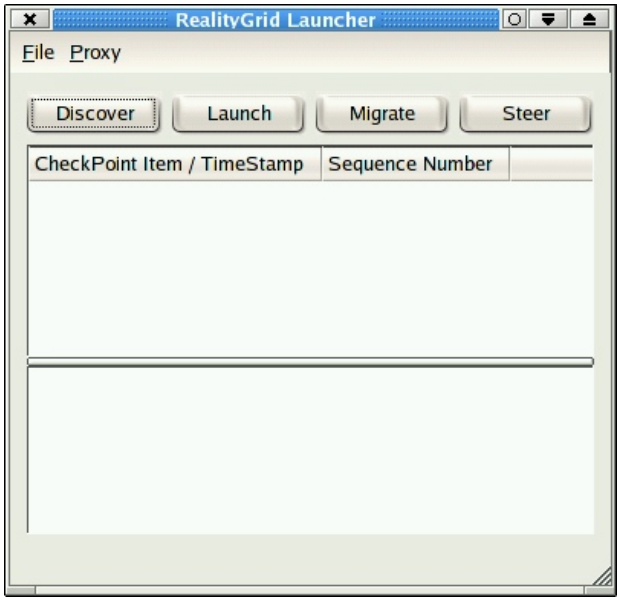


Figure 2: The main window of the launcher on startup.

Using the Launcher

The Main Window

The main window of the launcher, shown in Figure 2, is split into three conceptual sections:

- Buttons for initiating checkpoint browsing, job launching/migration and job steering
- Checkpoint tree browser
- Information panel

There are also a couple of menus, File and Proxy, and a status bar where information on launched jobs is displayed.

The File and Proxy menus

The File menu currently only contains a Quit option. The Proxy menu, shown in Figure 3, provides options for managing your proxy certificate. If you are intending to use globus for remote job launching then you must initialize your proxy via the Proxy->Init menu item (if you haven't already run grid-proxy-init on your machine). A proxy certificate typically remains valid for 12 hours.

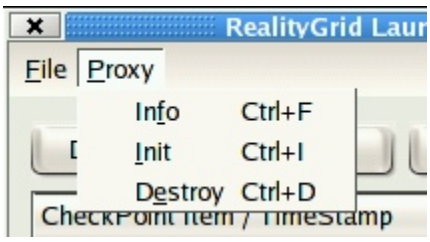


Figure 3: The Proxy menu on the main window.

Launching a job

All job launching is set-up by means of a wizard. Clicking the "Launch" button brings up the first page of this wizard, shown in Figure 4. The user is able to select the type of job they wish to launch from the drop down menu. (This menu is populated using the information supplied in the "applications" section of the configuration file.) If the selected job type requires an input file then the user must also specify that on this page. Finally, the length of the required run (wall-clock time in minutes) should be entered.

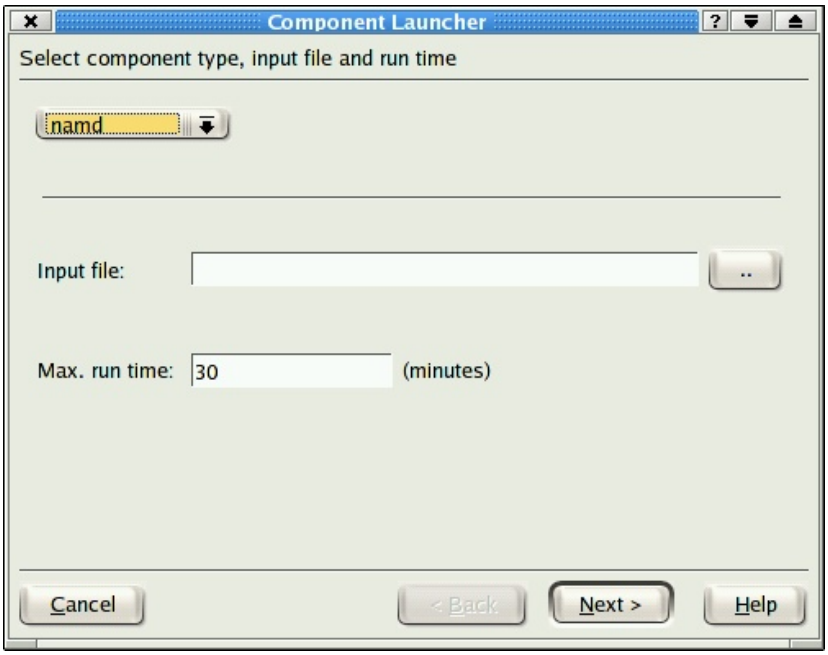


Figure 4: Launching wizard – job type, input file and run time entry

On page two of the wizard, the user selects the machine and number of processors (if appropriate) on which they wish to run the job. If the job is a visualization (as specified in the configuration file) then the user may also switch on the use of multicast and/or VizServer (from SGI) software. The latter is used to compress the output of the visualization and potentially allow collaborators to interact with the job. (Note that the user must have a VizServer client up and running on the machine with an active connection to the target machine before using the launching wizard.) Multicast is used to broadcast the output of the visualization to Access Grid.

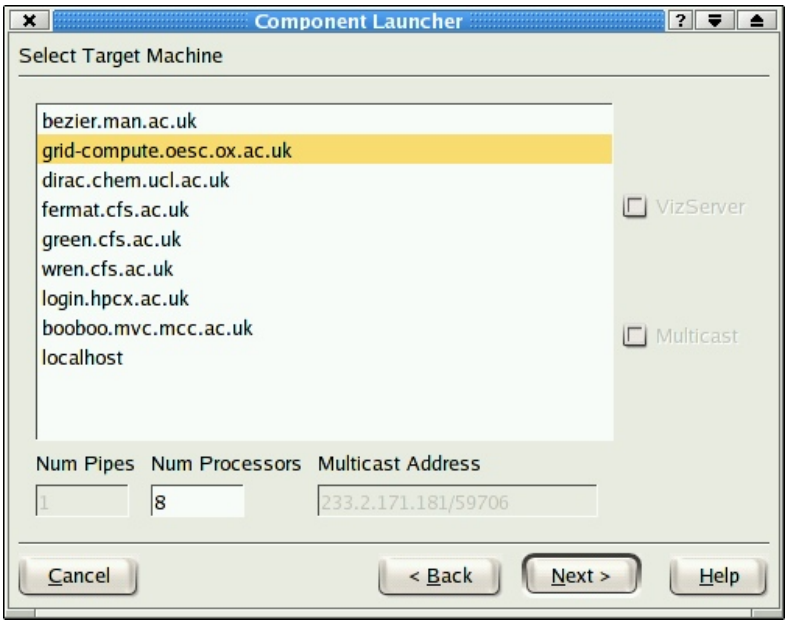


Figure 5: Launching wizard - target-machine page

Once the target machine and associated options have been selected, the user must choose which Grid-service container to use to host the Steering Grid Service (SGS) for the job that they are launching. For performance reasons, it is best to choose a container that has good network connectivity to the target machine. Ideally, it will also have good connectivity to the user's machine but that is less important since that does not affect the performance of the job itself.

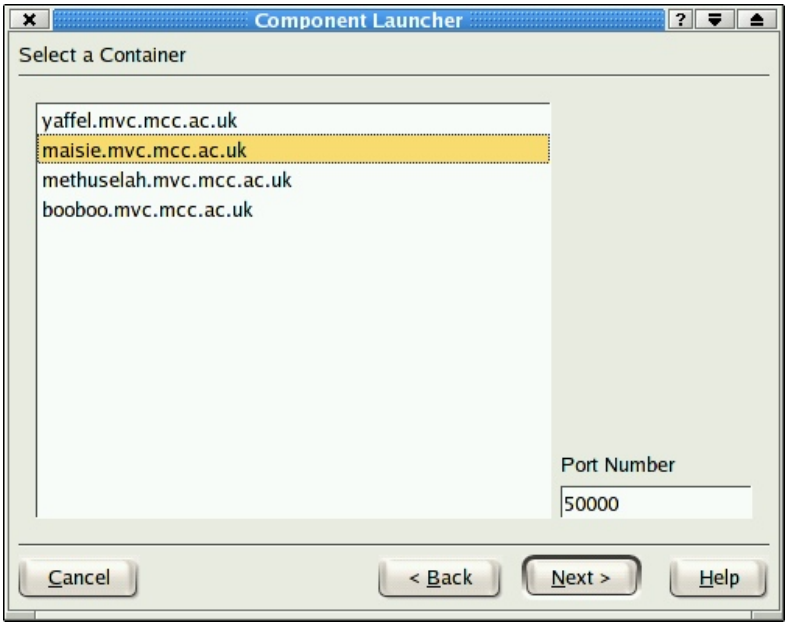


Figure 6: Launching wizard - container selection

The next page of the wizard (Figure 1Figure 7) captures some information on the job being launched. The first three fields are completed automatically leaving the user to enter the name of their organisation and some text describing the purpose of the job.

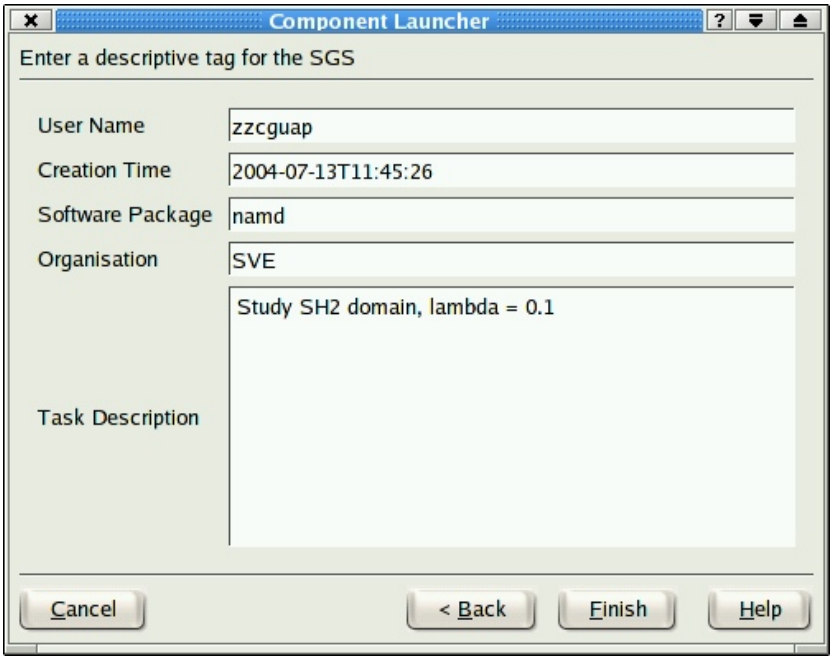


Figure 7: Launching wizard - job meta-data

If the application that the user has chosen to launch is not restartable then the job meta-data page completes the wizard. However, if the job is restartable and no checkpoint node has been selected (see the section on the Checkpoint Tree on page 11) then the user is given the option of creating a new checkpoint tree along with the job (Figure 8). To do this, they must enter some text

describing the checkpoint tree. If no checkpoint tree is required then this field should be left blank (simply press 'backspace' in the edit box).

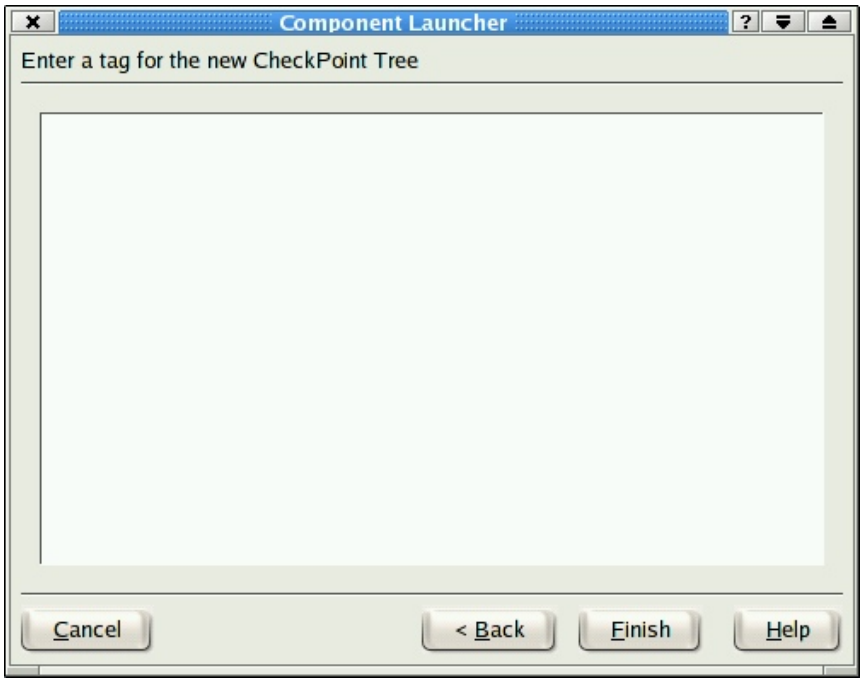


Figure 8: Checkpoint tree creation page.

Once the wizard is complete, pressing the Finish button initiates the launching process and closes the wizard. During the launching process, information on the Grid-service factory and SGS is output to the information panel in the main window of the launcher, as shown in Figure 9. Once an SGS has been created for the job, the launcher launches the job proper (using the appropriate scripts from the 'scripts' directory) via globus. Once this is complete, the launcher polls the SGS for the status of the job and the results are displayed in the status bar (bottom left of Figure 9).

Figure 9: Output from a launching process.

Once the status changes from "Job is Queued" to "Job is Running" then the job has successfully started up and contacted the Steering Grid Service. Clicking on the Steer button at the top right of the launcher main window will launch the steering client if it is available. By default, the launcher will instruct the steering client to connect to the last non-visualization job it has created. If the steering client fails to attach to the job then it is likely that the job has failed soon after startup. If the job status remains on "Job is Queued" and does not change then this indicates that the creation of the Grid service was successful but the launch of the job itself was not and it has therefore not

contacted the Grid service to signal that it has started execution. The use of the steering client itself is described in the RealityGrid Steering Application User Manual (available from *http://www.sve.man.ac.uk/Research/AtoZ/RealityGrid/*).

Launching a visualization job

The process of launching a visualization job is very similar to the process just described for a standard simulation. The principal difference is the need to specify a data source for the visualization. The wizard has an extra page for this (shown in Figure 10) which is displayed following the standard first page. This page displays a list of all the jobs that are currently running (as obtained by querying the registry specified in the launcher configuration file), enabling the user to select the one that they wish to use as a data source.

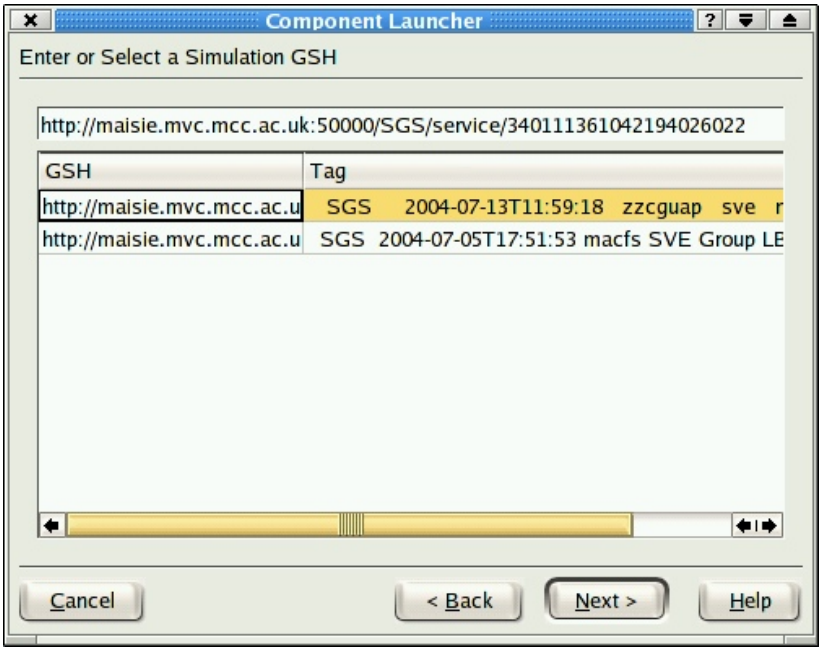


Figure 10: Choosing a data source.

Following this step, the only differences to the process of launching a standard simulation are:

1. It is possible to turn on VizServer and/or multicast output for the job – see Figure 11: Choosing a visualization machine.Figure 11. (This sets flags in the job script which must then be picked up by the remote visualization job and acted on as appropriate.)

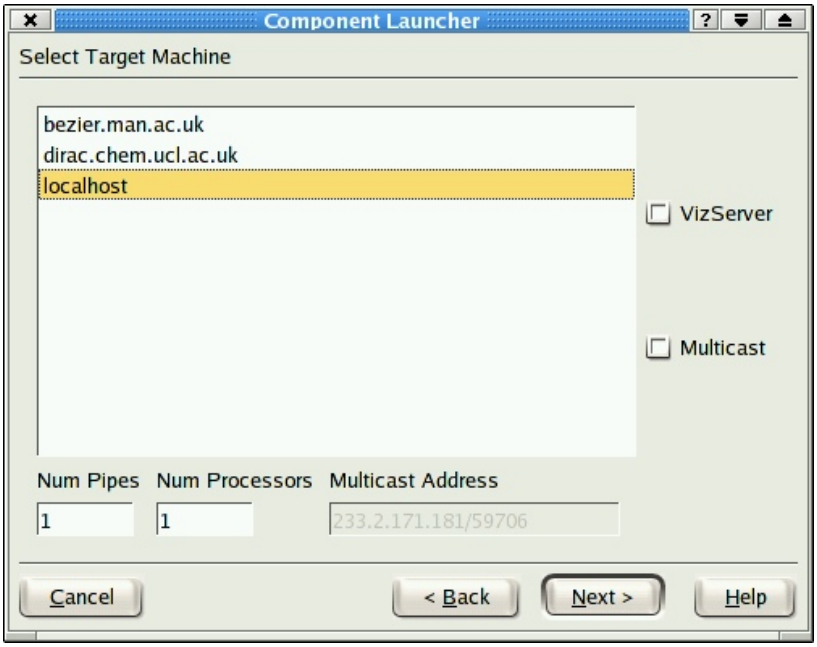


Figure 11: Choosing a visualization machine.

2. The user is prompted to select the type of visualization to perform (Figure 12). This step is only significant if the visualization to be launched is the vtk-based visualizer developed as part of the project. Any other visualization package (*e.g.* vmd) ignores this setting.

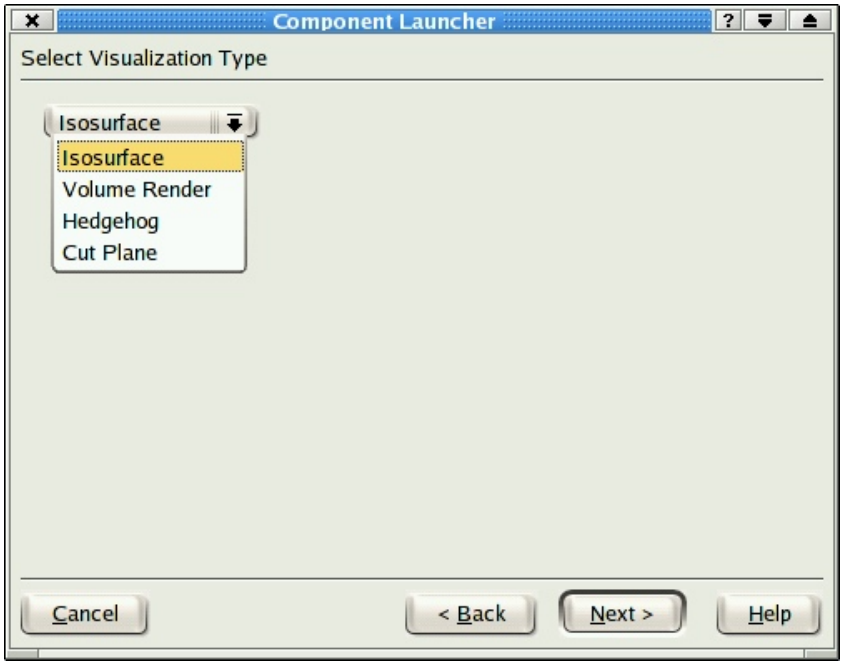


Figure 12: selecting the type of visualization.

Using the Checkpoint Tree

Browsing a tree

To browse the available checkpoint trees, click the Discover button on the main window. The information panel will show “Searching for CheckPoint Trees” and, after a short delay, the checkpoint tree window will display a list of checkpoint trees. Each of the displayed trees is the root node of a tree of checkpoints and each node in such a tree corresponds to a checkpoint created by an application.
A checkpoint tree may be expanded and explored by clicking the “+” symbols (just as in the tree view of a standard file-system browser). Figure 13 shows an example of expanding out a checkpoint tree.

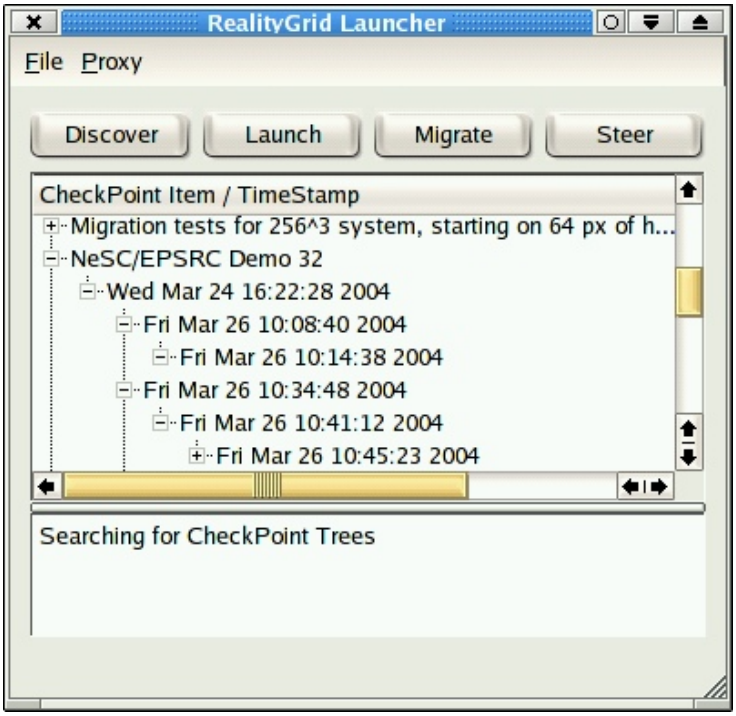


Figure 13: Exploring a checkpoint tree.

Each checkpoint node is labelled with the date and time at which it was created. In addition, the sequence number of the simulation at the point when the checkpoint was created is displayed in a column on the right-hand side of the checkpoint tree browser window (not visible in Figure 13).

Right-clicking on any node in a checkpoint tree brings up a context menu, as shown in Figure 14. This menu has four items:

- View Parameters;
- View GSH;
- View Input File;
- View CheckPoint Data.

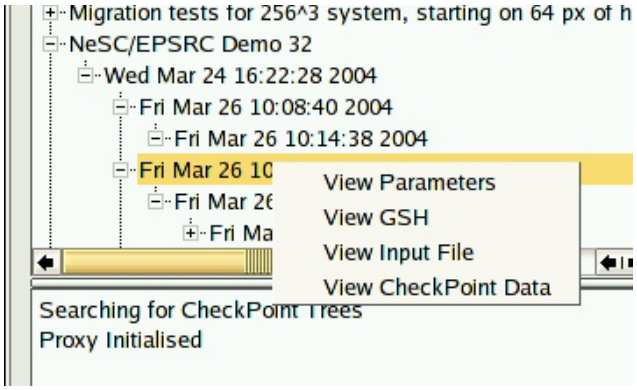


Figure 14: The checkpoint node context menu.

The View Parameters option brings up the window shown in Figure 15. This window displays the values of all of the parameters registered with the steering library (both steerable and monitored) at the time when the selected checkpoint was created.

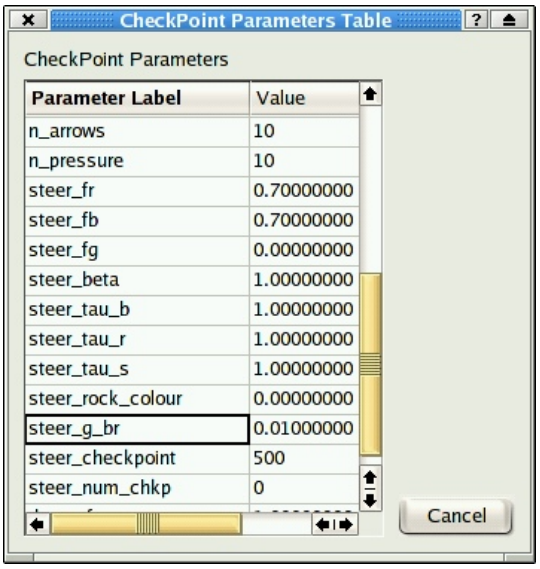


Figure 15: The checkpoint parameters window.

The View GSH option displays the GSH of the selected checkpoint node (required by the Qt steering client when rewinding a running code). To aid cut-&-paste, the GSH is displayed already selected:

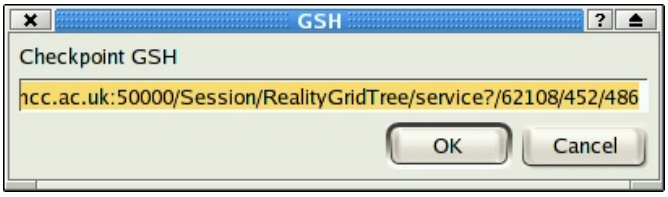


Figure 16: The checkpoint GSH window.

The View Input File option displays the input file (if any) that was used to start the job that created the selected checkpoint. An example is shown in Figure 17.

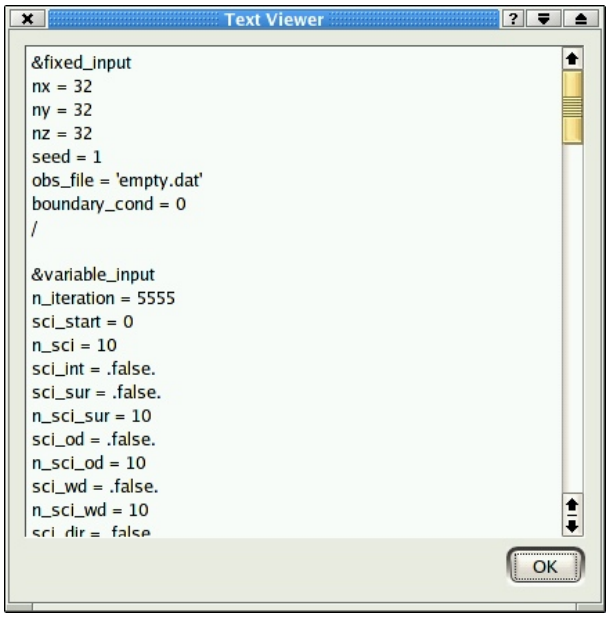


Figure 17: Example view of job input file.

Finally, the View CheckPoint Data option displays the xml document describing the make-up of the selected checkpoint. This enables the location of the checkpoint data to be viewed, as shown in Figure 18.

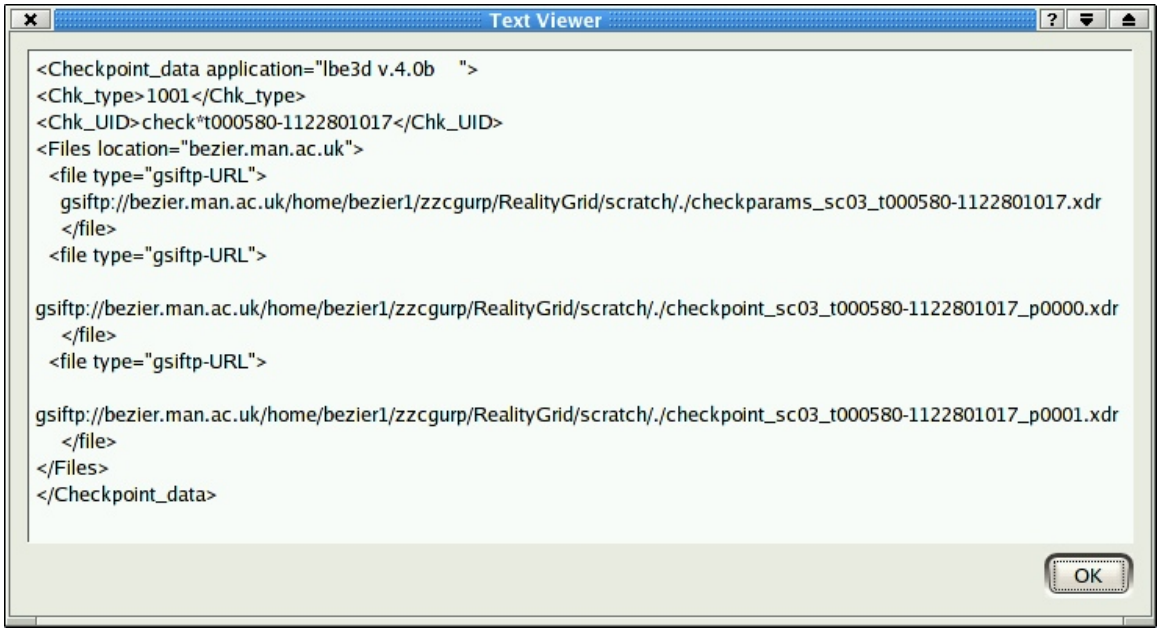


Figure 18: An example of checkpoint data. This checkpoint was created by an application called lbe3d, consists of three files and only exists on the machine bezier.man.ac.uk.

Restarting a job

Any valid node in a checkpoint tree may be used as the starting point for a new calculation by selecting (clicking on) it in the checkpoint tree browser window. This causes the “Launch” button

on the main window to be re-labelled as “Restart.” Clicking this button again brings up the launching wizard. However, since a restart has been requested the user does not need to select the type of job and therefore the job type and input-file selection parts of the first page of the wizard are greyed out. All the user need enter is the runtime for the job. The next page of the wizard now gives the user the option of editing the job input file (retrieved from the selected checkpoint node). From this point on, the steps are exactly the same as those described previously for launching a job from scratch (with the exception that the user is not given the option of creating a new checkpoint tree since that is not relevant).

Once the wizard is completed, the launcher checks whether the selected checkpoint is available on the target machine. If it isn’t then it initiates a third-party file transfer to copy the one or more files that make up the checkpoint to the target machine. (The code that does this is aware of the Atlantic and attempts to avoid copying files across it.) While this task is being completed, the launcher brings up an activity bar, shown in Figure 19. Since the file transfer is accomplished by running a shell script it is not possible to retrieve progress information and therefore the progress bar merely indicates that the launcher is carrying out the task.

Figure 19: The file-transfer activity bar.

Once the file transfer is complete, the job is started on the target machine. At this stage, the launcher main window should look something like that shown in Figure 20. Note the messages in the Information panel.

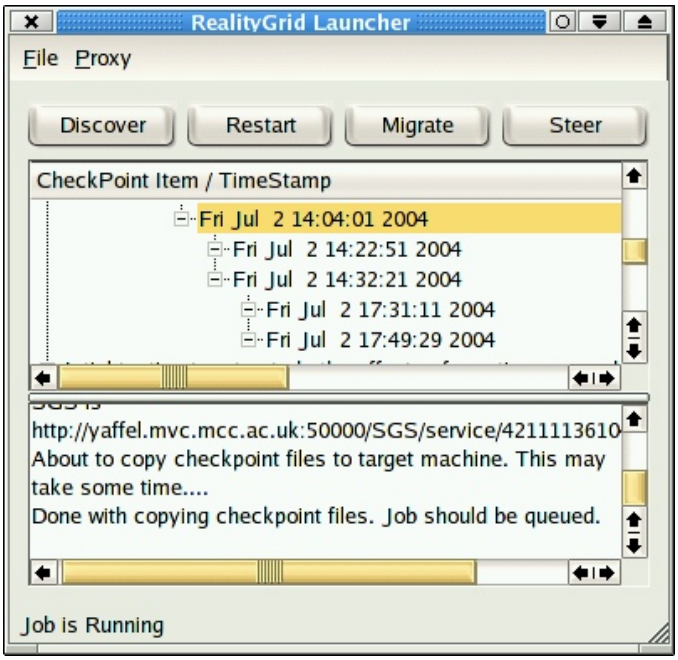


Figure 20: The main window following a successful restart.

Migrating a running job

In addition to creating new jobs, the launcher is capable of migrating a running job from one machine to another. This functionality is based upon the checkpoint tree and therefore the job must be restartable and instrumented such that it registers (via the RealityGrid steering library) the checkpoints it creates with the Checkpoint Tree.

In order to migrate a job, the user clicks the Migrate button on the main launcher window. As with launching a visualization component (Figure 10), this brings up a dialog showing the jobs currently stored in the registry. Once the user has selected a job to migrate, the launcher contacts it, instructs it to take a checkpoint and then stops it. Note that this process will fail if a steering client is connected to the job because concurrent steering by more than one client is not currently supported. During this process, the launcher should display messages like those shown in Figure 21.

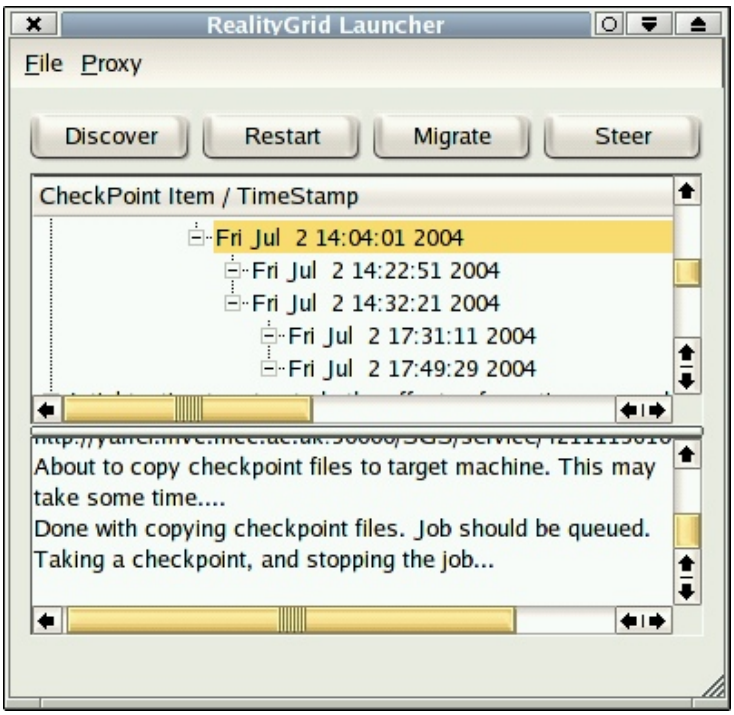


Figure 21: The launcher during a job migration.

Once this stage is complete, the launcher again brings up the wizard to allow the user to the desired runtime of the new job and to select the machine and number of processors to migrate the job to. Once the necessary information has been collected, the files making up the newly-created checkpoint are moved from the machine that hosted the original job to that chosen for the migration and a new instance of the application is launched.

Index