Contents

1 Introduction 4
   1.1 Downloading the Code and Documentation from CVS ............................ 4
   1.2 Philosophy Behind the RMs .................................................. 4

2 Prerequisites 4
   2.1 Perl Environment ........................................................................ 4
   2.2 HARC Account ........................................................................... 5
   2.3 ACL and SUDO Mode ................................................................. 5
   2.4 X509 Certificate ......................................................................... 5

3 Installation Procedure 5
   3.1 Creating the Configuration Directory ........................................... 5
   3.2 Creating the install.config ......................................................... 6
      3.2.1 Generic: Perl Location ....................................................... 7
      3.2.2 Generic: LD_LIBRARY_PATH/LIBPATH and PERL5LIB .......... 7
      3.2.3 Generic: Resource Manager URI ....................................... 7
      3.2.4 Generic: Resource Description ......................................... 8
      3.2.5 Generic: Mandatory Email and Project Specification ............ 8
      3.2.6 Generic: Authorization Module ......................................... 8
      3.2.7 Generic: Resource Manager Type ....................................... 8
      3.2.8 SimpleCompute: Nodename ............................................... 9
      3.2.9 SimpleCompute: Core Count and Memory ............................ 9
      3.2.10 SimpleCompute: Batch Scheduler and SUDO/ACL mode ...... 10
      3.2.11 SimpleCompute: Globus Jobmanagers .............................. 14
      3.2.12 SimpleNetwork ................................................................ 15
   3.3 CA Certificates ........................................................................... 15
   3.4 Acceptor Authorization File ....................................................... 15
   3.5 User Authorization File ............................................................. 15

4 Installing the Resource Manager 15
   4.1 Installing the RM Certificate .................................................... 16
4.2 Running Interactively to Test the RM Configuration ........................................ 17
4.3 Configuring the RM to Start via cron ................................................................. 17
4.4 Starting and Stopping the RM ............................................................................. 17
4.5 Updating the RM ................................................................................................. 18

5 Adding a New Module for a New Scheduler .......................................................... 18
  5.1 Creating the Module ............................................................................................ 18
  5.2 Modifying SimpleComputeRM to use it ............................................................ 19
  5.3 Overriding methods ........................................................................................... 19
    5.3.1 Constructor .................................................................................................. 19
    5.3.2 initialize ....................................................................................................... 20
    5.3.3 makeReservation ....................................................................................... 21
    5.3.4 cancelReservation ..................................................................................... 22
    5.3.5 getStatus ..................................................................................................... 23
  5.4 Changing the Installer ......................................................................................... 25
  5.5 Contributing back to the community ................................................................. 25

6 Adding a New RM for a New Type of Resource ..................................................... 25

7 RM Caveats ............................................................................................................. 26
1 Introduction

This document describes the installation of a HARC Compute RM, and the configuration necessary to make it talk to a scheduler.

The document assumes that the HARC environment variable points to the root of the HARC CVS repository, i.e. the “negotiation” directory.

1.1 Downloading the Code and Documentation from CVS

1. Go to the directory where you want to place the code, e.g.
   
  .cd $HOME/cvs

2. Set your CVSROOT up.
   
   export CVSROOT=:pserver:cvs_anon@cvs.cct.lsu.edu:/public

3. Log in, entering “anon” when prompted for your CVS password.
   
   cvs login

4. Check out the code and documentation for V1.9.4.
   
   cvs co -r RM_V1_9_4 negotiation

   A directory called negotiation will be created under the current directory, containing the Acceptor and RM sources, plus the documentation.

5. Set the environment variable HARC to point to the negotiation directory. Put this into your .bash_profile, etc. This variable will be referred to a lot in this and other documentation.

1.2 Philosophy Behind the RMs

The HARC RMs have been designed so that:

- New types of RMs can be written by others;
- Existing RMs can be customized;
- Interfaces can be enhanced or changed; and
- None of this means changing the acceptor code.

2 Prerequisites

2.1 Perl Environment

You should have already installed the required Perl modules. See the “HARC RM Perl Dependences” document.
2.2 HARC Account

The HARC RM should be installed under a separate account, e.g. harc, which should only be used for this purpose. This is particularly important if you’re using SUDO mode for the RM (see next section).

2.3 ACL and SUDO Mode

All of the batch scheduler modules currently in CVS (except Catalina) can be run in either ACL mode or SUDO mode. These operate as follows.

ACL mode  Reservations are made using the same account, and other users are given access to use those reservations using Access Control Lists which are given to the scheduler.

SUDO mode  The HARC account is given the power to make/query/cancel reservations as other users, via sudo.

SUDO mode is recommended, because this allows the HARC RM to be a more transparent interface to the scheduler. In addition, accounting and reservation quotas will be handled correctly by the scheduler.

ACL mode is useful primarily for testing the RM. ACL mode is not supported by the Catalina scheduler.

2.4 X509 Certificate

The HARC RM will need an X509 Certificate so that it can be Authenticated by the HARC Acceptors (clients only talk to RMs via the Acceptors, so they never authenticate the RMs).

We recommend getting a service certificate, rather than (re-)using a host certificate for the compute resource. If you are in the UK, the UK e-Science CA now issues “harccrm” service certificates for Compute RMs (CRMs), with DNs like:

/C=UK/O=eScience/OU=Manchester/L=MC/CN=harccrm/man2.nw-grid.ac.uk/emailAddress=...

3 Installation Procedure

There’s an installer which installs stuff from the source tree to a specified location, configuring this using information from a specified configuration directory. The goal was to allow the configuration for an RM to be easily stored in CVS, or some safe backed-up storage, separate from the code.

Where possible, the HARC installation should minimize dependence upon remote fileservers. Installing the software onto local disk, e.g. /usr/local/harc makes a lot of sense.

3.1 Creating the Configuration Directory

The configuration directory can be created anywhere. This will contain all the information, outside of the RM’s X509 certificate, that you need to install the RM from the code in CVS. The configuration directory will contain the following:
• A configuration file `install.config`;
• A directory of CA certificates that the RM will authenticate Acceptors against;
• The acceptor-mapfile against which the RM will authorize Acceptor; and
• The grid-mapfile against which the RM will authorize users.

These items are described in the following sections.

### 3.2 Creating the `install.config`

This file contains most of the configuration of the RM. It contains a simple set of assignments, in a form suitable to be sourced by a bash script, i.e. lines like the following, with no leading spaces, or spaces surrounding the “=” sign. If spaces appear in the RHS of the assignment, then the whole RHS should be surrounded by double quotes.

```
RM_PORT=9393
```

Blank lines, and comment lines (starting “#”) are permitted.

The following sections will deal with the creation of the necessary parameters in this file. In the end, you will have a file something like the following:

```
RM_HOST=130.88.200.242
RM_URL=man2-rm
RM_PORT=9393

RM_RESOURCE_DESCRIPTION='The Manchester NW-Grid node'
RM_MUST_GIVE_EMAIL=1
RM_MUST_GIVE_PROJECT=1
RM_INNER_TYPE=SimpleCompute

RM_COMPUTE_NODENAME=man2.nw-grid.ac.uk
RM_COMPUTE_CHECK_MEMORY=0
RM_COMPUTE_CPUS=8
RM_COMPUTE_CPU_BLOCK=4
RM_COMPUTE_MEMORY_MB_PER_CPU=4096
RM_COMPUTE_BATCH_TYPE=LoadLeveler
RM_COMPUTE_BATCH_USE_SUDO=1
RM_COMPUTE_BATCH_SUDO_COMMAND=/usr/bin/sudo
RM_LL_COMMAND_DIR=/usr/lpp/LoadL/full/bin

RM_COMPUTE_PRE_WSGRAM_RMC=man2.nw-grid.ac.uk/jobmanager-loadleveler
RM_COMPUTE_WSGRAM_ADDRESS=https://man2.nw-grid.ac.uk:8443/wsrf/services/ManagedJobFactoryService
RM_COMPUTE_WSGRAM_RESOURCE_ID=LoadLeveler
```
3.2.1 Generic: Perl Location

The installer assumes that perl is at /usr/bin/perl. If this is not the case, then set the variable PERL in the configuration file, e.g.

```
PERL=/usr/local/packages/perl-5.8.8/bin/perl
```

3.2.2 Generic: LD_LIBRARY_PATH/LIBPATH and PERL5LIB

If you have a non-standard PERL5LIB, e.g. resulting from a non-root CPAN installation, then you should set this as a parameter. Similarly, if you need to set LD_LIBRARY_PATH to run the RM test program in the Perl installation document, e.g. to pick up the Xerces-C library, then set this as well; on AIX, this is controlled by LIBPATH.

An example of this section would be:

```
PERL5LIB=/home/maclaren/lib/perl
LD_LIBRARY_PATH=/usr/local/compilers/Intel/intel_cc_90/lib:/usr/local/compilers/Intel/intel_fc_90/lib:/usr/local/packages/hdf5-1.6.4/lib:/usr/local/lib
```

3.2.3 Generic: Resource Manager URI

Resource Managers have URI’s like:

```
https://man2.nw-grid.ac.uk:9393/man2-rm/
```

This URI is controlled by three settings:

**RM_HOST** Which hostname/IP address should the RM listen on?

Although the hostname can be specified, it is usually better to specify the IP address, as on some clusters, the hostname may not resolve to the public IP address within the machine. The IP address should be the public IP address for the head node. If you look up the IP address for the machine on a different machine, using the `host` command, then this will be the right address.

**RM_PORT** Which port will the RM run on?

The convention for HARC RMs is port 9393. Use this if you can.

**RM_URL** What should the end of the RM’s URI be?

By convention, this is a short name for the resource (like the first part of the hostname), with “-rm” appended, e.g. bigben-rm.

So the three lines in this file might look like:

```
RM_HOST=130.88.200.242
RM_URL=man2-rm
RM_PORT=9393
```
3.2.4 Generic: Resource Description

The parameter \texttt{RM\_RESOURCE\_DESCRIPTION} should be set to a textual description of the resource, e.g.

\texttt{RM\_RESOURCE\_DESCRIPTION='The Manchester NW-Grid node'}

3.2.5 Generic: Mandatory Email and Project Specification

If present and set to 1, the parameter \texttt{RM\_MUST\_GIVE\_EMAIL} will make the RM reject any bookings that don’t specify a notification email address (via the \texttt{email} attribute on the \texttt{Work} element).

If present and set to 1, the parameter \texttt{RM\_MUST\_GIVE\_PROJECT} will make the RM reject any bookings that don’t specify a Project ID (via the \texttt{project} attribute on the \texttt{Work} element).

\begin{verbatim}
RM\_MUST\_GIVE\_EMAIL=1
RM\_MUST\_GIVE\_PROJECT=1
\end{verbatim}

3.2.6 Generic: Authorization Module

By default, every HARC RM has a grid-mapfile, which controls who may use the RM. Typically, for Compute RMs, this is linked to the Globus grid-mapfile in \texttt{/etc/grid-security}.

It is possible to change to using the UK NGS scheme for pool accounts, instead. To set this up, you need to configure the following parameters in your \texttt{install.config}.

\texttt{RM\_AUTHZ\_MODULE} Set this to “NGSPoolLookup”;

\texttt{RM\_AUTHZ\_NGS\_POOL\_LOOKUP\_SCRIPT} Set this to the ful path of the setuid script which looks up the user names for pool DN;

3.2.7 Generic: Resource Manager Type

\texttt{RM\_INNER\_TYPE} What type of RM is this?

At the moment, only two types are supported by classes in the CVS.

\texttt{SimpleCompute} For reserving time on supercomputers.

\texttt{SimpleNetwork} For reserving time on some GMPLS-controlled networks.

The configuration of SimpleCompute RMs is described in Sections 3.2.8 to 3.2.10; the configuration of SimpleNetwork RMs is described in Section 3.2.12.
For those who want to know more  The RM_INNER_TYPE parameter controls which subclass of InnerRM is instantiated in MainLoop.pm; see the initialize subroutine. People adding new types of RM would need to modify this loop in addition to creating a new module inheriting from InnerRM. The process of writing a new RM for a new type of resource is covered in Section 6 (currently very incomplete).

3.2.8 SimpleCompute: Nodename

All Simple Compute RMs accept messages from Acceptors for Resources that contain an element like the following:

<Compute>man2.nw-grid.ac.uk<Compute>

The RM_COMPUTE_NODENAME parameter should be set to the public name of the machine. This is the name that will typically be used to reference the machine from the command line.

3.2.9 SimpleCompute: Core Count and Memory

The current implementation of the Compute Resource Manager keeps its own timetable of reservations that have been made through that RM. The idea behind this was to allow the RM to answer questions about free time slots which is difficult to get from some schedulers. Experience has shown that this information tends not to be accurate anyway. Further, the current implementation checks to see that incoming bookings fit into the timetable before contacting the scheduler; those that don’t fit are rejected by the RM without consulting the scheduler. This is not ideal; there’s a danger that a configuration mistake would reject something that might’ve worked. In the future, the Core Count and the memory configuration will only be used for advertising the description of the resource; all incoming work will go to the scheduler, and no timetable will be kept in the RM. The parameter giving the number of Cores per node will still be used when talking to some schedulers.

The number of cores that the machine must be configured by setting the parameter RM_COMPUTE_CPUS. If your batch system gives out cores in some multiple (e.g. one cluster node at a time), then you should set the optional parameter RM_COMPUTE_CPU_BLOCK to this multiple, (e.g. the number of cores per node). RM_COMPUTE_CPUS must be a multiple of RM_COMPUTE_CPU_BLOCK.

There are currently two supported memory configurations, both very simple:

RM_COMPUTE_MEMORY_MB_PER_CPU Distributed memory cluster, with a set amount of memory per core; or

RM_COMPUTE_MEMORY_MB_ABSOLUTE Shared memory machine, with an overall amount of memory.

For clusters, you simply divide the memory per node by the number of cores per node. Obviously, this does not model heterogenous clusters. However, this can be set to a kind of upper bound. The specified memory is still sent to the scheduler, which would place the reservation on high-memory nodes if necessary. This configuration item is returned in the resource description, which might be used for
brokering. If you wish, it is possible to configure the RM to perform a memory check, rejecting jobs that request more memory than the number of cores multiplied by the memory per core; see below.

For shared memory (or Single System Image) machines, you specify the overall amount of memory. If the memory check is enabled (see below), HARC will require that all incoming jobs to such resources specify the amount of memory that is required; a tally of these will be kept. By default, the handling of memory requirements is left to the scheduler.

To enable the memory check, set **RM_COMPUTE_CHECK_MEMORY** to 1, i.e.

```
RM_COMPUTE_CHECK_MEMORY=1
```

*As noted at the start of this section, this feature will disappear in the next major version of the RM.*

### 3.2.10 SimpleCompute: Batch Scheduler and SUDO/ACL mode

The parameter **RM_COMPUTE_BATCH_TYPE** must be set to indicate the type of batch scheduler. Currently, the types supported by modules in the HARC CVS are:

- **PBSPro** The PBSPro scheduler;
- **TorqueMaui** The Torque scheduler, using Cluster Resource’s Maui scheduler for advance reservation;
- **TorqueMoab** The Torque scheduler, using Cluster Resource’s Moab scheduler for advance reservation;
- **TorqueCatalina** The Torque scheduler, using SDSC’s Catalina scheduler for advance reservation;\(^1\)
- **LoadLeveler** The LoadLeveler scheduler;
- **LSF** Platform’s LSF;
- **SunGridEngine** SGE 6.1 with Advance Reservation preview (not documented further here).\(^2\)

Each of these schedulers requires its own configuration parameters, as follows.

All of the batch scheduler modules currently in CVS can be run in either ACL mode or SUDO mode, except the **TorqueCatalina** scheduler, which only works in SUDO mode. In ACL mode, the reservations are made by the same user, who is running the service. Users are able to access the reservations because their username is specified in an ACL at the time the reservation is made.

In SUDO mode, the RM will use sudo to create/cancel/query the reservations. To make SUDO work, the user running the service must be given additional privileges, i.e. they must be able to run the commands to make/cancel/query reservations as any user of harc. For details of how to configure the sudoers file to do this, see the end of this section. SUDO mode works best in terms of accounting and quota systems, because the reservations are made by the correct user; this mode is recommended for production use.

The parameter **RM_COMPUTE_BATCH_USE_SUDO** controls which mode is used: 0 tells the RM to use ACLs; 1 tells the RM to use sudo. If this is set to 1, then the parameter **RM_COMPUTE_BATCH_SUDO_COMMAND** must be set too, to the full path of the sudo command, e.g. `/usr/bin/sudo`\(^1\)

---

\(^1\) As Catalina is slower than the other schedulers to make reservations, it can only be partially supported, until the RMs become multi-threaded.

\(^2\) Once the functionality is finalized and appears in 6.2, this documentation will be added.
PBSPro On clusters, it is important to set \texttt{RM\_COMPUTE\_CPU\_BLOCK} to the number of Cores per node in the cluster. This allows the RM to work out how many nodes to ask for, based on the requested CPU Count. (For Single System Image machines, this number must be set to 1.)

Set the parameter \texttt{RM\_PBSPRO\_COMMAND\_DIR} to the directory containing the Maui commands \texttt{pbs\_rsub}, \texttt{pbs\_rdel}, \texttt{pbs\_rstat}, e.g.

\begin{verbatim}
RM\_PBSPRO\_COMMAND\_DIR=/usr/local/pbs/bin
\end{verbatim}

\texttt{HARCRESV} commands for SUDO mode:

\begin{itemize}
\item \texttt{pbs\_rsub};
\item \texttt{pbs\_rdel}; and
\item \texttt{pbs\_rstat}.
\end{itemize}

In addition, if you are using groups to control which users may make reservations, so that all permitted users are in the same group, which is \textit{not} their primary group, then their is an extra parameter that, when set, will supply an argument to the \texttt{pbs\_rsub} command via the \texttt{"-g"} flag. This is set as follows:

\begin{verbatim}
RM\_PBSPRO\_RESERVATION\_GROUP=harcusers
\end{verbatim}

TorqueMaui On clusters, it is important to set \texttt{RM\_COMPUTE\_CPU\_BLOCK} to the number of Cores per node in the cluster. This allows the RM to work out how many nodes to ask for, based on the requested CPU Count. (For Single System Image machines, this number must be set to 1.)

Set the parameter \texttt{RM\_MAUI\_COMMAND\_DIR} to the directory containing the Maui commands \texttt{setres}, \texttt{releaseres}, \texttt{showres}, e.g.

\begin{verbatim}
RM\_MAUI\_COMMAND\_DIR=/usr/local/maui/bin
\end{verbatim}

\texttt{HARCRESV} commands for SUDO mode:

\begin{itemize}
\item \texttt{setres};
\item \texttt{releaseres}; and
\item \texttt{showres}.
\end{itemize}

TorqueMoab On clusters, it is important to set \texttt{RM\_COMPUTE\_CPU\_BLOCK} to the number of Cores per node in the cluster. This allows the RM to work out how many nodes to ask for, based on the requested CPU Count. (For Single System Image machines, this number must be set to 1.)

Set the parameter \texttt{RM\_MOAB\_COMMAND\_DIR} to the directory containing the Maui commands \texttt{mrsvctl}, \texttt{showres}, e.g.

\begin{verbatim}
RM\_MOAB\_COMMAND\_DIR=/usr/local/moab/bin
\end{verbatim}
HARCRESV commands for SUDO mode:

- mrsvctl; and
- showres.

**TorqueCatalina** On clusters, it is important to set `RM_COMPUTE_CPU_BLOCK` to the number of Cores per node in the cluster. This allows the RM to work out how many nodes to ask for, based on the requested CPU Count.

Set the parameter `RM_CATALINA_COMMAND_DIR` to the directory containing the Maui commands `user_set_res`, `show_res` and `user_cancel_res`, e.g.

```bash
RM_CATALINA_COMMAND_DIR=/usr/local/catalina/bin
```

HARCRESV commands for SUDO mode:

- `user_set_res`;
- `show_res`; and
- `user_cancel_res`.

Note that Catalina does not provide ACLs for reservation, and so, although ACL mode can be used to test your configuration options, it will not work properly; use SUDO mode.

**LoadLeveler** On clusters, it is important to set `RM_COMPUTE_CPU_BLOCK` to the number of Cores per node in the cluster. This allows the RM to work out how many nodes to ask for, based on the requested CPU Count.

Set the parameter `RM_LL_COMMAND_DIR` to the directory containing the LoadLeveler commands `llmkres`, `llrmres`, `llqres`, e.g.

```bash
RM_LL_COMMAND_DIR=/usr/lpp/LoadL/full/bin
```

HARCRESV commands for SUDO mode:

- `llmkres`;
- `llrmres`; and
- `llqres`.

**LSF** On clusters, it is important to set `RM_COMPUTE_CPU_BLOCK` to the number of Cores per node in the cluster. This allows the RM to work out how many nodes to ask for, based on the requested CPU Count.

Set the parameter `RM_LL_COMMAND_DIR` to the directory containing the LoadLeveler commands `brsvadd`, `brsvdel`, `brsvs`, e.g.
RM_LSF_COMMAND_DIR=/apps/lsf/6.2/linux2.6-glibc2.3-x86_64/bin

In addition, if the reservations must be made on a specific host, or within a specific host group (as sent to the brsvadd using the “-m” flag, then you may define this as follows:

RM_LSF_HOST_GROUP=ngs_group

In addition to these settings, the installer will copy the following environment variables from install.config which are important to making sure that LSF functions correctly:

- LSF_SERVERDIR;
- LSF_LIBDIR;
- LSF_BINDIR;
- XLSF_UIDDIR; and
- LSF_ENVDIR.

You should also make sure that LSF_LIBDIR is in your LD_LIBRARY_PATH.

The HARCRESV commands for SUDO mode are:

- brsvadd;
- brsvdel; and
- brsvs.

Configuring the sudoers file for SUDO mode. To run the RM in SUDO mode, you need the account that is running the RM to be able to run the reservation commands (specified above) as any user of HARC, without a password. It is possible to configure this accurately so that no other commands can be run, and no other user.

Here is an example of the three lines that are required to accurately configure the sudo command. To edit the file /etc/sudoers file, which controls sudo, you should use the command visudo. Here, the account harc will be used to run the RM, and we are using Torque/Maui as the example.

Runas_Alias HARCUSERS=%harcusers

Cmd_Alias HARCRESV=/usr/local/maui/bin/setres, \
/usr/local/maui/bin/releaseres, \
/usr/local/maui/bin/showres

harc ALL=(HARCUSERS) NOPASSWD: HARCRESV

The first line specifies the users that the RM should be able to run the commands as. This can either be a list of usernames (separated by commas), or you can set up a specific group as shown here (harcusers).
Using a group means that you only have to edit /etc/group to make changes, rather than editing the sudoers file, when adding a new user. If you want all users to be able to use HARC, then omit this line, and change HARCUSERS to ALL in the 3rd line.

The second line specified the commands that the RM can run under the other accounts. This line must specify only those commands listed in the sections above, and must specify the full paths for each.

The third line declares that the user harc can execute these commands using sudo for the specified users, without specifying a password.

For those who want to know more  The RM_COMPUTE_BATCH_TYPE parameter controls which subclass of SCBatch is instantiated in SimpleCompute.pm; see the initialize subroutine towards the bottom of the module. People adding new types of RM would need to modify this loop in addition to creating a new module inheriting from SCBatch. This process is described later in Section 5.

3.2.11 SimpleCompute: Globus Jobmanagers

It is possible to alter your Globus installation to support job submission to advance reservations, as described in a separate document. Once this has been done, you may wish to advertise the details, so that clients can easily see how to submit jobs to the reservations made with HARC.

The RM_COMPUTE_PRE_WSGRAM_RMC parameter specifies the pre-WS GRAM (i.e. GT2 style) Resource Manager Contact string that would be specified either as a command-line argument to globus-job-run, or in RSL supplied to globusrun. The two parameters RM_COMPUTE_WSGRAM_ADDRESS and RM_COMPUTE_WSGRAM_RESOURCE_ID specify the WS GRAM equivalent; the Endpoint of the Managed Job Factory Service, and the “resource ID”, which is typically the name of the scheduler, and maps down to the pre-WS GRAM jobmanager.

These parameters are all optional. It is possible to specify just the pre-WS GRAM settings, or just the WS GRAM settings (both WS GRAM parameters must be set), or both. This information is used to create an element in the resource description, which would look as follows:

```xml
<Description>
  <SimpleCompute>
    <CPUCount blockSize="4">8</CPUCount>
    <MemoryMB>
      <PerNode>4096</PerNode>
    </MemoryMB>
    <Globus>
      <PreWSGRAM>man2.nw-grid.ac.uk/jobmanager-loadleveler</PreWSGRAM>
      <WSGRAM>
        <Address xmlns="http://schemas.xmlsoap.org/ws/2004/03/addressing">
          https://man2.nw-grid.ac.uk:8443/werf/services/
          ManagedJobFactoryService</Address>
        <resourceID xmlns="http://www.globus.org/namespaces/2004/10/gram/job">
          LoadLeveler</resourceID>
      </WSGRAM>
    </Globus>
  </SimpleCompute>
</Description>
```

3 At the time of writing, this is still not supported as standard, although it is hoped that it will be soon. For now, there is a document describing how to do this at: http://www.cct.lsu.edu/~maclaren/HARC/Documentation/globus.pdf
3.2.12 SimpleNetwork

3.3 CA Certificates

In the configuration directory, create a subdirectory cacerts, and place the public part of each of the CA Certificates that have been used to sign any of the Acceptors that the RM is going to be configured to respond to. You will need all the CA Certificates going up to the self-signed CA at the top of the certificate chain.

Each certificate should be placed in the directory in PEM format, with a .crt suffix (the installer will create hashed names from these).

3.4 Acceptor Authorization File

The file acceptor-dns should be added to the configuration file. This is a list of the DNs of the Acceptors and the CA Certificates that sign them, i.e. all the DNs of the Acceptors and their certificate chains should be present in this file. This is similar to the format of the Globus Toolkit grid-mapfile, but only the field containing the DN is present.\(^4\) An example file is:

```plaintext
"/C=US/ST=Louisiana/L=Baton Rouge/O=LSU/OU=CCT/CN=EnLIGHTened HARC Acceptor Bodie/emailAddress=maclaren@ctt.lsu.edu"
"/C=US/ST=Louisiana/L=Baton Rouge/O=LSU/OU=CCT/CN=EnLIGHTened HARC Acceptor Currituck/emailAddress=maclaren@ctt.lsu.edu"
"/C=US/ST=Louisiana/L=Baton Rouge/O=LSU/OU=CCT/CN=EnLIGHTened HARC Acceptor Hatteras/emailAddress=maclaren@ctt.lsu.edu"
"/C=US/ST=Louisiana/L=Baton Rouge/O=LSU/OU=CCT/CN=EnLIGHTened HARC Acceptor Lookout/emailAddress=maclaren@ctt.lsu.edu"
"/C=US/ST=Louisiana/L=Baton Rouge/O=LSU/OU=CCT/CN=EnLIGHTened HARC Acceptor Ocracoke/emailAddress=maclaren@ctt.lsu.edu"
"/O=Louisiana State University/OU=CCT/OU=ca.cct.lsu.edu/CN=CCT CA"
```

3.5 User Authorization File

The grid-mapfile against which the RM will authorize users. Again, this is in the format of a Globus grid-mapfile. Rather than having its own copy of a grid-mapfile, it is common to want the RM to use a copy of the Globus grid-mapfile. If you do not create a grid-mapfile in your configuration directory, then the installer will create a symbolic link from:

```
/etc/grid-security/grid-mapfile
```

This means that the RM will allow all those users who are able to run Globus jobs to make reservations via HARC.

4 Installing the Resource Manager

Installing the RM should be trivial. This is done by the install-rm script in $HARC/rm-service/scripts. Note that when the script is initially checked out from CVS, it will probably not be set to be executable.\(^4\) This replaces the now-deprecated acceptor-mapfile which contained a second, redundant field of dummy user-names.
chmod a+x $HARC/rm-service/scripts/install-rm

To install the RM

$HARC/rm-service/scripts/install-rm $HOME/rm-config /usr/local/man2-rm

Be sure to specify the absolute path to the configuration directory; if you do not, it will be interpreted as being a subdirectory of $HARC/rm-service/config. Note that this script assumes that the openssl is in your PATH; if this is not the case, then add it to your PATH before you run the installer.

When you run this command, you will see an output similar to this:

    rm-service $ $HARC/rm-service/scripts/install-rm $HOME/rm-config /usr/local/man2-rm
    Makefile.crt ... Skipped
    cct-ca.crt ... 5fb2fc80.0
    old-uk-escience-ca.crt ... 01621954.0
    uk-escience-ca.crt ... adcbc9ef.0
    uk-escience-root.crt ... 8175c1cd.0
    Notice: Don’t forget to place your certificate and key files at:
              /usr/local/man2-rm/x509/server_cert.pem
              /usr/local/man2-rm/x509/server_key.pem

In addition to copying the Perl modules for the RM, the installer creates a number of scripts to help run the RM, as well as allowing the RM to be stopped and started for maintenance.

4.1 Installing the RM Certificate

Before the RM can be started, you will need to place the public and private parts of the RMs X509 credential into the locations given by the installer. Please make sure that the private key is not readable by anyone other than yourself, e.g.

    chmod 400 /usr/local/man2-rm/x509/server_key.pem

This key file must be passwordless. If the key file you have contains a password, you can create a passwordless version as follows:

    mv server_key.pem server_key.pem.pass
    openssl rsa -in server_key.pem.pass -out server_key.pem
    chmod 400 server_key.pem

If your signed credential exists as a PKCS#12 bundle, you can extract the public and private parts as follows:

    openssl pkcs12 -in bundle.p12 -clcerts -nokeys -out server_cert.pem
    openssl pkcs12 -in bundle.p12 -nocerts -nodes -out server_key.pem

You will need to enter a password to unlock the PKCS#12 bundle.
4.2 Running Interactively to Test the RM Configuration

The installer creates a script called commands in the install directory. This is usually run via cron (see below). But for testing purposes in can be run interactively, using:

   ./commands -i

4.3 Configuring the RM to Start via cron

The RM is designed to be started by cron. This also allows the RM to be quickly restarted in the case that it has crashed, or when the system is rebooted. The cron job will only start the RM if:

- The RM is not running; and
- The RM is not in a stopped state for maintenance.

The file crontab in the install directory will contain a line that can be added to your crontab. If you do not already have a crontab, then you can set this file to be your crontab by typing:

   crontab crontab

If you choose to cut and paste the file into your crontab, then you must take great care to ensure that the TAB character following the fifth asterisk/star does not get converted to a set of spaces.

Note that the RM will not yet start up. This is because the RM is initially put into a stopped state, and will not start until started with the start-rm script described in the following section.

4.4 Starting and Stopping the RM

There are two scripts created in the installation directory, start-rm and stop-rm. These can be used to start and stop the RM. start-rm relies upon the RM having been configured to start via cron (see the previous section).

To start the RM, go to the installation directory, and type:

   ./start-rm -w

This will set the RM to be restarted the next time that cron runs the rerun script (by removing the control file .do_not_restart), and will wait until the RM has been restarted.

To stop the RM, go to the installation directory, and type:

   ./stop-rm

This will stop the rerun script from restarting the RM (by creating a zero-byte file .do_not_restart in the installation directory), and then attempt to stop the RM, first by sending it SIGINT, and later, if the RM does not stop, with SIGKILL.
Both scripts are designed to be useful for inclusion in other scripts. They provide verbose, timestamped output, and provide useful return statuses. So that it does not wait forever, `start-rm` can be given a number of minutes to wait, e.g.

```
./start-rm -w 5
```

If the RM does not start at the end of the interval, it exits with a non-zero status. Zero is returned if the RM started or was already running.

Similarly, if it fails to bring down the RM, `stop-rm` will exit with a non-zero status. Zero implies that the RM was stopped, or was not running to begin with.

### 4.5 Updating the RM

To update an RM, for example with bug-fixes from CVS, it should be possible to:

1. Stop the RM with `stop-rm`;
2. Run the install command again, with the same configuration;
3. Start the RM with `start-rm`.

### 5 Adding a New Module for a New Scheduler

As stated in Section 3.2.10 (see “Those who want to know more), the module `SimpleComputeRM.pm` instantiates a module that inherits from `SCBatch` to manage the interactions with the scheduler. These modules are simple, containing less than 200 lines of Perl.

This section describes how to create a new module for interacting with a scheduler not supported in the current CVS.

#### 5.1 Creating the Module

Choose a good, descriptive name for the module, and have it start with “SCBatch”, e.g. `SCBatchMyScheduler.pm`. A stub is given here. But you should also look at the code for other schedulers in CVS; this will help a lot.

**Stub**

```perl
package SCBatchMyScheduler;
use base 'SCBatch';
```

---

5 Obviously this assumes that the patches being applied do not mandate some change in the configuration parameters, or the specification of additional parameters.

18
use warnings;
use strict;
use Carp;
use action_enum qw(:constants);
use Util qw(log_message);
use Date::Manip;
use status qw(:constants);
use Data::UUID;

....

return 1;

5.2 Modifying SimpleComputeRM to use it

SimpleComputeRM chooses the scheduler type based upon the value of the environment variable RM_COMPUTE_BATCH_TYPE. This is done in the initialize method, near the bottom of SimpleComputeRM.pm. In the following excerpt, the two added lines begin with an asterisk. Your scheduler module should be used if the $batch_type is set to the "MyScheduler".

if($batch_type eq "DUMMY") {
    $self->{BATCH}=SCBatchDummy->new();
} elsif($batch_type eq "TorqueMoab") {
    $self->{BATCH}=SCBatchTorqueMoab->new();
} elsif($batch_type eq "TorqueMaui") {
    $self->{BATCH}=SCBatchTorqueMaui->new();
*     $self->{BATCH}=SCBatchMyScheduler->new();
* } elsif($batch_type eq "MyScheduler") {
*     $self->{BATCH}=SCBatchMyScheduler->new();
} else {
    confess "Unknown RM_COMPUTE_BATCH_TYPE ".$batch_type."";
}

Also, near the top of SimpleComputeRM.pm, you will need to add a use statement like:

use SCBatchMyScheduler;

Look for the other use statements; it should be clear where this goes.

5.3 Overriding methods

5.3.1 Constructor

As there is a separate initialize method, you don’t need to put much here.

Stub
sub new {
    my $this = shift;
    my $class = ref($this) || $this;
    my (%params) = @_;

    log_message("SCBatchMyScheduler->new called");
    my $self = $class->SUPER::new();
bless ($self, $class);
    return $self;
}

5.3.2 initialize

Input Parameters  Only a single parameter, which is a handle to the object upon which the method was invoked.

Description  This method should read in any additional configuration items from environment variables. You should, for example, allow the specification of the location of the scheduler commands to be specified this way, and not hard code it. For example, in the variable 
RM_MY_SCHEDULER_COMMAND_DIR, to be consistent with the modules already in CVS.

Note that new modules for new schedulers should ideally support the two ways of making reservations that are explained above in Section 3.2.10, i.e. using ACLs, which is best for testing, and using sudo, which is best for production. If your scheduler does not provide ACLs, then you should make sure that your initialize method fails if the configuration parameter RM_COMPUTE_BATCH_USE_SUDO was set to 0 (for ACL), by inserting the following line just after the superclass initialize method is invoked.

    confess "This scheduler only supports SUDO mode, not ACL - please set RM_COMPUTE_BATCH_USE_SUDO to 1" unless ($self->{USE_SUDO} eq 1);

If you need to use UUIDs for your reservation name (see the next section), you should initialize a UUID generator here (see SCBatchTorqueMaui.pm).

Return Status  Returning at all means everything is fine. If something is missing, you should terminate the process using die, or, preferably confess.

Stub

    sub initialize {
        my $self=shift;

        $self->SUPER::initialize();

        ...
    }
5.3.3 makeReservation

Input Parameters  Following the first parameter, which is a handle to the object upon which the method was invoked, you will receive:

user_id the user who is making the reservation, based upon the DN that was presented, and the grid-mapfile;

cpu_blocks the number of blocks of cores being reserved;

cpu_block_size the number of cores per block;

memoryMB the amount of memory required, in MegaBytes;

start the start time for the job; and

end the end time for the job.

email the email address for any notifications. This may be undefined, unless the RM has been configured to make email notification addresses mandatory; see 3.2.5.

project the Project ID for this booking. This may be undefined, unless the RM has been configured to make project IDs; see 3.2.5.

Notes  The method makeReservation will receive start and end times which are specified absolutely (i.e. not relative to the current date). Checks have already been made to ensure that the start date is not in the past, and that the end date is later than the start date. These are correct with respect to timezones.

Both dates are objects from the Date::Manip package. For details on how to manipulate these, have a look at:

http://www.cise.ufl.edu/~sbeck/DateManip.html

However, the existing scheduler modules probably contain enough code so that you can cut and paste the code from there.

The reservation must be created for it to be accessible by the specified user. If the RM is being run in ACL mode, then you should create the reservation as the user running the RM, but use the ACL facility of the scheduler to allow the user (specified by $user_id) to access the reservation. (The user ID has already been looked up from the grid-mapfile by the RM.) If the scheduler does not support ACLs, then you should not allow the RM to run in this mode; see Section 5.3.2 above. If running in SUDO mode, you should instead prefix the command with sudo, and forget about the ACL. This can be done with code like the following (taken from SCBatchTorqueMaui):

```perl
my $command="";

if($self->{USE_SUDO} eq 1) {
    $command=$self->{SUDO_COMMAND}." -u ".$user_id." ";
}
```

...
The initialization of both the USE_SUDO and SUDO_COMMAND is dealt with in the SCBatch initialization method (i.e. don't worry about these).

It is important that the names of reservations do not repeat rapidly; this can cause confusion. For example, with Torque/Maui, if you make a reservation, the default name is username, plus a period, plus a number, e.g. harc.0. But, if you cancel this, and make another, it will also be harc.0. The same happens if the first reservation finishes, and then you make another. If your scheduler does this, then you should suggest more unique names, e.g. using UUIDs, as in SCBatchTorqueMaui.

You should use log_message to output the command you actually use to make the reservation, and probably log the response too. Be careful parsing things, please.

You should put the name of the command used to talk to the scheduler in a constant definition near the top of the module.

Return Status  makeReservation returns a tuple of two strings, the first representing the reservation ID, and the second an error. If there is any problem, undef MUST be returned for the ident string, and the error string MUST be defined. If the ident string is defined, SimpleComputeRM assumes success.

Stub

```perl
sub makeReservation {
    my $self=shift;
    my $user_id=shift;
    my $cpu_blocks=shift;
    my $cpu_block_size=shift;
    my $memoryMB=shift;
    my $start=shift;
    my $end=shift;
    my $email=shift;
    my $project=shift;

    my $ident=undef;
    my $error=undef;

    log_message("SCBatchMyScheduler - makeReservation called");

    ...

    return($ident,$error);
}
```

5.3.4 cancelReservation

Input Parameters  Following the first parameter, which is a handle to the object upon which the method was invoked, you will receive:
**Description**  This routine should cancel the reservation specified. This is being done on behalf of the identified user, although you probably won't need to refer to the user id.

As with `makeReservation`, put the command name at the top of the module; log the command you are issuing; log the output.

As with `makeReservation`, you should use `sudo`, if running in SUDO mode (see above).

**Return Status**  There isn’t one, as this is being done on the 2nd phase of the commit. You just have to try your best...

**Stub**

```perl
sub cancelReservation {
    my $self=shift;
    my $user_id=shift;
    my $ident=shift;
    my $email=shift;
    my $project=shift;

    log_message("SCBatchMyScheduler - cancelReservation called");

    ...
}
```

### 5.3.5 `getStatus`

**Input Parameters**  Following the first parameter, which is a handle to the object upon which the method was invoked, you will receive:

- **user_id** the user who is making the reservation, based upon the DN that was presented, and the grid-mapfile;
- **ident** the reservation ID being canceled;
- **email** the email address for any notifications. This may be undefined, unless the RM has been configured to make email notification addresses mandatory; see 3.2.5.
- **project** the Project ID for this booking. This may be undefined, unless the RM has been configured to make project IDs; see 3.2.5.
end  the end time for the job.

Description  There are a set of statuses defined in status.pm. If you copied the stub for the module, these are available to be used as written below, as constants (see SCBatchTorqueMaui for example). These constants have the following meaning:

STATUS_RESERVED  The reservation is known to the scheduler, but has not started;

STATUS_ACTIVATED  The reservation is known to the scheduler, and is currently active; and

STATUS_UNKNOWN  The reservation is not known to the scheduler, or has finished.

This method should interact with the scheduler to find out about the reservation. The start and end times are provided in case the scheduler doesn’t tell you much about the reservation, and you need to fake the results based on the current time. Have a look at SCBatchTorqueMoab.pm which does things properly, and SCBatchTorqueMaui.pm which is a bit of a hack.

As with makeReservation, put the command name at the top of the module; log the command you are issuing; log the output.

As with makeReservation, you should use sudo, if running in SUDO mode (see above).

Return Status  getStatus returns a tuple; the status and error message. If status is defined, the call is assumed to have succeeded, and the error message is discarded. If status is undefined, then the error string is passed back and is seen by the client.

Stub

sub getStatus {
    my $self=shift;
    my $user_id=shift;
    my $ident=shift;
    my $start=shift;
    my $end=shift;

    my $status=undef;
    my $error=undef;

    log_message("SCBatchMyScheduler - getStatus called");

    ...

    log_message("SCBatchTorqueMaui - status of $ident is $status");

    return($status,$error);
}
5.4 Changing the Installer

The installer script at $HARC/rm-service/scripts/install-rm should be modified so it can install your new scheduler RM module properly. This is typically very minor. The installer needs to propagate the extra environment variables you need in your initialize method from the install.config to the startup scripts for the RM.

To do this, search the installer for TorqueMaui. Just below that section, add lines for your scheduler. So, if we were using the examples from the previous sections, this part of the file should now look as follows (new lines starting with an asterisk):

```perl
TorqueMaui )
    add_to_commands "export RM_MAIUI_COMMAND_DIR=$RM_MAIUI_COMMAND_DIR"
    ;
* MyScheduler )
    add_to_commands "export RM_MY_SCHEDULER_COMMAND_DIR=$RM_MY_SCHEDULER_COMMAND_DIR"
    ;
* ) echo "$command: Unknown batch type: $RM_COMPUTE_BATCH_TYPE" >&2 ; exit 1;
```

That should be all that’s required (the Perl module will get copied correctly anyway).

5.5 Contributing back to the community

If you are contributing your scheduler module back to the community, then send me your new module, the altered SimpleComputeRM.pm and install-rm, and I will put them in. Put whatever copyright statement you wish at the top of your module; got to be an open source license. You can email the stuff to mailto:maclaren@cct.lsu.edu.

6 Adding a New RM for a New Type of Resource

This section is still to be written. (Sorry.)

See Section 3.2.7 ("Those who want to know more).

Steps for creating a new RM:

- Design your XML
- Resource element
- Work element
- Create a new subclass of InnerRM.pm
- Use the utility classes where possible
- To extend the API, create subclasses of
Customizing InnerRM. Override:

- Startup/shutdown
- initialize/remove
- Parsing (validating) the XML
- parseResourceElement
- parseWorkElement
- maybe parseScheduleElement
- Co-allocation
- tryMakeAction
- tryCancelAction
- addResourceBookings
- completeTransactionBookings
- Others for getTimetable/getStatus

7 RM Caveats

The RM code is pretty complete, but the following major things still need work:

- RMs currently need to be restarted to re-read the grid-mapfile
- RMs need to be restarted after the switch to/from Daylight savings time
- When restarted, they forget the bookings...
  Want to add persistence so that it’s trivial for RM developers to utilize