Owen™ Grid: Scheduler Administration

1. Notes for Owen™ Grid Administrators

This document is aimed at users who are responsible for the administration of an Owen™ Grid. In particular it describes the log files that are created both by the scheduler and each client which can be used to investigate problems if they occur.

1.1. Server Side

1.1.1. Log Files

The scheduler writes various events to a log to enable the user to see what it has been doing, this is mainly used for fault tracking and debugging purposes. The log can be a file or a directory. A log file is created as follows:

```
echo -n > /grid/master/log
```

All logged events will be appended to this file. You can instead create a log directory:

```
mkdir /grid/master/log
```

The scheduler will then create a series of log files within that directory. Each file name will be of the form `log.`*time* where *time* is a 10 digit timestamp of the time the file was created. This number can be given to the `date` command to see the date and time in conventional form. For example, given `log.1088030774`:

```
% date 1088030774
Wed Jun 23 23:46:14 BST 2004
```

It is usually better to use a log directory not a single log file, as it enables the scheduler to use the rollover function described below.

1.1.2. Rollover

Periodically, the scheduler starts logging to a new file. This partitions up the logs and enables the user to more easily find or remove a log for a particular time as well as preventing the build up of huge files. This only happens, however, when `/grid/master/log` is a directory. By default, rollover occurs once a week; this can be changed with the `-L` option to the scheduler:

`-L` *loginterval*

where *loginterval* is an integer followed by a letter. Possible letters are `ms` (milliseconds), `s` (seconds), `m` (minutes), `h` (hours) and `d` (days). The default time period is 1 week.

1.1.3. File Format

Each event is represented by a line in the log file, containing space separated fields. The first field on each line is a 10 digit field containing the timestamp of the event in number of milliseconds since the scheduler was started. The remaining fields describe the event. The first event in each log file is always `starttime` *time* where `time` is the time the scheduler was started in milliseconds since 1 January 1970. For example:

```
0000000000 starttime 1087832739784
0000002345 slave arrive 192.168.72.254!25984 inferno
0000002788 slave name bob 192.168.72.254!25984
0008727474 task new bob 192.168.72.254!25984 8c68c89f.4#3 tgid 2
```

To find the actual time of an event, simply add the event time offset to the time the scheduler was started (you will need to remove the last 3 digits of the result to pass it to `date`). Thus,

1087832739784 + 0008727474 = 1087841467(258)

and give the result to `date`:

```
% date 1087841467
Mon Jun 21 19:11:07 BST 2004
```

The format of each logged event depends on the event but as a general rule, all job related events will have at least the job's unique id associated with them and node events will hold the name and IP address of the node.

1.1.4. Debugging

The scheduler can be made to log events more verbosely by specifying the verbose flag `-v` in the command line arguments. This function is useful when troubleshooting as it provides a much more detailed picture of what is going on. However, be aware that the increased output can result in very large (hundreds of MB) log files, so it is recommended that this option is not used if disk space is limited.

1.1.5. Working Area

The `/grid/master/work` directory is the scratch directory for the scheduler. Each job has its own directory called `/grid/master/work/`*uniqueid*`.`*jobno* where *uniqueid* is a random hexadecimal number and *jobno* is the number of the job. These directories are used to store all the files required to execute the job: temporary data, results etc. The *uniqueid* ensures that if the scheduler is restarted with no dump to restore from and a new job is started with the same job number as one that had been running before, it will not overwrite the old data. This is important if, for example, the old data was from a semi-complete job and might provide information as to why it did not succeed.

1.1.6. Blacklisting

When a client repeatedly fails tasks, the scheduler will automatically *blacklist* the client and not give it any more work. This stops a bad client from ruining a job and enables the grid administrators to be able to easily identify problem clients. Once the client has been repaired, it may be removed from the blacklist using the Node Monitor and will start receiving work again.

Note: This scheme assumes that the only a small minority of tasks within a job can fail on a good client. If a job is started where most of the tasks fail even on good clients, then potentially all the clients could end up being blacklisted.

Periodically (by default, every 10 minutes) clients are automatically unblacklisted. This can be changed using the `-U` option:

`-U` *unblacklistinterval*

Where *unblacklistinterval* is the period of time before a client is automatically unblacklisted and uses the same format as the rollover interval `-L`.

1.1.7. Restarting the Scheduler

To restart the scheduler, simply kill off the old process and re-run the startup script. The scheduler will automatically restore its state from the most recent dump file and continue with the work it was doing.

1.1.8. Dump Files

The scheduler periodically dumps its current working state so that if it is shut down or crashes, it can carry on from the most recent dump. The interval between dumps is specified as an option to the scheduler of the form:

`-D` *dumpinterval*

where *dumpinterval* is the time between dumps and uses the same format as the log rollover interval option `-L`. By default, the state is dumped every 10 seconds.

The dump files are all stored in /grid/master and have one of three possible names:

`dump`	The most recent dump file.
`dump.old`	The previous dump file.
`dump.restore`*XXXXXXXXXX*	A copy of the dump file that was used to restore the scheduler. *XXXXXXXXXX* is a 10 digit timestamp of the time when the scheduler was restored.

When the scheduler is restarted, it automatically tries to restore its state from the dump files. It first tries `dump` and then, if that does not exist, or is corrupted, from `dump.old`.

The restore files may be used to restore the scheduler to a known point. For example, if the scheduler was restarted but a required file for one of the current jobs was missing, causing the restoration to fail then the original dump file could be overwritten. This would mean that even after the missing file had been retrieved, the job would be forgotten. However, copying the last restore file over the dump file and then restarting the scheduler would cause it to restart at the same point as before (i.e. with all the original jobs) but this time with the required file in place.

1.1.9. Troubleshooting

Here is a list of some potential problems along with the most likely solution:

- **Job just hangs**
  Check that there is at least one available client that is capable of processing the tasks for the job. Also, ensure that the task name is spelt correctly.
- **Clients all get blacklisted**
  There is probably a problem with the job. Make sure that all required executables and data files are installed and available to the clients.
- **No clients appear**
  Check that the scheduler and clients are all using the same authentication mode.

1.2. Client Side

1.2.1. Log Files

Each client records significant events in its own log file: `/grid/slave/log`. This file has the same structure as the scheduler log file with one event per line and the first field being the number of milliseconds since the client was started. The absolute time at which the client was last started is given by the most recent `start` event. For example:

```
0000000000 start 1089033266234
0000000000 ++++++++++++++ start Mon Jul 05 14:14:26 BST 2004
0000000031 new new version
0000000047 initial attributes
0000000047 attrs:
0000000047 jobtype2 update
0000000047 jobtype0 blast
0000000047 scriptslave: attempting to mount server
0000000547 scriptslave: mounted server
0000000594 scriptslave: resubmit succeeded
0000000594 scriptslave: get next task
0000460719 scriptslave: got task ’bab730a0.24#55’ (args: blast)
```

To find the time the slave got its task:

1089033266234 + 0000460719 = 1089033726(953)

and

```
% date 1089033726
Mon Jul 05 14:22:06 BST 2004
```

As with the scheduler, it is possible to force the client to produce more verbose output using the `-v` argument. Again, this will result in much larger log files sizes but is usually worth doing unless disk space is scarce.