RJE_QSub and RJE_IRIDIS Software Manual

Richard J. Edwards © 2012.

Contents

1. Introduction

This manual gives an overview of the RJE_QSub and RJE_IRIDIS utilities for running jobs on the University of Southampton's IRIDIS3 Supercomputer. These modules are not currently designed with another user in mind but feel free to try them if you like. They might even work on another HPC system running qsub. If anything is missing or needs clarification, please contact me.

The fundamentals are covered in Chapter 2, Fundamentals, including input and output details. General details about Command-line options can be found in the RJE Appendices document included with this download. Details of command-line options specific to RJE_QSub and RJE_IRIDIS can be found in the distributed readme.html file and online at bioware.soton.ac.uk.

Like the software itself, this manual is a 'work in progress' to some degree. If the version you are now reading does not make sense, then it may be worth checking the website to see if a more recent version is available, as indicated by the Version section of the manual. Options may have been added over the past few weeks and not all found their way into the manual yet. Check the readme on the website for up-to-date options *etc*. In particular, default values for options are subject to change and should be checked in the readme.

Good luck.

Rich Edwards, 2012.

1.1. Version

This manual is designed to accompany RJE_QSub version 1.4 and RJE_IRIDIS version 1.8.

The manual was last edited on 26 November 2012.

1.1.1. Known Issues with Current Version

Some of the features of RJE_IRIDIS are currently only implemented for SLiMFinder and QSLiMFinder. (These are the special options to control memory issues.) If running SLiMSearch or GOPHER, please contact the author.

1.2. Copyright, License and Warranty

1.3. Using this Manual

As much as possible, I shall try to make a clear distinction between explanatory text (this) and text to be typed at the command-prompt etc. Command prompt text will be `written in Courier New` to make the distinction clearer. Program options, also called 'command-line parameters', will be **`written in bold Courier New`** (and coloured **`red`** for fixed portions or **`dark`** for user-defined portions, such as file names etc.). Command-line examples will be given in (green) *`italicised Courier New`*. Optional parameters will (if I remember) be `[in square brackets]`. Names of files will be marked in coloured normal text.

1.4. Getting Help

Much of the information here is also contained in the documentation of the Python modules themselves. A full list of command-line parameters can be printed to screen using the **`help`** option, with short descriptions for each one:

```
python xxx.py help
```

General details about Command-line options can be found in the RJE Appendices document included with this download. Details of command-line options specific to XXX can be found in the distributed readme.html file and online at bioware.soton.ac.uk.

If still stuck, then please e-mail me (seqsuite@gmail.com) whatever question you have. If it is the results of an error message, then please send me that and/or the log file (see Chapter 2) too.

1.4.1. Something Missing?

As much as possible, the important parts of the software are described in detail in this manual. If something is not covered, it is generally not very important and/or still under development, and can therefore be safely ignored. If, however, curiosity gets the better of you, and/or you think that something important is missing (or badly explained), please contact me.

1.5. Citing RJE_QSub or RJE_IRIDIS

Until published, please cite the Seqsuite Website: https://sites.google.com/site/seqsuite/. If using RJE_IRIDIS to run (Q)SLiMFinder, SLiMSearch or GOPHER, please cite the relevant software. (See program-specific documentation for details.)

1.6. Availability and Local Installation

This software should work on any system with Python installed. With the exception of any external programs listed below (1.6.1), all the required files should have been provided in the downloaded zip file. Further information can be found at http://bioware.soton.ac.uk/ and the accompanying RJE Appendices document.

1.6.1. Programs Used by RJE_QSub and RJE_IRIDIS

These programs are designed to run a number of external programs. It is recommended that the user downloads and installs these programs according to the instructions given on the appropriate website. Instructions for external applications specific to other RJE Software will be found in the relevant manuals, or contact the author for advice.

2. Fundamentals

2.1. Running RJE_QSub

2.1.1. The Basics

Once logged on to the IRIDIS system, you should be able to run rje_qsub directly from the command line in the form:

```
python rje_qsub.py program="PROG_CALL"
```

To run with default settings, no other commands are needed. Otherwise, see the relevant sections of this manual.

IMPORTANT: If options contain spaces, they should be enclosed in double quotes: *program="example call"*. That said, it is recommended that files do not contain spaces as function cannot be guaranteed if they do.

2.1.2. Options

Command-line options are suggested in the following sections. General details about Command-line options can be found in the RJE Appendices document included with this download. Details of command-line options specific to XXX can be found in the distributed readme.txt and readme.html files. These may be given after the run command, as above, or loaded from one or more *.ini files (see RJE Appendices for details).

2.1.3. Using RJE_QSub to run programs

The most important input option for RJE_QSub is **program=X**, which determines which program call to enter into the qsub queue. If this is a standalone program, the full command should be given within double quotes. If it is part of the RJESuite of Python programs, the python call and path can be excluded so long as **rjepy=T** (the default) and **pypath=PATH** is given.

For example, these are effectively the same:

```
python program="rje_iridis.py" rjepy=T pypath=/home/re1u06/Tools/libraries/
python program="/home/re1u06/Tools/libraries/rje_iridis.py"
```

It is recommended that **pypath=PATH** is placed in the rje_qsub.ini file (or even rje.ini). See Chapter **Error! Reference source not found.** for details of running RJE_IRIDIS.

2.2. Input

RJE_QSub takes has no specific input, although it is recommended to give parameters as an INI file (see 2.4.1).

2.3. Output

Standard RJE_QSub output is a qsub job file, which will be output to the working directory, and also entered into the qsub queue. If **report=T**, a list of currently queued jobs with expected start and end times will also be output to the screen and log file (2.4).

2.3.1. Log Files

The GABLAM log file records information that may help subsequent interpretation of results or identify problems. Probably it's most useful content is any error messages generated. By default the log file is gablam.log but this can be changed with the **log=FILE** option. Logs will be appended unless the **newlog** option is used. (See the RJE Appendices document for details.)

2.4. Commandline Options

Commandline options are given in the appropriate sections. A full list of commandline options can be found in the readme file, online at bioware.soton.ac.uk or by running:

python xxx.py help

The key commandline options [and defaults] are:

- **program=X** : Program call for Qsub (and options) [None]
- **job=X** : Name of job file (.job added) [qsub]
- **qpath=PATH** : Path to change directory too [current path]
- **rjepy=T/F** : Whether program is an RJE *.py script (adds python PyPath/) [True]
- **pypath=PATH** : Path for RJE Python scripts [/rhome/re1u06/Serpentry/]
- **nodes=X** : Number of nodes to run on [4]
- **ppn=X** : Processors per node [12]
- **walltime=X** : Walltime for qsub job (hours) [60]
- **depend=LIST** : List of job ids to wait for before starting job (.blue30.iridis.soton.ac.uk added) []
- **pause=X** : Wait X seconds before attempting showstart [5]
- **report=T/F** : Pull out running job IDs and run showstart [False]

The main options used to control the resources requested by the job are **nodes=X**, **ppn=X** and **walltime=X**. Note that IRIDIS nodes have 12 processor (**ppn=12**) but sometimes it is better to use fewer ppn and have more memory available per processor. The maximum wall time is 60 hours (**walltime=60**).

2.4.1. Using INI Files

The best way to keep track of a run is to put all of the options into an ini file and call this. It is recommended to name the job and ini file the same.

3. Batch Farming with RJE_IRIDIS

RJE_QSub is set up predominantly to call RJE_IRIDIS, which is then used to actually farm out jobs to the nodes on IRIDIS. RJE_IRIDIS basically has three different run modes:

1. Standard batch job farming.
2. "Sequence by Sequence" job farming.
3. Special iRun batch farming of (Q)SLiMFinder, SLiMSearch, GOPHER or UniFake.

These are explored in more detail in the following sections.

3.1. Standard batch job farming.

In its most basic run mode, RJE_IRIDIS will read through a given list of jobs (**subjobs=LIST**) and farm them out one at a time. The total number of jobs running at one time will the number of nodes (**nodes=X**) multiplied by the ppn (**ppn=X**). The head node will always keep one processor free to controlling the batch run. To increase this and keep more memory free on the head node, use **keepfree=X**.

Once the initial batch of jobs has been farmed out, RJE_IRIDIS will cycle and check their progress. Each cycle of rje_iridis, it check to see whether any of the running jobs has finished. By default, it will check every second but this can be altered with **subsleep=X**. Jobs will only be spawned to nodes with at least 10% of their memory free. This can be altered using **memfree=X**. If the jobs being spawned are other RJESuite program calls, **rjepy=T** will add a commandline to each job to create a log file, which will be read and incorporated into the main RJE_IRIDIS log once the process is complete.

Standard run options:

- **rjepy=T/F** : Whether program is an RJE *.py script (adds log processing) [True]
- **subsleep=X** : Sleep time (seconds) between cycles of subbing out jobs to hosts [1]
- **subjobs=LIST** : List of subjobs to farm out to IRIDIS cluster []
- **iolimit=X** : Limit of number of IOErrors before termination [50]
- **memfree=X** : Min. proportion of node memory to be free before spawning job [0.1]
- **test=T/F** : Whether to produce extra output in "test" mode [False]
- **keepfree=X** : Number of processors to keep free on head node [1]

3.2. "Sequence by Sequence" job farming.

This will be explained in a later manual release. Basically, RJE_IRIDIS will use RJE_Seq to read in a set of sequences and farm them out one sequence at a time. The following commands are needed for this mode:

- **seqin=FILE** : Input sequence file to farm out [None]
- **seqbyseq=T/F** : Activate seqbyseq mode - assumes basefile=X option used for output [False]
- **basefile=X** : Base for output files – compiled from individual run results [None]
- **outlist=LIST** : List of extensions of outputs to add to basefile for output (basefile.*) []
- **pickup=X** : Header to extract from OutList file and used to populate AccNum to skip []
- **irun=X** : Name of RJESuite program to run each sequence on [None]
- **iini=FILE** : Ini file to pass to the called program [None]

3.3. Special iRun batch farming

One step beyond "Sequence by Sequence" job farming is the special iRun batch farming of specific SLiMSuite and SeqSuite tools. These work on the same principle but the output files etc. are handled automatically. There are also additional options to try and help regulate memory usage.

- **irun=X** : Exectute a special iRun analysis on Iridis (gopher/slimfinder/qslimfinder/slimsearch/unifake) []
- **pypath=PATH** : Path to python modules. **Must point to tools/**. ['/home/re1u06/Serpentry/']
- **runid=X** : Text identifier for iX run [None]
- **resfile=FILE** : Main output file for SLiMSuite iX run [islimfinder.csv]
- **sortrun=T/F** : Whether to sort input files by size and run big -> small to avoid hang at end [True]
- **loadbalance=T/F** : Whether to split SortRun jobs equally between large & small to avoid memory issues [True]
- **pickup=X** : Will pull out results from resfile with the correct runid and skip []
- **batch=LIST** : List of files (wildcards allowed) for batch input to SLiMSuite programs []

As with "Sequence by Sequence" job farming, additional program commands should be supplied using **iini=FILE**.

3.4. How to run (Q)SLiMFinder with RJE_QSub and RJE_IRIDIS

This section gives a brief overview of how best to run a large SLiMFinder batch job.

1. Make sure that RJE_Suite is setup on the system and the settings/ directory contains general system defaults.
2. Copy the set of input files to be run onto the system using rsync into a directory (e.g. "datadir/").
3. Create a run directory (e.g. "rundir/").
4. Within rundir/ create an rje_qsub.ini file with default settings numbers of nodes, walltime and ppn. This should also contain rjepy=T and pypath=PATH settings. If these are the same as all run defaults, this ini file can be in pypath/settings/. Note that at present, pypath=PATH for RJE_QSub **must point to the libraries/ directory**.
5. With rundir/ create an rje_iridis.ini file, which contains default settings for rje_iridis. This should contain any memory settings (sortrun/loadbalance/memfree/keepfree) as well as a **separate pypath=PATH setting that points to the tools/ directory**.
6. Create *.ini files for iRun program settings that might be reused for multiple different runs. E.g. disorder masking settings might be placed in dismask.ini.
7. For each run, create a specific *.ini file, e.g. "thisrun.ini". This should contain:
    a. the **irun=X** setting (e.g. *irun=slimfinder*)
    b. an **iini=FILE** self-reference to pass to the iRun program (*iini=thisrun.ini*).
    c. Additional **ini=X** commands for the iRun program, setting masking etc. (*ini=dismask.ini*)
    d. **runid=X**, **resfile=FILE**, **resdir=PATH** and **log=FILE** settings for this batch run (*runid=thisrun resfile=thisrun.csv resdir=ThisRun/ log=thisrun.log*)
    e. A program call for rje_qsub, **program=X** and matching job name (*program="rje_iridis.py ini=thisrun.ini" job=thisrun*)
    f. **pickup=T** if a large job that may not complete and could need multiple runs.
    g. Unless it is already given in one of the previous ini files, a **batch=LIST** command that specifies the input (*batch=../datadir/*.fas*)
8. Run rje_qsub, passing it the ini file created in Step 7:

```
python rje_qsub.py -ini thisrun.ini
```

This will then run rje_qsub, which will read in the rje_qsub.ini settings plus any over-riding settings in thisrun.ini. The latter ini file contains the job name and program call for rje_iridis, giving it the same ini file. RJE_IRIDIS will read in its own defaults from rje_iridis.ini (including, crucially,

pypath=PATH), and then get the specific run settings from thisrun.ini. This will include the key output settings, the program to run (irun=X) and will feed the same program the same ini file again (iini=thisrun.ini), thus keeping all of the run-specific information in one place. RJE_IRIDIS over-rides the output file names for the farmed jobs and then collates results at the end, so it will not interfere with the farming run to have these settings given to the iRun program in the ini file.

Finally, the status of the run can be monitored using `qstat`. Keeping an eye on thisrun.log and thisrun.pid, which are produced by RJE_IRIDIS during the batch farming, will also help monitor progress.