riss-doc Documentation

Release 1

RISS

April 12, 2016

Contents

RISS Tutorials and User Documentation

CHAPTER 1

Introduction to MSI resources

1.1 MSI Queues

Publically available queues at MSI

System	Queue	Nodes	Cores/node	Mem/node	Walltime	Running jobs
lab	lab (default)	73	8-32	15-128	72	6
	lab-long	?	8	15	150	6
	lab-600	?	8	128	600	1
	oc	4	12	23	72	3
Itasca	batch (default)	1086	8	22	24	2
	devel	32	8	22	2	?
	long	28	8	22	48	?
	jay	1	8	22	24	?
	sb	35	16	64	48	2
	sb128	8	16	128	96	2
	sb256	8	16	256	96	2
Mesabi	small (default)	256	24	64	96	?
	large (default)	360	24	64	24	?
	ram256g	32	24	256	96	?
	ram1t	16	32	1000	96	?
	k40	40	24	128	24	?

Notes:

- Jobs cannot request more than 1 node on the lab queue
- 8 nodes in the lab (default) queue have 128G of memory and 16-32 cores per node, the rest (65 nodes) have 15G of memory and 8 cores per node
- Jobs submitted to Mesabi's default queue automatically routes jobs requesting 10 or more nodes to the "large" queue, and smaller jobs to the "small" queue.

1.1.1 Lab Cluster Queues

The Lab Cluster is the oldest, slowest general-purpose cluster at MSI. All queues on the Lab cluster only allow single-node jobs, but up to six jobs can run simultaneously (per user). The Lab cluster hardware is very old and is not considered a high-performance system. Your jobs will run much faster on Itasca or Mesabi. The "isub" command launches interactive jobs on this cluster (useful for general-purpose command-line work).

lab (default) The is the main queue on the Lab cluster. Many nodes are available, but most are small. Job requesting 15G of memory or less and 8 nodes or less will have much shorter wait times in the queue than jobs requesting >15G memory or >8 nodes

lab-long Useful if your job needs more than 72 hours of walltime

lab-600 Useful if your job needs more than 150 hours of walltime

oc A queue for four overclocked nodes, particularly useful for serial (single-core) jobs. Also good for general use since these nodes are much newer than the other Lab nodes

1.1.2 Itasca Queues

Itasca is the second-tier general-purpose cluster at MSI, slower than Mesabi but faster than the Lab Cluster.

batch (default) This is the main queue on Itasca. A huge number of nodes are available, but each node is not particularly powerful. Great for jobs than can make use of many nodes, and for general use

devel This queue is for testing your pbs scripts. It works just like the batch queue, but you are limited to 2 hours of walltime and 32 nodes. The advantage is jobs on this queue have high priority, so jobs should start very quickly

long Useful if your job needs more than 24 hours of walltime

jay A queue for a special high-performance node with a high-speed internet connection

sb Sandybridge queue (64G memory). The Sandybridge nodes are much more powerful than the standard Itasca nodes in the batch queue, but there aren't very many of them. Great for single-node and smaller multi-node jobs. Large multi-node (>6) jobs tend to have long wait times in the queue. This queue has four times more nodes than the sb128 and sb256 queues, so use this queue unless you need more than 64G memory

sb128 Sandybridge queue (128G memory). Refer to sb queue

sb256 Sandybridge queue (256G memory). Refer to sb queue

1.1.3 Mesabi Queues

Mesabi is the newest, fastest general-purpose cluster at MSI that also contains some specialized hardware. MSI has not determined how queues will be set up on the new Mesabi system. However, it is likely that the queue structure will be derived from the hardware structure:

small (default) This is the main queue on Mesabi for jobs requesting fewer than 10 nodes. Jobs may request partial nodes (like on the lab queue). A large number of powerful nodes are available. Great for smaller multi-node jobs or for large numbers of small jobs (including single-node, single-core jobs)

large (default) This is the main queue on Mesabi for jobs requesting 10 or more nodes. A large number of powerful nodes are available. Great for jobs than can make use of many nodes.

ram256 (mid-mem) Queue for general-purpose 256G memory nodes

ram1t (high-mem) Queue for general-purpose 1T memory nodes. These nodes have 32 cores per node, so they are also good for jobs that scale well across multiple cores, but can't make use of multiple nodes.

k40 (gpu) Queue for nodes with NVidia Tesla GPUs, useful for jobs running software capable of using GPU accelerators

1.2 MSI Multiple Groups

1.2.1 MSI Groups

Your MSI user account is association with the PI who created the account, known as your primary group. You can have your user account added to additional "secondary" groups as well. This allows you to have access to another group's files, quota space, and SUs, which is useful if you are working with another PI on a project. To have your account added to Goldy Gopher's group have Goldy (the PI) email help@msi.umn.edu with a request to have your user account added to the gopherg group.

1.2.2 Quickstart

Add these commands to your .bashrc file (located in your home directory). The rest of this document explains these commands in detail. Replace "gopherg" with the name of your secondary group:

```
# define an environment variable to hold the active group
export MYGROUP=$(id -gn)

# group follows along with ssh
function ssg ()
{
\ssh $@ -t "newgrp \- $MYGROUP"
}
alias ssh=ssg

# switch groups by typing the name of the group (repeat this line for additional secondary groups)
# Replace gopherg with the name of your secondary group!
alias gopherg="newgrp \- gopherg"

# make new jobs run as, and bill SUs to,  your active group
alias qsub="qsub -A $MYGROUP -W group_list=$MYGROUP"
alias isub='isub -A $MYGROUP -W group_list=$MYGROUP'

# Add your active group to your command prompt (and add some color)
GROUP_NAME=$(id -gn)
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;36m\]\u-$GROUP_NAME@\h\[\033[00m\]:\[\033[01;33m\]\w
```

1.2.3 Switching Groups

When you log in to MSI your active group is your primary group. To switch to the gopherg group use the newgrp command:

```
$ newgrp - gopherg
```

Note: The dash in the newgroup command causes your environment to be reinitialized which bumps you back to your home directory (among other things), but is neccessary for the .bashrc modifications listed on this page to work. If you omit the dash and run "newgrp gopherg" you will stay in your current directory and keep your current environment, but alias definitions are lost.

To make it easier to switch to the group gopherg you can add an alias to your .bashrc:

```
alias gopherg="newgrp \- gopherg"
```

Then you can switch to the group just by typing its name:

```
$ gopherg
```

To switch back to your primary account type "exit" or press control-d

1.2.4 Files

When you create new files they will be owned by your active group. If your active group is not gopherg and you create files anywhere in gopherg's group directory the filesystem will immediately change the group ownership of the files to gopherg. This process occasionally fails resulting in a "Disk quota exceeded" error, so it is recommended you first change your active group before creating files in another group's home directory.

1.2.5 PBS Jobs

PBS jobs submitted with the qsub command draw SUs from your primary account, regardless of what your current active group is. To charge SUs to group gopherg you can use "#PBS -A gopherg" in the pbs script, or add "-A gopherg" to the qsub command line.

PBS jobs execute as your primary group, regardless of what your current active group is. This can result in the job encountering file permission and quota problems. To have a job execute as group gopherg use "#PBS -W group_list=gopherg" in the pbs script, or add "-W group_list=gopherg" to the qsub command line. If you always want jobs to bill SUs to, and run as, your current active group add these lines to your .bashrc file:

```
export MYGROUP=$(id -gn)
alias qsub="qsub -A $MYGROUP -W group_list=$MYGROUP"
alias isub="isub -A $MYGROUP -W group_list=$MYGROUP"
```

Note: You can always override aliases and run the original command by adding a "\" at the front of the command when you use it:

```
$ \qsub script.pbs
```

1.2.6 SSH

When you use ssh to connect to another MSI computer your active group on the new computer will be your primary group, regardless of what your active group was on the previous computer. To have your active group follow you around add these lines to your .bashrc file:

```
export MYGROUP=$(id -gn)
function ssg ()
{
\ssh $@ -t "newgrp \- $MYGROUP"
}
alias ssh=ssg
```

1.2.7 Command-Prompt

The default command-line prompt shows your username. Add these lines to your .bashrc to have the username and active group displayed so it's easy to see what group you are working in:

```
# Set the command prompt.
GROUP_NAME=$(id -gn)
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;36m\]\u-$GROUP_NAME@\h\[\033[00m\]:\[\033[01;33m\]\w
```

CHAPTER 2

Software

2.1 SOFTWARE Riss-util Module

Contents

The riss_util module contains a variety of small programs and scripts developed by RISS staff to perform various bioinformatics tasks. The module is available on the lab cluster, Itasca, and Mesabi. To load the module run:

```
$ module load riss_util
```

Most of the scripts are written in perl. After loading the module you can view the source code for the scripts at /soft/riss_util/1.0/bin/. If neccessary you can copy a script to your home directory and modify it to suite your needs.

2.1.1 profile.pl

NAME profile.pl - profile the cpu and memory usage of the computer

SYNOPSIS profile.pl [-s seconds] [-h] [-i] [-b bins] [-o logfile]

DESCRIPTION This script collects total memory and cpu usage information for the computer/node it is running on, and when the script is killed it prints ASCII plots to standard output summarizing memory and cpu usage across

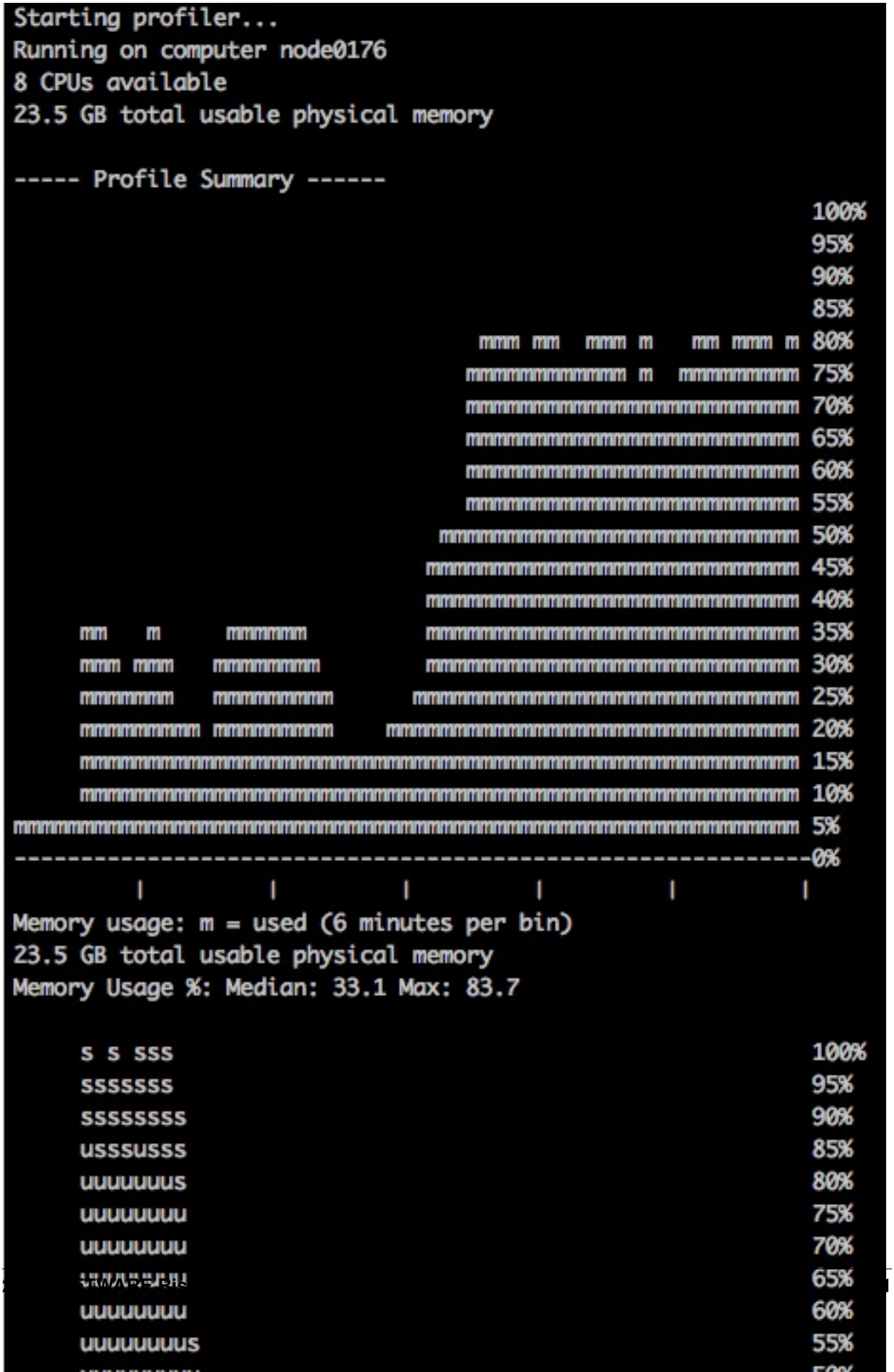
time. After the plots is a list showing the most active process in each bin in the plots.

Options:

-s seconds	The number of seconds between polling cpu and memory usage
-b bins	The number of bins in the summary histograms
-i	Interactive mode: print update to screen after every poll
-h	Display usage information
-o file	Print output to file instead of STDOUT

EXAMPLE Start profile.pl at the beginning of your pbs script (after loading the riss_util module) and put it in the background using "&". Check the standard output file (jobname.oXXXXX) for the results:

```
$ profile.pl &
```


Starting profiler...
Running on computer node0176
8 CPUs available
23.5 GB total usable physical memory
----- Profile Summary ------
100%
95%
90%
85%
80%
75%
70%
65%
60%
55%
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
Memory usage: m = used (6 minutes per bin)
23.5 GB total usable physical memory
Memory Usage %: Median: 33.1 Max: 83.7
100%
95%
90%
85%
80%
75%
70%
65%
60%
55%

2.1.2 profiles.pl

NAME profiles.pl - Run profile.pl on all nodes allocated to a job

SYNOPSIS profiles.pl [-s seconds] [-h] [-i] [-b bins]

DESCRIPTION Generates memory and cpu usage information for multiple nodes. One nodeXXXX.log file is created for each node allocated to the current job.

Options:

-s seconds	The number of seconds between polling cpu and memory usage
-b bins	The number of bins in the summary histograms
-h	Display usage information

EXAMPLE Start profiles.pl at the beginning of your pbs script (after loading the riss_util module) and put it in the background using "&":

```
$ profiles.pl &
```

2.1.3 multi-profile.pl

NAME multi-profile.pl - profile the cpu and memory usage of a multi-node job

SYNOPSIS multi-profile.pl [-s seconds] [-h] [-i] [-b bins] [-o logfile]

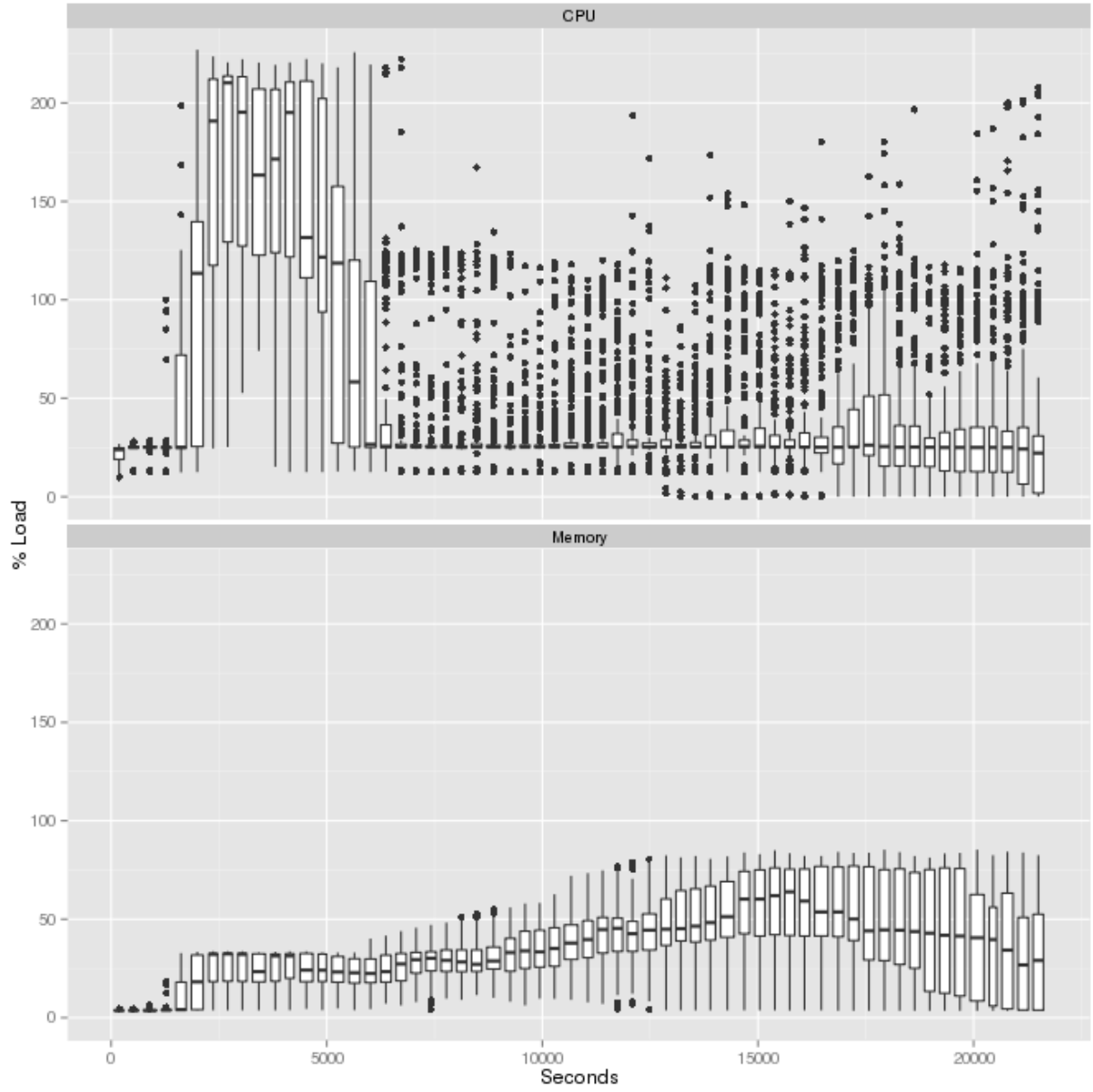
DESCRIPTION Generates one plot summarizing memory and cpu usage across all nodes in a multi-node job

Options:

-s seconds	The number of seconds between polling cpu and memory usage
-b bins	The number of bins in the summary histograms
-i	Interactive mode: print update to screen after every poll
-h	Display usage information
-o file	Print output to file instead of STDOUT

EXAMPLE Start multi-profile.pl at the beginning of your pbs script (after loading the riss_util module) and put it in the background using "&". View the profile.png image after the job finishes. The plots use boxplots to show the distribution of memory and cpu usage across all nodes at each timepoint bin. The top plot shows CPU load percentage, which is the number of threads running or ready to run, divided by the number of cores (thus the load can be higher than 100%):

```
$ multi-profile.pl &
```



2.1.4 cleanup

NAME cleanup - delete all but the most recent pbs.e and pbs.o output files

SYNOPSIS cleanup [-d]

DESCRIPTION Submitting the same pbs script to a queue multiple times results in many different standard error and standard out files. This script will delete all of the old files for you, leaving the most recent pair of files. This script finds all files ending in .pbs.e00000 and .pbs.o0000 and removes all but the most recent (as determined by the job number, not the file modification dates) .e and .o file for each .pbs file. Run without any options the script lists which files should be deleted and which should be kept. Run with the -d option the script will actually delete files.

Options:

-d Delete old .e and .o files

2.1.5 fastqqualityplot.pl

NAME fastqqualityplot.pl - Generate per-base quality plot for multiple fastq files

SYNOPSIS fastqqualityplot.pl -f /fastq/folder [-m mappingfile]

DESCRIPTION Generate per-base quality plot for multiple fastq files

-f folder A folder containing fastq files to process

-m file Full path to a mapping file

-o file Name of the output image file (fastqqualityplot.png)

-p integer Number of processors to use (number of threads to run) (this doesn't work yet...)

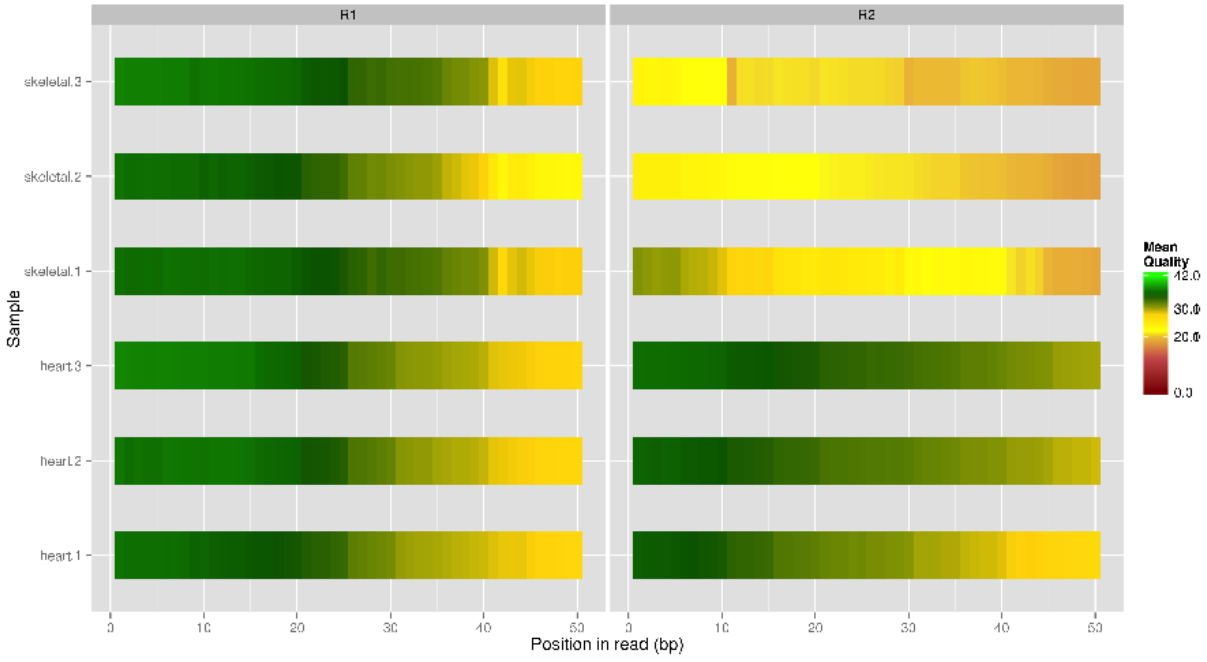
-s integer Subsample the specified number of reads from each fastq file. 0 = no subsampling

-h Print usage instructions and exit

-v Print more information while running (verbose)

EXAMPLE Run the script:

```
$ fastqqualityplot.pl -f /home/msistaff/public/garbe/sampledata/RNAseq/Hsap/fastq/ -s 4000 -o fa
```



2.1.6 insertsize.pl

NAME insertsize.pl - Calculate the insert size mean and standard deviation of a paired-end dataset

SYNOPSIS insertsize.pl [-m 1] bowtieindex R1.fastq R2.fastq

DESCRIPTION Calculate the insert size mean and standard deviation by aligning some reads from a pair of fastq files to a bowtie2 index

-b bowtieindex	A Bowtie2 index
-m integer	The first N million reads from the fastq files will be aligned (Default 1)
-p integers	Number of threads to use (Default $PBS_NUM_PPN or 1);
-h	Print usage instructions and exit
-v	Print more information while running (verbose)

EXAMPLE Run the script:

```
$ insertsize.pl bowtieindex R1.fastq R2.fastq
```

Runtime: 15 seconds using "-m .1 -p 8" on Itasca, 102 seconds using "-m 1 -p 8" on Itasca

2.1.7 insertplot.pl

NAME insertplot.pl - generate a fragment-length plot from Picard output

SYNOPSIS insertplot.pl insert_summary1.txt [insert_summary2.txt ...] insertplot.pl -f filelist.txt

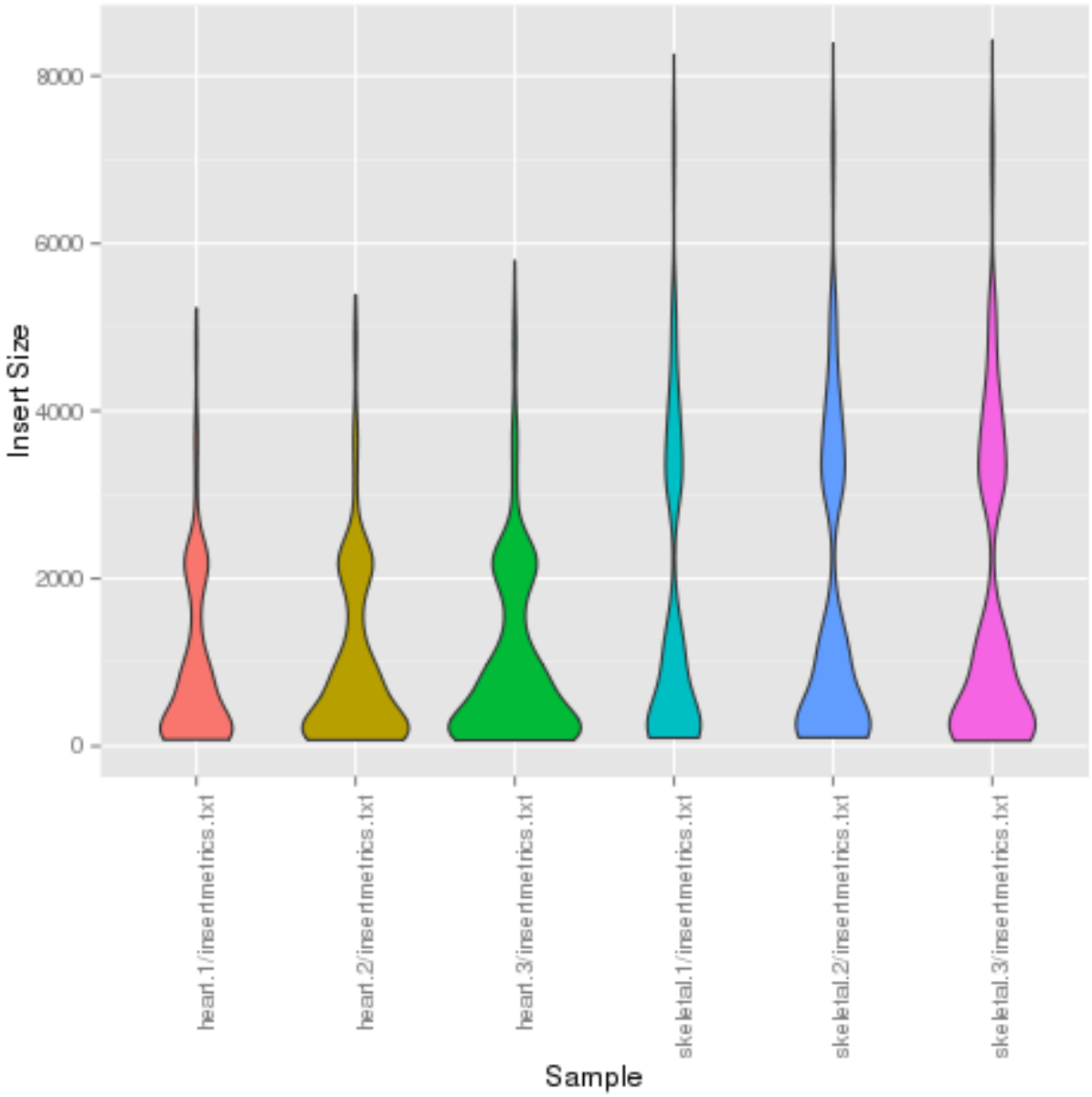
DESCRIPTION Generate a plot summarizing multiple Picard-tools insert-size-metrics output files. R is required, as well as the R package ggplot2.

Options: -f filelist.txt : provide a file with a list of picard insert-size-metrics output files, one per line. A

second tab-delimited column may be included containing sample names -h : Print usage instructions and exit -v : Print more information whie running (verbose)

EXAMPLE Generate a plot from six different picard output files:

```
$ cd /home/msistaff/public/garbe/sampledata/RNAseq/Hsap/analysis
$ insertplot.pl heart.1/insertmetrics.txt heart.2/insertmetrics.txt heart.3/insertmetrics.txt sk
```



2.1.8 fastq-species-blast.pl

NAME fastq-species-blast.pl - Given a fastq file, blast a sample of the sequences and count how many hits there are to each species.

SYNOPSIS fastq-species-blast.pl [-n number_of_sequences_to_blast] [-t num_threads] [-d blast_database(s)] input.fastq

DESCRIPTION fastq-species-blast.pl can be used to blast a small number of fastq reads against a BLAST database in order to determine what species the fastq file contains, and if there are significant amounts of contaminating sequence from other species. The -n option is used to specify how many reads from the input.fastq file shoule be BLASTed (default is 10). The -t option specifies how many processor cores to use (default is 1, this script cannot run across multiple nodes). The -d option specifies which BLAST database to use (default is htgs). Any

database installed with the local NCBI Blast installation can be used (the taxdb must be installed). Multiple databases can be blasted against: fastq-species-blast.pl input.fastq -d "human_genomic vector"

EXAMPLE Blast 10 fastq sequences (the default) against the htgs database (the default):

```
$ fastq-species-blast.pl
/home/msistaff/public/garbe/sampledata/RNAseq/Hsap/fastq/heart-1_R1.fastq
6 out of 10 sequences (60%) have a hit in the htgs blast database
   Common name               Scientific name      # of sequences
   grivet                    Chlorocebus aethiops 1
   cattle                    Bos taurus           1
   white-tufted-ear marmoset Callithrix jacchus   1
   human                     Homo sapiens         3
```

2.1.9 fastq-cat.pl

NAME fastq-cat.pl - Concatenate FastQC files

SYNOPSIS fastq-cat.pl /fastq/folder

DESCRIPTION This script identifies samples spread across multiple fastq files and generates cat commands to concatenate them together. Symlink commands are generated for single-file samples. This script only generates the commands to concatenate and link files. Run "fastq-cat.pl FOLDER | bash" to generate the concatenated and linked files.

Options:

-f FOLDER Folder containing fastq files

EXAMPLE Create a directory to contain the concatenated files:

```
$ mkdir fastq-cat
$ cd fastq-cat
```

Generate the concatenation commands:

```
$ fastq-cat.pl ~/fastq-files > fastq-commands.txt
```

Execute the concatenation commands:

```
$ bash fastq-commands.txt
```

2.1.10 redup.pl

NAME redup.pl - Remove exact duplicate reads from paired-end fastq files

SYNOPSIS redup.pl [-n N] sample1_R1.fastq sample1_R2.fastq > topdups.fasta

Options:

-n integer Print out this many of the most duplicated sequences

-h Display usage information

DESCRIPTION This script removes duplicate paired-end reads from the input files sample1_R1.fastq and sample1_R2.fastq and prints out unique reads to the files sample1_R1.fastq.unique and sample2_R2.fastq.unique. Reads must have the exact same sequence to be called duplicates, quality scores are ignored. The top N (default 20) most duplicated sequences are printed out in fasta format, making it convenient for using BLAST to identify them.

2.1.11 resync.pl

NAME resync.pl - Resynchronize a pair of paired-end fastq files.

SYNOPSIS resync.pl sample1_R1.fastq sample1_R2.fastq [sample1_R1_synced.fastq sample1_R2_synced.fastq]

DESCRIPTION Programs that process paired-end fastq files usually require that the Nth read in the R1 fastq file and the Nth read in the R2 fastq file are mates. Using trimming or filtering programs that aren't paired-end aware often results in reads being removed from one paired-end fastq file but not the other, resulting in "unsyncronized" files. This program reads in two unsynchronized fastq files and writes out two synchronized fastq files. The synchronized files have properly paired reads, with singleton reads removed. Casava 1.7 and 1.8 read ID formats are supported. This program shouldn't use much memory (<1GB), but maximum memory use could be equivalent to the size of one input file in a worst-case scenario.

Options: -h : Display usage information -s : Save singletons to .singleton files

2.1.12 fasterqc.pl

NAME fasterqc.pl - Combine FastQC output images

SYNOPSIS fasterqc.pl [-s 100] [-o fasterqc.png]

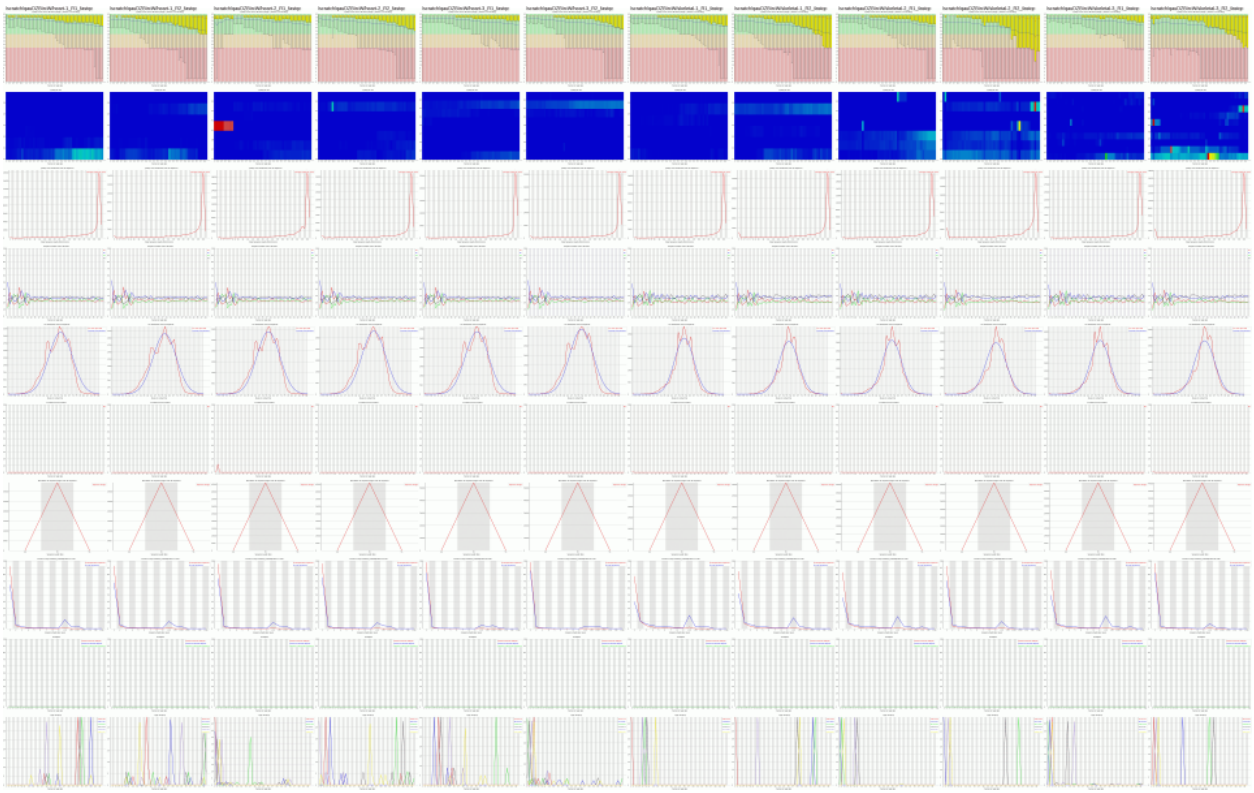
DESCRIPTION This script combines FastQC output images into one large png image to make it easy to quickly assess the FastQC output from many samples. When FastQC is run it generates a zip file named SAMPLENAME_fastqc.zip. Run this script in a folder containing one or more of these SAMPLENAME_fastqc.zip files and it will generate a single image containing all of the FastQC images from all samples. It also prints out the "overrepresented sequences" for each sample to the file fasterqc.overrep.txt. Recommended maximum number of fastqc folders is 50. This script works with older and newer versions of FastQC, but won't work with a mix of old and new version FastQC output files.

Options:

-s percent	Scale the final image by the specified percent (valid range 5-100, default 100). Files larger than 5000 pixels wide are automatically scaled to 5000 pixels wide
-o file	Save the final image in the specified file (default fasterqc.png)

EXAMPLE Consolidate the results from 12 FastQC runs into one tiny image:

```
$ cd /home/msistaff/public/garbe/sampledata/RNAseq/Hsap/fastq/fastqc
$ fasterqc.pl -s 10 -o fasterqc-sample.png
```



2.1.13 tophatplot.pl

NAME tophatplot.pl - Generate plots from tophat align_summary.txt output files

SYNOPSIS tophatplot.pl align_summary1.txt [align_summary2.txt ...] tophatplot.pl -f filelist.txt

DESCRIPTION Generate a plot summarizing mapping percentage for multiple samples

Options:

-f file Provide a file with a list of align_summary.txt files, one per line. A second tab-delimited column may be included containing sample names. A third column may be included containing bam files from mapping unmapping reads against a spike-in control reference

-h Display usage information

EXAMPLE

```
software/alignments-pairs.png
```

2.1.14 expressiontableplot.pl

NAME expressiontableplot.pl - Given a table of expression data, generate a series of summary plots including:

-MDS plot -Dendogram -Expression distribution violin plots -Expressed genes plot

SYNOPSIS expressiontableplot.pl data.txt

DESCRIPTION Generate a series of plots summarizing a table of expression data. The input file should be tab delimited with a header. There should be a row for each feature (gene, transcript, exon, etc), and a column for each sample. The first row should contain sample names and the first column feature IDs.

Options:

-n	Normalize expression values: 75% quartile normalization
-m integer	Minimum expression value
-t string	Feature type (gene, transcript, exon, etc)
-h	Display usage information
-v	Verbose output

EXAMPLE

software/mdsplot.png

2.1.15 Deprecated scripts

These scripts are no longer supported:

tophatstatsPE.pl Tophat now produces a file name align_summary.txt containing alignment statistics. Use tophatplot.pl to summarize multiple align_summary.txt files

cuffplot.pl Use cuffdiffplot.pl instead, it genreates more plots and uses ggplot2 instead of gnuplot

cuffdiff2_mds_plot.pl Use cuffdiffplot.pl instead, it generates an mds plot as well as several other useful plots

2.1.16 Support

There is a discussion thread for the riss_util module in the MSI google group: https://groups.google.com/a/umn.edu/forum/#!categories/msi-user-questions/software Updates and changes to programs in the riss_util module are posted to the thread, and you may post feature requests or bug reports to the thread. You may also email RISS at help@msi.umn.edu

2.2 SOFTWARE GNU Parallel

GNU Parallel is a great tool for executing commands in parallel on one or more nodes. If you put all of your commands in a file named commands.txt you can have GNU Parallel dynamically distribute the commands across all of the nodes and cores that were requested by a pbs job.

2.2.1 Single-node Examples

You have a file named commands.txt containing a list of commands and want to run one command per core:

```
$ module load parallel
$ parallel < commands.txt
```

GNU Parallel automatically identifes the number of cores on the node and runs one command per core. Use the *–jobs* option to specify a different number of concurrent commands:

```
$ module load parallel
$ parallel --jobs 2 < commands.txt
```

You want to run the same command (FastQC) on many (fastq) files, running one command per core:

```
$ module load parallel
$ module load fastqc
$ find ~/fastqfolder -name *.fastq | parallel "fastqc {}"
```

You want to run the same command (wc -l) on many files, running one command per core, saving the output to files named EXAMPLE.fastq.out *in the same directory* as the fastq files:

```
$ module load parallel
$ find ~/fastqfolder -name *.fastq | parallel "wc -l {} > {}.out"
```

You want to run the same command (wc -l) on many files, running one command per core, saving the output to files named EXAMPLE.fastq.out *in a different directory*:

```
$ module load parallel
$ find ~/fastqfolder -name *.fastq | parallel "wc -l {} > ~/output/{/}.out"
```

You want to run the same command (wc -l) on many files, running one command per core, saving the output to files named EXAMPLE.out in the current working directory:

```
$ module load parallel
$ find ~/fastqfolder -name *.fastq | parallel "wc -l {} > {/.}.out"
```

2.2.2 Multi-node Examples

If you pass GNU Parallel a file with a list of nodes it will run jobs on each node. The PBS environment variable PBS_NODEFILE points to a file that lists all nodes allocated to the current job, however each node is listed once for each core on the node. Therefore you need to either tell GNU Parallel to run one job per node, or remove duplicate node names from the node file.

You have a file named commands.txt containing a list of single-threaded commands and want to run one command *per core* on multiple nodes:

```
$ module load parallel
$ parallel --jobs 1 --sshloginfile $PBS_NODEFILE --workdir $PWD < commands.txt
```

Which is equivalent to:

```
$ module load parallel
$ sort -u $PBS_NODEFILE > unique-nodelist.txt
$ parallel --sshloginfile unique-nodelist.txt --workdir $PWD < commands.txt
```

You have a file named commands.txt containing a list of multi-threaded commands and want to run one command *per node* on multiple nodes:

```
$ module load parallel
$ sort -u $PBS_NODEFILE > unique-nodelist.txt
$ parallel --jobs 1 --sshloginfile unique-nodelist.txt --workdir $PWD < commands.txt
```

Loading modules

Multi-node jobs are a little tricky because the remote nodes do not inherit the environment from the head node, so any modules loaded by the pbs script won't be present on the remote nodes. Also, the module command is really just a shell alias, and aliases don't work in the non-interactive bash sessions that are created on the remote nodes. One workaround is to include this environment variable defenition in your PBS script after you have loaded your modules, but before you run GNU Parallel:

```
$ module load parallel
$ module load another_module
$ export PARALLEL="--workdir . --env PATH --env LD_LIBRARY_PATH --env LOADEDMODULES --env _LMFILES_ -
$ parallel --jobs 1 --sshloginfile $PBS_NODEFILE --workdir $PWD < commands.txt
```

2.2.3 Additional Resources

GNU Parallel man page.

GNU Parallel Tutorial.

CHAPTER 3

Support

If you are having issues, please let us know. Email help@msi.umn.edu, include RISS in the subject line