VK4) Job script configuration

Single-processor job

This script can serve as the template for single-processor applications.

```
#!/bin/bash
#SBATCH --job-name=basic_job_test        # Job name
#SBATCH --mail-type=END,FAIL             # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=usr1@york.ac.uk      # Where to send mail
#SBATCH --ntasks=1                       # Run on a single CPU
#SBATCH --mem=1gb                        # Job memory request
#SBATCH --time=00:05:00                  # Time limit hrs:min:sec
#SBATCH --output=basic_job_%j.log        # Standard output and error log
#SBATCH --account=dept-proj-2018         # Project account

echo My working directory is `pwd`
echo Running job on host:
echo -e '\t'`hostname` at `date`
echo

module load lang/Python/3.7.0-foss-2018b

python hello.py

echo
echo Job completed at `date`
```

A Slurm job script has the format:

- Slurm directives section
- Command section

Line 1 sets the shell for the job.

Lines 2-9 are the Slurm directives requesting resources for the job. All directives start with "#SBATCH". Any directives must come before any executable commands in the job script.

Lines 10 - 21 are the commands executed by the job.

Job directives and resource requests

The full list of #SBATCH directives can be found in the sbatch man page.

```
[usr1@login1(viking) scratch]$ man sbatch
```

Any of the command line parameters can be specified in a job script as an #SBATCH directive.

Managing time

The *time* directive is the time that the job take to run. It can be omitted in which case the default time is 8 hours. The job will be killed if it exceeds this time. Specifying a shorter and more accurate *time* will allow your job to start execution sooner. The *time* can be specified in two ways:

1. In the job script using the "#SBATCH --time=02:00:00" directive (RECOMMENDED)
2. On the command line using the "--time=02:00:00" option

The form of the time is "HH:MM:SS".

Controlling memory usage

You need to understand the memory requirements of your program. A job has a default memory allocation and sometimes you will need to request more memory so your program can run.

--mem=<*size[units]*> Specify the real memory required per node. Default units are megabytes.

For example:

    #SBATCH --mem=1gb

Redirecting output

The %j in the -o (--output) line tells SLURM to substitute the job ID in the name of the output file. You can also add a -e or --error with an error file name to separate output and error logs.

Filename patterns

There are several useful placeholders that can be used in filenames which will be automatically filled in by SLURM. The full list of these can be found in the sbatch man page, under the *`filename pattern`* heading.

Specifying a project code

Please use the following line in your batch scripts, so that we can associate your work with your project:

    #SBATCH --account=YOUR-PROJECT-CODE

Specifying a project account code is mandatory.

Threaded/multi-process jobs

This is used for programs that are commonly referred to as *threaded*, *OpenMP*, *PTHREADS*, or shared memory applications. While they can use multiple processors on the same node, they cannot make use of multiple nodes.

```
#!/bin/bash
#SBATCH --job-name=threaded_job_test       # Job name
#SBATCH --mail-type=END,FAIL               # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=usr1@york.ac.uk        # Where to send mail
#SBATCH --ntasks=1                         # Run a single task...
#SBATCH --cpus-per-task=4                  # ...with four cores
#SBATCH --mem=1gb                          # Job memory request
#SBATCH --time=00:05:00                    # Time limit hrs:min:sec
#SBATCH --output=threaded_job_%j.log       # Standard output and error log
#SBATCH --account=dept-proj-2018               # Project account

echo My working directory is `pwd`
echo Running job on host:
echo -e '\t'`hostname` at `date`
echo $SLURM_CPUS_ON_NODE CPU cores available
echo

module load lang/Python/3.7.0-foss-2018b

python hello.py

echo
echo Job completed at `date`
```

Remember:

> set --ntasks=1, and then set --cpus-per-task to the number of threads you wish to use.

MPI jobs

This is used for programs that can use multiple cpus or processors that may, or may not, be on multiple compute nodes. Tasks are packed onto available nodes.

```
#!/bin/bash
#SBATCH --job-name=mpi_example               # Job name
#SBATCH --mail-type=END,FAIL                 # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=usr1@york.ac.uk          # Where to send mail
#SBATCH --ntasks=16                          # Run sixteen tasks...
#SBATCH --cpus-per-task=1                    # ...with one core each
#SBATCH --mem-per-cpu=600mb                  # Memory per processor
#SBATCH --time=00:05:00                      # Time limit hrs:min:sec
#SBATCH --output=mpi_example_%j.log          # Standard output and error log
#SBATCH --account=dept-proj-2018                 # Project account

echo "Running mpi_example on ${SLURM_NTASKS} CPU cores"

mpiexec -n ${SLURM_NTASKS} ./mpi_example
```

More control over which nodes are used and where tasks are placed, can be specified.

```
#!/bin/bash
#SBATCH --job-name=mpi_example               # Job name
#SBATCH --mail-type=END,FAIL                 # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=andrew.smith@york.ac.uk  # Where to send mail
#SBATCH --ntasks=80                          # Run 80 tasks
#SBATCH --cpus-per-task=1                    # Number of CPU cores per task
#SBATCH --nodes=2                            # Number of nodes
#SBATCH --ntasks-per-node=40                 # How many tasks on each node
#SBATCH --ntasks-per-socket=20               # How many tasks on each CPU or socket
#SBATCH --distribution=cyclic:cyclic         # Distribute tasks cyclically on nodes and sockets
#SBATCH --mem-per-cpu=600mb                  # Memory per processor
#SBATCH --time=00:05:00                      # Time limit hrs:min:sec
#SBATCH --output=mpi_example_%j.log          # Standard output and error log
#SBATCH --account=dept-proj-2018                 # Project account

echo "Running mpi_example on ${SLURM_NTASKS} CPU cores"

mpiexec -n ${SLURM_NTASKS} ./mpi_example
```

The following directives are important:

Directive	Use	Comment
-c, --cpus-per-task=<ncpus>	Inform Slurm that ensuing job steps will require ncpus (cores) per task	Usually set to 1, ie, 1 task to 1 core
-m, --distribution=arbitrary|<block|cyclic|plane=<options>[:block|cyclic|fcyclic]>	Specify alternate distribution methods for remote processes	Recommend cyclic:cyclic, distribute tasks cyclically over nodes and sockets
-N, --nodes=<minnodes[-maxnodes]>	minimum of nodes be allocated	
-n, --ntasks=<number>	Number of tasks (MPI ranks)	
--ntasks-per-node=<ntasks>	ntasks be invoked on each node	
--ntasks-per-socket=<ntasks>	Request the maximum ntasks be invoked on each socket	Viking has mostly 2 socket nodes, with 20 cores per socket

Array jobs

Array jobs offer a mechanism for submitting and managing collections of identical jobs (for instance for parameter sweeps, or a large number of input files), quickly and easily. Job arrays with thousands of tasks can be submitted in milliseconds. All jobs must have the same initial options (e.g. size, time limit, etc.).

```
#!/bin/bash
#SBATCH --job-name=basic_job_test        # Job name
#SBATCH --mail-type=END,FAIL             # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=usr1@york.ac.uk      # Where to send mail
#SBATCH --ntasks=1                       # Run a single task
#SBATCH --mem=1gb                        # Job memory request
#SBATCH --time=00:05:00                  # Time limit hrs:min:sec
#SBATCH --output=array_job_%A_%a.log     # Standard output and error log
#SBATCH --account=dept-proj-2018             # Project account
#SBATCH --array=1-5                      # Array range

echo My working directory is `pwd`
echo Running array job index $SLURM_ARRAY_TASK_ID, on host:
echo -e '\t'`hostname` at `date`
echo

module load lang/Python/3.7.0-foss-2018b

python hello.py

echo
echo Job completed at `date`
```

Job arrays are only supported for batch jobs and the array index values are specified using the --array or -a option of the sbatch command. The option argument can be specific array index values, a range of index values, and an optional step size as shown in the examples below. Note that the minimum index value is zero and the maximum value is a Slurm configuration parameter (MaxArraySize minus one, which is currently set to 4096 as seen in `scontrol show conf`).

Jobs which are part of a job array will have the environment variable SLURM_ARRAY_TASK_ID set to its array index value.

Multiple copies of a job script can be submitted by using the --array argument to sbatch.

```
# Submit a job array with index values between 0 and 31
[usr1@login1(viking) scratch]$ sbatch --array=0-31 basic.job
```

Task Concurrency - limiting the number of concurrent tasks

To *throttle* a job array by keeping only a certain number of tasks active at a time use the `%N` suffix where *N* is the number of active tasks. For example

```
#SBATCH --array 1-150%5
```

will produce a 150 task job array with only 5 tasks active at any given time.

Using scontrol to modify throttling of running array jobs

If you want to change the number of simultaneous tasks of an active job, you can use scontrol:

```
scontrol update ArrayTaskThrottle=<count> JobId=<jobID>
```

Set ArrayTaskThrottle=0 to eliminate any limit.

Array Indexing

The array stride can be a value other that 1.

```
# Submit a job array with index values of 1, 3, 5 and 7
[usr1@login1(viking) scratch]$ sbatch --array=1,3,5,7 basic.job
```

The default step size of one can be changed.

```
# Submit a job array with index values between 1 and 7
# with a step size of 2 (i.e. 1, 3, 5 and 7)
[usr1@login1(viking) scratch]$ sbatch --array=1-7:2 basic.job
```

Job ID and Environment Variables

Job arrays will have additional environment variable set.

Variable	Value
SLURM_ARRAY_JOB_ID	job ID of the array
SLURM_ARRAY_TASK_ID	job array index value
SLURM_ARRAY_TASK_COUNT	number of tasks in the job array
SLURM_ARRAY_TASK_MAX	highest job array index valu
SLURM_ARRAY_TASK_MIN	lowest job array index value

For example a job submission of this type:

sbatch --array=1-3 basic.job

will generate a job array containing three jobs. If if the job id is 36, then the environment variables will be set as follows:

SLURM_JOB_ID=36
SLURM_ARRAY_JOB_ID=36
SLURM_ARRAY_TASK_ID=1
SLURM_ARRAY_TASK_COUNT=3
SLURM_ARRAY_TASK_MAX=3
SLURM_ARRAY_TASK_MIN=1

SLURM_JOB_ID=37
SLURM_ARRAY_JOB_ID=36
SLURM_ARRAY_TASK_ID=2
SLURM_ARRAY_TASK_COUNT=3
SLURM_ARRAY_TASK_MAX=3
SLURM_ARRAY_TASK_MIN=1

SLURM_JOB_ID=38
SLURM_ARRAY_JOB_ID=36
SLURM_ARRAY_TASK_ID=3
SLURM_ARRAY_TASK_COUNT=3
SLURM_ARRAY_TASK_MAX=3
SLURM_ARRAY_TASK_MIN=1

All Slurm commands and APIs recognise the SLURM_JOB_ID value. Most commands also recognise the SLURM_ARRAY_JOB_ID plus SLURM_ARRAY_TASK_ID values separated by an underscore as identifying an element of a job array. Using the example above, "37" or "36_2" would be equivalent ways to identify the second array element of job 36. A set of APIs has been developed to operate on an entire job array or select tasks of a job array in a single function call. The function response consists of an array identifying the various error codes for various tasks of a job ID. For example the job_resume2() function might return an array of error codes indicating that tasks 1 and 2 have already completed; tasks 3 through 5 are resumed successfully, and tasks 6 through 99 have not yet started.

scancel and array jobs

If the job ID of a job array is specified as input to the scancel command then all elements of that job array will be cancelled. Alternately an array ID, optionally using regular expressions, may be specified for job cancellation.

```
# Cancel array ID 1 to 3 from job array 20
[usr1@login1(viking) scratch]$ scancel 20_[1-3]

# Cancel array ID 4 and 5 from job array 20
[usr1@login1(viking) scratch]$ scancel 20_4 20_5

# Cancel all elements from job array 20
[usr1@login1(viking) scratch]$ scancel 20
```

squeue command use

When a job array is submitted to Slurm, only one job record is created. Additional job records will only be created when the state of a task in the job array changes, typically when a task is allocated resources, or its state is modified using the *scontrol* command. By default, the *squeue* command will report all of the tasks associated with a single job record on one line and use a regular expression to indicate the "array_task_id" values as shown below.

```
[usr1@login1(viking) scratch]$ squeue -u usr1
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
       9_[102-500]     nodes array_jo     usr1 PD       0:00      1 (None)
               9_1     nodes array_jo     usr1  R       0:01      1 node169
               9_2     nodes array_jo     usr1  R       0:01      1 node169
               9_3     nodes array_jo     usr1  R       0:01      1 node169
               9_4     nodes array_jo     usr1  R       0:01      1 node169
               9_5     nodes array_jo     usr1  R       0:01      1 node169
               9_6     nodes array_jo     usr1  R       0:01      1 node169
               9_7     nodes array_jo     usr1  R       0:01      1 node169
               9_8     nodes array_jo     usr1  R       0:01      1 node169
               9_9     nodes array_jo     usr1  R       0:01      1 node170
              9_10     nodes array_jo     usr1  R       0:01      1 node170
...
```

GPU jobs

```
#!/bin/bash
#SBATCH --job-name=cuda_job                    # Job name
#SBATCH --mail-type=END,FAIL                   # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=a.user@york.ac.uk          # Where to send mail
#SBATCH --ntasks=1                             # Run a single task...
#SBATCH --cpus-per-task=1                      # ...with a single CPU
#SBATCH --mem=128gb                            # Job memory request
#SBATCH --time=01:00:00                        # Time limit hrs:min:sec
#SBATCH --output=cuda_job_%j.log               # Standard output and error log
#SBATCH --account=dept-proj-2018                   # Project account
#SBATCH --partition=gpu                        # Select the GPU nodes...
#SBATCH --gres=gpu:1                           # ...and a single GPU

module load system/CUDA/10.0.130

echo `date`: executing gpu_test on host $HOSTNAME with $SLURM_CPUS_ON_NODE cpu cores
echo
cudaDevs=$(echo $CUDA_VISIBLE_DEVICES | sed -e 's/,/ /g')
echo I can see GPU devices $CUDA_VISIBLE_DEVICES
echo

./cuda_example
```

WARNING: Missing out the line "#SBATCH --gres=gpu:1", will cause the job to be queued but it will not run, and will sit in the queue with a reason code of *QOSMinGRES*.

Selecting more than one GPU

```
#!/bin/bash
#SBATCH --job-name=gpu_test                    # Job name
#SBATCH --mail-type=END,FAIL                   # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=usr1@york.ac.uk            # Where to send mail
#SBATCH --ntasks=4                             # Run four tasks...
#SBATCH --cpus-per-task=1                      # ...with one CPU core each
#SBATCH --mem=28gb                             # Job memory request
#SBATCH --time=00:15:00                        # Time limit hrs:min:sec
#SBATCH --output=logs/gpu_test_%j.log          # Standard output and error log
#SBATCH --partition=gpu                        # Select the GPU nodes...
#SBATCH --gres=gpu:3                           # ...and three GPUs

module load system/CUDA/10.0.130

echo `date`: executing gpu_test on host $HOSTNAME with $SLURM_CPUS_ON_NODE cpu cores
echo
cudaDevs=$(echo $CUDA_VISIBLE_DEVICES | sed -e 's/,/ /g')
echo I can see GPU devices $CUDA_VISIBLE_DEVICES
echo

./add
```

```
Fri 22 Feb 14:20:22 GMT 2019: executing gpu_test on host gpu02 with 4 cpu cores

I can see GPU devices 1,2,3

[0]Found 3 gpu devices
Device Number: 0
  Device name: Tesla V100-SXM2-32GB
  Memory Clock Rate (KHz): 877000
  Memory Bus Width (bits): 4096
  Peak Memory Bandwidth (GB/s): 898.048000

Device Number: 1
  Device name: Tesla V100-SXM2-32GB
  Memory Clock Rate (KHz): 877000
  Memory Bus Width (bits): 4096
  Peak Memory Bandwidth (GB/s): 898.048000

Device Number: 2
  Device name: Tesla V100-SXM2-32GB
  Memory Clock Rate (KHz): 877000
  Memory Bus Width (bits): 4096
  Peak Memory Bandwidth (GB/s): 898.048000

3 + 4 is 7
```

Best Practices

Large Job Counts

Consider putting related work into a single Slurm job with multiple job steps both for performance reasons and ease of management. Each Slurm job can contain a multitude of job steps and the overhead in Slurm for managing job steps is much lower than that of individual jobs.

Job arrays are an efficient mechanism of managing a collection of batch jobs with identical resource requirements. Most Slurm commands can manage job arrays either as individual elements (tasks) or as a single entity (e.g. delete an entire job array in a single command).

Selecting irregular file names in array jobs

Often your data files can not be referenced via a unique numerical index - the file names will be dates or unrelated strings. This example demonstrates processing a set of files with no numerical index. We first make a file containing a list of the filenames. A short "awk" script returns the line as indexed by the numerical argument.

```
[usr1@login1(viking) scratch]$ ls idata/
Wed Nov 19 14:22:26 GMT 2014  Wed Nov 19 14:25:14 GMT
2014
Wed Nov 19 14:22:32 GMT 2014  Wed Nov 19 14:25:21 GMT
2014
Wed Nov 19 14:22:37 GMT 2014  Wed Nov 19 14:25:28 GMT
2014
Wed Nov 19 14:22:45 GMT 2014  Wed Nov 19 14:25:36 GMT
2014
Wed Nov 19 14:22:52 GMT 2014  Wed Nov 19 14:25:43 GMT
2014
Wed Nov 19 14:23:00 GMT 2014  Wed Nov 19 14:25:51 GMT
2014
Wed Nov 19 14:23:07 GMT 2014  Wed Nov 19 14:25:58 GMT
2014
Wed Nov 19 14:23:15 GMT 2014  Wed Nov 19 14:26:06 GMT
2014
```

```
Wed Nov 19 14:23:22 GMT 2014  Wed Nov 19 14:26:13 GMT
2014
Wed Nov 19 14:23:29 GMT 2014  Wed Nov 19 14:26:20 GMT
2014
Wed Nov 19 14:23:37 GMT 2014  Wed Nov 19 14:26:28 GMT
2014
Wed Nov 19 14:23:44 GMT 2014  Wed Nov 19 14:26:35 GMT
2014
Wed Nov 19 14:23:52 GMT 2014  Wed Nov 19 14:26:43 GMT
2014
Wed Nov 19 14:23:59 GMT 2014  Wed Nov 19 14:26:50 GMT
2014
Wed Nov 19 14:24:07 GMT 2014  Wed Nov 19 14:26:57 GMT
2014
Wed Nov 19 14:24:14 GMT 2014  Wed Nov 19 14:27:05 GMT
2014
Wed Nov 19 14:24:21 GMT 2014  Wed Nov 19 14:27:12 GMT
2014
Wed Nov 19 14:24:29 GMT 2014  Wed Nov 19 14:27:20 GMT
2014
Wed Nov 19 14:24:36 GMT 2014  Wed Nov 19 14:27:27 GMT
2014
Wed Nov 19 14:24:44 GMT 2014  Wed Nov 19 14:27:35 GMT
2014
Wed Nov 19 14:24:51 GMT 2014  Wed Nov 19 14:27:42 GMT
2014
Wed Nov 19 14:24:59 GMT 2014  Wed Nov 19 14:27:50 GMT
2014
Wed Nov 19 14:25:06 GMT 2014  Wed Nov 19 14:27:57 GMT
2014
[usr1@login1(viking) scratch]$ ls -1 idata/ > data.
files
[usr1@login1(viking) scratch]$ cat data.files
Wed Nov 19 14:22:26 GMT 2014
Wed Nov 19 14:22:32 GMT 2014
Wed Nov 19 14:22:37 GMT 2014
Wed Nov 19 14:22:45 GMT 2014
Wed Nov 19 14:22:52 GMT 2014
Wed Nov 19 14:23:00 GMT 2014
Wed Nov 19 14:23:07 GMT 2014
Wed Nov 19 14:23:15 GMT 2014
Wed Nov 19 14:23:22 GMT 2014
Wed Nov 19 14:23:29 GMT 2014
Wed Nov 19 14:23:37 GMT 2014
Wed Nov 19 14:23:44 GMT 2014
Wed Nov 19 14:23:52 GMT 2014
Wed Nov 19 14:23:59 GMT 2014
Wed Nov 19 14:24:07 GMT 2014
Wed Nov 19 14:24:14 GMT 2014
Wed Nov 19 14:24:21 GMT 2014
Wed Nov 19 14:24:29 GMT 2014
Wed Nov 19 14:24:36 GMT 2014
Wed Nov 19 14:24:44 GMT 2014
Wed Nov 19 14:24:51 GMT 2014
Wed Nov 19 14:24:59 GMT 2014
Wed Nov 19 14:25:06 GMT 2014
Wed Nov 19 14:25:14 GMT 2014
Wed Nov 19 14:25:21 GMT 2014
Wed Nov 19 14:25:28 GMT 2014
Wed Nov 19 14:25:36 GMT 2014
Wed Nov 19 14:25:43 GMT 2014
Wed Nov 19 14:25:51 GMT 2014
Wed Nov 19 14:25:58 GMT 2014
Wed Nov 19 14:26:06 GMT 2014
Wed Nov 19 14:26:13 GMT 2014
Wed Nov 19 14:26:20 GMT 2014
Wed Nov 19 14:26:28 GMT 2014
Wed Nov 19 14:26:35 GMT 2014
Wed Nov 19 14:26:43 GMT 2014
Wed Nov 19 14:26:50 GMT 2014
Wed Nov 19 14:26:57 GMT 2014
```

```
Wed Nov 19 14:27:05 GMT 2014
Wed Nov 19 14:27:12 GMT 2014
Wed Nov 19 14:27:20 GMT 2014
Wed Nov 19 14:27:27 GMT 2014
Wed Nov 19 14:27:35 GMT 2014
Wed Nov 19 14:27:42 GMT 2014
Wed Nov 19 14:27:50 GMT 2014
Wed Nov 19 14:27:57 GMT 2014
```

```
#!/bin/bash
#SBATCH --job-name=basic_job_test        # Job name
#SBATCH --mail-type=END,FAIL             # Mail events
(NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=usr1@york.ac.uk      # Where to
send mail
#SBATCH --ntasks=1                       # Run on a
single CPU
#SBATCH --mem=1gb                        # Job memory
request
#SBATCH --time=00:05:00                  # Time limit
hrs:min:sec
#SBATCH --output=array_job_%A_%a.log     # Standard
output and error log
#SBATCH --account=dept-proj-2018             # Project
account
#SBATCH --array=1-46                     # Array range

filename=$(awk NR==$SLURM_ARRAY_TASK_ID data.files)
Rscript analyseData.R "idata/$filename" "results
/$filename.rst"
```

Environment Variables

There are a number of environment variables which get set by SLURM and are available for use by your job script. The full list of these can be found in the sbatch man page, in the *`INPUT ENVIRONMENT VARIABLES`* and *`OUTPUT ENVIRONMENT VARIABLES`* sections.

These allow you to write more generic job scripts, and let SLURM fill in the specifics later on.

To use one of the variables in your script, simply precede it with a *`$`* as normal.

```
echo I am job $SLURM_JOB_NAME running in the $SLURM_JOB_PARTITION queue.
```

Task distribution

Distributing your tasks together on one node can sometimes improve the performance. You should not use this unless you have an in-depth understanding of how the software distributes its workload.

```
#SBATCH --distribution block
```

For more information, please read the sbatch man page.