Slurm Script

Basic Slurm script commands

Slurm script command	Description
#!/bin/bash	Sets the shell that the job will be executed on the compute node, this must be the first line of your job script
#SBATCH --mail-user= <email>	Sets the email address for sending notifications about your job state.
#SBATCH --mail-type=BEGIN	Sends an email when the job begins
#SBATCH --mail-type=FAIL	Sends an email if the job fails
#SBATCH --mail-type=REQUEUE	Sends an email if a job is requeued (restarted)
#SBATCH --mail-type=END	Sends an email when a job ends
##SBATCH --mail-type=ALL	Sends an email when the job: begins, fails, ends, or is restarted

Slurm script commands

Slurm script command	Description
#SBATCH --job-name=nameofjob	Sets the Jobs name
#SBATCH -o output_file	Sets the file name that collects output the job.
#SBATCH -o output_file_**%j_%x**	Sets the file name that collects output the job. %j is replaced with the jobs if number %x is replaced with th jobs name
#SBATCH -e error_file	If set sends any error messages (Standard error) to a file with the specified name. If not set the std err is sent to the single output file instead.
#SBATCH -e error_file_**%j_%x**	If set sends any error messages (Standard error) to a file with the specified name. If not set the std err is sent to the single output file instead. %j is replaced with the jobs number %x is replaced with the jobs name

Slurm script commands

Slurm script command	Description
#SBATCH --time=0-05:00 #SBATCH -t 0-05:00	Sets the maximum runtime of 5 hours for your job
#SBATCH --ntasks=1 #SBATCH --n1	Requests for 1 processors on task, usually 1 cpu as 1 cpu per task is default.
#SBATCH --ntasks=X	Requests for X tasks. When cpus-per-task=1 (and this is the default) this requests X cores When not otherwise constraint these CPUs may be running on any node
#SBATCH --nodes=X	Request that a minimum of X nodes be allocated to this job
#SBATCH --nodes=X-Y	Request that a minimum of X nodes and a maximum of Y nodes be allocated to this job
#SBATCH --cpus-per-task=X	Request that a minimum of X CPUs per task be allocated to this job
#SBATCH --tasks-per-node=X	Requests minimum of X task be allocated per node

Slurm script commands

Slurm script commands	Description of effects
#SBATCH --ntasks=1 #SBATCH --cpus-per-task=1	Requests 1 CPU (Serial) cpus-per-task is set to 1 by default and may be omitted.
#SBATCH --cpus-per-task=X #SBATCH --ntasks=1 #SBATCH --nodes=1	Requests for X CPUs in 1 task on 1 node (OpenMP) Both ntasks and nodes are set to 1 by default and may be omitted
#SBATCH --ntasks=X #SBATCH --tasks-per-node=X #SBATCH --cpus-per-task=1	Requests for X CPUs and tasks on 1 node (OpenMP) cpus-per-task is set to 1 by default and may be omitted.
#SBATCH --ntasks=X #SBATCH --nodes=1 #SBATCH --cpus-per-task=1	Requests for X CPUs and tasks on 1 node (OpenMP) cpus-per-task is set to 1 by default and may be omitted.

Slurm script commands

Slurm script commands	Description of effects
#SBATCH --ntasks=X #SBATCH --cpus-per-task=1	Requests X CPUs and tasks (MPI) cpus-per-task is set to 1 by default and may be omitted.
#SBATCH --ntasks=X #SBATCH --ntasks-per-node=Y #SBATCH --cpus-per-task=1	Requests for X CPUs and tasks with Y CPUs and tasks per node (MPI) cpus-per-task is set to 1 by default and may be omitted.
#SBATCH --ntasks=X #SBATCH --nodes=1 #SBATCH --cpus-per-task=1	Requests for X CPUs and tasks on the same node, cpus-per-task is set to 1 by default and may be omitted.
#SBATCH --ntasks=X #SBATCH --nodes=1 #SBATCH --cpus-per-task=1	Requests for X CPUs and tasks on the 1 node cpus-per-task is set to 1 by default and may be omitted.

Slurm script commands

PBS script command	Description
#SBATCH --mem=4000	Requests 4000 MB of memory in total
#SBATCH --mem-per-cpu=4000	Requests 4000 MB of memory per cpu
#SBATCH --licenses=sas:2	Requests 2 SAS licenses
#SBATCH --gres=gpu:1	Requests that your job get 1 GPU allocated per node
#SBATCH --exclusive	Requests that your job run only on nodes with no other running jobs
#SBATCH --dependency=after:job_id1	Requests that the the job start after job (jobid1) has **started**
#SBATCH --dependency=afterany:job_id1, job_i2	Requests that the the job start after ether job (jobid1) or job (jobud2) has **finished**
#SBATCH --dependency=afterok:job_id1	Requests that the the job start after job (jobid1) has **finished successfully**

SLURM Environment Variables

Environment Variable	Description
SLURM_JOB_NAME	User specified job name
SLURM_JOB_ID	Unique slurm job id
SLURM_NNODES	Number of nodes allocated to the job
SLURM_NTASKS	Number of tasks allocated to the job
SLURM_ARRAY_TASK_ID	Array index for this job
SLURM_ARRAY_TASK_MAX	Total number of array indexes for this job
SLURM_MEM_PER_CPU	Memory allocated per CPU
SLURM_JOB_NODELIST	List of nodes on which resources are allocated to Job
SLURM_JOB_CPUS_PER_NODE	Number of CPUs allocated per Node
SLURM_JOB_PARTITION	List of Partition(s) that the job is in.
SLURM_JOB_ACCOUNT	Account under which this job is run.

Slurm script commands

PBS script command	Description
#SBATCH --mem=4000	Requests 4000 MB of memory in total
#SBATCH --mem-per-cpu=4000	Requests 4000 MB of memory per cpu
#SBATCH --gres=gpu:1	Requests 1 GPU to be allocated per node
#SBATCH --licenses=sas:2	Requests 2 SAS licenses
#SBATCH --dependency=after:job_id1	Requests that the the job start after job (jobid1) has **started**
#SBATCH --dependency=afterany:job_id1, job_id2	Requests that the the job start after ether job (jobid1) or job (jobid2) has **finished**
#SBATCH --dependency=afterok:job_id1	Requests that the the job start after job (jobid1) has **finished successfully**

Slurm script commands

PBS script command	Description
#SBATCH --account=acc_name	To submit a job to a specific accounting group such as RAC/RAS allocation or different role
#SBATCH --exclusive	Requests that your job run only on nodes with no other running jobs
#SBATCH --tmp=200G	Asks for 200Gb of temporary disk space
#SBATCH --constraint=blue	To ask for a node feature or constraint set by cluster admin. Here we are looking for “blue” nodes.
#SBATCH --partition=partition_name	To ask for the job to run in a specific partition or queue by name, (unlike Moab there can be more than 1 partition per Job)
--prolog=<executable>	Run by srun only, runs the executable before the step
--epilog=<executable>	Run by srun only, runs the executable after the step finishes

SLURM Environment Variables

Environment Variable	Description
SLURM_JOB_NAME	User specified job name
SLURM_JOB_ID	Unique slurm job id
SLURM_NNODES	Number of nodes allocated to the job
SLURM_NTASKS	Number of tasks allocated to the job
SLURM_ARRAY_TASK_ID	Array index for this job
SLURM_ARRAY_TASK_MAX	Total number of array indexes for this job
SLURM_MEM_PER_CPU	Memory allocated per CPU
SLURM_JOB_NODELIST	List of nodes on which resources are allocated to Job
SLURM_JOB_CPUS_PER_NODE	Number of CPUs allocated per Node
SLURM_JOB_PARTITION	List of Partition(s) that the job is in.
SLURM_JOB_ACCOUNT	Account under which this job is run.

Getting information on your Job

Command	What its used for
squeue -u <username>	List all current jobs for a user
squeue -u <username> -t PENDING	List all pending jobs for a user
squeue -u <username> -t RUNNING	List all running jobs for a user
Squeue -p <partitionname>	List all the jobs in a partition
scontrol show job <jobid>	List information on Job
scontrol show jobid -dd <jobid>	List detailed information on Job
Squeue -o "%.18i %.30P %.8j %.8u %.2t %.8p %.10M %.6D %R "	Formatted output of squeue: we added priority and made the partition field bigger (30 characters)

Controlling jobs

Command	What its used for
salloc --nodes=1 --time=0-01:20 (srun -- pty bash)	Allocate and run a interactive job
srun --jobid <jobid> --pty	Interactively join a running job
srun --jobid <jobid> --pty --x11	Interactively join a running job with GUI
scancel <jobid>	Cancel job
scancel -u <username>	Cancel all the jobs for a user
scancel -t PENDING -u <username>	Cancel all the pending jobs for a user:
Scancel -name JobName	Cancel one or more jobs by name
scontrol hold <jobid>	Hold a job, prevent it form starting
scontrol resume <jobid>	Release a job hold, allowing the job to try to start
scontrol requeue <jobid>	Requeue a running, suspended or finished job into pending state
scontrol requeuehold<jobid>	First requeue the job than put a hold on it.
squeue -u <username> -ho %A -t R	List running jobs by user
squeue --start	Show expected start time of jobs. (This can change)

Getting information on your Job

Command	What its used for
sstat --format=AveCPU,MaxRSS,MaxVMSize,JobID -j <jobid>	List info resource used by your completed job : average cpu time, Max memory, Max virtual memory, JobId
sacct –u <username> --format=JobID,JobName,AveCPU,MaxRSS,MaxVMSize,JobID,Elapsed	List resources used by all jobs of a user
sprio	List job priority information
srun –jobid <jobid> --pty	Interactively join a running job and get a console inside the job. Use standard Unix commands to get information.

Getting information on you and your group

Command	What its used for
sacctmgr list Users USERS=<username>	List user and their default account (accounting group)
sacctmgr show user <username> withassoc	List user and their default account (accounting group) and shows more extensive information
sshare	Shows usage info for user usage and priority.
sshare -l	Shows even more info for user usage and priority.

Getting information on your Cluster

Command	What its used for
sinfo --states=idle	Show idle node on cluster
sinfo -R	Show down, drained and draining nodes and their reason
sinfo --Node --long	Show detailed node info.
scontrol show reservation	Shows reservations on the cluster
partition-stats	Compute Canada script to show jobs and nodes by partition
scontrol create reservation user=root starttime=now duration=infinite flags=maint nodes=<nodeid>	Used by cluster administrators to create a reservation