SLURM – Advanced Usage

March 12, 2025

Bad Job Practices

- job submissions within a loop (take a long time)

```
for i in {1..1000}
do
    sbatch job.sh $i
done
```

- loop inside job script (sequence of mpirun commands):

```
for i in {1..1000}
do
    mpirun -np 16 my_program $i
done
```

VIENNA SCIENTIFIC CLUSTER

Array Jobs

- submit/run a series of independent jobs via a single SLURM script

Array Jobs

- submit/run a series of independent jobs via a single SLURM script
- each job in the array gets a unique identifier (SLURM_ARRAY_TASK_ID) based on which various workloads can be organized

Array Jobs

- submit/run a series of independent jobs via a single SLURM script
- each job in the array gets a unique identifier (SLURM_ARRAY_TASK_ID) based on which various workloads can be organized
- example (job_array_vsc5.sh), 10 jobs, SLURM_ARRAY_TASK_ID=1,2,3 . . . 10

```
#!/bin/bash
#SBATCH -J array
#SBATCH -N 1
#SBATCH --array=1-10

echo "Hi, this is array job number" $SLURM_ARRAY_TASK_ID
sleep $SLURM_ARRAY_TASK_ID
```

Array Jobs

- independent jobs: 1, 2, 3 . . . 10

VSC-5 > squeue -u $user

499514_3	zen3_0512	array	sh	R	INVALID	1	n3504-057
499514_4	zen3_0512	array	sh	R	INVALID	1	n3506-047
499514_5	zen3_0512	array	sh	R	INVALID	1	n3507-013
499514_6	zen3_0512	array	sh	R	INVALID	1	n3509-016
499514_7	zen3_0512	array	sh	R	INVALID	1	n3511-029
499514_8	zen3_0512	array	sh	R	INVALID	1	n3503-010
499514_9	zen3_0512	array	sh	R	INVALID	1	n3503-011
499514_10	zen3_0512	array	sh	R	INVALID	1	n3503-028

Array Jobs

- independent jobs: 1, 2, 3 . . . 10

VSC-5 > squeue -u $user

```
499514_3    zen3_0512  array  sh  R  INVALID  1  n3504-057
499514_4    zen3_0512  array  sh  R  INVALID  1  n3506-047
499514_5    zen3_0512  array  sh  R  INVALID  1  n3507-013
499514_6    zen3_0512  array  sh  R  INVALID  1  n3509-016
499514_7    zen3_0512  array  sh  R  INVALID  1  n3511-029
499514_8    zen3_0512  array  sh  R  INVALID  1  n3503-010
499514_9    zen3_0512  array  sh  R  INVALID  1  n3503-011
499514_10   zen3_0512  array  sh  R  INVALID  1  n3503-028
```

- corresponding SLURM output files

VSC-5 > ls slurm-*

```
slurm-499514_10.out  slurm-499514_2.out  slurm-499514_4.out  slurm-499514_6.out  slurm-499514_8.out
slurm-499514_1.out   slurm-499514_3.out  slurm-499514_5.out  slurm-499514_7.out  slurm-499514_9.out
```

Array Jobs

- independent jobs: 1, 2, 3 . . . 10

VSC-5 > squeue -u $user

```
499514_3     zen3_0512  array  sh  R  INVALID  1  n3504-057
499514_4     zen3_0512  array  sh  R  INVALID  1  n3506-047
499514_5     zen3_0512  array  sh  R  INVALID  1  n3507-013
499514_6     zen3_0512  array  sh  R  INVALID  1  n3509-016
499514_7     zen3_0512  array  sh  R  INVALID  1  n3511-029
499514_8     zen3_0512  array  sh  R  INVALID  1  n3503-010
499514_9     zen3_0512  array  sh  R  INVALID  1  n3503-011
499514_10    zen3_0512  array  sh  R  INVALID  1  n3503-028
```

- corresponding SLURM output files

VSC-5 > ls slurm-*

```
slurm-499514_10.out  slurm-499514_2.out  slurm-499514_4.out  slurm-499514_6.out  slurm-499514_8.out
slurm-499514_1.out   slurm-499514_3.out  slurm-499514_5.out  slurm-499514_7.out  slurm-499514_9.out
```

- explicit content of a single SLURM output file

VSC-5 > cat slurm-499514_8.out
Hi, this is array job number 8

Array Jobs

- fine-tuning via builtin variables (SLURM_ARRAY_TASK_MIN, SLURM_ARRAY_TASK_MAX etc)

Array Jobs

- fine-tuning via builtin variables (SLURM_ARRAY_TASK_MIN, SLURM_ARRAY_TASK_MAX etc)

- example of going in chunks of a certain size, e.g. 5, SLURM_ARRAY_TASK_ID=1,6,11,16

```
#SBATCH --array=1-20:5
```

Array Jobs

- fine-tuning via builtin variables (SLURM_ARRAY_TASK_MIN, SLURM_ARRAY_TASK_MAX etc)
- example of going in chunks of a certain size, e.g. 5, SLURM_ARRAY_TASK_ID=1,6,11,16

```
#SBATCH --array=1-20:5
```

- example of limiting number of simultaneously running jobs to 2 (perhaps for licences)

```
#SBATCH --array=1-20:5%2
```

Single Core Jobs

- use an entire compute node for several independent jobs

Single Core Jobs

- use an entire compute node for several independent jobs
- example single_node_multiple_jobs_vsc5.sh:

```
#!/bin/bash
#SBATCH -J snglcre
#SBATCH -N 1
#SBATCH -p zen3_0512
#SBATCH --qos=zen3_0512

for ((i=1; i<=128; i++))
do
    stress --cpu 1 --timeout $i &
done
wait
```

Single Core Jobs

- & is important ! sends the process into the background so that the script can continue

Single Core Jobs

- & is important ! sends the process into the background so that the script can continue
- "wait" is also important ! waits for all processes in the background to terminate before moving on

Combination of Array and Single Core Job

- example combined_array_multiple_jobs_vsc5.sh:

```
...
#SBATCH -N 1
#SBATCH --array=1-384:128

j=$SLURM_ARRAY_TASK_ID
((j+=127))

for ((i=$SLURM_ARRAY_TASK_ID; i<=$j; i++))
do
    stress --cpu 1 --timeout $i &
done
wait
```

Exercises

- files are located in folder examples/05_submitting_batch_jobs
- look into "job_array_vsc[4, 5].sh" and modify it such that the considered range is from 1 to 20 but in steps of 5
- look into "single_node_multiple_jobs_vsc[4, 5].sh" and also change it to go in steps of 5
- run "combined_array_multiple_jobs_vsc[4, 5].sh" and check whether the output is reasonable

Job Script Enhancements

- usage of corresponding environmental variables

#SBATCH	Environmental Variable
-N	SLURM_JOB_NUM_NODES
--ntasks-per-core	SLURM_NTASKS_PER_CORE
--ntasks-per-node	SLURM_NTASKS_PER_NODE
--ntasks [-n]	SLURM_NTASKS

Job Script Enhancements

- usage of corresponding environmental variables

#SBATCH	Environmental Variable
-N	SLURM_JOB_NUM_NODES
--ntasks-per-core	SLURM_NTASKS_PER_CORE
--ntasks-per-node	SLURM_NTASKS_PER_NODE
--ntasks [-n]	SLURM_NTASKS

- email notifications

```
...
#SBATCH --mail-user=yourmail@example.com
#SBATCH --mail-type=BEGIN,END
```

Submission Scripts Tuning

- using time constraints less than runtime limits

```
...
#SBATCH --time=DD-HH[:MM[:SS]]
```

Submission Scripts Tuning

- using time constraints less than runtime limits

```
...
#SBATCH --time=DD-HH[:MM[:SS]]
```

- backfilling:
  the specified time is an estimate of your required computing time; if this is shorter than the default runtime limit (mostly 24h), SLURM may squeeze it in on idle nodes waiting for a larger job;

Submission Scripts Tuning

- using time constraints less than runtime limits

```
...
#SBATCH --time=DD-HH[:MM[:SS]]
```

- backfilling:
  the specified time is an estimate of your required computing time; if this is shorter than the default runtime limit (mostly 24h), SLURM may squeeze it in on idle nodes waiting for a larger job;

- max runtime limit is 72h

```
...
#SBATCH --time=03-00:00:00
```

Reservation of Compute Nodes

- core-h accounting is done for the entire period of reservation

VIENNA SCIENTIFIC CLUSTER

Reservation of Compute Nodes

- core-h accounting is done for the entire period of reservation
- contact support@vsc.ac.at

VIENNA SCIENTIFIC CLUSTER

Reservation of Compute Nodes

- core-h accounting is done for the entire period of reservation
- contact support@vsc.ac.at
- reservations are named after the project id

Reservation of Compute Nodes

- core-h accounting is done for the entire period of reservation
- contact support@vsc.ac.at
- reservations are named after the project id
- check for reservations

```
VSC-5> scontrol show reservations
```

Reservation of Compute Nodes

- core-h accounting is done for the entire period of reservation
- contact support@vsc.ac.at
- reservations are named after the project id
- check for reservations

```
VSC-5> scontrol show reservations
```

- using reservations

```
...
#SBATCH --reservation=MyRsrv
```

Job Dependencies

1. Submit first job and get its <job_id>

Job Dependencies

1. Submit first job and get its <job_id>

2. Submit dependent job using the just received parent <job_id>

```
#!/bin/bash
#SBATCH -J myjb
#SBATCH -N 2
#SBATCH --dependency=afterany:<job_id>
mpirun -np 256 my_prog
...
```

Job Dependencies

1. Submit first job and get its <job_id>

2. Submit dependent job using the just received parent <job_id>

```
#!/bin/bash
#SBATCH -J myjb
#SBATCH -N 2
#SBATCH --dependency=afterany:<job_id>
mpirun -np 256 my_prog
...
```

3. continue with **2.** for further dependent jobs

MPI + Pinning

- important issue to place various processes correctly on individual cores

MPI + Pinning

- important issue to place various processes correctly on individual cores
- use only a few processes per node if memory demand is high

MPI + Pinning

- important issue to place various processes correctly on individual cores
- use only a few processes per node if memory demand is high
- details: `https://wiki.vsc.ac.at/doku.php?id=doku:slurm_corepinning`

MPI + Pinning

- important issue to place various processes correctly on individual cores
- use only a few processes per node if memory demand is high
- details: `https://wiki.vsc.ac.at/doku.php?id=doku:slurm_corepinning`
- "srun" example 2 nodes with two MPI processes each

```
#!/bin/bash
#SBATCH -J myjb
#SBATCH -N 2
#SBATCH --tasks-per-node=2

srun --cpu-bind=map_cpu:0,64 ./my_mpi_program
```

MPI + Pinning

- "INTEL MPI" example 2 nodes with two MPI processes each

```
#!/bin/bash
#SBATCH -J myjb
#SBATCH -N 2
#SBATCH --tasks-per-node=2

export I_MPI_PIN_PROCESSOR_LIST=0,64
mpirun -np 4 ./my_mpi_program
```

Exercises-2

- check for available reservations. If there is one available, use it
- specify an email address that notifies you when your job has finished