جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

Job scheduling on Shaheen III

What is SLURM?

- A resource manager
  - Manages more work than the resource by scheduling queues of work
- Supports complex scheduling of algorithms

- Provides:
  - Way to describing and submitting a resource request
  - Way to prescribing how to run workload on allocated resource
  - Way to monitoring the state of the submitted ”job”
  - Way to account for the resources used (charging system)

Querying system resources

sinfo

- A concise view of the system resources and their availability
  - Shows state of resources in every `paritition/queue` on the system
- `partition`: a pool of homogenoeus compute resources adhering to a policy

```
> sinfo
PARTITION   AVAIL JOB_SIZE  TIMELIMIT   CPUS  S:C:T NODES STATE
workq*      up    1-4.50K  1-00:00:00    384 2:96:2     1 mixed
workq*      up    1-4.50K  1-00:00:00    384 2:96:2   699 allocated
workq*      up    1-4.50K  1-00:00:00    384 2:96:2  3908 idle
72hours     up    1-128    3-00:00:00    384 2:96:2     1 mixed
72hours     up    1-128    3-00:00:00    384 2:96:2   699 allocated
72hours     up    1-128    3-00:00:00    384 2:96:2  3908 idle
debug       up    1-4         4:00:00    384 2:96:2     4 idle
shared      up    1-infini 1-00:00:00    384 2:96:2     1 mixed
shared      up    1-infini 1-00:00:00    384 2:96:2    15 idle
ppn         up    1-15     1-00:00:00    256 2:64:2    15 idle
dtn         up    1-4      1-00:00:00     48 1:24:2     4 idle
```

squeue

- Shows the list of jobs in different SLURM partitions

```
> squeue | head -n 10
JOBID      USER ACCOUNT         NAME   ST REASON         START_TIME       TIME  TIME_LEFT NODES
157319    ******  k#####  interactive    R None  2024-03 31T15:46:43    1:10:00    1:50:00   16
151927    ******  k#####   gaussian16    R None  2024-03-31T00:39:14   16:17:29    7:42:31    1
151928    ******  k#####   gaussian16    R None  2024-03-31T00:39:17   16:17:26    7:42:34    1
151938    ******  k#####   gaussian16    R None  2024-03-31T01:15:11   15:41:32    8:18:28    1
155825    ******     k##      SN_feat    R None  2024-03-31T15:21:52    1:34:51    8:25:09    1
```

squeue

- Shows the list of jobs in different SLURM partitions
- You can filter the list by:
  - User `squeue -u $USER`
  - Partition `squeue -p debug`
  - Job ID `squeue -j  12345678`

Specifying resources

Requestable resources

- CPUs
- GPUs (pre-post processing)
- Memory
- Wall time

Requesting resources

- You may run your workload as batch job or interactive job
- Implications:

Batch job

- You will write a jobscript with resource request and the commands to run when resources available
- Submit to the scheduler
- Scheduler will run the jobscript on your behalf once the resources are available
- Jobs can have requests for long durations (24-72 hours)

Interactive job

- Resource requests are usually small and short
- You run each command yourself one after the other
- Useful for prototyping and debugging workflows and creating jobscripts

Sample jobscript

```
#!/bin/bash

#SBATCH --account=k01
#SBATCH --job-name=myfirstjob
#SBATCH --time=01:00:00
#SBATCH --partition=workq
#SBATCH --ntasks=192
#SBATCH --hint=nomultithread

module load my-package

srun --hint=nomultithread -n ${SLURM_NTASKS} myapp

> sbatch jobscript.slurm
```

sbatch

- Command to submit jobscripts to SLURM
  - Upon successful submission a unique jobID is assigned
  - Job is queued in a paritition and awaits allocation of the requested resources
  - A priority value is assigned to each job – increases with age of job in the paritition

scancel

- SLURM command to cancel a queued job:

```
> squeue --me
JOBID       USER ACCOUNT        NAME    ST REASON  START_TIME           TIME  TIME_LEFT NODES
232870   #######     k##  sleep.slurm    R   None  2024-04-11T15:43:50  0:09      59:51     1

> scancel 232870

> squeue --me
JOBID       USER ACCOUNT        NAME    ST REASON  START_TIME           TIME  TIME_LEFT NODES
```

salloc

- Comman to request allocation of resource for *interactive use*:

```
> salloc -p debug --hint=nomultithread
salloc: Granted job allocation 232879
salloc: Waiting for resource configuration
salloc: Nodes nid04605 are ready for job

> srun -l -n4  -c 48 bash -c 'taskset -cp $$'
0: pid 468435's current affinity list: 0-47
1: pid 468436's current affinity list: 96-143
2: pid 468437's current affinity list: 48-95
3: pid 468438's current affinity list: 144-191

> exit
exit
salloc: Relinquishing job allocation 232879
```

Using allocated resources

srun

Once allocated, `srun` command can be used to launch apps on compute nodes

```
> salloc -p debug
salloc: Granted job allocation 232893
salloc: Waiting for resource configuration
salloc: Nodes nid04605 are ready for job
```

```
> srun -l -n 1 -c 1 echo 'sequential step'
0: sequential step
> srun -l -n 4 -c 1 echo 'parallel step' | sort -n
0: parallel step
1: parallel step
2: parallel step
3: parallel step
```

srun

- Each srun command is accounted as a “job step” by SLURM
- After completion of a ”job step” the allocation continues to exist
- This implies you can run multiple job steps in on an allocated resource

srun (in jobscript)

```
#!/bin/bash
#SBATCH --account=k##
#SBATCH --time=01:00:00
#SBATCH --partition=workq
#SBATCH --ntasks=8
#SBATCH --cpus-per-task=4
#SBATCH --hint=nomultithread

# sequential step
srun -n 1 ./serial_step
# parallel step
srun -n 8 -c 4 ./parallel_step
```

```
#!/bin/bash
#SBATCH --account=k##
#SBATCH --time=01:00:00
#SBATCH --partition=workq
#SBATCH --ntasks=32
#SBATCH --cpus-per-task=1
#SBATCH --hint=nomultithread

for (( i=0; i< ${SLURM_NTASKS}; i++))
do
  srun -n 1 -c 1 ./a.out task_index &
done
wait
```

Monitoring and accounting

squeue

- You can query the state of your job using `squeue`
- Common filters to modify output
  - -u : by <username> (`--me`)
  - -j : by <job-id>
  - -n : by <job-name>

```
> squeue --me
JOBID       USER ACCOUNT           NAME  ST REASON    START_TIME   TIME  TIME_LEFT NODES
233767   shaima0d     k01    sleep.slurm  PD None       N/A         0:00    1:00:00     1
> squeue -j 233767
JOBID       USER ACCOUNT           NAME  ST REASON    START_TIME   TIME  TIME_LEFT NODES
233767   shaima0d     k01    sleep.slurm  PD None       N/A         0:00    1:00:00     1
```

scontrol

- `scontrol` with subcommand `show job` shows parameters of the job allocation in detail
- Recommended to add in jobscripts to collect useful information when seeking support
- Query only works for the lifetime of the job

```
> scontrol show job 232876
JobId=232876 JobName=monitor
   UserId=maj0a(165718) GroupId=g-maj0a(1165718)
MCS_label=N/A
   Priority=1 Nice=0 Account=k10014 QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
……
   ReqTRES=cpu=32,mem=12337760M,node=32,billing=32
   AllocTRES=cpu=12288,node=32,billing=12288
   Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=*
 …..
```

sacct

- Displays historic accounting information on user level for jobs and job-steps.

```
> sacct -j 121011
JobID           JobName  Partition    Account    NNodes      State ExitCode
------------ ---------- ---------- ---------- -------- ---------- --------
121011            uni-ST      workq     k#####        8  COMPLETED      0:0
121011.batch       batch                k#####        1  COMPLETED      0:0
121011.exte+      extern                k#####        8  COMPLETED      0:0
121011.0     synthetic+                k#####        8  COMPLETED      0:0
```

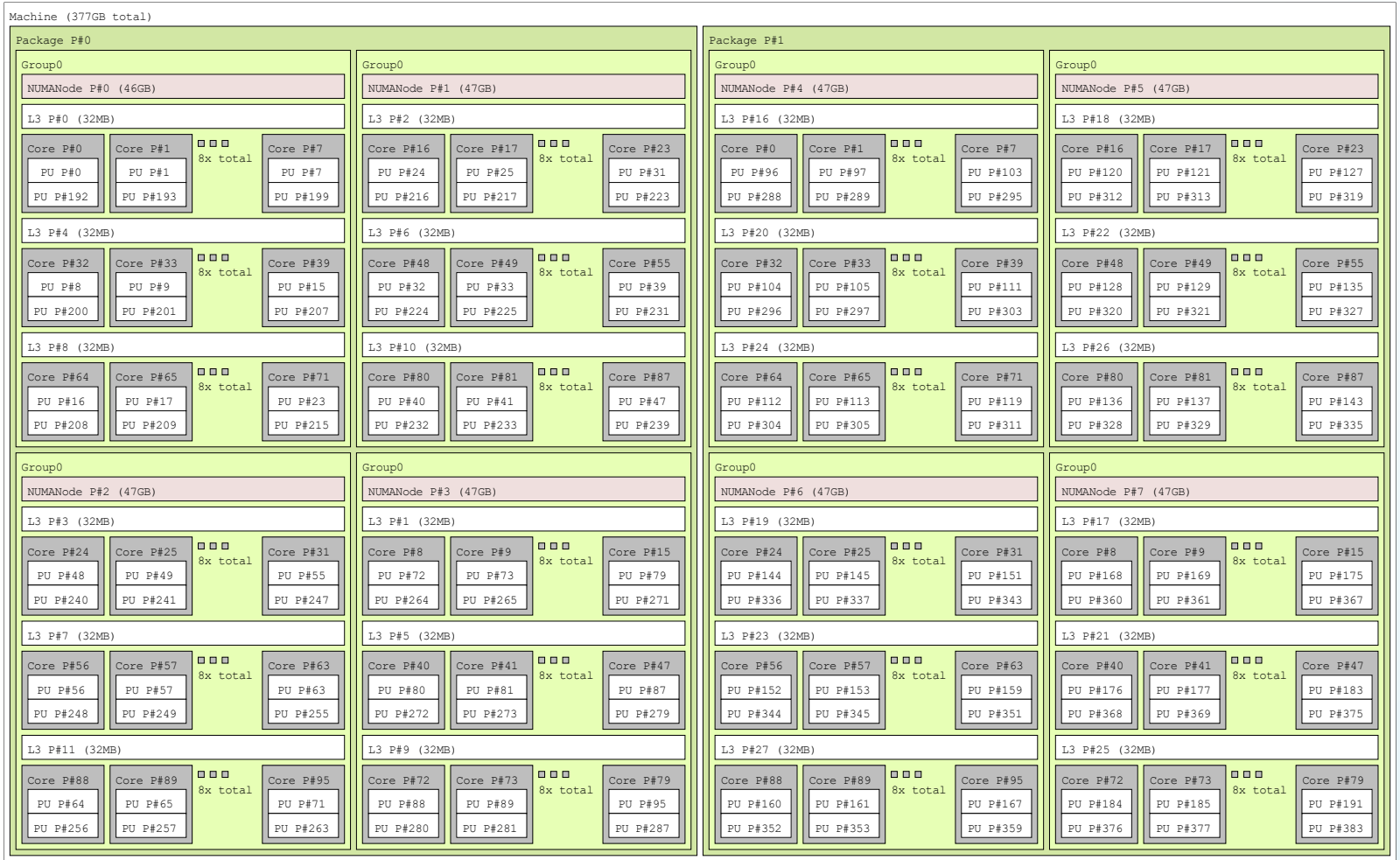
- Information can be retrieved even after the lifetime of the job

sacct

- Common filters to modify output
  - -u : by <username>
  - -j : by <job-id>
  - -n : by <job-name>
- Collect metrics on a completed job

`-o  jobid,elapsed`	for reporting elapsed time of a job
`-o jobid,nodelist%100`	for listing all node IDs allocated to the job
`-o jobid,ntasks`	for listing number of tasks requested for a jobstep
`-o jobid,CPUTime,MaxRss`	for reporting CPUtime for each jobstep (elapsed * CPU count)

Node Architecture


Machine (377GB total)
Package P#0
Group0
NUMANode P#0 (46GB)
L3 P#0 (32MB)
Core P#0 PU P#0 PU P#192
Core P#1 PU P#1 PU P#193
8x total
Core P#7 PU P#7 PU P#199
L3 P#4 (32MB)
Core P#32 PU P#8 PU P#200
Core P#33 PU P#9 PU P#201
8x total
Core P#39 PU P#15 PU P#207
L3 P#8 (32MB)
Core P#64 PU P#16 PU P#208
Core P#65 PU P#17 PU P#209
8x total
Core P#71 PU P#23 PU P#215
Group0
NUMANode P#1 (47GB)
L3 P#2 (32MB)
Core P#16 PU P#24 PU P#216
Core P#17 PU P#25 PU P#217
8x total
Core P#23 PU P#31 PU P#223
L3 P#6 (32MB)
Core P#48 PU P#32 PU P#224
Core P#49 PU P#33 PU P#225
8x total
Core P#55 PU P#39 PU P#231
L3 P#10 (32MB)
Core P#80 PU P#40 PU P#232
Core P#81 PU P#41 PU P#233
8x total
Core P#87 PU P#47 PU P#239
Group0
NUMANode P#2 (47GB)
L3 P#3 (32MB)
Core P#24 PU P#48 PU P#240
Core P#25 PU P#49 PU P#241
8x total
Core P#31 PU P#55 PU P#247
L3 P#7 (32MB)
Core P#56 PU P#56 PU P#248
Core P#57 PU P#57 PU P#249
8x total
Core P#63 PU P#63 PU P#255
L3 P#11 (32MB)
Core P#88 PU P#64 PU P#256
Core P#89 PU P#65 PU P#257
8x total
Core P#95 PU P#71 PU P#263
Group0
NUMANode P#3 (47GB)
L3 P#1 (32MB)
Core P#8 PU P#72 PU P#264
Core P#9 PU P#73 PU P#265
8x total
Core P#15 PU P#79 PU P#271
L3 P#5 (32MB)
Core P#40 PU P#80 PU P#272
Core P#41 PU P#81 PU P#273
8x total
Core P#47 PU P#87 PU P#279
L3 P#9 (32MB)
Core P#72 PU P#88 PU P#280
Core P#73 PU P#89 PU P#281
8x total
Core P#79 PU P#95 PU P#287
Package P#1
Group0
NUMANode P#4 (47GB)
L3 P#16 (32MB)
Core P#0 PU P#96 PU P#288
Core P#1 PU P#97 PU P#289
8x total
Core P#7 PU P#103 PU P#295
L3 P#20 (32MB)
Core P#32 PU P#104 PU P#296
Core P#33 PU P#105 PU P#297
8x total
Core P#39 PU P#111 PU P#303
L3 P#24 (32MB)
Core P#64 PU P#112 PU P#304
Core P#65 PU P#113 PU P#305
8x total
Core P#71 PU P#119 PU P#311
Group0
NUMANode P#5 (47GB)
L3 P#18 (32MB)
Core P#16 PU P#120 PU P#312
Core P#17 PU P#121 PU P#313
8x total
Core P#23 PU P#127 PU P#319
L3 P#22 (32MB)
Core P#48 PU P#128 PU P#320
Core P#49 PU P#129 PU P#321
8x total
Core P#55 PU P#135 PU P#327
L3 P#26 (32MB)
Core P#80 PU P#136 PU P#328
Core P#81 PU P#137 PU P#329
8x total
Core P#87 PU P#143 PU P#335
Group0
NUMANode P#6 (47GB)
L3 P#19 (32MB)
Core P#24 PU P#144 PU P#336
Core P#25 PU P#145 PU P#337
8x total
Core P#31 PU P#151 PU P#343
L3 P#23 (32MB)
Core P#56 PU P#152 PU P#344
Core P#57 PU P#153 PU P#345
8x total
Core P#63 PU P#159 PU P#351
L3 P#27 (32MB)
Core P#88 PU P#160 PU P#352
Core P#89 PU P#161 PU P#353
8x total
Core P#95 PU P#167 PU P#359
Group0
NUMANode P#7 (47GB)
L3 P#17 (32MB)
Core P#8 PU P#168 PU P#360
Core P#9 PU P#169 PU P#361
8x total
Core P#15 PU P#175 PU P#367
L3 P#21 (32MB)
Core P#40 PU P#176 PU P#368
Core P#41 PU P#177 PU P#369
8x total
Core P#47 PU P#183 PU P#375
L3 P#25 (32MB)
Core P#72 PU P#184 PU P#376
Core P#73 PU P#185 PU P#377
8x total
Core P#79 PU P#191 PU P#383
Host: nid00799
Indexes: physical
Date: Wed 03 Apr 2024 11:53:18 PM +03

Example Jobscripts

Serial jobs

```
#!/bin/bash
#SBATCH --partition=workq
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --hint=nomultithread
#SBATCH --account=k#####
#SBATCH --time=01:00:00

scontrol show job ${SLURM_JOBID}
# load your software environment here:
srun --hint=nomultithread ./a.out
```

OpenMP jobs

```
#!/bin/bash
#SBATCH --partition=workq
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=48
#SBATCH --hint=nomultithread
#SBATCH --account=k#####
#SBATCH --time=01:00:00
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
scontrol show job ${SLURM_JOBID}
# load your software environment here:
srun --hint=nomultithread -c ${OMP_NUM_THREADS} \
   --cpu-bind=verbose,cores ./a.out
```

MPI jobs

```
#!/bin/bash
#SBATCH --partition=workq
#SBATCH --ntasks=192
#SBATCH --ntasks-per-node=192
#SBATCH --hint=nomultithread
#SBATCH --account=k#####
#SBATCH --time=01:00:00
export OMP_NUM_THREADS=1
scontrol show job ${SLURM_JOBID}
# load your software environment here:
srun --hint=nomultithread -n ${SLURM_NTASKS} -N ${SLURM_NNODES} \
--cpu-bind=verbose,cores ./a.out
```

MPI and OpenMP jobs

```
#!/bin/bash
#SBATCH --partition=workq
#SBATCH --ntasks=4
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=48
#SBATCH --hint=nomultithread
#SBATCH --account=k#####
#SBATCH --time=01:00:00
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
scontrol show job ${SLURM_JOBID}
# load your software environment here:
srun --hint=nomultithread -n ${SLURM_NTASKS} -N ${SLURM_NNODES} \
-c ${OMP_NUM_THREADS} --cpu-bind=verbose,cores ./a.out
```

CPU Affinity

- CPU affinity binds application's processes and threads to compute unit, e.g. hardware threads, cores, sockets, NUMA domains etc.
- Biniding can have non-trivial effect on the performance of an application, depnding on its runtime characteristics

Managing CPU affinity with SLURM

- Use --cpu-bind:
- Syntax

```
srun -n ## -c ## --cpu-bind=verbose,cores ./a.out
```

Managing CPU affinity with SLURM

- Demo

```
> salloc
srun -n 192 -c 1  bash -c 'echo -n "Proc $SLURM_PROCID of Node $SLURM_NODEID ";
taskset -cp $$' | sort -n | head -n 20

srun -n 192 -c 1 --hint=nomultithread bash -c 'echo -n "Proc $SLURM_PROCID of
Node $SLURM_NODEID "; taskset -cp $$' | sort -n | head -n 20

srun -n 192 -c 1 --cpu-bind=cores bash -c 'echo -n "Proc $SLURM_PROCID
of Node $SLURM_NODEID "; taskset -cp $$' | sort -n | head -n 20

srun -n 192 -c 1 --cpu-bind=rank bash -c 'echo -n "Proc $SLURM_PROCID
of Node $SLURM_NODEID "; taskset -cp $$' | sort -n | head -n 20
```

Managing CPU affinity with SLURM

- Demo

```
> salloc --nodes=2 --hint=nomultithread
srun -n 16 -N 2 -c 1 bash -c 'echo -n "Proc $SLURM_PROCID of Node $SLURM_NODEID ";
taskset -cp $$' | sort -k 2 -n


srun -n 16 -N 2 -c 1 -m block:block bash -c 'echo -n "Proc $SLURM_PROCID of Node
$SLURM_NODEID "; taskset -cp $$' | sort -k 2 -n


srun -n 16 -N 2 -c 1 --cpu-bind=cores -m block:block bash -c 'echo -n "Proc
$SLURM_PROCID of Node $SLURM_NODEID "; taskset -cp $$' | sort -k 2 -n
```

Pre-post processing jobs

- Large memory node

```
#!/bin/bash
#SBATCH --partition=ppn
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=256
#SBATCH --mem=2T
#SBATCH --hint=nomultithread
#SBATCH --account=k#####
#SBATCH --time=01:00:00
#SBATCH --nodelist=ppn9
srun lsmem
```

- Node with graphics GPU

```
#!/bin/bash
#SBATCH --partition=ppn
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=128
#SBATCH --gres=gpu:1
#SBATCH --hint=nomultithread
#SBATCH --account=k#####
#SBATCH --time=01:00:00


nvidia-smi
```

72 hour queue

- User needs to be added to 72 hour QoS before the job can run

```
#!/bin/bash
#SBATCH --partition=72hours
#SBATCH --qos=72hours
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=192
#SBATCH --hint=nomultithread
#SBATCH --account=k#####
#SBATCH --time=01:00:00

srun ........
```

Shared queue

- By default, 2 cpus and 1GB mem is allocated

```
#!/bin/bash
#SBATCH --partition=shared
#SBATCH --account=k#####
#SBATCH --time=01:00:00

scontrol show job $SLURM_JOBID
srun ....
```

- A maximum of 4 cpus and full node memory can requested

```
#!/bin/bash
#SBATCH --partition=shared
#SBATCH --account=k#####
#SBATCH --time=01:00:00
#SBATCH -c 4
#SBATCH --mem=370G
scontrol show job $SLURM_JOBID
srun ....
```

Data mover jobs

```
#!/bin/bash
#SBATCH --partition=dtn
#SBATCH --ntasks=8
#SBATCH --account=k#####

module swap PrgEnv-cray PrgEnv-gnu
module load mpifileutils
module list
time -p srun -n ${SLURM_NTASKS} dcp --verbose --progress 60 --preserve
<source_dir> <dest_dir>
```

Job arrays

```
> cat jobarray.slurm
#!/bin/bash
#SBATCH --partition=workq
#SBATCH --array=0-10
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=48
#SBATCH --account=k01
#SBATCH --time=00:10:00


echo "ARRAY_ID: ${SLURM_ARRAY_JOB_ID}  TASK_ID: ${SLURM_ARRAY_TASK_ID}"
sleep 20
```

Job arrays

```
> sbatch jobarray.slurm
Submitted batch job 235300
> squeue -me
JOBID        USER   ACCOUNT         NAME   ST REASON           START_TIME     TIME TIME_LEFT NODES
235300_0    ####      k01    jobarray.sh    R None    2024-04-14T01:53:40     0:02  9:58     1
235300_1    ####      k01    jobarray.sh    R Prolog  2024-04-14T01:53:40     0:02  9:58     1
235300_2    ####      k01    jobarray.sh    R None    2024-04-14T01:53:40     0:02  9:58     1
235300_3    ####      k01    jobarray.sh    R None    2024-04-14T01:53:40     0:02  9:58     1
..........
235300_8    ####      k01    jobarray.sh    R None    2024-04-14T01:53:40     0:02  9:58     1
235300_9    ####      k01    jobarray.sh    R None    2024-04-14T01:53:40     0:02  9:58     1
235300_10   ####      k01    jobarray.sh    R None    2024-04-14T01:53:40     0:02  9:58     1
```

Job arrays

```
> ls
jobarray.slurm    slurm-235300_10.out  slurm-235300_2.out  slurm-235300_4.out  slurm-
    235300_6.out  slurm-235300_8.out     slurm-235300_0.out  slurm-235300_1.out  slurm-
    235300_3.out  slurm-235300_5.out  slurm-235300_7.out  slurm-235300_9.out
> cat slurm-235300_0.out
ARRAY_ID: 235300  TASK_ID: 0
> cat slurm-235300_1.out
ARRAY_ID: 235300  TASK_ID: 1
> cat slurm-235300_10.out
ARRAY_ID: 235300  TASK_ID: 10
```

Job dependencies

- JobA.slurm

```
#!/bin/bash
#SBATCH --partition=dtn
#SBATCH --ntasks=8
#SBATCH --cpus-per-task=1
#SBATCH --account=k01
#SBATCH --time=00:10:00
echo "Hi, I will move some data from
   project to scratch"
sleep 60
echo "Job A is finished successfully"
```

- JobB.slurm

```
#!/bin/bash
#SBATCH --partition=workq
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=48
#SBATCH --account=k01
#SBATCH --time=00:10:00
echo "Hi, I launch a solver that
   required data from jobA moved in
   scratch"
sleep 60
echo "Job B is finished successfully"
```

Job dependencies

```
> sbatch jobA.slurm
Submitted batch job 235318
> squeue --me -l
Sun Apr 14 02:17:04 2024
JOBID PARTITION     NAME     USER    STATE     TIME TIME_LIMI  NODES NODELIST(REASON)
235318       dtn  jobA.sh     ####  RUNNING     0:04      10:00      1 dtn1
> sbatch --dependency=afterok:235318 jobB.slurm
Submitted batch job 235319
```

Job dependencies

```
> squeue --me -l
Sun Apr 14 02:18:05 2024
JOBID PARTITION     NAME     USER    STATE   TIME TIME_LIMI  NODES NODELIST(REASON)
235319  workq    jobB.slurm ####  PENDING   0:00     10:00      1     (Dependency)
235318    dtn    jobA.slurm #### COMPLETI   1:02     10:00      1      dtn1
> squeue --me -l
Sun Apr 14 02:18:10 2024
JOBID PARTITION     NAME     USER    STATE   TIME TIME_LIMI  NODES NODELIST(REASON)
235319  workq    jobB.slurm  ####  RUNNING   0:03     10:00      1      nid00023
```

Job packing

- srun &
- srun --multi-prog
- srun ./wrapper.sh --> Using $SLURM_PROCID

Things to consider

- Always submit jobs from /scratch
  - HOME is not-mounted on compute node
    - On a compute node : `HOME=/scratch/$USER`
  - Project is read-only on compute node (can’t write so no slurm-#####.out will be created