Job Scheduling on KSL HPC Systems

Saber Feki, Bilel Hadri,
Mohsin Ahmed Shaikh, Nagarajan Kathiresan
Computational Scientists @ KSL

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

SHAHEEN
SUPERCOMPUTING LABORATORY

KSL Systems – Shaheen Supercomputer

- Allocatable resources
  - 6174 nodes of Intel Haswel, each node in exclusive mode, each with 32 cores and 128 GB RAM.
  - 4 nodes with 256 GB RAM
  - 1.5 PB total capacity of Burst Buffer(BB) with 368 GB of granularity, with 268 BB nodes.

- Queue
  - Default queue: workq: 24hours limit
  - Debug queue: debug : 4 nodes max for 30min maximum
  - 72hours queue, 72hours, 512 nodes maximun shared with all users for 72hours maximum

KSL Systems -- Ibex Cluster

- **Heterogeneous CPU architecture**
  - ✓ Intel Cascadelake – 44 nodes (48 cores, 2 sockets per node)
  - ✓ Intel Skylake – 128 nodes (32-40 cores, 2 sockets per node)
  - ✓ Intel Haswell – 16 nodes (36 cores, 2 sockets per node)
  - ✓ Intel Ivy Bridge – 163 nodes (20 cores, 2 socket per node)
  - ✓ Intel Sandy Bridge – 96 nodes (16 cores, 2 sockets)
- **He**terogeneous GPU architecture
  - ✓ Volta (v100) - 38 nodes (4 or 8 GPUs cards per node)
  - ✓ Turing (rtx2080ti) - 4 nodes (8 GPUs cards per node)
  - ✓ Pascal (gtx1080ti, p100, p6000) - 20 nodes (2-8 GPUs per node)
- Large memory nodes
  - ✓ 3 TB nodes – 18 nodes (32- 48 cores per node, Intel Skylake and Cascadelake)
  - ✓ 2 TB nodes – 3 nodes (32 – 80 cores per nodes, Intel Skylake and Westmear)
  - ✓ 750GB – 1.5TB nodes – 13 nodes (64 cores per node, AMD Abu Dhabi)

What is a Job Scheduler?

Our goal? -- Maximizing utility of KSL systems

- HPC resources at KSL must be busy at maximum capacity including nights and weekends
  - Fair distribution of resource
- How we achieve it:
  - Queues and schedulers help increase utilization
- You may need to change your workflow:
  - You may need to wait for your turn because resources in use
  - Automate where ever possible for scheduler to run on your behalf
  - Have your work queued to maximize utilization

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 state/availability

On Shaheen:

```
> sinfo
PARTITION   AVAIL JOB_SIZE  TIMELIMIT   CPUS  S:C:T NODES STATE
workq*      up    1-6158   1-00:00:00     64 2:16:2  6016 allocated
workq*      up    1-6158   1-00:00:00     64 2:16:2   142 idle
72hours     up    1-512    3-00:00:00     64 2:16:2  6016 allocated
72hours     up    1-512    3-00:00:00     64 2:16:2   142 idle
debug       up    1-4            30:00    64 2:16:2    16 idle
```

sinfo

- A concise view of the system resources and their state/availability

On Ibex:

```
> sinfo -Nel
Mon Dec  9 13:11:27 2019
NODELIST      NODES   PARTITION       STATE CPUS    S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE
REASON
besest514-03      1      batch*       mixed   80   8:10:1 154000        0   1540 ibex2017 none
cn603-02-l        1      batch*       mixed   40   2:20:1 375618        0    100 dragon,c none
cn603-02-r        1      batch*       mixed   40   2:20:1 375618        0    100 dragon,c none
cn603-03-l        1      batch*   allocated   40   2:20:1 375618        0    100 dragon,c none
cn603-03-r        1       c2034   allocated   40   2:20:1 375618        0    100 dragon,c none
…..
sdlm111-20        1      batch*        idle   64   4:16:1 101423        0   1014 ibex2017 none
sdlm111-22        1      batch*        idle   64   4:16:1 101423        0   1014 ibex2017 none
sdlm112-18        1      batch*     drained   64   4:16:1 510000        0    511 ibex2017 ZHARDWARE:[2019-
11-2
```

ginfo

An in-house tool developed to query the status of GPU resources on Ibex cluster

```
> ginfo -d
Current idle GPU Nodes:
-----------------------------------------
Node Name  | Total GPU   | Mem GB     | CPU
---------   ----------   ----------   ---
dgpu501-02 | 4 gtx1080ti | 246 GB     | 36
...
dgpu501-26 | 4 p100      | 246 GB     | 36
dgpu501-30 | 4 p100      | 246 GB     | 36
...
dgpu609-14 | 2 p6000     | 246 GB     | 36
gpu104-12  | 1 v100      | 118 GB     | 16
gpu208-10  | 8 v100      | 745 GB     | 48
...
gpu214-14  | 8 v100      | 745 GB     | 48
gpu504-37  | 8 gtx1080ti | 366 GB     | 32
gpu510-02  | 8 rtx2080ti | 366 GB     | 32
...
-----------------------------------------
```

squeue

- Shows the list of jobs in the queue along with information about the request and its current state

```
> squeue
   12125632    user123   k1363  imperial_wing   R None      2019-12-08T14:29:38   22:45:08    1:14:52   256
   12125727    user103   k1373           2S_BS   R None      2019-12-08T14:48:48   22:25:58    1:34:02     4
   12125776    user223   k1182 MB_bromobenzen   R None      2019-12-08T14:58:06   22:16:40    1:43:20     1
   12127627    user423   k1056 inversionORTHO   R None      2019-12-09T00:34:43   12:40:03    2:19:57    60
   12128619    user101   k1206 2019120906_pos   R None      2019-12-09T09:40:07    3:34:39    2:25:21     1
 12127113_1    user121   k1068        ago-107   R None      2019-12-08T17:05:27   20:09:19    3:50:41     8
 12127114_1    user121   k1068        ago-107   R None      2019-12-08T17:05:27   20:09:19    3:50:41     8
 12127115_1    user121   k1068        ago-107   R None      2019-12-08T17:05:27   20:09:19    3:50:41     8
 12127116_1    user121   k1068        ago-107   R None      2019-12-08T17:05:27   20:09:19    3:50:41     8
 12127580      user125   k1056             tu   R None      2019-12-09T00:21:56   12:52:50   11:07:10   228
```

- You can use filters on the list:
  - -u $USERNAME – show jobs by a user
  - -p $USERNAME – show jobs on a specific partition
  - -j $JOBID -- show status of a pariticular job

Specifying resources

Resource pools

- Partitions
  - Queues with names adhering to a scheduling policy
    - e.g. workq, debug , smc_… , gpu_wide, group-csim ….
- Features/Constraints
  - Some partitions have types of one resources, e.g.
  - CPU architectures
    - various kinds of GPUs
    - collections of node with various amount of memory

```
> sinfo -o %P,%f
PARTITION,AVAIL_FEATURES
batch*,ibex2017,nolmem,cpu_intel_e5_2699_v3,ssh,gpu,mpi_intel,intel_gpu,local_200G,gpu_gtx1080ti,gtx1080ti
batch*,ibex2017,nogpu,cpu_intel_e7_2830,dragonlmn,intel,largemem,local_1T,local_200G,local_2T,local_400G,local_500G,zram
…...
debug,ibex2017,nolmem,cpu_intel_e5_2699_v3,ssh,gpu,mpi_intel,intel_gpu,local_200G,local_400G,local_500G,p6000
debug,dragon,nolmem,cpu_intel_e5_2670,intel,ssh,gpu,intel_gpu,local_200G,local_400G,gpu_v100,v100
…..
gpu_wide,dragon,ibex2018,nolmem,cpu_intel_platinum_8260,intel,gpu,intel_gpu,local_200G,local_400G,local_500G,gpu_v100,v100,no
ssh
```

Requestable resources

- CPUs
- Memory
- GPUs
- Burst Buffer
- Wall time

Requesting resource

You can either run your jobs in batch mode or interactive mode:

Implications:

- Batch mode
  - You will need a script with resource request and the commands to run on that resource once allocated
  - Scheduler will run the script on your behalf once the requested resources are available
  - Resources can be requests for longer durations (several hours)
- Interactive
  - Resource requests are usually small and short
  - You run each command by typing it interactively
  - Useful for prototyping and debugging

JobScript (Shaheen)

```
--------- jobscirpt.slurm ------
#!/bin/bash -l
#SBATCH --account=k01
#SBATCH --job-name=myfirstjob
#SBATCH --time=04:00:00
#SBATCH --partition=batch
#SBATCH --ntasks=40
#SBATCH --hint=nomultithread
#DW jobdw type=scratch access_mode=striped capacity=2TiB

#load your modules here
srun –n 40 my_application [blah]
-----------------------------

> sbatch jobscript.slurm
```

JobScript (Ibex)

```
--------- jobscirpt.slurm ------
#!/bin/bash -l

#SBATCH --job-name=myfirstjob
#SBATCH --time=04:00:00
#SBATCH --partition=batch
#SBATCH --ntasks=1
#SBATCH --gres=gpu:1
#SBATCH --constraint=v100
module load cuda
./my_application [blah]
-------------------------------

> sbatch jobscript.slurm
```

sbatch

Command to submit your **jobscript** to SLURM:

- Upon successful submission a unique job ID is assigned
- Job is queued and awaits allocation of the requested resources
- On Shaheen where the usage is accounted for, a priority is assigned to each job based on first come basis
- In general, shorter and smaller jobs are easier to scheduler

salloc (Shaheen)

Command to request allocation of resource for interactive use:

- Primary be used for prototyping and/or debugging your workflow

```
shaima0d@cdl2:> salloc --partition=debug –N 2
salloc: Granted job allocation 12130189
shaima0d@gateway2:> srun -n 64 ./helloworld

Hello from rank  0 of 64
Hello from rank  1 of 64
….
Hello from rank 63 of 64
```

salloc (Ibex)

Command to request allocation of resource for interactive use:

- Primary be used for prototyping and/or debugging your workflow

```
> salloc -p debug --gres=gpu:1 --constraint=v100 -t 00:10:00
salloc: Pending job allocation 7329981
salloc: job 7329981 queued and waiting for resources
salloc: job 7329981 has been allocated resources
salloc: Granted job allocation 7329981
salloc: Waiting for resource configuration
salloc: Nodes gpu104-12 are ready for job
> module load cuda
> srun -n 1 deviceQuery/deviceQuery
deviceQuery/deviceQuery Starting...
 CUDA Device Query (Runtime API) version (CUDART static linking)
Detected 1 CUDA Capable device(s)
Device 0: "Tesla V100-PCIE-32GB"
  CUDA Driver Version / Runtime Version          10.1 / 9.2
  CUDA Capability Major/Minor version number:    7.0
  Total amount of global memory:                 32480 MBytes (34058272768 bytes)
  (80) Multiprocessors, ( 64) CUDA Cores/MP:     5120 CUDA Cores
….
  deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 10.1, CUDA Runtime Version = 9.2, NumDevs = 1
Result = PASS
```

Using allocated resources

srun

Once allocated, `srun` command can be used to launch your application on to the compute resources

```
>user123@cdl2:~> salloc --partition=debug -N 2
salloc: Granted job allocation 12140840
> srun  --ntasks=4 --cpus-per-task=10 ./my_application
```

- Each srun command is considered as "job step" for the corresponding allocation by SLURM
- When a job step completes, the allocation does not automatically terminate
- This means you can run multiple job steps with different configurations.

NOTE: you can only run one job step at a time. (backgrounding `srun` will run sequentially)

srun (in jobscript)

```
--------- jobscirpt.slurm ------
#!/bin/bash --login
#SBATCH --account=k01
#SBATCH --job-name=myfirstjob
#SBATCH --time=04:00:00
#SBATCH --partition=batch
#SBATCH --ntasks=40

srun -n 1 ./serial_step          # quick but sequential preprocessing step
srun -n 8 -c 5 ./parallel_step   # computationally expensive/parallel step

-----------------------------
```

Monitoring and account your jobs

squeue

You can query the state of your job using squeue

- Common filters include
  - by user
  - by job ID

```
> squeue -u mylogin
             JOBID       USER ACCOUNT           NAME  ST REASON    START_TIME                 TIME  TIME_LEFT NODES
          12127214    mylogin    k1320 IMe-H-RuCl3-fr   R None      2019-12-08T20:05:20   21:13:44    2:46:16     1
          12128314    mylogin    k1320 IMe-H-RuCl3-fr   R None      2019-12-09T08:32:51    8:46:13   15:13:47     1
          12129938    mylogin    k1320 IMe-H-RuCl3-fr  PD AssocGrpC N/A                       0:00 1-00:00:00     1
          12129939    mylogin    k1320 IMe-H-RuCl3-fr  PD AssocGrpC N/A                       0:00 1-00:00:00     1

> squeue -j 12127214
             JOBID       USER ACCOUNT           NAME  ST REASON    START_TIME                 TIME  TIME_LEFT NODES
          12127214    mylogin    k1320 IMe-H-RuCl3-fr   R None      2019-12-08T20:05:20   21:14:44    2:45:16     1
```

scontrol

- scontrol command, amongst other things, allows user to show parameters of request and allocated resource for a job in queue (in any state i.e running, pending, etc)

```
> scontrol show job 12130305
   JobId=12130305 JobName=profiling_jags
   UserId=shaima0d(174988) GroupId=g-shaima0d(1174988) MCS_label=N/A
   Priority=1 Nice=0 Account=k01 QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:55:25 TimeLimit=10:00:00 TimeMin=N/A
   SubmitTime=2019-12-09T15:04:12 EligibleTime=2019-12-09T15:04:12
   AccrueTime=2019-12-09T15:04:12
   StartTime=2019-12-09T15:04:12 EndTime=2019-12-10T01:04:12 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2019-12-09T15:04:12
   Partition=workq AllocNode:Sid=cdl2:38945
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=nid00107
   BatchHost=nid00107
   NumNodes=1 NumCPUs=64 NumTasks=1 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
   TRES=cpu=64,mem=128448M,node=1,billing=64
```

sacct

Displays accounting command which tell about the resources used by the job and its job steps.

```
> sacct -j 12130040
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
12130040     profiling+      workq        k01         64  COMPLETED      0:0
12130040.ba+      batch                   k01         64  COMPLETED      0:0
12130040.0      Rscript                   k01         12  COMPLETED      0:0
```

The accounting information persists after the life of the job

Common filters include `-u` for "by user" and `-j` for "by jobID"

Example Jobscripts

Example – OpenMP jobs on Shaheen

- A jobscript running an OpenMP code on a single Shaheen node with 4 OpenMP threads

```
#!/bin/bash
#SBATCH --account=k01
#SBATCH --time=00:10:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --hint=nomultithread

export OMP_NUM_THREADS=4
srun -n 1 -c 4 ./my_omp_application
```

Example – MPI jobs on Shaheen

- A jobscript running MPI code on a Shaheen with 32 MPI tasks

```
#!/bin/bash -l
#SBATCH --account=k01
#SBATCH --time=00:10:00
#SBATCH --ntasks=32
#SBATCH --hint=nomultithread

# load your modules here …
srun -n 32 ./my_mpi_application
```

Example – MPI+OpenMP jobs on Shaheen

- A jobscript running a hybrid code on Shaheen parallelized with both OpenMP and MPI (requires 32 x 4 = 128 core in total)

```
#!/bin/bash
#SBATCH --account=k01
#SBATCH --time=00:10:00
#SBATCH --ntasks=32
#SBATCH --cpus-per-task=4
#SBATCH --hint=nomultithread

export OMP_NUM_THREADS=4
srun -n 32 -c 4 ./my_hybrid_application
```

Example – serial jobs on Ibex

- A jobscript running an OpenMP code on a Ibex with 4 OpenMP threads
  - NOTE: nodes are shared on Ibex (must define the request properly)

```
#!/bin/bash
#SBATCH --time=00:10:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4

module load gcc/6.4.0

export OMP_NUM_THREADS=4
export OMP_PLACES=cores
Export OMP_PROC_BIND=close
./my_omp_application
```

Example – MPI jobs on Ibex

- A jobscript running MPI code on a Ibex with 32 MPI tasks on same node
  - NOTE: nodes are shared on Ibex (must define the request properly)

```
#!/bin/bash
#SBATCH --time=00:10:00
#SBATCH --ntasks=32
#SBATCH --ntasks-per-node=32

module load gcc/6.4.0
module load openmpi
mpirun -n 32 ./my_omp_application
```

Example – large memory jobs on Ibex

- Normal compute nodes have memory up to ~ **360GB** per node.
- “large memory job” is a label that’s assigned to your job by SLURM if you ask for memory => **370G**

  - ✓ Use `--mem=####G` to request nodes with large memory.
  - ✓ When you don't specify `--mem`, the **default memory** allocation will be **2GB**
  - ✓ Upon submission via `sbatch` or `salloc`, SLURM will notify the following message:
  - ✓ `sbatch`: job tagged as large memory

Example – 1 GPU jobs on Ibex

- Running a CUDA code on a single GPU
  - NOTE: nodes are shared on Ibex (must define the request properly)
  - All GPU jobs should be submitted from `glogin.ibex.kaust.edu.sa` login node

```
#!/bin/bash
#SBATCH --time=00:10:00
#SBATCH --ntasks=1
#SBATCH --mem=64G
#SBATCH --gres=gpu:1
#SBATCH --constraint=v100

module load gcc/6.0.4
module load cuda/10.1.105

./my_omp_application
```

Example – 4 GPUs jobs on Ibex

- Running a CUDA code on 4 GPUs on single Ibex node
  - NOTE: nodes are shared on Ibex (must define the request properly)
  - All GPU jobs should be submitted from `glogin.ibex.kaust.edu.sa` login node

```
#!/bin/bash
#SBATCH --time=00:10:00
#SBATCH --ntasks=4
#SBATCH --mem=64G
#SBATCH --gres=gpu:4
#SBATCH --constraint=v100

module load gcc/6.0.4
module load cuda/10.1.105

mpirun -n 4 ./my_omp_application
```

Example – Running containers on Ibex

Help from job script generator: https://www.hpc.kaust.edu.sa/ibex/job

Any Intel Architecture:

```
#SBATCH --constraint=[intel]
```

Intel Skylake:

```
#SBATCH --constraint=[cpu_intel_gold_6148]
```

Intel IveyBridge:

```
#SBATCH --constraint=[cpu_intel_e5_2680_v2|cpu_intel_e5_2670_v2]
```

Any GPU Architecture:

```
#SBATCH --gres=gpu:1
#SBATCH --constraint=[gpu]
```

Volta V100, 8 GPUs per node

```
#SBATCH --gres=gpu:8
#SBATCH --constraint=[v100]
```

Large Memory

```
#SBATCH --mem=2T
```

Local storage

```
#SBATCH --constraint=local_500G
```

Shaheen specifics

Accounting on Shaheen

- Shaheen access is granted through project proposal allocations lead by faculty with core hours.
- Cost is free of charge for KAUST and in Kingdom Universities
- One hour of run on one node is charged 32. 1 hour x 32 cores.
- 3 commands to check allocation:
  - groupies, sb kxxx, sb_users kxxxx

Accounting on Shaheen

3 commands to check allocation:

- groupies $USER: which accounts I belong

```
~> groupies alimen
Name: Ali Men
Group Name                                                PI
----- -------------------------------------------------- ------------------------------
k2019  Simulations of reacting flows                      Ali Men
```

- sb kxxxx: show the balance of the project

```
Project k2019: Simulations of reacting flows
PI: Ali Men
Allocations     Core hours
--------------------------
2019-01-06         6000000
--------------------------
Expired on      2020-01-06
--------------------------
Allocated          6000000
Shaheen            5966859
Neser                    0
--------------------------
Balance              33141
--------------------------
```

- sb_users kxxxx: show the utilization by all the users

Status of my runs

```
sacct -u user123 -S2019-12-08-09:11 -E2020-12-01-00:00  -X -ojobid,submit,start,end,jobname,elapsed,state,exit,nnodes,nodelist,partition,account
```

JobID	Submit	Start	End	JobName	Elapsed	State	ExitCode	NNodes	NodeList	Partition	Account
12125680	2019-12-08T14:39:03	2019-12-08T14:39:04	2019-12-08T14:39:17	Tandem_g4+	00:00:13	COMPLETED	0:0	1	nid00575	workq	k1363
12125702	2019-12-08T14:43:52	2019-12-08T14:43:53	2019-12-08T14:44:10	Tandem_g4+	00:00:17	COMPLETED	0:0	1	nid00166	workq	k1363
12125704	2019-12-08T14:46:49	2019-12-08T14:46:50	2019-12-08T14:49:02	Tandem_g4+	00:02:12	CANCELLED+	0:0	1	nid00073	workq	k1363
12125764	2019-12-08T14:55:46	2019-12-08T14:55:46	2019-12-08T14:56:59	Tandem_g4+	00:01:13	CANCELLED+	0:0	2	nid00[585-586]	workq	k1363
12125784	2019-12-08T14:59:12	2019-12-08T14:59:12	2019-12-08T17:11:22	Tandem_g4+	02:12:10	COMPLETED	0:0	1	nid00247	workq	k1363
12125791	2019-12-08T15:01:55	2019-12-08T15:01:56	2019-12-08T15:02:29	Tandem_sp+	00:00:33	CANCELLED+	0:0	64	nid00[716-767,+	workq	k1363
12125793	2019-12-08T15:04:46	2019-12-08T15:04:46	2019-12-08T15:07:55	Tandem_p8+	00:03:09	COMPLETED	0:0	64	nid00[767,776-+	workq	k1363
12125795	2019-12-08T15:07:30	2019-12-08T15:07:31	2019-12-09T03:07:56	Tandem_g4+	12:00:25	TIMEOUT	0:0	1	nid00073	workq	k1363
12129779	2019-12-09T11:44:38	2019-12-09T11:45:29	2019-12-10T02:45:21	Tandem_p8+	14:59:52	COMPLETED	0:0	32	nid0[0989-1020]	workq	k1363
12131202	2019-12-09T16:30:21	2019-12-09T16:30:22	Unknown	imperial_+	17:40:25	RUNNING	0:0	256	nid0[3424-3455+	workq	k1363
12131216	2019-12-09T16:37:04	2019-12-09T16:37:04	2019-12-09T16:37:19	delta_win+	00:00:15	FAILED	6:0	128	nid0[2094-2111+	workq	k1363
12132767	2019-12-09T18:39:13	2019-12-09T18:39:23	2019-12-09T18:41:44	delta_win+	00:02:21	COMPLETED	0:0	128	nid0[3961-3967+	workq	k1363
12132809	2019-12-09T18:42:26	2019-12-09T18:42:31	2019-12-09T18:43:26	delta_win+	00:00:55	COMPLETED	0:0	128	nid0[3961-3967+	workq	k1363
12132846	2019-12-09T18:47:48	2019-12-09T18:47:51	2019-12-09T18:49:38	delta_win+	00:01:47	COMPLETED	0:0	128	nid0[3961-3967+	workq	k1363
12132928	2019-12-09T19:25:52	2019-12-09T19:25:54	2019-12-09T20:26:04	delta_win+	01:00:10	TIMEOUT	0:0	128	nid0[4189-4223+	workq	k1363
12133644	2019-12-09T21:14:31	2019-12-09T21:14:32	Unknown	delta_win+	12:56:15	RUNNING	0:0	128	nid0[4838-4863+	workq	k1363

Example – using Burst Buffer on Shaheen

- Burst Buffer adds a layer between the compute nodes and the parallel file system.
- Shaheen has 268 DataWarp nodes, 368 GB of granularity.
- Test case: checkpointing, I/O improvements, Shared BB allocation for multiple jobs...
- Check resources available

```
dwstat
    pool units quantity    free   gran
wlm_pool bytes   1.5PiB 1.21PiB 368GiB
```

- Check status of usage and allowed access. If not listed, Send a justification to help@hpc.kaust.edu.sa

```
scontrol show burst
Name=datawarp DefaultPool=wlm_pool Granularity=368GiB TotalSpace=1541TiB FreeSpace=1272544GiB
UsedSpace=204608GiB
  Flags=EnablePersistent
  StageInTimeout=1800 StageOutTimeout=1800 ValidateTimeout=5 OtherTimeout=300
  AllowUsers=user1,user123,user324,user126,user843
```

Example – using Burst Buffer on Shaheen

- A user requests 60TB of DW nodes, how many DW nodes is he going to reserve ?
- We have 268 DW nodes, each nodes provides initially one DW instance
- When all of them are used, then it starts from the first DW node again.
- The allocation occurs under round -robin basis
- Requested_DW_nodes= 60*1024/368 = 166.95, so we will reserve 167DW nodes.
- In case we reserve more 96TB (268*368/1024), then some DW nodes will be used twice .

Example – using Burst Buffer on Shaheen

Lustre run

```
#!/bin/bash
#SBATCH --partition=workq
#SBATCH -t 10:00:00
#SBATCH --nodes=32
#SBATCH --ntasks=1024
#SBATCH -J slurm_test

srun –n 1024 ./my_exe
```

BB run (2TB of DW space)

```
#!/bin/bash
#SBATCH --partition=workq
#SBATCH -t 10:00:00
#SBATCH --nodes=32
#SBATCH --ntasks=1024
#SBATCH -J slurm_test
#DW jobdw type=scratch access_mode=striped capacity=2TiB
#DW stage_in type=directory source=/scratch/user1/bb_dest=$DW_JOB_STRIPED
#DW stage_out type=directory destination=/scratch/user1/back_lustre=$DW_JOB_STRIPED/

cd $DW_JOB_STRIPED

chmod +x my_exe

srun –n 1024 ./my_exe
```

Best Practices for using SLURM

Why my job is not running?

When the estimated start time of your pending job is not available, you can get more details and reasons for your job not running:

By typing *squeue --job <jobid >–l* , you will get the following output along with the reason for your job not running.

```
squeue --job 12132574 -l

        JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
     12132574 workq 8-tuned_   user1  PENDING       0:00 3-00:00:00      1 (AssocGrpCPUMinutesLimit)
```

Why my job is not running?

Here are the most common reasons. These codes identify the reason that a job is waiting for execution. A job may be waiting for more than one reason, in which case only one of those reasons is displayed.

AssocGrpCPUMinutesLimit
Core hours left in project are not enough to complete the run

Cleaning
The job is being requeued and still cleaning up from its previous execution.

Dependency
This job is waiting for a dependent job to complete.

JobHeldAdmin
The job is held by a system administrator

JobHeldUser
The job is held by the user

NodeDown
A node required by the job is down.

Priority
One or more higher priority jobs exist for this partition or advanced reservation. Other jobs in the queue have higher priority than yours.

QOSGrpNodeLimit
The maximum number of nodes available to the partition are in use.

QOSUsageThreshold
Required QOS threshold has been breached

ReqNodeNotAvail
No nodes can be found satisfying your limits, for instance because maintenance is scheduled and the job can not finish before it

Reservation
The job is waiting for its advanced reservation to become available.

Resources
The job is waiting for resources (nodes) to become available and will run when Slurm finds enough free nodes.

SystemFailure
Failure of the SLURM system, a file system, the network, etc.

Best Practices

- Check the balance of your account(sb), the status of the queue(sinfo). (Check announcement of maintenance...)
- Check your job status before leaving your session
- Don’t' perform watch squeue... Use email notifications instead
- If your application has the capability to checkpoint and restart, consider submitting your job for shorter time periods.
- On a system like Shaheen, there are many opportunities for backfilling jobs.
- If there is a large job at the top of the queue, the system will need to drain resources in order to schedule that job. During that time, short jobs can run. Jobs that request short wall clock times are good candidates for backfill.

Best Practices

- Checks examples in /sw/ for 3rd party software
  - /sw/xc40cle6/espresso/6.4.1/cle6.05_intel18.0.1.163/example/01/z_jobs_shaheen
- Use jobs generators
  - https://www.hpc.kaust.edu.sa/job on Shaheen
  - https://www.hpc.kaust.edu.sa/ibex/job on Ibex

- Make sure the wall clock time you request is accurate.
  - Many users unnecessarily enter the largest wall clock time possible as a default and therefore, this will increase your waiting time, especially for runs that could be executed in a few minutes.

- Run jobs before scheduled system maintenance.
  - The queues must be drained of all jobs before a maintenance session so at this time there is an opportunity for good turn-around for shorter jobs.
  - Make sure that the wall clock time is smaller than the remaining time before the maintenance session starts.

Summary

Resource	Flag Syntax	Description	Notes
partition	--partition=	Partition is a queue for jobs. Workq/72hours/debug (Shaheen)	default workq
qos	--qos=72hours	QOS is quality of service value	required
time	--time=01:00:00	Time limit for the job.	1 hour; default is 24 hours
nodes	--nodes=2	Number of compute nodes for the job.	default is 1; compute nodes
cpus/nodes	--ntasks-per-node=8	Corresponds to number of cores on the compute node.	default is 1
cpus/socket	--ntasks-per-socket=4	Corresponds to number of socket on the compute node.	default is 1
node type	--constraint=intel	Node type feature.	default is no node type specified
resource feature	--gres=gpu:2	Request use of GPUs on compute nodes	default is no feature specified;
memory	--mem=24000	Memory limit per compute node for the job. Do not use with mem-per-cpu flag.	memory in MB;
memory	--mem-per-cpu=400	Per core memory limit. Do not use the mem flag,	memory in MB;
account	--account=kxxxx	Users may belong to groups or accounts.	default is the user's primary group.
job name	--job-name="hello_test"	Name of job.	default is the JobID
output file	--output=test.out	Name of file for stdout.	default is the JobID
email address	--mail-user=username@kaust.edu.sa	User's email address	required
email notification	--mail-type=ALL --mail-type=START/END	When email is sent to user.	omit for no email
BB usage (only Shaheen)	#DW	pragma for capacity , stage in/out	required

Contact us !

For any issue, contact the team:

- Please share the job id, the system, the error message....
- For Shaheen help@hpc.kaust.edu.sa.
- For Ibex ibex@hpc.kaust.edu.sa

Check our training website : hpc.kaust.edu.sa/training

Thanks !

Q&A