Job Scheduling on IBEX

Mohsin Ahmed Shaikh

Computational Scientists @ KSL

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

SHAHEEN
SUPERCOMPUTING LABORATORY

KSL Systems -- Ibex Cluster

- **Heterogeneous CPU architecture**
  - ✓ Intel Cascade Lake – 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 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
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           9204779     batch NO_MoSe2 nadhrega  R   14:46:40      1 dbn404-16-r
           9118758  gpu_wide     bash     zhup  R 2-22:03:07      1 gpu214-14
           9118404     batch     bash  zhanb0b  R 2-22:26:04      1 gpu208-02
           9195595     batch dtchFlow   parawr  R   19:13:39      1 gpu510-12
           9194968     batch      aug   parawr  R   19:29:43      1 gpu510-07
         9234033_7     batch   s3knn2    qiang  R    1:42:16      1 gpu211-02
         9234033_8     batch   s3knn2    qiang  R    1:42:16      1 gpu211-02
         9234033_9     batch   s3knn2    qiang  R    1:42:16      1 gpu211-02
        9234033_10     batch   s3knn2    qiang  R    1:42:16      1 gpu609-02
           9087308     batch Haplotyp alsaedsb PD       0:00      1 (DependencyNeverSatisfied)
           9087317     batch Haplotyp alsaedsb PD       0:00      1 (DependencyNeverSatisfied)
```

- You can use filters on the list:
  - -u $USERNAME – show jobs by a user
  - -p $PARTITION – 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. batch, 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
- 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 requested 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 (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 Ibex jobs accounting is not enabled. Although a priority is assigned to each job, it is not used. This may change in future.

- In general, short and small jobs are to schedule

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)

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 alsaedsb
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           9087308     batch Haplotyp alsaedsb PD       0:00      1 (DependencyNeverSatisfied)
           9087317     batch Haplotyp alsaedsb PD       0:00      1 (DependencyNeverSatisfied)
           9197195     batch Joint-GV alsaedsb  R   18:38:25      1 dbn302-31-r
```

```
> squeue -j 9197195
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           9197195     batch Joint-GV alsaedsb  R   18:39:17      1 dbn302-31-r
```

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 9197195
   JobId=9197195 JobName=Joint-GVCFs
   UserId=alsaedsb(157323) GroupId=g-alsaedsb(1157323) MCS_label=N/A
   Priority=81 Nice=0 Account=default QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=18:40:26 TimeLimit=1-00:00:00 TimeMin=N/A
   SubmitTime=2020-02-08T17:07:42 EligibleTime=2020-02-08T17:07:42
   AccrueTime=2020-02-08T17:07:42
   StartTime=2020-02-08T17:07:43 EndTime=2020-02-09T17:07:43 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2020-02-08T17:07:43
   Partition=batch AllocNode:Sid=dbn503-33-r.ibex.kaust.edu.sa:6787
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=dbn302-31-r
   BatchHost=dbn302-31-r
   NumNodes=1 NumCPUs=16 NumTasks=1 CPUs/Task=16 ReqB:S:C:T=0:0:*:*
   TRES=cpu=16,mem=115G,node=1,billing=16
```

sacct

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

```
> sacct -u shaima0d
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
9230749      jupyter_s+      debug    default         36     FAILED      1:0
9230752      jupyter_s+      debug    default         16    TIMEOUT      0:0
9230752.bat+      batch                default         16  CANCELLED     0:15
9230752.ext+     extern                default         16  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 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 `vlogin.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
```

Common Constraints on Ibex

For CPU only jobs, use jobscript generator: https://www.hpc.kaust.edu.sa/ibex/job

For advice on templates for GPU jobs (DL training) please attend the afternoon session on Deep Learning Best Practices

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
```

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 -j 9235760 -l
Sun Feb  9 11:57:39 2020
             JOBID PARTITION     NAME     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
           9235760     batch     vasp  agapevk  PENDING       0:00 3-00:00:00      4 (Resources)
```

To understand the reason more:

```
> man squeue
```

And search for **JOB REASON CODES**

Resources The job is waiting for resources to become available.

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 status of the queue(sinfo). (Check announcement of maintenance...)
- Check your job status before leaving your session
- Don’t run `watch squeue` ... Use email notifications instead
- For GPU jobs, allocate at least 2 times the memory for each GPU, e.g. for NVIDIA `V100 GPUs`, allocate `--mem=64GB` (refer to the afternoon session on DL best practices for more advice)
- If your application has the capability to checkpoint and restart, consider submitting your job for shorter time periods.
- For **multi-gpu** jobs for ML training, try using gpu-wide queue.
- It is always a good idea to prototype your workflow for guesstimating your resource requirement appropriately

Best Practices

- 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.

Summary

Resource	Flag Syntax	Description	Notes
partition	--partition=	Partition is a queue for jobs. e.g. batch, gpu_wide, debug	default batch queue
time	--time=01:00:00	Time limit for the job.	1 hour
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
cpus/task	--cpus-per-task=4	Corresponds to number of CPUs per mpi task. Useful for multithreaded applications.	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;
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

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