IMSc Kamet HPC – Users Documentation

Table of Contents

IMSc Kamet HPC – Users Documentation

1 About the Kamet HPC

Kamet cluster consists of 58 nodes of Dell EMC PowerEdge R650 servers.

- Each node consists of 2 x Intel Xeon Platinum 8358 @ 2.6GHz, 32 Cores. Therefore, each node has 64 Cores and overall Kamet has 3,712 Cores.
- Each node has 256GB RAM with a configuration of 4GB per core with maximum memory throughput.
- Nodes are connected together using Mellanox HDR100 (100G) InfiniBand Interconnect.
- Two master nodes are configured in High-Availability mode to support user logins and other cluster related operations.
- This cluster has Lustre PFS storage of 150TB for faster access and this should be used as storage space.
- It also has 150TB of ZFS based storage for /home area to store long- term data.
- IP Address of Kamet is 10.96.72.200

2 Login to Kamet HPC

- Get necessary username and credentials from System Admin. Usually the username will be your mail id.
- Use one of the Terminal App to login to kamet (10.96.72.200) via ssh command.
    - `ssh <username>@kamet` or `ssh <username>@10.96.72.200`

3 Understanding Kamet HPC Storage

- Kamet HPC has directories /home/<username> and /lustre/<username> to store data.
- /lustre/<username> is the scratch space and it's based on high throughput Lustre PFS.
- /home/<username> is the user home area to store long term data.
- Users are allowed to store 1TB on /lustre/<username> for 6 months. Files older than 6 months will be deleted automatically to keep the higher throughput.
- Users are allowed to store 1.5TB on /home/<username> and users are advised to take regular backup.
- Users are advised to submit their jobs from /lustre/<username> and finalized results and data can be moved into /home/<username> area.

4 Understanding kamet HPC resource management system

SLURM(v 22.05) job scheduler is installed and configured to be job scheduler application with following partition and restriction table

Sl No	Partition Name	Time Limit	Nodes	Extended Nodes	Minimum Allowed cores	Maximum Allowed Cores
1	kshort*	12 Hrs	10	8	16	192
2	kmedium	2 days	38	5	16	768
3	klong	7 days	10	0	16	192

Users must submit the jobs via SLURM job scheduler only.

To submit jobs to SLURM jobs scheduler, user needs to prepare job script file

4.1 Preparing job script

SLURM requires a job-script to be created to submit a job. A sample job scheduler for medium a <queue> (kshort, medium, klong) is provided below

```
#!/bin/bash
#SBATCH -J <JobName>
#SBATCH -o Out_%j.txt
#SBATCH -e Err_%j.txt
#SBATCH -p <queue>
#SBATCH --mail-user=<user>@imsc.res.in
## Memory requirement per cpu (in MB), leave blank if default memory per cpu is sufficient for your run. Total memory per node can be specified by --mem=. Use responsibly, excess memory usage will be monitored and penalized.
#SBATCH --mem-per-cpu=<Memory>
## Specify the number of processors needed, not node
#SBATCH -n <N>
## For hybrid jobs only, specify the number of openmp processes needed per mpi processes
#SBATCH -c <p>
## Specify the time in DD-HH:MM:SS format, example, for a 2 day run, specify <time> as 2-00:00:00
#SBATCH -t <time>
#SBATCH --export=all
#SBATCH --mail-user=<username>@imsc.res.in
```

```
#SBATCH --mail-type=ALL
#SBATCH -D <directory>
echo $SLURM_JOB_NODELIST > hostfile_$SLURM_JOBID
export OMP_NUM_THREADS=<P>
module purge
module try-load <module1>
module try-load <module2>

mpirun -np <n> <directory>/<executable>
## <n> is the number of mpi processes, N = <n> x <p>
```

1. Giving instructions to SLURM to allocate how many processors, how long the program to be run and where to store its STDOUT and STDERR messages.
2. Setting Running path and storing allocated list of nodes to a file for feature reference.
3. Necessary instructions for running OpenMP based jobs
4. Necessary instruction for running MPI based jobs

Note:

1. We have configured SLURM to provide the necessary -np , --nodes, and --ppn values to mpirun. However, users are requested to set only the number of cores following the script above and not nodes.
2. Refer this link for additional Reference:

   https://slurm.schedmd.com/sbatch.html

4.2 Submitting Jobs (sbatch)

Use "sbatch" command to submit jobs to SLURM

Syntax: sbatch <jobscript.sh> [additional squeue options]

Example:

```
$: sbatch jobscript.sh
submitted batch 2021
```

Upon successful submission of a job, a unique job ID (i.e. 2021 in this example) is assigned by SLURM, which may be referred to for job management such as job status checking and cancellation.

4.3 Job Status Checking (squeue)

Use "squeue" command to see the status of the submitted job

Syntax: squeue [additional sbatch options]

Example:

```
$: squeue
```

```
JOBID PARTITION     NAME     USER ST       TIME NODES NODELIST(REASON)
   33    kshort hpl-test   hpctest R       0:08      1 kcpu1-2
   34    kshort hpl-test   hpctest R       0:08      1 kcpu13-4
   35    kshort hpl-test   hpctest R       0:08      1 kcpu5-6
   32    klong hpl-test    hpctest PD      0:00      1 (Resources)
   31    kshort hpl-test   hpctest R       0:14      1 kcpu7-8
```

In the above output

ST means Status of the Job in queue

R means Running,

P means Pending,

CE means Completing and etc

Note: Refer this link for additional Reference: https://slurm.schedmd.com/squeue.html

4.4 Canceling the Jobs (scancel)

The command *scancel* is used to delete or cancel the jobs from the job queue. While executing this command, slurm will kill all the processes executed under the given jobid and safely clean it from the job scheduling queue.

Syntax: squeue [additional scancel options] [jobid]
Example 1: Cancel single selecting jobs

```
$ scancel 2021
```

Above command will cancel the 2021 from the job queue.

Example 2: Cancel all the users jobs

```
$ scancel -u hpcuser
```

Above command will cancel all the jobs of the user “hpcuser” from the job queue

Example 3: Cancel all the pending jobs of a particular user

```
$ scancel -u hpcuser -t PENDING
```

Above command will cancel all the pending jobs of the user “hpcuser”. This command will not impact the running jobs.

Note: Refer this link for additional Reference: https://slurm.schedmd.com/scancel.html

4.5 Job Reporting/Job Controlling (scontrol)

4.5.1 Job Report

To know full status with detailed information of a submitted job
Syntax: $ scontrol show jobid <joobid>

Example:

```
$ scontrol show jobid 2021
```

```
   JobId=1726 JobName=mix_100c
UserId=mangal(2526) GroupId=sysadm(1014) MCS_label=N/A Priority=4564 Nice=0
Account=kamet QOS=normal JobState=RUNNING Reason=None Dependency=(null)
Requeue=1 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0RunTime=00:00:16
TimeLimit=03:30:00 TimeMin=N/A SubmitTime=2024-06-09T15:50:53
EligibleTime=2024-06-09T15:50:53 AccrueTime=2024-06-09T15:50:53
StartTime=2024-06-09T15:50:53 EndTime=2024-06-09T19:20:53 Deadline=N/A
SuspendTime=None SecsPreSuspend=0 LastSchedEval=2024-06-09T15:50:53
Scheduler=Backfill
Partition=kmedium AllocNode:Sid=kamet2:2372297 ReqNodeList=(null)
ExcNodeList=(null) NodeList=kcpu[11-12]
BatchHost=kcpu11
NumNodes=2 NumCPUs=100 NumTasks=100 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
TRES=cpu=100,mem=400000M,node=2,billing=100
Socks/Node=* NtasksPerN:B:S:C=0:0:*:* CoreSpec=* MinCPUsNode=1
MinMemoryCPU=4000M MinTmpDiskNode=0 Features=(null) DelayBoot=00:00:00
OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
Command=/lustre/mangal/slurm/job-script.sh WorkDir=/lustre/mangal/slurm
StdErr=/lustre/mangal/slurm/mix_100c-job.1726.out StdIn=/dev/null
StdOut=/lustre/mangal/slurm/mix_100c-job.1726.out Power=
MailUser= <userid>@imsc.res.in
MailType=INVALID_DEPEND,BEGIN,END,FAIL,REQUEUE,STAGE_OUT
```

Above command will report the information about the following

- When the job is submitted (Submit Time)
- When the job is started (Start time)
- How long the job is running (RunTime)
- On which compute nodes its running (NodeList & BatchHost)
- How many core are allocated (NUMCPUs/TRES)
- How much RAM is allocated (TRES)
- Which path its running (WorkDir)
- etc

4.5.2 Holding a Job

To hold the list of submitted job not to start until its released by user or admin

Syntax: scontrol hold <joblist>

Example:

```
$ scontrol hold 1726
```

```
$ scontrol hold 1726,1727
```

Note:

- There are two types of holding in SLURM. 1 hold by user and 2 hold by administrator.
- If a job is held by a user then the same job can be released by both Administrator or the same user, but if the job is held by an administrator the administrator only can be released.

4.5.3 Release the hold job

To release the hold jobs to allow to start as per the availability of resources Syntax: scontrol release <joblist>

Example:

```
$ scontrol release 1726
$ scontrol release 1726,1727
```

4.6 View information about SLURM node and Partition (sinfo)

The command "sinfo" is used to view partition and node status information of the SLURM job scheduler on this cluster.

Syntax: sinfo [addition sinfo options] Example 1:

```
$ sinfo
```

```
PARTITION   AVAIL TIMELIMIT  NODES  STATE NODELIST
kshort*       up   12:00:00     18   idle kcpu[1-10,41-48]
kmedium       up 2-00:00:00     43   idle kcpu[11-48,54-58]
klong         up 7-00:00:00     10   idle kcpu[49-58]
```

Following is the possible node states

- idle – all the core of the nodes are free and not allocated
- alloc – all the cores of the nodes is allocated and no cores are free.
- mix – It's a mixed of idle and alloc states of a node, which means partial core are free
- down – the node is down due to issue
- drain – the node having jobs but its not able to kill the job or signal the job.

Example 2:

```
$ sinfo -o %n,%C,%e,%m
```

```
HOSTNAMES,CPUS(A/I/O/T),FREE_MEM,MEMORY
kcpu1,    0/64/0/64,     250522, 256000
kcpu2,    0/64/0/64,     250517,  256000
kcpu3,    0/64/0/64,     250625,  256000
kcpu4,    0/64/0/64,     250843,  256000
kcpu5,    0/64/0/64,     250126,  256000
kcpu6,    0/64/0/64,      250131, 256000
kcpu7,    0/64/0/64,     250362,  256000
kcpu8,    0/64/0/64,     250434,  256000
kcpu9,    0/64/0/64,     250460,  256000
kcpu10,   0/64/0/64,       250436, 256000
etc..
```

The above command will display the Allocated/Idle CPUs and free memory information of each node. Here in (A/I/O/T) A – Allocated, I – Idle, O – offline, T – total

5 Environment Modules

Environment Modules provide a convenient way to dynamically change the users' environment through modulefiles. This includes easily adding or removing directories to the PATH environment variable and LD_LIBRARY_PATH environment variables.

A modulefile contains the necessary information to allow a user to run a particular application or provide access to a particular library. All of this can be done dynamically without logging out and back in. Modulefiles for applications modify the user's path to make access easy. Modulefiles for Library packages provide environment variables that specify where the library and header files can be found.

5.1.1 Sample Module file

```
#%Module1.0 ####################################################################
proc ModulesHelp { } { puts stderr " "
puts stderr "This module loads the gromacs program" puts stderr "application."
puts stderr "\nVersion 2024.1\n"

}
module-whatis "Name: gromcs" module-whatis "Version: 2024.1"
module-whatis "Category: runtime library"
module-whatis "Description: example independant module" module-whatis "URL
http://www.google.com/"
set   version     2024.1
set GMXPREFIX     /opt/apps/gromacs/2024.1
prepend-path      PATH  ${GMXPREFIX}/bin prepend-path LD_LIBRARY_PATH
${GMXPREFIX}/lib64 prepend-path     MANPATH     ${GMXPREFIX}/share/man
prepend-path      PKG_CONFIG_PATH   ${GMXPREFIX}/lib64/pkgconfig
append-path PATH  /opt/ohpc/pub/compiler/gcc/12.3.0/bin append-path
LD_LIBRARY_PATH
/opt/ohpc/pub/compiler/gcc/12. 3.0/lib64
setenv      GMXBIN            ${GMXPREFIX}/bin setenv GMXLDLIB
${GMXPREFIX}/lib64
setenv      GMXMAN      ${GMXPREFIX}/share/man setenv GMXDATA
${GMXPREFIX}/share/gromacs setenv   GMXTOOLCHAINDIR
${GMXPREFIX}/share/cmake setenv     GROMACS_DIR ${GMXPREFIX}
```

5.2 Listing Available Modules

Use the “module avail” command to list all the available modules:

Syntax: module avail

Example:

```
$ module avail
```

```
---------------------------        /opt/ohpc/pub/moduledeps/intel-i        -------------------
         mpi adios/1.13.1     hypre/2.18.1      netcdf-cxx/4.3.1
                 (D)          petsc/3.18.1
      ptscotch/7.0.1          superlu_dist/6.4.0
         boost/1.81.0         mfem/4.4                                    phdf5/1.10.8
                              netcdf-fortran/4.6.0 (D)
      scalapack/2.2.0         trilinos/13.4.0
         fftw/3.3.10          mumps/5.2.1                                 pnetcdf/1.12.3
                              netcdf/4.9.0              (D)               slepc/3.18.0


---------------------------------        ------------------------------
        advisor/2024.0                    compiler32/2023.2.1             dpct/2024.0.0
      intel_ippcp_intel64/2021.9          oclfpga/2023.2.1
        ccl/2021.11.2                     compiler32/2024.0.2             dpl/2022.3

      itac/2022.0                         oclfpga/2024.0.0 (L,D)
        compiler-rt/2023.2.1              dal/2024.0.0                    icc/2023.2.1
      mkl/2023.2.0                        tbb/2021.10.0
        compiler-rt/2024.0.2              debugger/2023.2.0               icc32/2023.2.1


      mkl/2024.0                          tbb/2021.11 (L,D)



        compiler-rt32/2023.2.1            debugger/2024.0.1               ifort/2024.0.2
      mkl32/2023.2.0                      vtune/2024.0
        compiler-rt32/2024.0.2            dev-utilities/2021.10.0         ifort32/2024.0.2
      mkl32/2024.0


        compiler/2023.2.1                 dev-utilities/2024.0.0          inspector/2024.0

                              /opt/ohpc/pub/moduledeps/oneapi


       mpi/2021.10.0
         compiler/2024.0.2    (L,D)                        dnnl/3.3.0
       intel_ipp_intel64/2021.10 mpi/2021.11                     (L,D)

---------------------------        /opt/ohpc/pub/moduledeps/intel        -------------------
         gsl/2.7.1                    impi/2021.10.0
       metis/5.1.0                 mvapich2/2.3.7          netcdf- fortran/4.6.0
       openmpi4/4.1.6              scotch/6.0.6
         hdf5/1.10.8              impi/2021.11              (L,D)
       mpich/3.4.3-ofi            netcdf-cxx/4.3.1          netcdf/4.9.0 plasma/21.8.29
       superlu/5.2.1
```

```
---------------------------   /home/mangal/modulefiles   ---------------------------
      ohpc (L)

---------------------------   /opt/apps/modulefiles   ---------------------------
-------------------------

         diagham/4346          gromacs/2024.1
      intelpython/2024.0.2.1   lammps/02-08-2023 namd2/2.14

-------------------------------------   ------------------------------------
       EasyBuil       cmake/         hwloc/2.       intel/2                os
       d/4.8.2        3.24.2         7.2            024.0.2


     ucx/1.15.0
           (L)
       autotool       gnu12/         intel/20       libfabr                pr
       s              12.3.0         23.2.1         ic/1.19                un
                                                    .0                     /2
                                                                           .2
     valgrind/3
     .19.0
                          /opt/ohpc/pub/modulefiles


        Where:
         D: Default
         Module L:
         Module is
         loaded

      If the avail list is too long consider trying:

      "module --default avail" or "ml -d av" to just list the
      default modules. "module overview" or "ml ov" to display the
      number of modules for each name.

      Use "module spider" to find all possible modules and extensions.
      Use "module keyword key1 key2 ..." to search for all possible modules
      matching any of the "keys".
```

5.3 Listing Loaded Modules

Use the “module list” command to list only the loaded module in the current shell environment.

Syntax: module list

Example:

```
$ module list
```

```
Currently Loaded Modules:
  1) tbb/2021.11       3) oclfpga/2024.0.0          5)
                                                    mkl/2024.0
mpi/2021.11 9) ucx/1.15.0
  2) compiler-rt/2024.0.2      4)                   6)
  compiler/2024.0.2                                 intel/2024.0
                                                    .2
impi/2021.11 10) ohpc
```

Note:

kamet hpc cluster is configured to load Intel OneAPI compiler and IMPI (Intel MPI)

module by default to all the users.

If the users do not want to load the intel module by default, then they should comment the following line in /home/<username>/modulefiles/ohpc

```
$ module try-load intel
$ module try-load impi
$ module try-load ucx
```

or change intel and impi to gnu12 and openmpi4 respectively to load the gnu compiler.

```
$ module try-load gnu12
$ module try-load openmpi4
$ module try-load ucx
```

5.4 Loading Module to the Environment

Use the “module load “ command to load the specified module to the current environment.

Syntax: module load <module list>

Example 1:

```
$ module load lammps, gromacs
```

The above command will load the lammps and gromacs module to the current environment

Example 2:

```
$ module swap intel gnu12
```

The above command will swap the same family of modules with a new one. Here the Intel and gnu12 are same family of modules and intel got swapped by gnu2

5.5 Unloading/Removing module from Environment.

Use the “module remove” command to unload or remove the specific modules from the current environment.

Syntax: module rm <module list>

Example:

```
$ module rm lammps
```

The above command will remove the module lammps from the environment.

5.6 Show modules

Use the “module show” command to view its targeted environment variable and its value.

Syntax: module show

Example:

```
$ module show lammps
---------/opt/apps/modulefiles/lammps/02-08-2023:------
      whatis("Name: lammps")
-------whatis("Version: 02-08-2023")--------------------
      whatis("Category: runtime
      library")
      whatis("Description: example independant module")
      whatis("URL http://www.google.com/")
      prepend_path("PATH","/opt/apps/lammps/02-08-2023/bin
      ")
      prepend_path("MANPATH","/opt/apps/lammps/02-08-2023/share/man
      ") setenv("LAMMPS_POTENTIALS","/opt/apps/lammps/02-08-
      2023/share/lammps/potentials")
      help([[
      This module loads the lammps program
      application.

      Version 02-08-2023
    ]])
```

6 Don’ts

- Do not run ls -l frequently on /lustre location
- Do not run du -sh command on /lustre location
- Do not store many small files in number in /lustre
- Do not keep the files more than 6 months in /lustre
- Do not submit more jobs in pending status.