We have the following cluster :

Fermi Cluster : 75 computations nodes, total 2,104 cores. Currently, Fermi Cluster are integrated, so the host name of head node (namely, master node & login node) is 'Fermi', and the name of the computation nodes are 'Dirac'.

You can login to fermi through this command: ssh –XCY –p 4022 username@fermi.ibs.re.kr

After login to the fermi head node, you can submit the jobs ('qsub' command) to the computation nodes. Namely, this means that you don't have to login to the each computation nodes. On the head node, if you submit your jobs using 'qsub', automatically, the job manager of the cluster would send them to free computation nodes.

The computation nodes (Dirac01 to Dirac16) are connected to the FDR IB card, and the other computation nodes (Dirac17 to Dirac75) are connected to the EDR InfiniBand (IB) card. The specification for each node are as follows.

1. Dirac01 ~ Dirac32 : 24 cores (12 cores * 2 CPU) with 64 GB memory per 1 node
2. Dirac33 ~ Dirac64 : 28 cores (14 cores * 2 CPU) with 64 GB memory per 1 node
3. Dirac65 ~ Dirac75 : 40 cores (20 cores * 2 CPU) with 256 GB high memory per 1 node

In this year (in 2022), We plan to add one more cluster login node in order to prevent from bottleneck due to many user's simultaneous access to fermi as well as to upgrade to login speed. Also, For the future, we are considering add more high cores in the cluster such as 64 / 100 cores.

1. System environment information

- OS: CentOS 7.4
- Job Scheduling and Resource Management: SGE

2. Queue

Based on the each node's specifications, we have 8 queues as follows.

1. debug.q → Only use node [dirac01], wall time 2 hours
2. single.q → Only use nodes [dirac02 ~ dirac08], unlimited wall time
3. short_24.q →Only use nodes [dirac09 ~ dirac32], wall time 2 days, 200 cores limit per user for running job
4. short_28.q → Only use nodes [dirac33 ~ dirac64], wall time 2 days, 600 cores limit per user for running job
5. long_24.q → Only use nodes [dirac09 ~ dirac32] , wall time 2 weeks, 220 cores limit per user for running job
6. long_28.q → Only use nodes [dirac33 ~ dirac64], wall time 2 weeks, 100 cores limit per user for running job
7. hm_short_40.q → Only use nodes [dirac65 ~ dirac75], wall time 2 days, 120 cores limit per user for running job
8. hm_long_40.q → Only use nodes [dirac65~ dirac75], wall time 2 weeks, 120 cores limit per user for running job

3. Application

We are using **"Module"** to use the installed SW. Module is a tool to easily configure the application environment to improve the usability of the application.

"module avail" command shows available modules (Namely, the installed SW) as follows.

```
[root@fermi:~]# module avail

------------------------------------------------ /usr/share/Modules/modulefiles ------------------------------------------------
dot         module-git  module-info modules     null        use.own

------------------------------------------------------ /opt/Modules/scheduler ------------------------------------------------------
sge/8.1.8

------------------------------------------------------ /opt/Modules/compiler -------------------------------------------------------
gcc/7.3.0                                intel-oneapi/dev-utilities/latest         intel-oneapi/intel_ippcp_ia32/2021.5.1
gcc/8.2.0                                intel-oneapi/dnnl/2022.0.2                intel-oneapi/intel_ippcp_ia32/latest
gcc/9.2.0                                intel-oneapi/dnnl/latest                  intel-oneapi/intel_ippcp_intel64/2021.5.1
intel/17.0.1                             intel-oneapi/dnnl-cpu-gomp/2022.0.2       intel-oneapi/intel_ippcp_intel64/latest
intel/18.0.3                             intel-oneapi/dnnl-cpu-gomp/latest         intel-oneapi/itac/2021.5.0
intel/mkl-2019u5                         intel-oneapi/dnnl-cpu-iomp/2022.0.2       intel-oneapi/itac/latest
intel-oneapi/advisor/2022.0.0            intel-oneapi/dnnl-cpu-iomp/latest         intel-oneapi/mkl/2022.0.2
intel-oneapi/advisor/latest              intel-oneapi/dnnl-cpu-tbb/2022.0.2        intel-oneapi/mkl/latest
intel-oneapi/ccl/2021.5.1                intel-oneapi/dnnl-cpu-tbb/latest          intel-oneapi/mkl32/2022.0.2
intel-oneapi/ccl/latest                  intel-oneapi/dpct/2022.0.0                intel-oneapi/mkl32/latest
intel-oneapi/clck/2021.5.0               intel-oneapi/dpct/latest                  intel-oneapi/mpi/2021.5.1
intel-oneapi/clck/latest                 intel-oneapi/dpl/2021.6.0                 intel-oneapi/mpi/latest
intel-oneapi/compiler/2022.0.2           intel-oneapi/dpl/latest                   intel-oneapi/oclfpga/2022.0.2
intel-oneapi/compiler/latest             intel-oneapi/icc/2022.0.2                 intel-oneapi/oclfpga/latest
intel-oneapi/compiler-rt/2022.0.2        intel-oneapi/icc/latest                   intel-oneapi/tbb/2021.5.1
intel-oneapi/compiler-rt/latest          intel-oneapi/icc32/2022.0.2               intel-oneapi/tbb/latest
intel-oneapi/compiler-rt32/2022.0.2      intel-oneapi/icc32/latest                 intel-oneapi/tbb32/2021.5.1
intel-oneapi/compiler-rt32/latest        intel-oneapi/init_opencl/2022.0.2         intel-oneapi/tbb32/latest
intel-oneapi/compiler32/2022.0.2         intel-oneapi/init_opencl/latest           intel-oneapi/vpl/2022.0.0
intel-oneapi/compiler32/latest           intel-oneapi/inspector/2022.0.0           intel-oneapi/vpl/latest
intel-oneapi/dal/2021.5.3                intel-oneapi/inspector/latest             intel-oneapi/vtune/2022.0.0
intel-oneapi/dal/latest                  intel-oneapi/intel_ipp_ia32/2021.5.2      intel-oneapi/vtune/latest
intel-oneapi/debugger/2021.5.0           intel-oneapi/intel_ipp_ia32/latest        llvm/8.0.0
intel-oneapi/debugger/latest             intel-oneapi/intel_ipp_intel64/2021.5.2
intel-oneapi/dev-utilities/2021.5.2      intel-oneapi/intel_ipp_intel64/latest

------------------------------------------------------ /opt/Modules/language -------------------------------------------------------
julia/gcc-4.8.5/1.0.5   julia/gcc-4.8.5/1.8.0   python/gcc-4.8.5/3.4.9  python/gcc-4.8.5/3.7.11 python/gcc-4.8.5/3.9.7
julia/gcc-4.8.5/1.6.2   python/gcc-4.8.5/2.7.15 python/gcc-4.8.5/3.5.6  python/gcc-4.8.5/3.8.11

------------------------------------------------------ /opt/Modules/library --------------------------------------------------------
armadillo/gcc-4.8.5/9.200.7                 gl2ps/gcc-4.8.5/1.4.0
arpack-ng/gcc-4.8.5/openmpi-2.1.5/3.7.0     glfw/3.3
arpack-ng/gcc-4.8.5/serial/3.7.0            gmp/6.1.2
arpack-ng/gcc-9.2.0/imkl-2019u5/serial/3.7.0 gsl/gcc-4.8.5/2.5
arpack-ng/gcc-9.2.0/openmpi-4.0.2/3.7.0     guile/gcc-4.8.5/2.2.4
```

You can see the modules which listed up through the "module avail" command, and select the module to use through **"module load"** command as follows.

```
[root@fermi:~]# module load intel-oneapi/advisor/2022.0.0
Loading advisor version 2022.0.0
[root@fermi:~]#
```

For the details, Please refer to the attached manuals.

- How to use modulefiles .pdf
- pcs_application_installed_list-20220726