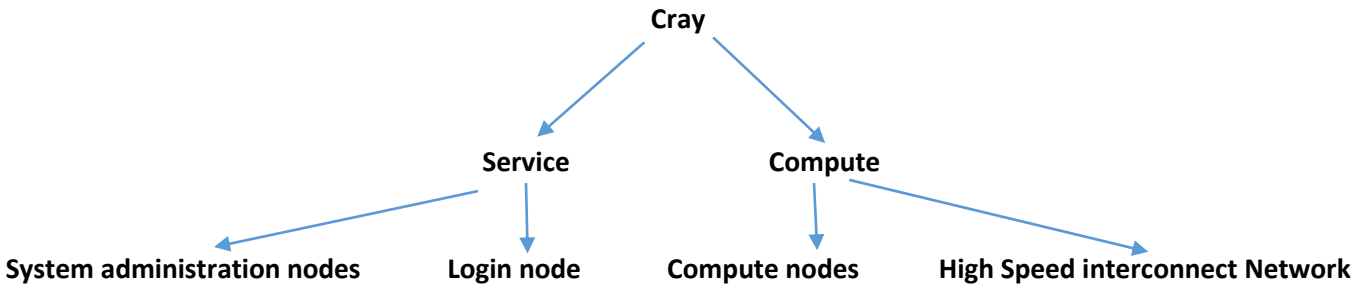
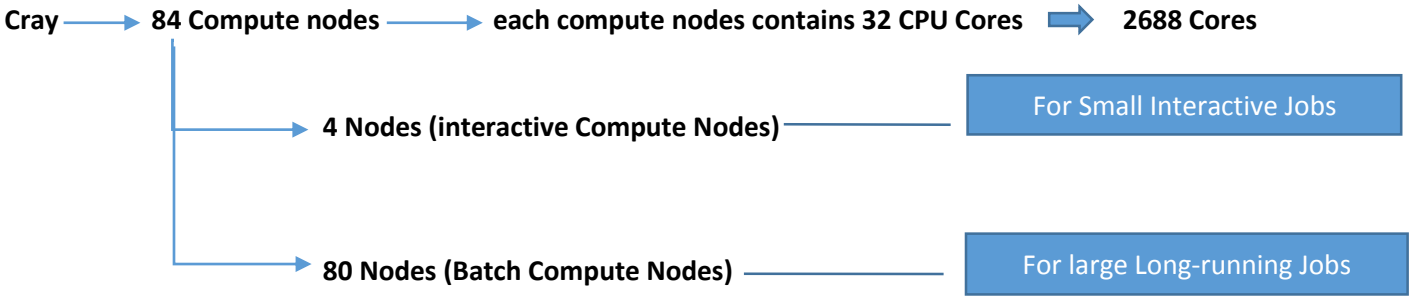
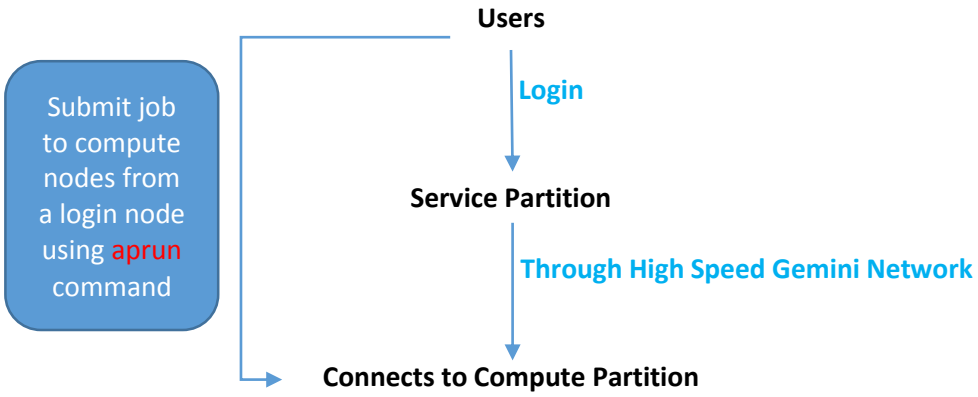
Cray High Performance Computer (XE6)







Access:

Popular client applications to access the Cray:

PuTTY: http://www.chiark.greenend.org.uk/~sgtatham/putty/

FileZilla: https://filezilla-project.org/

General Commands:

Script raven_demo.log: Script started and the file is raven_demo.log
Cat /etc/motd: Summary of: Number of nodes which are on login node, 20 compute nodes and 8 XK6 compute nodes which are GPU enable nodes.
Speed of processors and memory

Hostname –a: name of the machine that we are on
Who: How many people are on the system and how busy the system is
Stprocadmin: provides a snapshot of all the nodes on the machine (64 nodes)
Whether they are service nodes or compute nodes
Whether they are up and down
Whether they are interactive mode or batch mode

Parse_xtprocadmin.sh: Full representation of xtprocadmin (more columns about GPU and clock bits and so on)
Module list: provides list of modules
Module avail: shows all available modules
Module avail cce: All different versions of cray compiler including the default version
Module swap PrgEnv-cray PrgEnv-pgi: You will be able to switch between modules
Module unload PrgEnv-pgi: Unload a module
Module load PrgEnv-cray: load the cray programming environment
Pwd: directory that we are in
Ls ${Home}: everything on out home directory
Df –h: How many file systems are currently mounted
Ps –ef | grep username: all the processes I am running

Login node: when you are on the system, you are on the login node or service node and that where you are doing all your activity. By using launch command, you can run your job.
Cat /proc/ cpuinfo: gives us the info About processors associated with this login node
Cat /proc/ cpuinfo | grep processor : list of processors and cores associated with this node
Cat /proc/ cpuinfo | grep MHz: Parse out the processor speed
Cat /proc/ meminfo:
Cat /proc/ meminfo | grep MemTotal: how much memory on this node
Ls /proc/cray_xt: cray_xt has some specific info about cray such as cname and nid which are respectively name of the node and its id
More /proc/cray_xt/cname: provides the name of the node which is a kind of representor for the location of node into the machine if you can have this type of info by typing xtnodestat command
More /proc/cray_xt/nid: provides the id of the node

Qstat –q: how many queues are available?
Qstat –a: list of jobs running
Aprun –n 1 cat /proc/cpuinfo | grep "model name" | tail -1: if we use cat /proc/cpuinfo we will get info about login node, but if we need to get info about compute nodes, we have to use aprun command. –n shows how many cores you are going to run. And the rest is the same. So, by this command we launch the job which is a cat on that processor

Aprun –n 1 cat /proc/cpuinfo | grep processor: list of processors on the compute node (32 processors)

Aprun –n 1 cat /proc/cpuinfo | grep MHz: speed f processors
Aprun –n 1 cat /proc/meminfo: same thing for memory
Aprun –n 1 cat /proc/cray_xt/nid : net id on compte nodes
Apron –n 1 pwd: home directory
cc: Compiler for compiling c codes example: cc hello.c
CC: Compiler for compiling C++ codes example: CC hello.cpp
Ftn: Compiler to compile Fortran code example: ftn hello.f90
Qsub filename: for submitting the job where filename is the name of a batch text file
Qstat –q: list all available queues (brief)
Qstat –qf: list all available queues (full)
Qstat: show the status of jobs in all queues
Qstat –u username: show only the status of jobs corresponding to the written user account
Qsub filename: submit a job to the default batch queue
Qdel jobid: delete a job from a batch queues

Explanation of Batch Script:

#!/bin/bash → **Specifying the shell environment to use the batch file (It is not requires but its more professional to mention the name of the shell)**

#PBS –N result → **-N renames the output file to whatever name we mention**

#PBS –j oe → **For combining standard output and standard error in a single file**

#PBS –l mppwidth=32 → **-l mppwidth specifies the number of cores to allocate the job (It has to be less than 1792 where 1792 is the number of cores in a largest batch queue)**

#PBS –l walltime=1:00:00 → **-l walltime specifies the maximum amount of time in hours: minutes: seconds in which the job may take to run**

Cd $PBS_O_WORKDIR → **path to the directory from which you submitted your job**

aprun –n 32 exacutablefile → **example: apron –n 32 ./a.out**

Sample Batch Script:

Suppose you have a C++ code called: SampleCode.cpp
Compile your code: CC SampleCode.cpp
Open a batch file: vim test.sh
Write your batch file as:

```
#!/bin/bash
#PBS -N result
#PBS -j oe
#PBS -l mppwidth=30
#PBS -l walltime=00:10:00


cd $PBS_O_WORKDIR
aprun -n 30 ./a.out
```

Save your batch file: ESC → type :wq
Submit your job: qsub test.sh
Check the status of your job: qstat
Make sure the output file has been created: type ls and see if you there is a file named result.o1023
(1023 is an example job id)
See the result: cat result.o1023

Note: make sure that the number of cores in #PBS -l mppwidth=30 and the number of cores in aprun -n 30 ./a.out are equal to each other.