Concepts: resources, batch queues, virtual resource pools

Dr. Clement Onime

onime@ictp.it

Overview

- Resources
  - Sharing
  - Security
- Concepts
  - Resource management
  - Policies
- Advanced configurations/examples

RESOURCES

HPC resources

Common resources

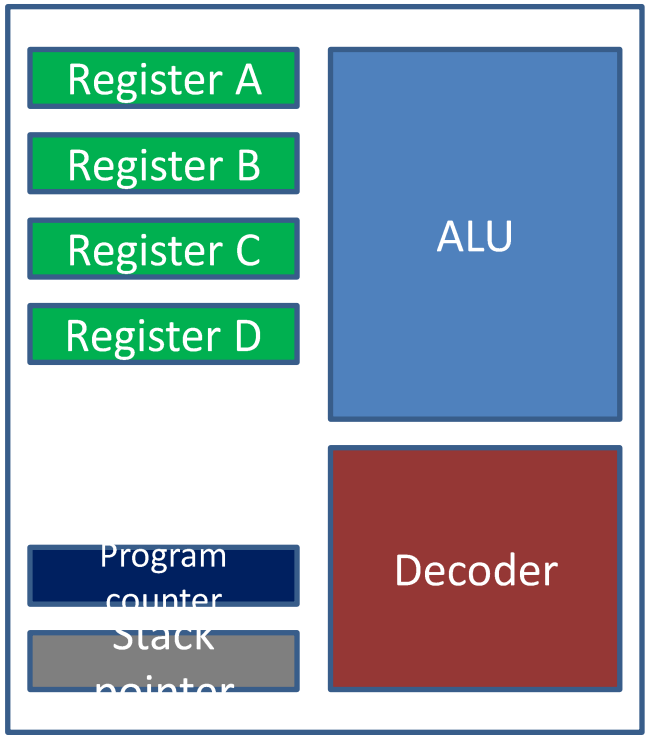
- Compute nodes
  - CPU
  - Memory
  - Network
  - Storage devices
  - Visualizations/display
- Central Storage
  - occupancy

Management concepts

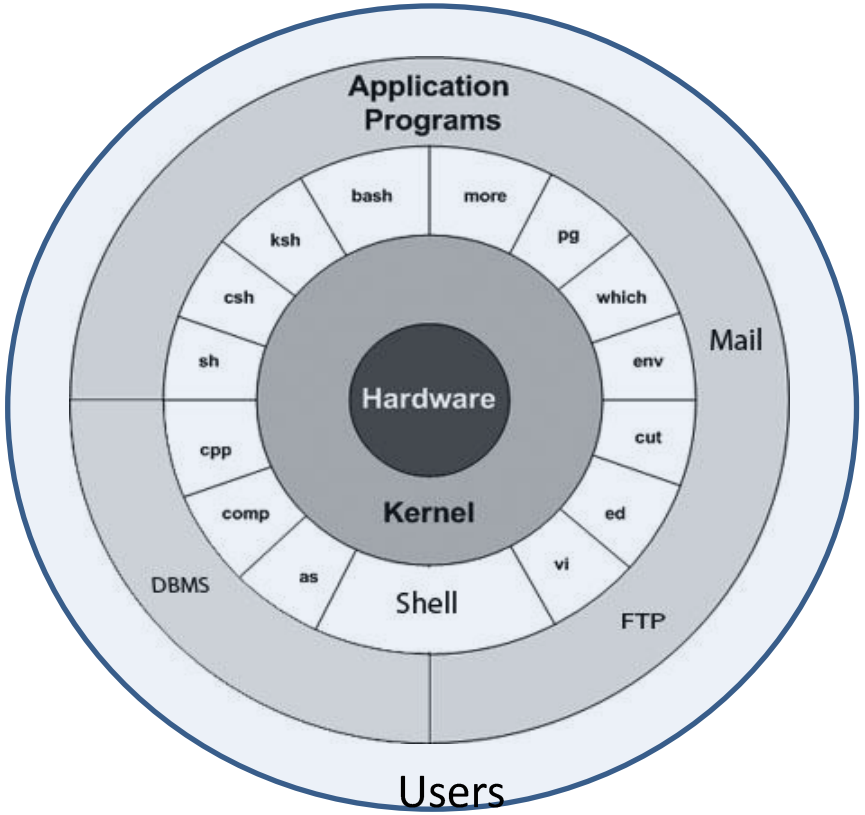
- Allocation/de-allocation
  - Whole or partial
  - Schedule/queues
- Access control/security
  - Multi-user nature of Unix
- Setting boundaries
  - Policies
  - Priorities
- Avoiding violations
- Accounting

Sharing resources

- Resources are always limited
  - Challenge is optimizing usage
  - Multi-tasking in a single CPU
  - Scheduling
    - Improving multi-tasking based on other external factors
    - On-demand scheduling (Event/IRQ based)



Securing resources



- Operating System manages access to key hardware resources
- Resources
  - CPU, memory and peripherals
- Jobs/tasks
  - Consumption of resources

Organizing resources

- Grouping resources into pools
  - Defining entry points (queues) and requirements
  - Easier to plan/coordinate utilization/consumption
  - Easier mitigation of faults
  - Better accounting
- *In HPC, **queues** refer to the main entry points through which users send their requests. While **jobs** refer to the requests by users for consumption of resources.*

CONCEPTS

Management: resources

- Creating groups and associated entry points
  - Also deleting/removal
- Defining and enforcing entry requirements
  - Security, conditions for use, FIFO, fair-share
- Scheduling of available resources
  - Off-line individual resources for maintenance
- Monitoring
  - Availability, reporting and accounting

Management: requests

- Accepting new requests
  - Submission, suspend, cancel or delete
- Managing requests
  - Matching requests to available resources
  - Scheduling/queuing requests based on availability of resources
  - Start, stop, restart and cleaning-up
- Monitoring
  - Reporting and accounting

Policies

- Definition
  - Advanced mechanisms that automate the goals and control of the utilization of resources on a long term basis
  - General rules of behaviour for the HPC system to be implemented by the batch queue system
- Affects the whole HPC system
  - Could be abstract
  - Usually automated

Share based utilization policy

- Shares
  - User (individuals)
  - Departments (groups of users)
  - Projects (working teams)
- Hierarchical nature (mirrors organisational structure)
- Optionally implements history for detecting/punishing over-usage
- Useful when multiple organisations fund the HPC system

Functional utilization policy

- Shares
  - By importance of organisational role or function
- Complex & versatile nature
  - as individual jobs/classes have to implemented
- Does not maintain a history or punish or compensate under-usage and may be pre-emptive
- Useful for situations levels of importance are fixed

Deadline utilization policy

- Shares
  - Normal “day-to-day” usage
  - Jobs that must be completed by certain dates
- Fractional/Percentage nature
  - On-demand reservation/allocation of some HPC resources. May terminate day-to-day jobs in order to satisfy deadlines.
- Attempts to balance between two extremes
- Useful for when deadlines exists

Manual override utilization policy

- Shares
  - Fixed or percentage allocation/de-allocation to individual jobs, users, departments.
- Non- automated nature
  - As specified by administrator
  - Has to balanced against other policies in place
- Useful when some “temporary” administrative goal has changed or must be met

Scheduling policies

- Policies that affect only the start-up of jobs
- Static by nature
- Common algorithms
  - First Come First Serve (FIFO)
  - Least Loaded
  - Weighted (by no of cores, etc)..
  - Fixed sequence or Sequential
  - Combination of the above

ADVANCED CONFIGURATIONS/ EXAMPLES

Heterogeneous hardware

- Node definitions should reflect hardware differences
  - Use short tags to reflect
    - CPU type/no of cores/families
    - Amount of RAM
    - Network devices (Infiniband vs non-infiniband)
- Grouping into queues
  - Use predefined tags select which nodes may be grouped together.
  - *Defaults should use minimum capabilities*

Torque (nodes file)

```
node05 np=8 12gb ib0 nh4core cmsp_guest
node25 np=8 24gb ib0 nh4core any
node39 np=20 64gb ivy10core any
node40 np=20 64gb ivy10core cmsp1
testing01 np=8 testing
serial01 np=7 serial
node45 np=12 24gb west6core any
node79 np=12 32gb sb6core esp
node123 np=12 32gb sb6core cmsp
node124 np=12 32gb sb6core cmsp
node157 np=16 32gb sb8core any
node190 np=20 64gb ivy10core esp1
gpu01 np=16 gpus=2 32gb sb6core 4992gpucores gpu
```

Torque (queues definitions)

```
set queue cmsp1 resources_default.neednodes = cmsp1
set queue testing resources_default.neednodes = testing
set queue esp1 resources_default.neednodes = esp1
set queue esp resources_default.neednodes = esp
set queue arc resources_default.neednodes = any
set queue gpu resources_default.neednodes = gpu
set queue short resources_default.neednodes = any
set queue esp_guest resources_default.neednodes = esp_guest
set queue cmsp resources_default.neednodes = cmsp
set queue cmsp_guest resources_default.neednodes = cmsp_guest
set queue long resources_default.neednodes = any
set queue serial resources_default.neednodes = serial
```

Example configuration

- Maui (maui.cfg)

```
NODESETLIST nh4core west6core sb6core sb8core ivy10core serial testing
QOSCFG[esp] QFLAGS=DEDICATED OMAXPROC[USER]=4096 FLAGS=IGNMAXJOB
QOSCFG[long] QFLAGS=DEDICATED
QOSCFG[cmsp] QFLAGS=DEDICATED OMAXPROC[USER]=4096 FLAGS=IGNMAXJOB
QOSCFG[serial] MAXJOB=32
CLASSCFG[serial]  PRIORITY=1
NODECFG[serial01] MAXJOB=7
NODECFG[serial02] MAXJOB=7
CLASSCFG[long]    PRIORITY=10000 QDEF=long
CLASSCFG[esp]   PRIORITY=20000 QDEF=esp
CLASSCFG[esp_guest]     PRIORITY=20000 QDEF=esp_guest
RCFG[serial] CLASSLIST=serial ACCESS=SHARED PERIOD=INFINITY GROUPLIST=ALL HOSTLIST=serial01,serial02
SRCFG[esp] CLASSLIST=esp PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=esp
SRCFG[esp1] CLASSLIST=esp1 PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=esp1
SRCFG[esp_guest] CLASSLIST=esp_guest PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=esp_guest
SRCFG[cmsp] CLASSLIST=cmsp PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=cmsp
SRCFG[cmsp1] CLASSLIST=cmsp1 PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=cmsp1
SRCFG[cmsp_guest] CLASSLIST=cmsp_guest PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=cmsp_guest
SRCFG[long] CLASSLIST=long PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=any
```

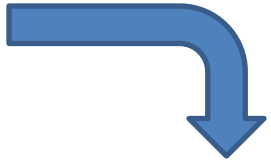
Example configuration

Virtual queues/resources

- Virtual queues can be used to present different views/policies of resources
  - Routing queue work by accepting jobs and forwarding them for execution to a different queue.

Torque (routing queue)

```
# Create and define queue short
create queue short
set queue short queue_type = Route
set queue short resources_max.walltime = 06:00:00
set queue short resources_default.neednodes = any
set queue short resources_default.walltime = 01:00:00
set queue short acl_group_enable = True
set queue short acl_groups = HPCUsersGroup
set queue short acl_group_sloppy = True
set queue short route_destinations = long
set queue short enabled = True
set queue short started = True
set server default_queue = short
```



Torque (normal queue)

```
# Create and define queue long
create queue long
set queue long queue_type = Execution
set queue long resources_max.walltime = 24:00:00
set queue long resources_default.neednodes = any
set queue long resources_default.walltime = 01:00:00
set queue long acl_group_enable = True
set queue long acl_groups = HPCUsersGroup
set queue long acl_group_sloppy = True
set queue long enabled = True
set queue long started = True
set server default_queue = short
```

Example configuration

Remote vs local jobs

- All Jobs should be executed on compute (worker) nodes not on master.
- Securing master and/or login nodes
  - Do not run resource manger client daemon
  - Limit consumption of local CPU resources
  - Allocate separate limited resource for short testing
  - Use CPU-SETS to partition/allocate cores for user tasks.

/etc/security/limits.conf

```
#NO logins allowed on argo for normal users
root            -       maxlogins       10
onime           -       maxlogins       10
mverina         -       maxlogins       10
@atmosGroup     -       maxlogins       0
*               -       maxlogins       0
```

Cpu-sets

```
#/etc/cset.conf
#
[default]
mountpoint = /cgroup/cpuset
```

Cpu-sets

```
#FOR /etc/rc.local
[ -d /dev/cpuset ] || mkdir /dev/cpuset
mount -t cpuset none /dev/cpuset
#
UDIR="/cgroup/cpuset/users"
OSDIR="/cgroup/cpuset/os"
cat "${UDIR}/cpuset.cpus" | grep '0-5' || echo '0-5' > "${MYDIR}/cpuset.cpus"
[ -e "${UDIR}/cpuset.cpu_exclusive" ] && echo 1 > "${MYDIR}/cpuset.cpu_exclusive"
[ -e "${UDIR}/cpuset.mems" ] && echo 0 > "${MYDIR}/cpuset.mems"
[ -e "${UDIR}/cpuset.sched_relax_domain_level" ] && echo -1 > "${MYDIR}/cpuset.sched_relax_domain_level"
#OS
cat "${OSDIR}/cpuset.cpus" | grep '6,7' || echo '6,7' > "${OSDIR}/cpuset.cpus"
[ -e "${OSDIR}/cpuset.cpu_exclusive" ] && echo 1 > "${OSDIR}/cpuset.cpu_exclusive"
[ -e "${OSDIR}/cpuset.mems" ] && echo 0 > "${OSDIR}/cpuset.mems"
[ -e "${OSDIR}/cpuset.mem_hardwall" ] && echo 1 > "${OSDIR}/cpuset.mem_hardwall"
[ -e "${OSDIR}/cpuset.sched_load_balance" ] && echo 0 > "${OSDIR}/cpuset.sched_load_balance"
[ -e "${OSDIR}/cpuset.sched_relax_domain_level" ] && echo -1 > "${OSDIR}/cpuset.sched_relax_domain_level"

#Force tasks to CPU by echoing PID into ${UDIR}/tasks OR ${OSDIR}/tasks
```

Security

- Compute nodes are main work-horse
  - Prevent un-authorized access/direct-logins to compute nodes
    - Except for root user and owner of current job during execution
  - Perform node health checks
    - Schedule automatic clean-up of local storage space/devices
    - Schedule periodic maintenance/reboots
  - No firewalls

/etc/security/access.conf

Idle

```
-:ALL EXCEPT root:ALL
```

During job execution

```
-:ALL EXCEPT root egbobani:ALL
```

Activation via PAM

```
#/etc/pam.d/sshd
#
#NEXT 2 LINES NEEDED FOR ICTP CLUSTER CONTROL
account    required      pam_access.so debug
session    required      pam_access.so debug
```

Health-check script

```
$clienthost     master
$logevent       0x7f
$usecp          *:/home /home
$node_check_script   /opt/bin/ictp_pbs_health_check
$node_check_interval 40,jobend
```

Example configuration

Example health checks

```
#!/bin/sh
#Clean-up local disks
tmpwatch --mtime 48 /var/spool/torque/{aux,spool,undelivered}
#For local scratch
tmpwatch --ctime 48 /local_scratch
#
#Handle KERNEL panics
MYERR=""
l=`dmesg | egrep  "BUG:|Code:|Trace:" | grep ':$' | sort | uniq | cut -f1 -d':'`
[ -n "${l}" ] && MYERR="CODE"
#Check for Orphaned processes causing high-load
MYCPUS=`cat /proc/cpuinfo | grep -i processor | wc -l`
MYLOAD=`cat /proc/loadavg | awk '{print $1}' | cut -f1 -d'.'`
MYX=`expr ${MYCPUS} + 3`
[ ${MYX} -gt 0 -a ${MYLOAD} -gt ${MYX} ] && MYERR="LOADAVG"
#
#Weekly reboot based on uptime
MYUP=`cat /proc/uptime | awk '{print $1}' | cut -f1 -d'.'`
[ ${MYUP} -gt 0 -a ${MYUP} -gt 604800 ] && MYERR="UPTIME"

#Check Infiniband device is OK
X=`ifconfig -a 2>/dev/null | grep ib0`
[ -z "${X}" ] && MYERR="INFINIBAND"
#REPORT
[ -n "${MYERR}" ] &&  echo "${MYERR}"
```

Non-batch jobs

- Serial jobs
  - Computer node is shared concurrently by several host
    - reserve 1 core for O.S. housekeeping (multi-tasking & I/O) duties .
- Interactive and parallel jobs
  - Password-less SSH key access must be set-up

Maui.conf

```
NODESETLIST nh4core west6core sb6core sb8core ivy10core serial testing
QOSCFG[esp] QFLAGS=DEDICATED OMAXPROC[USER]=4096 FLAGS=IGNMAXJOB
QOSCFG[long] QFLAGS=DEDICATED
QOSCFG[cmsp] QFLAGS=DEDICATED OMAXPROC[USER]=4096 FLAGS=IGNMAXJOB
QOSCFG[serial] MAXJOB=32
CLASSCFG[serial]  PRIORITY=1
NODECFG[serial01] MAXJOB=7
NODECFG[serial02] MAXJOB=7
CLASSCFG[long]    PRIORITY=10000 QDEF=long
CLASSCFG[esp]   PRIORITY=20000 QDEF=esp
CLASSCFG[esp_guest]     PRIORITY=20000 QDEF=esp_guest
RCFG[serial] CLASSLIST=serial ACCESS=SHARED PERIOD=INFINITY GROUPLIST=ALL HOSTLIST=serial01,serial02
SRCFG[esp] CLASSLIST=esp PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=esp
SRCFG[esp1] CLASSLIST=esp1 PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=esp1
SRCFG[esp_guest] CLASSLIST=esp_guest PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=esp_guest
SRCFG[cmsp] CLASSLIST=cmsp PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=cmsp
SRCFG[cmsp1] CLASSLIST=cmsp1 PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=cmsp1
SRCFG[cmsp_guest] CLASSLIST=cmsp_guest PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=cmsp_guest
SRCFG[long] CLASSLIST=long PERIOD=INFINITY HOSTLIST=ALL NODEFEATURES=any
```

/etc/profile.d/ssh_keys.sh

```
#!/bin/sh
# /etc/profile.d/firstlogin.sh

if [ ! -r ~/.ssh/id_dsa ] ; then
        echo -n "Generating DSA keys..."
        ssh-keygen -t dsa -f ~/.ssh/id_dsa -N '' -C "`hostname -s` cluster internal key" -q
        if [ $? -eq 0 ] ; then
                echo -ne "\t\tdone\n"
                cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
                chmod go-rwx ~/.ssh -R
        else
                echo -ne "\t\tFAILED\n"
        fi
fi
```

```
#!/bin/csh
# /etc/profile.d/firstlogin.csh

if (! -r ~/.ssh/id_dsa ) then
        echo -n "Generating DSA keys..."
        ssh-keygen -t dsa -f ~/.ssh/id_dsa -N '' -C "`hostname -s` cluster internal key" -q
        if ( $? == 0 )  then
                echo -ne "\t\tdone\n"
                cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
                chmod go-rwx ~/.ssh -R
        else
                echo -ne "\t\tFAILED\n"
        endif
endif
```

Example configuration

Summary

- Sharing of resources is fundamental in HPC, policies provide a guaranteed quality of service and avoid problems such as
  - Non-optimal usage of resources
  - Starvation of resources especially for important or urgent jobs
  - Unauthorized users dominating resources or grossly exceed their resource quota e.g by sending large amount of jobs.