Environment Canada Environnement Canada

Canada

Linux and Other Clusters at Dorval

John Marshall
IB/SSS
2008-07-15
v1.1

Overview

- Cells/Clusters
- Storage
- Gridengine
- Monitoring

Cells/Clusters

- Terminology
  - Cluster – a collection of computers, typically homogeneous (all the same kind)
  - Cell – a collection of computers, not necessarily homogeneous; taken from Gridengine naming
- Cell identification
  - Fully qualified host name of the head node
    - e.g., dorval-ib.cmc.ec.gc.ca, maia.cmc.ec.gc.ca
  - All cells/clusters may be identified in this way

Proper Usage (1)

- Frontend/head nodes
  - All cells have head nodes
    - Usually 2: master, backup
  - For
    - Interactive use (you may log on)
    - Data transfers
    - Minor testing
  - Not for
    - Running servers (e.g., apache)
    - Large scale testing (e.g., running a model)
    - Cron jobs

Proper Usage (2)

- Backend/compute nodes
  - For
    - Batch jobs
  - Not for
    - Interactive use (do not log on)
    - Running servers (e.g., apache)
    - Large data transfers
      - Offload to frontend
      - Offload to transfer queue (e.g., dev.s.ib-xfer)
  - Do not use more resources than you were allocated
    - E.g., Do not run 8 concurrent processes having asked for 4 slots
  - Exclusive use are special case (explained later)
    - If you have been allocated the whole node, you may use it as you wish

IBM Supercomputers

- Clusters – all machines are the same
- Cell names:
  - maia.cmc.ec.gc.ca
  - naos.cmc.ec.gc.ca
  - saiph.cmc.ec.gc.ca (available this Fall)
- Purpose: Operational and development use
- Run AIX
- Queuing system: LoadLeveler
- Dedicated, high-speed networking
- Dedicated, high-speed, high-capacity storage

Linux Cells

- Running Debian Linux (Etch) as of 2008-05-08
- Cell names:
  - dorval-ib.cmc.ec.gc.ca
  - alef.cmc.ec.gc.ca
  - beth.cmc.ec.gc.ca (future)
- Queuing system: Gridengine 6.x
  - Computing/backend nodes **not** for interactive use!
- Networking
  - Outgoing access to regular network
  - Incoming access via head nodes only
  - Private networks
    - 1GigE
    - High-speed network (e.g., Infiniband or 10GigE)

dorval-ib.cmc.ec.gc.ca (1)

- Purpose: Operational and development use
- Machines
  - Frontends: 2x Dell 2850, 2x dual-core, 3.4GHz, 4GB
  - Backends
    - Old: 40x Dell 1850, dual-core, 3.0GHz, 4GB
    - New: 40x Dell 1950, 2x quad-core, 2.33GHz, 16GB, low-power!
- High speed network
  - 3x 24-port Infiniband switches (4x)
- Availability
  - Currently operational

alef.cmc.ec.gc.ca

- Purpose: Development use. Operational backup.
- Machines:
  - Frontends: 2x Dell 2950, 2x dual-core, 3.2GHz, 12GB
  - Backends
    - 58x Dell 1850, 2x dual-core, 3.2GHz, 4GB
    - 10x Dell 1950, 2x quad-core, 1.86GHz, 16GB, 64-bit
    - 11x Sunfire 4400, 2x dual-core (AMD Opteron), 2.6GHz, 3.5GB
- High speed network
  - 1x Mellanox 132 (max 144) port Infiniband switch (4x)
- Storage: Details later
- Availability: For testing

beth.cmc.ec.gc.ca

- Purpose: Development use.
- Machines:
  - Frontends: 2x Dell 1950, 2x quad-core, 2.6GHz, 16 GB, low power
  - Backends:
    - TBDx Dell 1950, 2x quad-core, 2.6GHz, 16 GB, low power
- High speed network (planned)
  - 1x Mellanox 132 (max 144) port Infiniband switch (4x)
- Availability: Date not set

dor-ib.cmc.ec.gc.ca

- Purpose: Development use.
- Machines:
  - Frontends: 2x Dell 1950, 2x quad-core, 1.86GHz, 16 GB
  - Backends: 8x Dell 1950, 2x quad-core, 1.86GHz, 16 GB
- Batch system: Gridengine ?
- Network
  - 1x Cisco 24-port Infiniband switch (4x)
  - GigE interconnects
    - No NATting to access non-private network
- Availability: Not yet

Mixed

- Running IRIX and Debian Linux (Etch)
- Cell names:
  - sgemaster.cmc.ec.gc.ca (deprecated)
  - dorval-dev.cmc.ec.gc.ca
  - dorval-ops.cmc.ec.gc.ca
- Queuing system
  - sgemaster.cmc.ec.gc.ca: Gridengine 5.3
  - Others: Gridengine 6.x
- Networking: Standard

sgemaster.cmc.ec.gc.ca (gfxcl0[12])

- Purpose: Operational use.
- Machines:
  - Frontends: 2x Dell 2850, 2 cpu, 1.266GHz, 2GB
  - Compute:
    - Operational and development machines: castor, pollux, asep-[12], faucon, gfxcl*
- Availability:
  - Development: Until 2008-07-14!
  - Operations: Until transition to dorval-ops and dorval-dev is completed

dorval-dev.cmc.ec.gc.ca

- Purpose: Development use
- Machines:
  - Frontends: 2x Dell 2850, 2x dual-core, 3.4GHz, 4GB (3GB used)
  - Compute:
    - Development type machines: pollux, desktops (2nd, 4th, and 5th floor)
- Queuing system
  - Takes of development queues from sgemaster
  - Desktop queues:
    - Tailored in conjunction with user rep
- Availability: Now

dorval-ops.cmc.ec.gc.ca

- Purpose: Operational use
- Machines:
  - Frontends: 2x Dell 2850, 2x dual-core, 3.4GHz, 4GB (3GB used)
  - Compute:
    - Operational type machines: castor, asep-[12], faucon
- Queuing system
  - Takes of operational queues from sgemaster
- Availability:
  - Now

Storage

- Commonly at standard locations
  - `/home`
  - `/data`
- Allocations
  - Based on “official” storage allocations
  - User reps may ask for space
    - Capacity
    - Performance
    - Accessbility

Netapp, EMC, Bobcat

- Netapp (netapp/vesta) servers
  - NFS
  - Typically at `/home` and `/data`
  - Availability: Now
- EMC
  - Raw devices over FC
  - Availability: Now
- Bobat
  - NFS
  - `/data`
  - Availability: Now

Rapidscale/Terrascale “Bricks”

- Purpose: Provide high performance, high capacity storage to dorval-ib Linux cluster
- High performance
  - Runs over IP, IPoIB, SDP
- High capacity
  - 10x ~4TB
- Accessibility
  - Local to dorval-ib only!
  - Under `/fs`
- Allocations: Some have already been made
- Availability: Now

Cava (1)

- Purpose: Provide high-perfomance, high capacity storage to all Linux clusters *and* accessibility across the site network
- High performance
  - Direct attach to Infiniband-based clusters
  - Direct attach to 10GigE network
    - Therefore, accessible to regular network
  - Multiple storage servers
- High capacity: 10s of TBs

Cava (2)

- Accessibility
  - Linux machines
    - glusterfs
    - NFS
  - General: NFS
  - Typically under `/data`
- Allocations: None yet
- Availability: Summer/Fall

Gridengine

- Gridengine – A batch queuing system for managing resources and running jobs across a distributed computing environment
  - Aka: SGE, Sun Grid Engine, GE, Grid Engine
  - Allows users to run jobs in a cooperative way
- Current deployment:
  - 5.3 on sgemaster.cmc.ec.gc.ca only (deprecated)
  - 6.1 on all other cells
- Naming:
  - Changes because of transition from 5.3 to 6.1
  - Attempt to bring uniformity across all gridengine cells

Queues – Review

- Gridengine manages resources using queues
- A queue is associated with zero or more hosts
- Each (queue,host) pair is assigned slots,
- A slot is allocated to run a job
  - Serial job: allocated 1 slot (1 process)
  - Parallel job: allocated >=2 slot (multiple processes)

- Do not ask for 1 slot if your job will run multiple processes *concurrently*

Queues – Names

- Format
  - <useClass>.<type>.<specifier>[.<mach>]
  - useClass
    - dev – development
    - prod – production
  - Type
    - s – shared resource usage
    - e – exclusive resource usage
- Examples
  - prod.s.ib – production, shared, on dorval-ib
  - dev.s.pollux-hi – development, shared, on pollux, hi priority jobs
  - dev.e.alef.x86-8-1860 – development, exclusive, alef cluster, 8-core nodes with x86 arch running at 1.86GHz

Parallel Environments – Review

- Gridengine schedules parallel jobs using PEs
- A PE is associated with zero or more queues
- A PE is used to configure the job environment to run parallel jobs
  - I.e., to generate a list of nodes on which to run
- A PE determines how slots are allocated
  - From the same node
  - From different nodes

Parallel Environments – Names

- Format
  - <useClass>.<type>.<specifier>[.<mach>]
  - Generally follow that of queue names
- Examples
  - `dev.e.alef.x86-8-1860` – development, exclusive, alef cluster, 8-core x86 nodes running at 1.86GHz
  - `dev.s.pollux` – development, shared, helper PE for multiple pollux queues `dev.s.pollux-hi`, `dev.s.pollux-med0`, `dev.s.pollux-med1`, `dev.s.pollux-lo`

Using Queues and PEs (1)

- When to use a queue?
  - To run a serial job in a specific queue or from a list of queues
    - E.g.,
      ```
      #$ -q dev.s.pollux-hi,dev.s.pollux-med0
      ```
- When to use a PE?
  - To run a parallel job (naturally)
    - E.g.,
      ```
      #$ -pe dev.e.ib.x86-8-1860 8
      ```
  - To run a serial job using a PE as a *helper*
    - I.e., Some PEs have been created to facilitate requesting that a job run in a queue selected from a list of queues
    - E.g.,
      ```
      #$ -pe dev.s.pollux 1
      ```

Using Queues and PEs (2)

- What if I specify a queue and a PE?
  - The queue specification will limit which queues a job may be run in
  - The following will only consider `dev.s.pollux-hi`, even if the `dev.s.pollux` PE is configured to select from other queues

```
#$ -q dev.s.pollux-hi
#$ -pe dev.s.pollux 1
```

- How many slots may I request?
  - The larger the number the less likely your job will be scheduled
  - Request multiples of what the PE supports
    - Currently, this info can be determined by the PE name
    - E.g., `dev.e.x86-8-2330` supports multiples of 8 slots

Using Queues and PEs (3)

- How do I choose a PE?
  - *Exclusive* PEs have been set up to enforce using nodes of the same type
    - Helps syncronization of parallel job running across nodes
- Can I specify more than one PE?
  - The PE specification may include the * wildcard, but only one PE will ultimately be chosen
  - E.g.,
    ```
    #$ -pe dev.e.x86-8-*
    ```

Using Queues and PEs (3)

- Best practice?
  - Parallel jobs must use PEs, whether OpenMP or MPI
  - The helper PEs (e.g., `dev.s.pollux`) are preferable because they simplify queue selection

Limits

- Jobs exceeding limits are automatically killed
- May be specified with job script
  - Helps select suitable queues
  - May improve your chances to run earlier
- Cpu time (h_cpu)
  - Usually applies to `dev.s.*` queues
- Wallclock time (h_rt)
  - Usually applies to `dev.e.*` queues
- Memory (h_vmem)
  - Applies to all queues

dorval-dev.cmc.ec.gc.ca Queues

- Desktop
  - dev.s.desktop
- General use for pollux
  - dev.s.pollux-hi
  - dev.s.pollux-lo
  - dev.s.pollux-med0
  - dev.s.pollux-med1
- Data transfer for pollux
  - dev.s.pollux-xfer
- Other
  - dev.s.uniord

dorval-dev.cmc.ec.gc.ca PEs

- General use for pollux
  - dev.s.pollux
    - Schedules to dev.s.pollux-{hi,lo,med0,med1} queues
- Other
  - dev.s.uniord
    - Schedules to dev.s.uniord

dorval-ib.cmc.ec.gc.ca Queues

- General use for ib
  - dev.s.ib
- General use for ib (to replace dev.s.ib)
  - dev.s.ib-hi
  - dev.s.ib-lo
  - dev.s.ib-med0
  - dev.s.ib-med1
- Data transfer for ib
  - dev.s.ib-xfer

dorval-ib.cmc.ec.gc.ca PEs

- Information will be on web site
  - http://cmiss.cmc.ec.gc.ca/computing_cells-clusters

Submitting a Job (1)

- Multiple ways of submitting a job
  - `soumet` – job submission wrapper from RPN
  - `ocsub_unify` – job submission wrapper from Operations
  - Native `qsub` from Gridengine (equivalent to `llsubmit` for LoadLeveler)

Submitting a Job (2)

- Multiple ways of submitting a job (cont'd)
  - `gmjob` wrapper from SSS (part of gmtools package)
    - Works across Gridengine and LoadLeveler cells
    - *Delivers* the appropriate command and information to the target cell
      - Does not translate job script directives
      - Does not provide equivalents for every native option
    - Takes fully qualified cell name (aka IP alias hostname of frontends/head nodes)
      - E.g., `gmjob dorval-ib.cmc.ec.gc.ca submit myjob`
    - `gmjob` is used by `soumet` and `ocsub_unify` to eliminate calls to rsh/ssh
    - Requires access to cell frontend/head node from submitting machine
    - Requires proper setup of ssh keys, `known_hosts`, and `config`

Submitting a Job (3)

- Best practice?
  - `soumet` and `obsub_unify` may be preferable
    - They provide a managed layer between the queuing system and the user which can hide certain complications
    - They may integrate with your run environment
  - `gmjob` is almost certainly preferable over the native tools because it manages the task of getting the job request to the destination in a uniform way
    - Make sure your ssh “stuff” is set up!!!

Running a Serial Job

- Job script (iam.job)

```
#! /bin/sh
#$ -pe dev.s.pollux 1
echo "my name is `whoami`"
```

- Submission

```
gmjob dorval-dev.cmc.ec.gc.ca submit iam.job
```

Running a Parallel Job

- Job script (hellompi.job)

```
#! /bin/sh
#$ -pe dev.e.ib 4
echo "Got $NSLOTS slots."
echo "Machines file:"
cat $TMPDIR/machines
export P4_RSHCOMMAND=${TMPDIR}/rsh
mpirun -v -np $NSLOTS \
    -machinefile $TMPDIR/machines \
    ~/hellompi
```

- Submission (requires a hellompi binary)

```
gmjob dorval-ib.cmc.ec.gc.ca submit hellompi.job
```

Monitoring

- Queuing system specific tools
  - `qstat` – Gridengine
  - `llq` – Loadleveler
- Other tools
  - Cellmon
  - Gridmon

Cellmon

- Monitors job, queue, and host information for a single cell
  - Gridengine and LoadLeveler
- Web-based
  - Accessible via browser or command line (`wget`, `curl`)
- Supports querying/filtering using URI
- Accessible from anywhere
- Starting point for users
  - `http://cellmon.cmc.ec.gc.ca/`

Gridmon

- Monitors job, queue, and host information for multiple cells
- Contacts cellmon for data
- Personal web application
  - Runs on user machine
  - Display in browser
- Allows control of jobs, queues, hosts
  - E.g., Delete job
- Starting point for users
  - http://gridmon.cmc.ec.gc.ca/
  - Provides help your own gridmon up and running

References

- Main web page for computing cells/clusters (apart from IBM clusters)
  - http://cmiss.cmc.ec.gc.ca/computing_cells-clusters/
- Starting points:
  - http://cellmon.cmc.ec.gc.ca/
  - http://gridmon.cmc.ec.gc.ca/