The CNMS Computer Cluster

This page describes the CNMS Computational Cluster, how to access it, and how to use it.

- (**16 August 2010**) N.B. The latest block of the CNMS Computer Cluster is still undergoing debugging and improvements to its configuration. Although the hardware is fixed, the software configuration is likely to change. In particular, final decisions on the queue configuration have not been taken.
- (**2008**) Previous blocks of CNMS Cluster
- (**2006**) First block of the CNMS cluster

Introduction

The computer cluster is a "Beowulf" style machine purchased in 2010 for the CNMS. It is provided with a light level of support. It is NOT provided with the level of support of a user facility like NERSC or NCCS.

Please take care to

1. make suggestions and report problems – you might be the first person to spot an issue
2. not to abuse the resource: check with your CNMS contact if you need e.g. to run an extraordinary large number or size of jobs.

Hardware

The CNMS Computer Cluster is actually two racks of the 2010 additions to the Oak Ridge Institutional Cluster, two racks of the 2008 additions, and 1 rack of the original units (http://oic.ornl.gov). The software and hardware of the 2010 additions are configured very similarly to earlier racks. Experience, codes, and scripts should carry over. The CNMS units are the "2008 and 2010 CNMS" units.

In summary the 2010 units are an Intel/Linux/PBS/MPI/Infiniband system with 336 processor cores total, 3GB memory/core, delivering nearly 3 million CPU hours per year. CNMS has two of these units.

The 2008 blocks are two units consisting of XE310 boxes (3 Ghz) from SGI that contain one login node, one storage node and 28 compute nodes within 14 chassis. Each node has eight cores. There are 16GB of memory per node. The login and storage nodes are XE240 boxes from SGI; Two units consisting of XE240 SGI nodes and contain one login node, one storage node and 20 compute nodes within 20 separate chassis. Each node has 8 cores. There are 16GB of memory per node. These nodes contain larger node-local scratch space and a much higher I/O to this scratch space because the scratch space is a volume from 4 disks.

The 2006 block is one unit of the original blocks that consist of a bladed architecture from Ciara Technologies called VXRACK. Each VXRACK contains two login nodes, three storage nodes, and 80 compute nodes. Each compute node has Dual Intel 3.4GHz Xeon EM64T processors, 4GB of memory and dual Gigabit Ethernet Interconnects.

Compute nodes

2010: 2x42 nodes each with two (2) Quad Core 2.26 GHz Intel Xeon E5520 (Nahelem-EP) processors with 24GB 1066MHz DDR3 memory per node and 500GB of local disk:

2008 2x48 nodes each with two (2) Quad Core 3.0 GHz Xeon processors;

2006 1x80 nodes each with Dual Core 3.4 Ghz Xeon EM64T processors.

The interconnect is quad data rate (4X) Infiniband for 2010 and 2008 units and gigabit for the 2006 unit.

Head node

2010: Two (2) Quad Core 2.26 GHz Intel Xeon E5520 (Nahelem-EP) processors with 24GB 1066MHz DDR3 memory and redundant power supplies.

2008: Two (2) Quad Core 3.0 GHz Intel Xeon XE340 processors and Two (2) Quad Core 3.0 GHz Intel Xeon XE240 processors

2006: Dual core 3.4 Ghz Xeon EM64T processors.

Access

The machine is “behind” the ORNL firewall. A SecureID token and OIC account are required for access.

You must have an **active CNMS user project or be a CNMS staff**. If you have a project or are staff and do not have ORNL computer access, write Erica Lohman (lohmanem@ornl.gov) and provide the CNMS user project number and PI and ask for access to the CNMS computer cluster.

Obtaining Help

Account problems should be sent to the main ORNL helpline, help@ornl.gov. Be sure to clearly state that you are referring to the OIC and how to reproduce the problem.

For simulation software issues the OIC administrators might be able to help if the problem seems hardware related. Contacting other CNMS staff members or OIC users is likely to be more productive for more general issues.

Logging in

If you are attempting access from outside ORNL, first use VPN or login via the gateway machine login1.ornl.gov. You need a SecurID token for this step.

From inside ORNL

```
ssh myuserid@ccsd2.oic.ornl.gov
```

Use your UCAMS password.

Using the head nodes

The head nodes are intended for compiling and submitting jobs. Do not run long simulations or analysis jobs on the head nodes. These should be submitted as PBS jobs.

Software

The CNMS unit shares software with the other OIC units. See http://oic.ornl.gov
Most significant software packages (compilers, mpi, some applications) have been configured via the modules environment. The environment is also installed e.g. at NERSC and NCCS. module list shows currently loaded modules module avail shows available modules module load abc loads the abc module module unload abc unload (removes) the abc module These commands can be added to shell startup files if you wish.

Recommended software

Paul Kent recommends using the following setup unless you have technical reasons to use otherwise. Be sure to “unload” other loaded mpi and compiler modules. Many older compiler and mpi versions are also available, but understand that they are more buggy and help will be harder to obtain if you are doing something exotic.

Please use the latest Intel C++ and Fortran Compilers

```
module load icce/11.1
```

Notice that this includes the fortran compiler

Please use the latest OpenMPI

```
module load mpi/openmpi-1.3.2-intel
```

If you have used the OIC before and changed the defaults, you might also need to “switcher mpi = openmpi-1.3.2-intel”

If you choose to use the gnu compilers you will need to use e.g. mpi/openmpi-1.3.2-gcc4

Please use the Intel MKL BLAS/LAPACK

Linked by specifying ‘-mkl’ on the compile/link line using the v11 Intel compilers. (See below for more options including Scalapack). The actual libraries are under /opt/intel/Compiler /11.1/072/mkl/ . Unfortunately the link lines can be quite complex if you opt to manually link...

By default, the MKL library uses threads. For single core jobs, this setting can improve performance considerably. However, If you are using all the cores on a node for MPI tasks, this risks actually reducing performance. Set OMP_NUM_THREADS=1 to avoid this.

Running jobs

Jobs are submitted and monitored via standard PBS queueing commands: qsub, qstat, qdel, qhold, qrls.

Abuse

Respect other users. We aim for a permissive configuration of this cluster so that the resource can be maximally and flexibly utilized. If you see problems please let the computational thrust leaders know.

Note that the main userspace filesystem is mounted via NFS. Large amounts of I/O will flood the network slowing access for everyone. Instead write to local scratch $PBS_SCRATCH and copy large files back, or change/reconfigure your software to write infrequently. This scratch space is not shared between nodes.

(October 2010) The 2010 units feature a shared/global scratch filespace for each job. This means that files written in this space can be read by all nodes with high performance. Use $GLOBAL_SCRATCH within your job for access. These directories are deleted at the end of each job, so be sure to copy back the files you need. While a job is running, you can access these directories on the head nodes via /global_scratch/your_job_id/…

A very good way to end up the black books of the administrators and other users is to run a quantum chemistry code (e.g. NWCHEM) and write a large integrals file to your homespace. Please don’t do this.

Sample job

The job below requests 2 hours on 16 cores

```
#PBS -N TESTJOB
#PBS -j oe
#PBS -M my_email@somewhere.inter.net
#PBS -m abe
#PBS -q cnms08fq
#PBS -l walltime=2:00:00,nodes=2:ppn=8

 NCORES=16

EXEC=./supercapacitor_simulation

# In case of unresolved symbols, good candidates are: wrong mpi module loaded &/or missing LD_LIBRARY_PATH
to
# compiler runtime libraries or Intel MKL library. Check your shell startup files or e.g.
# export LD_LIBRARY_PATH=/opt/intel/Compiler/11.1/072/mkl/lib/em64t/:${LD_LIBRARY_PATH}
#echo $LD_LIBRARY_PATH

cd $PBS_O_WORKDIR
```

```
mpirun -v --mca mpi_leave_pinned 1 --mca mpool_base_use_mem_hooks 1 --mca bt1 openib,self -np ${NCORES}
${EXEC}
```

Queue structure

Queue name	Core count limit	Node limit	Run limit	Time limit	Unit	Global scratch	Local scratch (not shared between nodes)
cnms10fq	8	2	3	72 hours	2010	$GLOBAL_SCRATCH	
cnms10tq	None	None	2	None	2010	$GLOBAL_SCRATCH	
cnms08fq	8	1	3	72 hours	2008	None	$PBS_SCRATCH
cnms08tq	None	None	2	None	2008	None	$PBS_SCRATCH
cnmsq	None	None	2	None	2006	None	$PBS_SCRATCH

The queue configuration will be adjusted based on experience and feedback.

Specifying the number of cpus, nodes

Please choose a processor core count appropriate for your calculation.

The quantum of allocation is a single node. Each node has eight cores. Consequently a single core job locks up the resources of an eight core job. Even a barely parallel job will make better use of resources than a serial run. Also note that each 2010 node has 24GB memory, more than machines at NERSC or NCCS. "Large" jobs might fit on fewer cores than on other machines.

To check the parallelization efficiency of your simulation on the cluster you will have to run benchmarks. Efficiencies should be reasonable, but not as good as e.g. a Cray XT that has a better interconnect and tuned software stack.

For jobs requiring 8 or fewer cores, vary the ppn (processors per node) request:

e.g. a four core run

```
#PBS –l nodes=1:ppn=4
mpirun –np 4 ./a.out
```

For jobs requiring multiples of 8 cores:

e.g. a sixty-four core run

```
#PBS –l nodes=8:ppn=8

mpirun –np 64 ./a.out
```

Submitting jobs

```
qsub myjob.job
```

Monitoring jobs

```
qstat
```

Note that the default queue status report includes every job on every queue on the cluster. To see only your jobs

```
qstat –u $USER
```

Deleting jobs

```
qdel –W 0 jobnumber
```

Queue dependencies

Standard PBS job dependencies are supported enabling very long running simulations or automatic running of analysis:

```
qsub first.job
12345.b15l01.oic.ornl.gov
qsub –W depend=afterok:12345 second.job
12346.b15l01.oic.ornl.gov
qsub –W depend=afterok:12346 third.job
```

The second job will only start after the first one has completed with no errors. Similarly the third will only start after the second has completed with no errors. By using a (very) little scripting VASP, LAMMPS, NAMD etc. jobs can be automatically restarted and continued for long simulation times. **Thus there is no effective limit on the length of simulation that can be run.**

Building codes

The OpenMPI library provides wrappers for the C, C++, and Fortran compilers. If you are building a standard MPI application, look for an Intel/MPI configuration.

The following should compile and link, producing a hello_world executable

```
cat >hello_world.f <<EOF
 program hello
   include 'mpif.h'
   integer rank, size, ierror, tag, status(MPI_STATUS_SIZE)
   call MPI_INIT(ierror)
   call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierror)
   call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierror)
   print*, 'node', rank, ': Hello world'
   call MPI_FINALIZE(ierror)
 end

EOF
mpif90 –o hello_world hello_world.f
```

Numerical libraries

See the available modules

The latest (v11) Intel compilers can link agains the Intel Math Kernel Library (MKL) by specifying the '–mkl' on the link line

```
mpif90 –mkl –o numerical.exe numerical.f90
```

The libraries can be directly linked from

```
MKLPATH=/opt/intel/Compiler/11.1/072/mkl/lib/em64t
```

To link the parallel Scalapack library in combination with OpenMPI use

```
-L$(MKLPATH) -lmkl_scalapack_lp64 -lmkl_blacs_openmpi_lp64
```

Applications

LAMMPS

NWChem

VASP

Paul Kent has compiled versions of 4.6 and 5.2.8 for the OIC unit using the latest compilers, MKL, the MKL FFTW, and MKL scalapack. Access is only provided to authorized license holders of vasp.

History

16 August 2010 -Initial version by Paul Kent
21 October 2010 – Updated by Bobby Sumpter and Paul Kent