EMSL

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

U.S. DEPARTMENT OF
ENERGY

Users' Introduction to EMSL's Cascade Supercomputer

Amity Andersen, Ph.D.

Scientist, MSC Consulting, Interfacial Sciences & Simulation

September 22, 2017

EMSL’s Cascade Supercomputer

- 1440 nodes x 16 conventional Intel Xeon processor cores per node = 23,040 processor cores
- Each node has 2 Intel Xeon Phi (a.k.a. MIC) coprocessors
  - 60 cores per coprocessor, 4 threads per core
  - 1440 nodes x 2 coprocessors x 60 coprocessor cores = 172,800 coprocessor cores
- Total machine cores: 195,840
- 128 GB memory per compute node
- FDR Infiniband network
- 2.7 petabyte shared parallel file system
- Linux operating system
- Recently added: 8-node Intel Xeon Phi Knight’s Landing (KNL) partition
  - Coprocessor is main processing unit (1 coprocessor per node)
  - 68 cores per coprocessor, 4 threads per core

Getting Access: Becoming a User

- Check out proposal opportunities at:
  - https://www.emsl.pnl.gov/emslweb/proposal-opportunities
- Submit proposals through User Portal:
  - https://eusi.emsl.pnl.gov/Portal/
- For questions, contact EMSL User Support Office:
  - emsl@pnnl.gov

Available Installed Science Applications

- Quantum Chemistry
  - NWChem†
  - VASP*
  - ADF*
  - Molpro
  - GAMESS
  - CP2K
  - Quantum Espresso
  - CPMD
  - Siesta
- Electron Microscopy Image Processing
  - Relion
  - EMAN
- Classical Molecular Dynamics
  - AMBER
  - GROMACS
  - NAMD
  - LAMMPS†
- Biology/Bioinformatics
  - DIAMOND
  - ScalaBLAST†
  - USEARCH
  - Biocellion
- Environmental/Hydrology
  - PFLOTRAN

†Can utilize Intel Xeon Phi
*Requires user license

Supporting Software

- Interpretive languages: Python, NCL
- Data analysis: GNUPLOT, Python, Matlab*, NCL
- Data formats: NetCDF, HDF5, H5hut
- Data visualization: VisIt, H5hut, NCL, Python, Matlab*, GNUPLOT
- Math packages: Python, Matlab*

*Requires user license

Compiling an Application: Available compilers, MPI, math libraries, tools

- Compilers:
  - Intel (v 13.0.1, 14.0, 14.0.3, 15.0.090, 16.1.150, ips_16_u3, ips_17_u4, ips_17, ips_18)
    - `icc` - C compiler
    - `icpc` - C++ compiler
    - `ifort` - FORTRAN compiler
  - gcc (v 4.4.7, 5.2.0, 6.3.0)
    - `gcc` – C compiler
    - `g++` - C++ compiler
    - `gfortran` - FORTRAN compiler
  - NAG (v 5.3.1)
- Message Passing Interface (MPI)
  - IntelMPI (v 4.1.0.024, 4.1.1.036, 4.1.2.040, 5.1.2.150)
  - MVAPICH2(v 1.9, 2.3a_mt, 2.3a)
  - OpenMPI (v 1.6.5, 1.10.2)
  - When loaded, the follow compiler wrappers are available:
    - `mpicc`, `mpiicc` - C compiler wrapper
    - `mpif90` - FORTRAN compiler wrapper
    - `mpic++`, `mpicxx`, `mpiicpc` - C++ compiler wrappers
    - `-show` (e.g., '`mpicc -show`') shows details of the wrapper
- Math libraries
  - MKL [BLAS, LAPACK, FFT, SCALAPACK] (v 13.0.1, 14.0, 14.0.3, 15.0.090, 16.1.150)
  - PETSc (sparse linear algebra)
- Development Tools
  - `gdb`

Compiling an Application: Modules

- ‘`module list`’ to view current loaded modules
- ‘`module purge`’ to purge loaded modules
- ‘`module load`’ to load a module
- ‘`module avail`’ to list available modules
- Example:
  - `module purge`
  - `module load intel`
  - `module load impi`
  - `module load mkl`

  Loads default Intel compiler (icc for C, ifort for Fortran, default IntelMPI, and default MKL (lib paths will be included in **LD_LIBRARY_PATH**)
- **Loading order is important**

File Systems: Where to keep your data

- `/home`
  - Backed up
  - Every user has their own directory
- `/dtemp`
  - Lustre global shared temporary directory
  - Not backed up
  - Currently, every user has their own directory
  - Moving toward user project directories where users on the project will have access
- `/scratch`
  - Local scratch per node
  - Purged at the end of a job
- `/archive`
  - Aurora archive for long-term data storage
  - Storage requirement needs to be requested

Submitting Jobs to the Queue: Introduction

- Formerly MOAB/SLURM, now just SLURM
  - '**msub**' command can still be used to launch jobs
  - '**sbatch**' SLURM command for launching jobs
- For popular applications, interactive submission commands available:
  - **submit_nwchem**
  - **submit_vasp**
  - **submit_adf**
  - **submit_molpro**
  - **submit_gamess**
  - **submit_amber**
  - **submit_lammps**
  - **submit_namd**
  - **submit_cp2k**

  Specific versions, modules of the above available: **/home/scicons/cascade/bin**

Submitting Jobs to the Queue: Creating a Submission Script

- Top of the script:
  - Legacy MOAB msub directives:

```
#!/bin/csh -f
#MSUB -l nodes=16:ppn=16,walltime=48:00:00
#MSUB -A <ACCOUNT_NUMBER>
#MSUB -o <JOBNAME>.output.%j
#MSUB -e <JOBNAME>.err.%j
#MSUB -N <JOBNAME>
#MSUB -V
#MSUB -m ea
#MSUB -M <EMAIL_ADDRESS>
```

  - SLURM sbatch directives:

```
#!/bin/csh -f
#SBATCH -A <ACCOUNT_NUMBER>
#SBATCH -N 16
#SBATCH -ntasks-per-node=16
#SBATCH -t 48:00:00
#SBATCH -J <JOBNAME>
#SBATCH -o <JOBNAME>.output.%j
#SBATCH -e <JOBNAME>.err.%j
#SBATCH --mail-user=<EMAIL_ADDRESS>
#SBATCH --mail-type=END
```

Submitting Jobs to the Queue: Creating a Submission Script (continued)

- Echo some job information for debugging and refund purposes

```
echo "refund: UserID = <USERNAME>"
echo "refund: SLURM Job ID = ${SLURM_JOBID}"
echo "refund: Number of nodes          = 16"
echo "refund: Number of cores per node = 16"
echo "refund: Number of cores          = 256"
echo "refund: Amount of time requested = 48:00"
echo "refund: Directory = ${PWD}"
echo " "
echo Processor list
echo " "
echo "${SLURM_JOB_NODELIST}"
echo " "
```

Submitting Jobs to the Queue: Creating a Submission Script (continued)

- Set up the runtime environment

```
source /etc/profile.d/modules.csh
module purge
source/msc/apps/compilers/intel/15.0.090/composer_xe_2015.0.090/bin/compilervars
.csh intel64
source /msc/apps/compilers/intel/impi/5.0.1.035/intel64/bin/mpivars.csh intel64

cd /scratch

setenv ARMCI_DEFAULT_SHMMAX 32768
setenv NWCHEM_BASIS_LIBRARY "/home/scicons/cascade/apps/nwchem-
6.6//src/basis/libraries/"
setenv NWCHEM_NWPW_LIBRARY "/home/scicons/cascade/apps/nwchem-
6.6//src/nwpw/libraryps/"
setenv ARMCI_OPENIB_DEVICE mlx4_0
setenv OFFLOAD_INIT on_offload
```

- Run the program

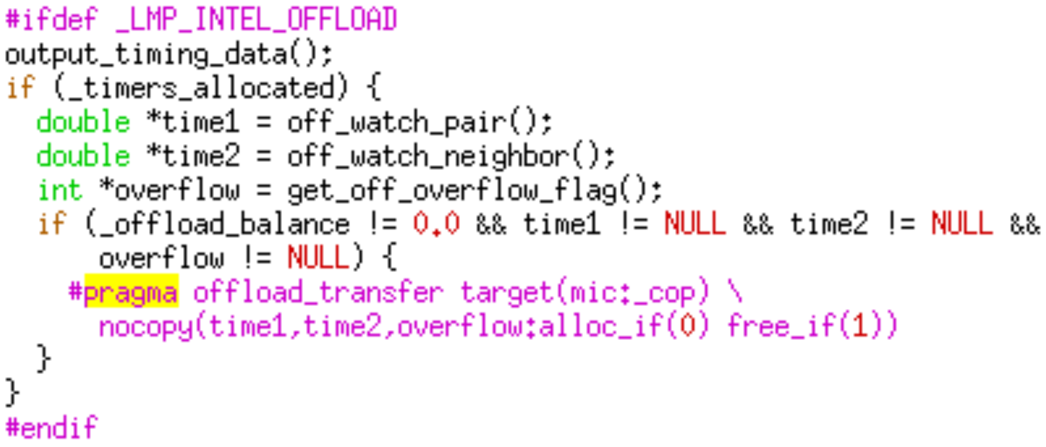
```
srun --mpi=pmi2 -n $SLURM_NPROCS -K1 /dtemp/scicons/bin/nwchem6.6 <JOBNAME>.nw
```

Some Useful Commands

- Checking computer time account balance and user usage:
  - **`gbalance`**
  - **`gusage`**
- Job status on the queue
  - **`showq`** or **`squeue`** for brief status of job(s) on queue
    - **`showq –u <USERNAME>`**
    - **`squeue –u <USERNAME>`**
  - **`scontrol`** to get detailed information about jobs
    - **`scontrol show job –v <JOBNUMBER>`**
  - **`canceljob`** or **`scancel`** to cancel jobs
  - **`backfill`** to see number of available nodes
- Opening an interactive session (X-windows required)
  - **`isub`**
    - '**`ssh –X`**' or '**`ssh –Y`**' needed for Cascade login
    - '**`isub –A <ACCOUNT_NUMBER> –N <number of nodes> -W <time limit HH:MM> -s <SHELL>`**'

Intel Xeon Phi Usage through LAMMPS Code

- USER_INTEL module (Mike Brown, Intel) must be compiled with LAMMPS (Sandia National Laboratory)
- LAMMPS can offload some tasks to MICs
  - Neighbor lists
  - Pairwise interactions
  - Long-range Coulombic interactions
  - Ghost atoms
- LAMMPS, written in C++, uses OpenMP-like #pragma blocks for offloading
  - Code example:



```
#ifdef _LMP_INTEL_OFFLOAD
output_timing_data();
if (_timers_allocated) {
  double *time1 = off_watch_pair();
  double *time2 = off_watch_neighbor();
  int *overflow = get_off_overflow_flag();
  if (_offload_balance != 0.0 && time1 != NULL && time2 != NULL &&
      overflow != NULL) {
    #pragma offload_transfer target(mic:_cop) \
      nocopy(time1,time2,overflow:alloc_if(0) free_if(1))
  }
}
#endif
```

Intel Xeon Phi Usage through LAMMPS Code: Example

- Molecular dynamics simulation
  - Water solvated Gb1 protein over goethite (100) mineral surface [Andersen et al., *Langmuir*, 32, 6194-6209 (2016)]
  - 74,923 atoms
  - AMBER (protein), SPC/E (water), ClayFF (goethite) force-fields
  - Particle-particle, particle-mesh (PPPM) method for long-range electrostatics
  - Time step 2 fs with SHAKE for H-bearing bonds
- Conventional usage (no MICs)
  - 8 node x 16 processor cores per node = 128 cores
  - 40 ns simulation time in <2 days
- Heterogeneous usage (with MICs)
  - MPI: 4 nodes x 8 cores per node
  - 2 OpenMP threads per core
  - 2 MICs per node
  - 40 ns simulation time in <2 days

	Time (s) to complete 10000 MD time steps
Conventional	71.43
1 MIC	77.80
2 MICs	65.57

Intel Xeon Phi usage through LAMMPS Code: Job submission

- Header information

```
#!/bin/csh -f
#MSUB -A <ACCOUNT_NUMBER>
#MSUB -l nodes=4:ppn=8,walltime=48:00:00
#MSUB -o solvatedgb1-goethite-100-1.log.%j
#MSUB -e solvatedgb1-goethite-100-1.err.%j
#MSUB -N lammps-phi-test
#MSUB -m ea
#MSUB -M amity.andersen@pnnl.gov
#MSUB -V
```

- Setting up the environment

```
source /etc/profile.d/modules.csh
module purge
module load python/2.7.9
module load intel/16.1.150
module load impi/5.1.2.150
module load mkl

setenv VORO_LIB /home/scicons/cascade/apps/lammps/voro++-0.4.6/src
setenv MEAM_LIB /home/scicons/cascade/apps/lammps/lammps-7Dec15/lib/meam
setenv REAX_LIB /home/scicons/cascade/apps/lammps/lammps-7Dec15/lib/reax
setenv POEMS_LIB /home/scicons/cascade/apps/lammps/lammps-7Dec15/lib/poems
setenv COLVARS_LIB /home/scicons/cascade/apps/lammps/lammps-7Dec15/lib/colvars
setenv LD_LIBRARY_PATH
${VORO_LIB}:${MEAM_LIB}:${REAX_LIB}:${POEMS_LIB}:${COLVARS_LIB}:${LD_LIBRARY_PATH}

limit stacksize unlimited

setenv OMP_NUM_THREADS 2
```

Intel Xeon Phi usage through LAMMPS Code: Job submission (continued)

- Running the program:

```
mpirun -n 32 ./lmp_cascade -sf hybrid intel omp -pk intel 2 omp 2 -pk omp 2 -in solvatedgb1-
goethite-100-1.inp > solvatedgb1-goethite-100-1.out.${SLURM_JOBID}
```

- Controlling MIC usage in the input file:

```
…
package intel 2 omp 2 mode mixed balance -1 ghost yes
…
```

- Output stats

```
----------------------------------------------------------
Using Intel Coprocessor with 4 threads per core, 59 threads per task
Precision: mixed
----------------------------------------------------------
------------------------------------------------
               Offload Timing Data
------------------------------------------------
  Data Pack/Cast Seconds    0.044084
  Host Neighbor Seconds     0.000887
  Host Pair Seconds         0.000000
  Offload Neighbor Seconds  0.025141
  Offload Pair Seconds      0.755232
  Offload Wait Seconds      1.061217
  Offload Latency Seconds   0.007667
  Offload Neighbor Balance  1.000000
  Offload Pair Balance      1.000000
  Offload Ghost Atoms       Yes
------------------------------------------------
```

Intel Xeon Phi Knight’s Landing (KNL) Node Partition

- After logging into Cascade, log into ‘ghep1’ login node
- KNL partition is msub or sbatch batch script must be specified:

```
#!/bin/csh -f
#MSUB -A <ACCOUNT_NUMBER>
#MSUB -l nodes=4:ppn=64,walltime=0:30:00
#MSUB -o <JOBNAME>.output.%j
#MSUB -e <JOBNAME>.err.%j
#MSUB -N <JOBNAME>
#MSUB -q knl (for sbatch, ‘#SBATCH -p knl’)
#MSUB -m ea
#MSUB <EMAIL_ADDRESS>
#MSUB -V
```

- For Intel compiler, MPI, and MKL, specify:

```
source /msc/apps/compilers/IPS_2017_U1/bin/compilervars.sh intel64
source /msc/apps/compilers/IPS_2017_U1/impi/2017.1.132/bin64/mpivars.sh intel64
```

- For some codes (e.g., LAMMPS’ LRT), specify KMP_AFFINITY:

```
setenv KMP_AFFINITY none
```

- Example LAMMPS mpirun launch:

```
mpirun -np 64 -ppn 16 ./lmp_knl -in in.intel.rhodo -log none -pk intel 0 omp 3
lrt yes -sf intel
```

Where to get help

- mscf-consulting@emsl.pnl.gov
- EMSL-MSC User Guide: http://www.emsl.pnl.gov/MSC/UserGuide/index.html
- NWChem Intel Xeon Phi support: http://www.nwchem-sw.org/index.php/Compiling_NWChem#How-to:_Intel_Xeon_Phi
- LAMMPS Intel Xeon Phi support: http://lammps.sandia.gov/doc/accelerate_intel.html

Our Consultants

Lee Ann McCue (Capability Lead)
Doug Baxter
Edo Apra
Kurt Glaesemann
Kevin Glass
Jun Li
Angela Norbeck
Seunghwa Kang
Amity Andersen

Questions?