SUPERCOMPUTING WALES
UWCHGYFRIFIADURA CYMRU

Guide to Running ***NWChem***

What is NWChem?

NWChem is a general-purpose computational chemistry code specifically designed to run on distributed memory parallel computers. The core functionality of the code focuses on Hartree–Fock theory, density functional theory methods for both plane-wave basis sets as well as Gaussian basis sets, tensor contraction engine-based coupled cluster capabilities, combined quantum mechanics/molecular mechanics descriptions and molecular dynamics. It was realized from the beginning that scalable implementations of these methods required a programming paradigm inherently different from what message-passing approaches could offer. In response, a global address space library, the Global Array toolkit, was developed.

The programming model it offers is based on using predominantly one-sided communication. This model underpins most of the functionality in NWChem, and the power of it is exemplified by the fact that the code scales to tens of thousands of processors. The core capabilities of NWChem are described in the paper “NWChem: scalable parallel computational chemistry”, as well as their implementation to achieve an efficient computational chemistry code with high parallel scalability (see References).

NWChem has been developed by the Molecular Sciences Software group of the Theory, Modeling & Simulation program of the Environmental Molecular Science Laboratory (EMSL) at the Pacific Northwest National Laboratory (PNNL). The early implementation was funded by the EMSL Construction Project.

Step 1 - Log in

The examples used in this guide are configured to run on the Cardiff Skylake *Hawk* cluster. Connect to *hawklogin.cf.ac.uk* with your Supercomputing user credentials using your preferred method (e.g. PuTTY from a Windows machine or ssh from any Linux terminal, thus

```
ssh –l your_username hawklogin.cf.ac.uk
```

The steps below involve typing commands (**in bold font**) in the terminal window.

Step 2 - Load a NWChem module

In common with most of the application guides in this series, we assume at the outset that the module of choice would be selected from those originally available on HPC Wales. Thus, a necessary first step would be to gain access to that module set by issuing the commands

```
module purge
module load hpcw
```

Note that the “module purge” command is required to prevent any unintentional collision with pre-existing modules. Subsequently issuing the module command

```
module avail
```

will provide visibility of and access to the entire set of HPCW modules.

A number of NWChem binary packages are available. Note that in common with most other software packages on the system, these are built with the Intel compiler.

- List pre-installed NWChem versions:
  ```
  module avail nwchem
  ```
- Load your preferred version (6.3-gpu is used in this guide):
  ```
  module load nwchem/6.3-gpu
  ```
- Confirm the loaded modules. Note that this is currently the latest version of the code available on Supercomputing Wales, and can be run with or without GPU support. The examples below are without accelerator support. All dependencies are handled automatically via the module file:
  ```
  module list
  ```

Step 3 - Create a directory

From your home directory, create a directory to hold the NWChem data:

```
cd ~
mkdir nwchem
```

Step 4 - Obtain a test case

A number of tests case are provided with the installation at:

```
/app/chemistry/nwchem/example
```

This directory contains the required NWChem input files, as well as SLURM job scripts. Copy the input file and job script for the aump2 example to your user space:

```
cd ~/nwchem
cp /app/chemistry/nwchem/example/aump2.nw .
cp /app/chemistry/nwchem/example/nwchem.aump2.SLURM.q .
```

The test case conducts an MP2 calculation, followed by a single point coupled cluster CCSD(T) calculation on the Au^+ ion using pseudopotentials.

Step 5 - Submit a parallel job

You are now ready to run this test case with the supplied job script *nwchem.aump2.SLURM.q.* The first thing to note is the somewhat modest usage of cores on each node – in this case just 16 compared to the 40 available. These are requested through lines 5 and 6 of the script:

```
#SBATCH --ntasks=64                # number of parallel processes (tasks)
#SBATCH --ntasks-per-node=16       # tasks to run per node
```

This is driven by the somewhat limited capabilities of the Global Array (GA) tools on high core count nodes – attempts to utilise more than 32 cores per node will almost certainly lead to an error condition. We expect to address this shortcoming in later builds of the code.

- From your working directory, submit the job using: `sbatch nwchem.aump2.SLURM.q`
- Check the job queue using: `squeue`

- The job should complete within a minute if it is not held in a queue. When completed, the job output will appear in your nwchem directory in the file *aump2.Hawk.n64.out.<JobID>* – the NWChem database (*Au+.db*), the CCSD T2 amplitude file (*Au+.t2*) and the vectors (*Au+.movecs*) will reside in */scratch/$USER/nwchem.<Job_ID>*, the directory created by the job, and may be copied back into your nwchem directory *(<Job_ID>* is the ID generated by the system). Note that many other files are generated and routed to this scratch directory.
- Compare your job output with the reference output file

  /app/chemistry/nwchem/example/aump2.Hawk.n64.out.14834

Step 6 - More Test Cases

Two additional test cases are provided in the */app/chemistry/nwchem/example* directory, both running zeolite fragments of increasing size –

- *nwchem.siosi6.SLURM.q* and the associated input file siosi6.nw
- *nwchem.siosi7.SLURM.q* and the input file siosi7.nw

Repeat the procedure outlined above for these two cases. Note again the restricted deployment of cores in both examples. The siosi6 DFT calculation should take under 1 minute on 64 cores, using 4 nodes with 16 cores per node.

The more extensive siosi7 calculation requires ca. 3 minutes on the same number of cores. Compare your results with the output files *siosi6.Hawk.n64.out.14835* and *siosi7.Hawk.n64.out.14836* provided in the */app/chemistry/nwchem/example* directory.

The impact of varying the node core count can be seen by comparing the two 128 core siosi7 output files, with that using 8 nodes (*siosi7.Hawk.n128.ppn=16.out.27247*,16 cores per node) taking under 3 minutes minutes compared to the 7 minutes when running with 4 nodes, and 32 cores per node (*siosi7.Hawk.n128.ppn=32.out.27250*).

More test cases are distributed with NWChem. They can be found at

```
/app/chemistry/nwchem/6.3/sb/intel-13.0/intel-4.1/examples
```

Adapt the job script above to run these cases.

To create a new case, refer to the section on Examples, Sample Inputs of the NWChem Manual, which is widely available online

http://www.nwchem-sw.org/index.php/Release65:NWChem_Documentation

A PDF version of the documentation is also available at: http://www.nwchem-sw.org/images/NWChem6.5_Documentation.pdf

References

- Official NWChem website: http://www.nwchem-sw.org/index.php/Main_Page
- "*NWChem: scalable parallel computational chemistry*", H.J.J. van Dam, W.A. de Jong, E. Bylaska, N. Govind, K. Kowalski, T.P. Straatsma and M. Valiev,

Computational Molecular Science, 1 (6) pages 888–894, November/December 2011, DOI: 10.1002/wcms.62

- NWChem User Documentation, http://www.nwchem-sw.org/index.php/Release65:NWChem_Documentation