GROMACS GPU Performance at HPC Wales

Introduction

Included as part of HPC Wales' hardware portfolio are two clusters containing NVidia GPUs. The clusters are comprised of 16 nodes, each with two 8-core Intel Xeon E5-2670 (Sandy Bridge) processors with 64GB memory and attached to a single NVidia Tesla M2090 GPU. It has been demonstrated in many different scientific application domains that GPUs may be used to accelerate embarrassingly parallel kernels to achieve impressive speedups.

One key area of interest for HPC Wales customers is computational chemistry, and GROMACS is one of the most popular packages in this area. GROMACS is a suite of executables for managing molecular dynamics simulations, and its developers have invested a considerable effort in parallelising the computations it performs using MPI, OpenMP and CUDA. In order to get the best performance out of GROMACS it may be necessary to utilise all three of these parallel options. In this report we set out to understand where enabling CUDA-GPU parallelism in GROMACS pays off and by how much, when running on the HPC Wales GPU clusters.

Setup

GROMACS 4.6.5 has been installed using the following cmake options:

```
cmake .. -DGMX_FFT_LIBRARY=mkl -DGMX_GPU=ON
-DCUDA_TOOLKIT_ROOT_DIR=/usr/local/cuda-5.5 -DGMX_MPI=ON -DGMX_DOUBLE=OFF
-DGMX_DEFAULT_SUFFIX=ON
-DCMAKE_INSTALL_PREFIX=/app/chemistry/gromacs/4.6.5/gpu/intel-12.1/intel-
4.1/mpi_s -DCMAKE_PREFIX_PATH=/app/SOURCES/chemistry/gromacs
```

This includes the Intel compiler 12.1, Intel's MPI 4.1 and FFT (from MKL 10.3) libraries and the CUDA toolkit 5.5. This build is available to HPC Wales users as the module gromacs/4.6.5-mpi_s-gpu. Note that the GPU version of GROMACS operates in single precision.

GROMACS GPU Acceleration

The details of how GROMACS makes use of GPUs are provided on this Web page: http://www.gromacs.org/GPU_acceleration. In short, the non-bonded force calculations are offloaded to one or more GPUs on every time-step, and the CPU is concurrently free to compute bonded forces and do lattice summation (PME), most likely multithreading this work with OpenMP. The work within a single node can be performed in parallel with other nodes, contributing to the same simulation, by using MPI communication.

Ewrop & Chymru: Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales: Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government

Thus the results below first assess performance within a single node (i.e. 1 MPI task) to see how well multithreading with OpenMP scales with and without GPU acceleration, and then we choose the best number of threads and see how well MPI parallelism scales with and without GPU acceleration.

Benchmarking Results

This report shows results for 3 test cases: ADH, alcohol dehydrogenase protein (taken from the Web page above) which contains 130K atoms, ion channel which contains 141K atoms, and lignocellulose which contains 3.3M atoms. In all results below we plot performance against parallel configuration, where performance is as given in the md.log output file, ns/day.

ADH

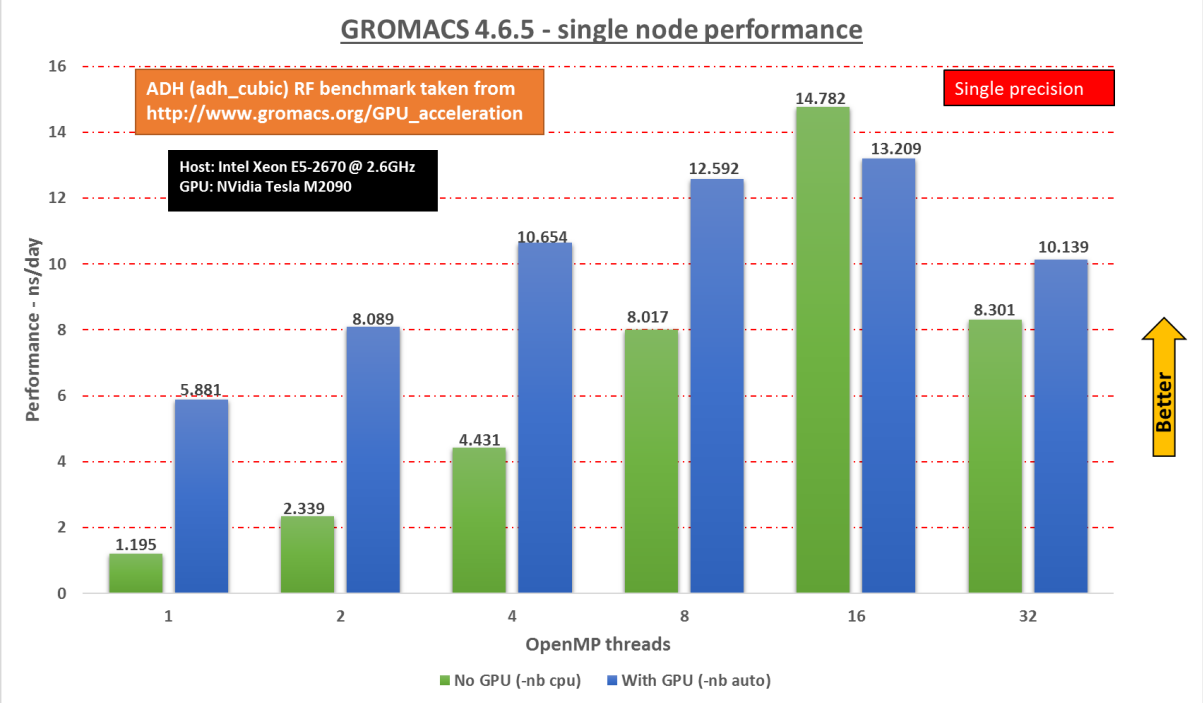

Figure 1: Performance for ADH using a single node with varying numbers of OpenMP threads, with (blue) and without (green) GPU acceleration.

Figure 1 shows that within a single node using a GPU pays off until all cores are being used (16 threads), at which point the overheads of using the GPU dominate. The overheads are because the CPU threads finish calculating the bonded forces and PME before the GPU can finish the non-bonded calculations, and this is evidenced by a high “Wait GPU local” figure of 69.7% in the md.log file. Note that using 32 threads here corresponds to using 2 threads per core, which although supported in hardware with Intel’s Hyper Threading technology, does not pay off for CPU-bound codes (since the 2 threads are sharing a single floating point unit). The evidence here is that GROMACS does not benefit from using these virtual cores.

Ewrop & Chymru: Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales: Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government

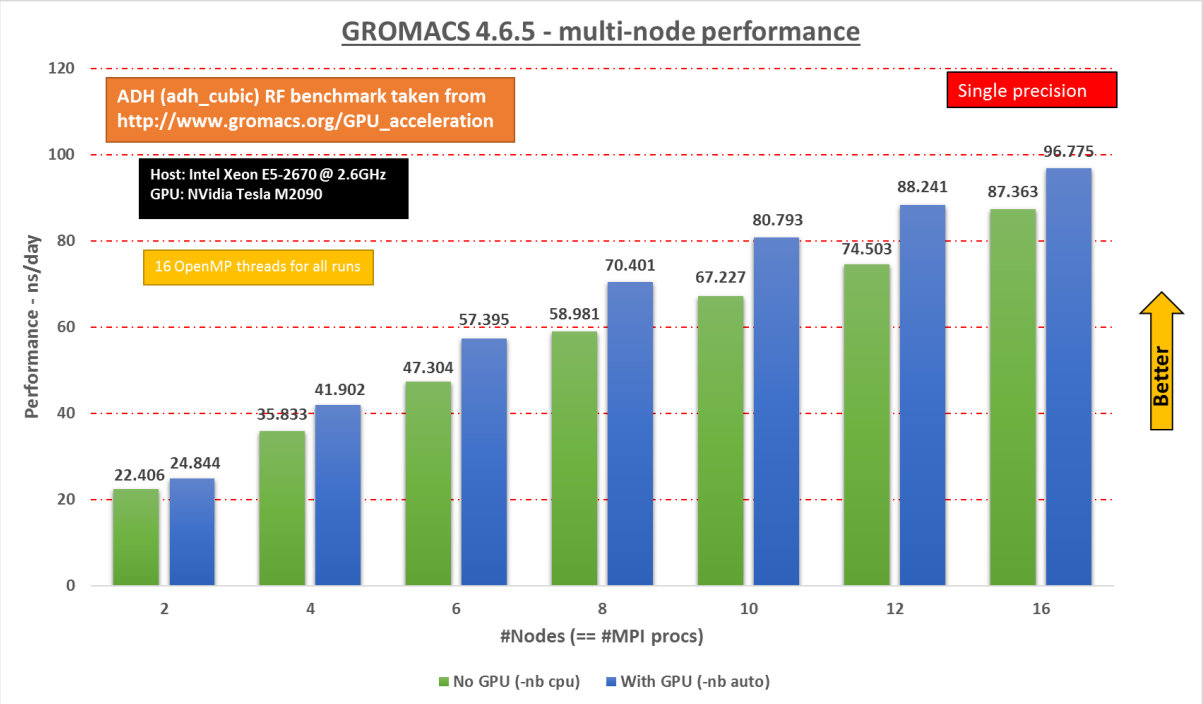

Figure 2: Performance for ADH with multiple nodes each using 16 OpenMP threads, with and without GPU acceleration.

Figure 2 shows that it always faster to enable the GPU when using more than one node, although the best speedup is fairly modest: 1.2x at 6 MPI tasks. Interestingly, the slowdown seen when using a single node with a GPU and 16 threads does not occur for more than 1 node. From the profile given in the md.log files this appears to be explained by the fact that the cost of overheads scale at different rates on the CPUs and GPUs for increasing numbers of nodes. This is shown in Figure 3, where the blue line is the CPU time and the red line is time spent waiting for the GPU: when increasing the number of nodes the amount of work being done on each host is reduced by a smaller amount than the amount of work being done on each GPU. Or equivalently, compared to the CPU, there are fewer overheads on the GPU which come with increasing the number of MPI tasks.

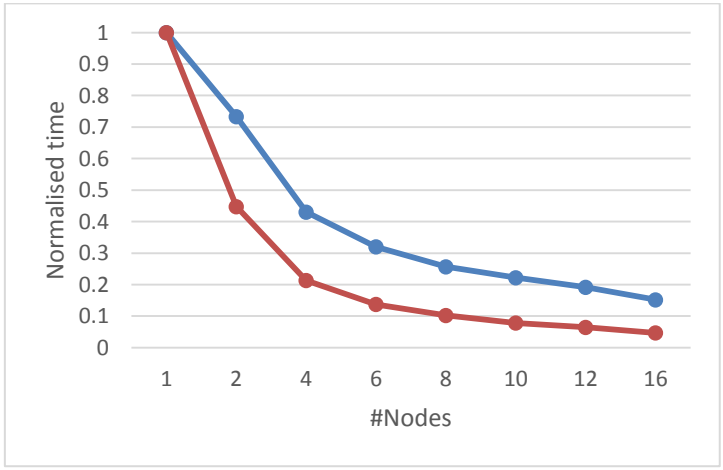

Figure 3: wall time of CPU calculations (blue) and time spent waiting for results from GPU (red).

This shows that there is a small benefit to utilising a GPU for this test case at HPC Wales in the presence of MPI and OpenMP parallelism. Next we consider two larger test cases with more work for the GPU.

Ion Channel

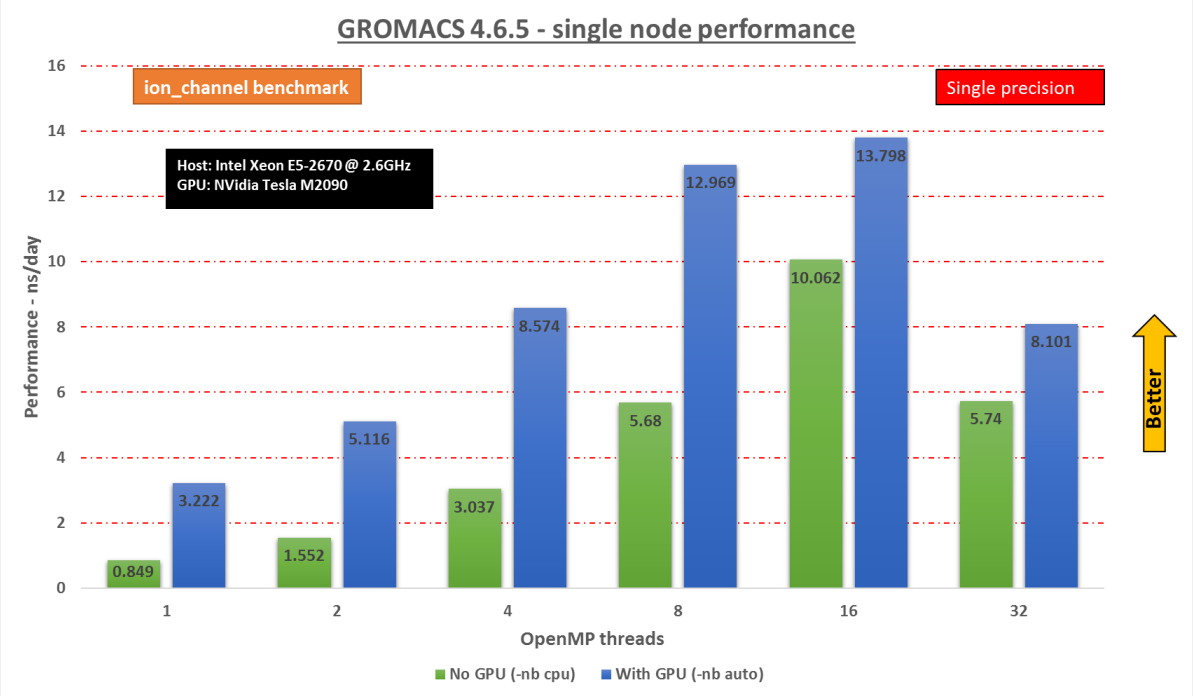

Figure 4: Performance of ion channel within a single node.

In this case it always pays to use the GPU, although with diminishing returns for increasing numbers of threads as the relative load on the host decreases. This can be seen in Figure 5 which plots the force evaluation time ratio GPU:CPU. A value of 1 is optimal; < 1 means the CPU is doing more force work; >1 the GPU is doing more.

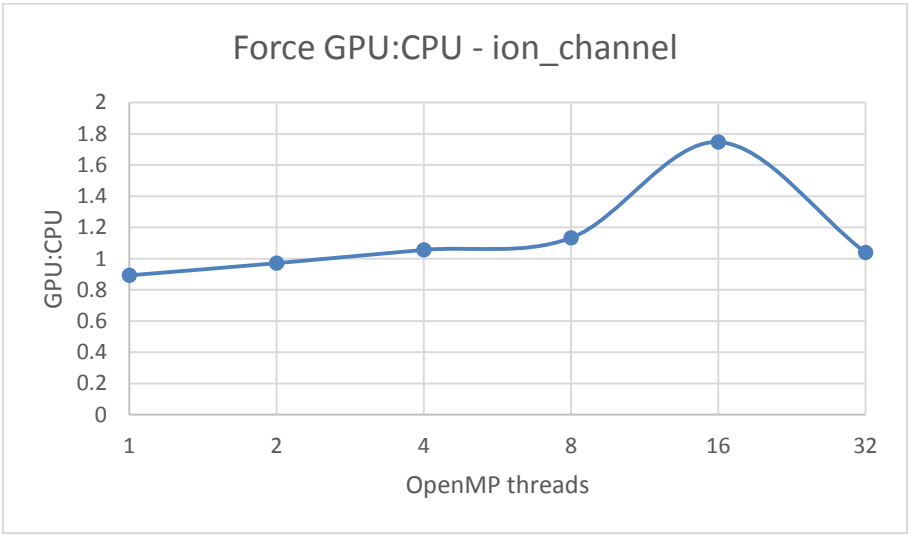

Figure 5: Ratio of force calculation times spent on GPU and CPU.

Ewrop & Chymru: Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales: Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government

Figure 5 shows that the load balance is almost optimal up to and including 8 threads. At 8 threads we have a speedup due to using the GPU of 2.8x.

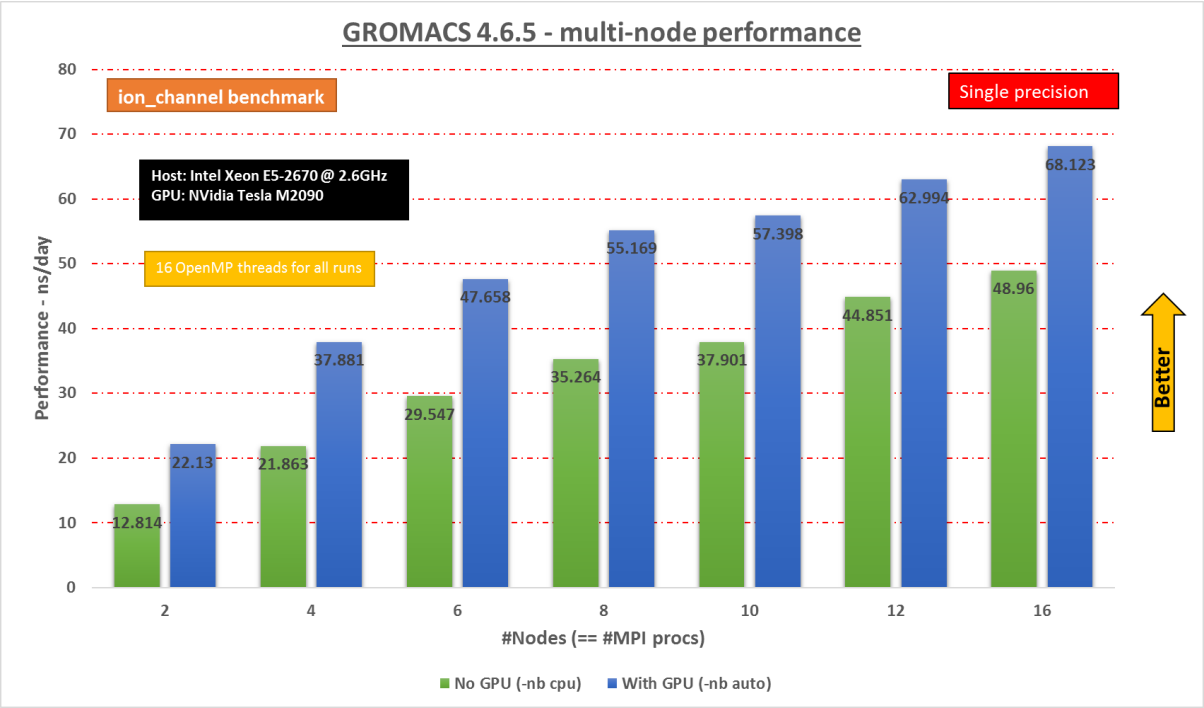

Figure 6: Performance of ion channel with multiple nodes, 16 OpenMP threads per node.

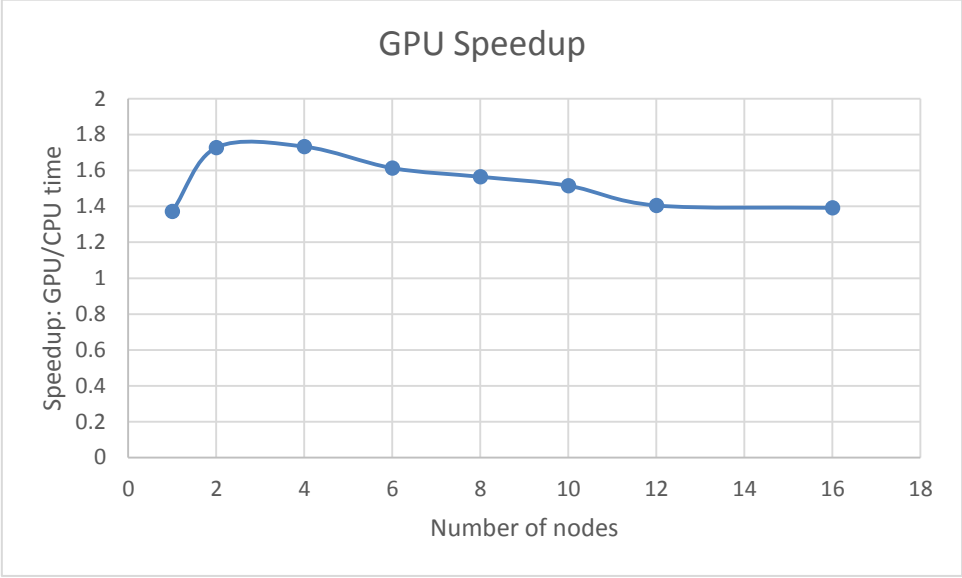

Figure 7: Plot of speedup due to using GPUs against number of nodes.

Figures 6 and 7 show that it is always advantageous to use the GPU up to 16 nodes. As seen in the ADH test case the speedup due to using a GPU is again higher for > 1 node, and this can be attributed to the same observations made for ADH with respect to Figure 3.

Ewrop & Chymru:
Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales:
Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government

Lignocellulose

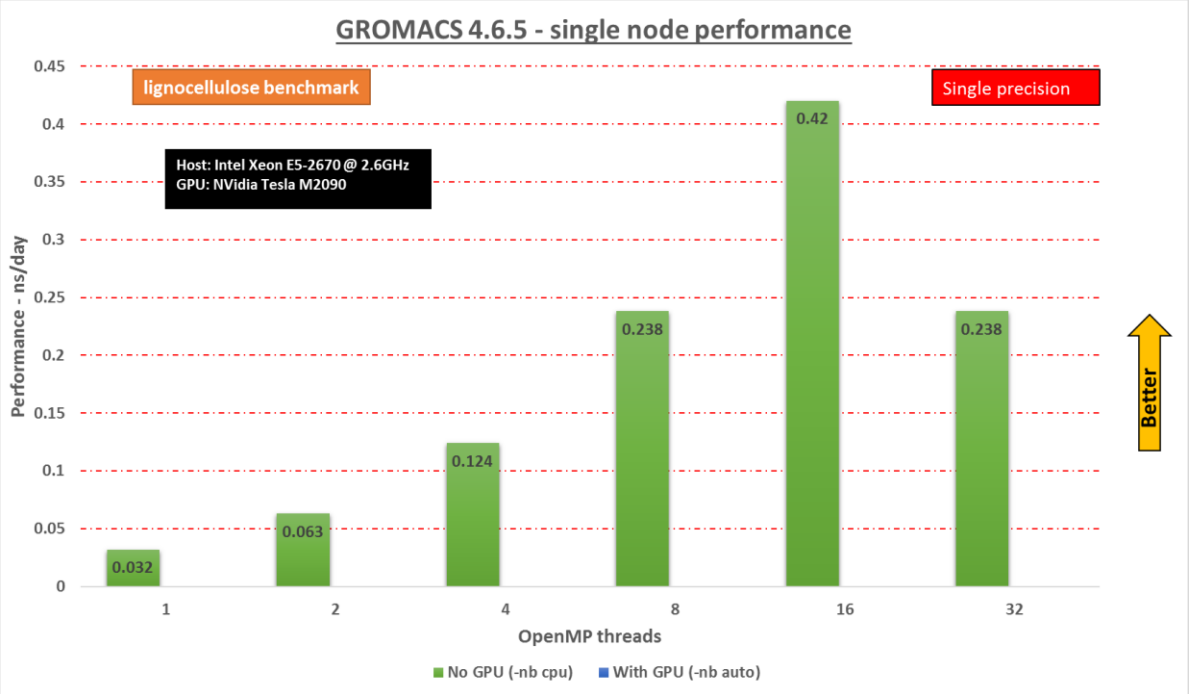

Figure 8: Performance of lignocellulose within a single node.

For the lignocellulose test case running in a single node and enabling the GPU caused the runs to abort with the following error message:

```
Fatal error:
Watch out, the input system is too large to simulate!
The number of nonbonded work units (=number of super-clusters) exceeds the
maximum grid size in x dimension (65670 > 65535)!
```

This happens with the –nbauto GROMACS flag, and indicates that the amount of non-bonded work is too much for a single GPU.

Figure 8 does show that OpenMP scaling is very good up to 16 threads, as seen in all of the benchmarks.

Ewrop & Chymru:
Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales:
Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government

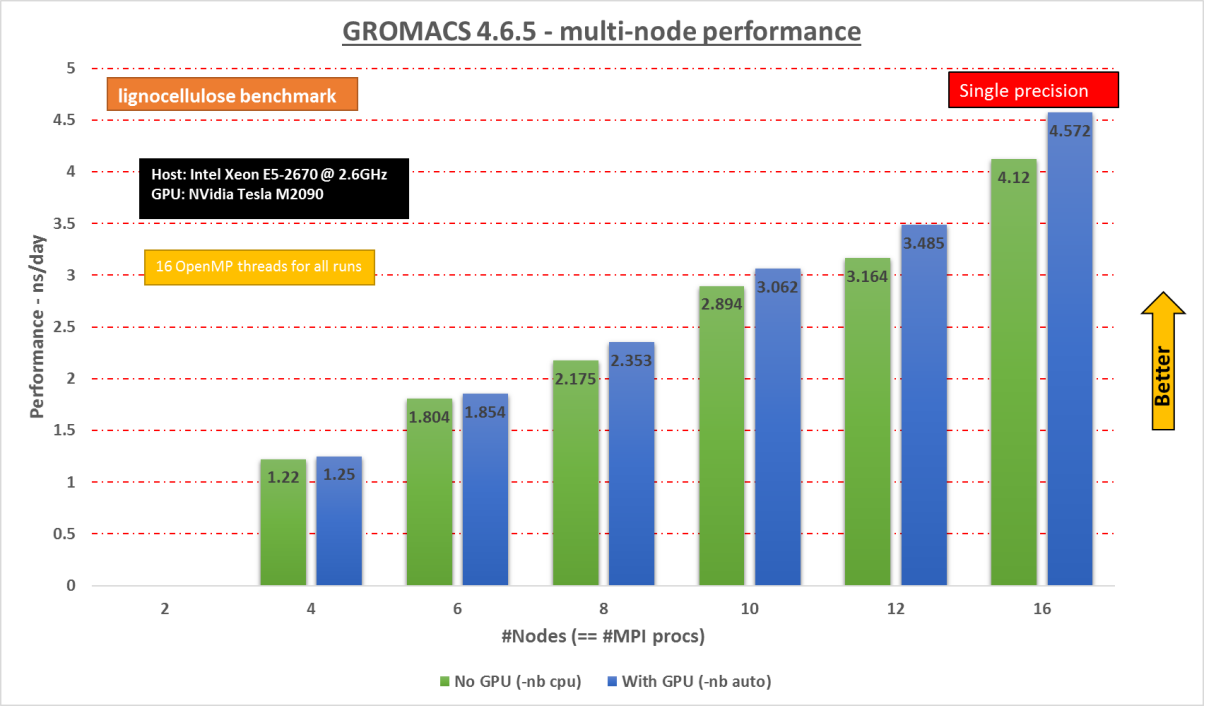

Figure 9: Performance of lignocellulose with multiple nodes, 16 OpenMP threads per node.

When using two nodes runs failed with and without the GPU enabled, giving the following error message:

```
Fatal error:
Too many LINCS warnings (48854)
If you know what you are doing you can adjust the lincs warning threshold
in your mdp file or set the environment variable GMX_MAXCONSTRWARN to -1,
but normally it is better to fix the problem
```

Setting the suggested environment variable does not help, and we do not have access to the mdp file (only the tpr input file).

For more than two nodes there is little to no benefit to using the GPU in this case. Taking into account the error seen when running with the GPU in a single node (i.e. that there is too much non-bonded work for a single GPU) and the fact that for the smallest job we've managed to run with GPUs enabled (4 nodes) the time spent waiting for the GPU is 71% of the overall, it appears that there is too much non-bonded forces work for the GPUs and not enough bonded forces work for the CPU, resulting in a large load imbalance for this case. This comes down to 62% at 16 nodes, hence the minor improvement, but this probably only because of the increased overheads on the CPU due to managing more MPI tasks.

Summary

This report shows that GROMACS can make effective use of the GPU hardware at HPC Wales in order to speed up molecular dynamics simulations. However, the magnitude and scalability of the speedups to be expected depend heavily on the characteristics of the simulation: there needs to be a fair balance between bonded force/PME calculations (done on the CPU) and non-bonded force calculations (done on the GPU) and there needs to be enough work to make offloading worthwhile at all (users are advised to look for the line

"`Force evaluation time GPU/CPU`" in the md.log output file of a GPU enabled run for this information). These characteristics need to be understood when planning to run a GROMACS job in order to avoid wasting compute resources. For the test cases studied here:

- the ADH test case is too small to give impressive speedups with more than 8 threads in a single node;
- the lignocellulose test case contains too many non-bonded interactions so that the CPU is mostly idle whilst it waits for the GPU to complete;
- the ion channel test case is just right for GPU utilization – it can achieve a fair balance of time spent on force calculations by the CPU and GPU as reported for the single node runs in Figure 5 and this scales fairly well as the number of nodes increases.

It is worth noting that, in each case, this balance may be different when running the test case on a different machine with different hardware characteristics. For example, it may prove more beneficial to use a GPU when running the lignocellulose test case on a machine with fewer, or slower, CPU cores, or a machine with more, or more powerful, GPUs. Thus it is always worthwhile running some short benchmark test cases in order to find an appropriate job configuration for your problem on the hardware you are using.