Horizon 2020
European Union Funding
for Research & Innovation

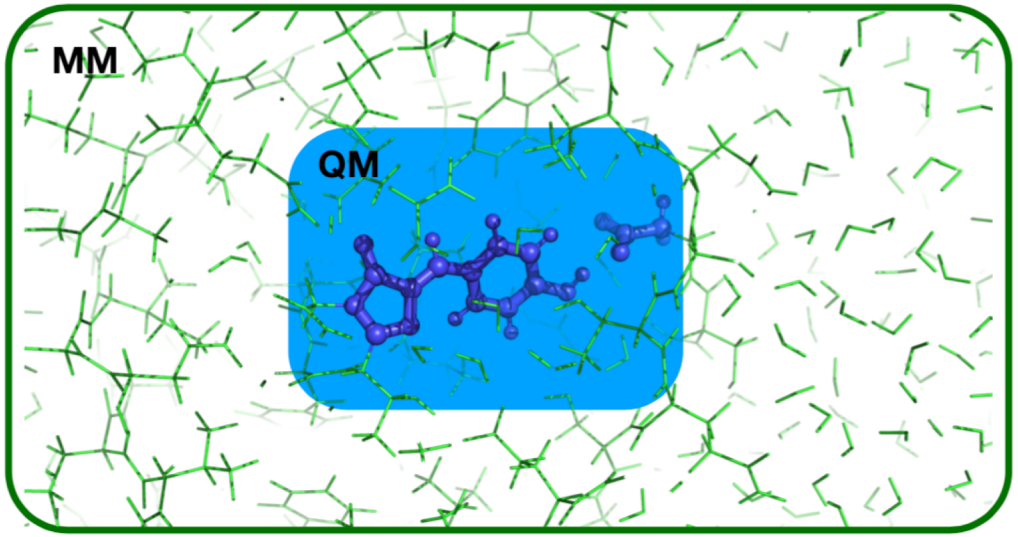
CP2K – QM/MM Practical

Aims

- Run a QM/MM simulation using CP2K on multiple nodes with MPI+OpenMP
  - Look at the effect of threading across multiple nodes
- Learn about profiling CP2K – what can we profile and why is it useful.
- Look at the communication (Message Passing) overheads and how they affect the scaling.
- Think about the different sorts of overheads that can limit scaling.

QM/MM

- MD simulation where you want the accuracy of QM in some region(s), but need classical forces (MM) in general as the system is large.
- An example of this is a protein in a liquid.
- QM energy is calculated using DFT. CP2K uses the QUICKSTEP method: a mixture of Gaussians and Plane wave basis sets.
- The coupling between the QM/MM regions is calculated using GEEP (Gaussian expansion of the Electrostatic Potential).



$$E = E_{QM} + E_{MM} + E_{QM/MM}$$

QuickStep GEEP

CP2K

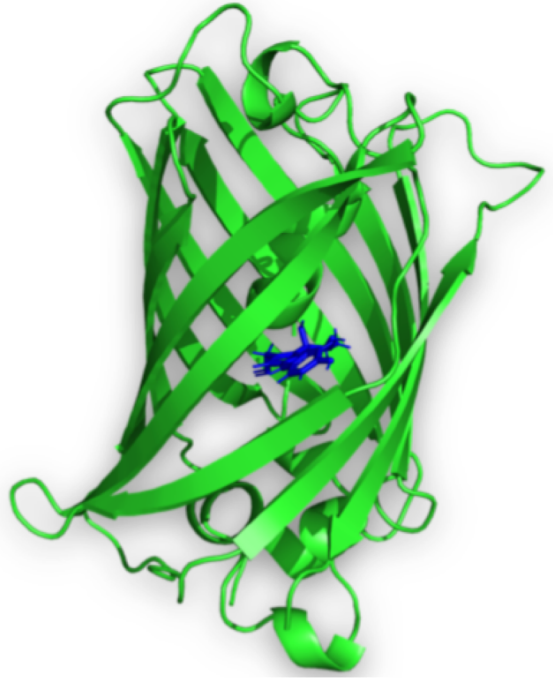
Used to perform atomistic simulations – mainly using density functional theory (DFT)

Can be run with pure MPI– **cp2k.popt,** or MPI+OpenMP - **cp2k.psmp**

Features

- Energy and Forces
- Optimisation
  - Geometry optimisation
  - Nudged elastic band
- Molecular Dynamics
  - Born-Oppenheimer MD
- Properties
  - Atomic charges (RESP, Mulliken..)
  - Spectra
  - Frequency calculations

Green Fluorescent Protein with 20 QM atoms



CP2K QM/MM best practice guide - https://docs.bioexcel.eu/qmmm_bpg/en/main/

CP2K - Profiling

```
 -------------------------------------------------------------------------------
 -                                                                             -
 -                                T I M I N G                                  -
 -                                                                             -
 -------------------------------------------------------------------------------
 SUBROUTINE                       CALLS  ASD         SELF TIME        TOTAL TIME
                                MAXIMUM       AVERAGE  MAXIMUM  AVERAGE  MAXIMUM
 CP2K                                1  1.0    1.012    1.093  579.165  579.166
 qs_mol_dyn_low                      1  2.0    0.004    0.005  566.513  566.935
 qs_forces                           2  3.5    0.000    0.001  465.518  465.519
 qs_energies                         2  4.5    0.000    0.000  457.119  457.120
 scf_env_do_scf                      2  5.5    0.000    0.000  453.759  453.759
 scf_env_do_scf_inner_loop          90  6.2    0.002    0.006  437.466  437.466
 rebuild_ks_matrix                  92  8.2    0.000    0.000  365.177  365.220
 qs_ks_build_kohn_sham_matrix       92  9.2    0.016    0.017  365.177  365.220
 qs_ks_update_qs_env                94  7.2    0.001    0.001  356.828  356.870
```

- CP2K output file gives timings of the called routines – see https://www.cp2k.org/dev:profiling
- SELF TIME – time spent only in this routine
- TOTAL TIME – time spent in this routine, including subroutines called by it
- AVERAGE – averaged over ranks. MAXIMUM – max time of all ranks
- Difference between AVERAGE time and MAXIMUM time indicates load imbalance or synchronisation.

Exercise – Scaling of QM/MM simulations with CP2K

- Download the input files
- Create a suitable job script for running CP2K.
- Explore running the QM/MM simulation (qmmm-1.inp) on a single node of ARCHER2 with MPI+OpenMP.
- Identify the time spent in message passing routines.
- What fraction of the total time is spent in these routines?
- How does this change as the number of nodes is increased?
- Advanced: Repeat for the qmmm-4.inp system which has 77 QM atoms. How is this different?

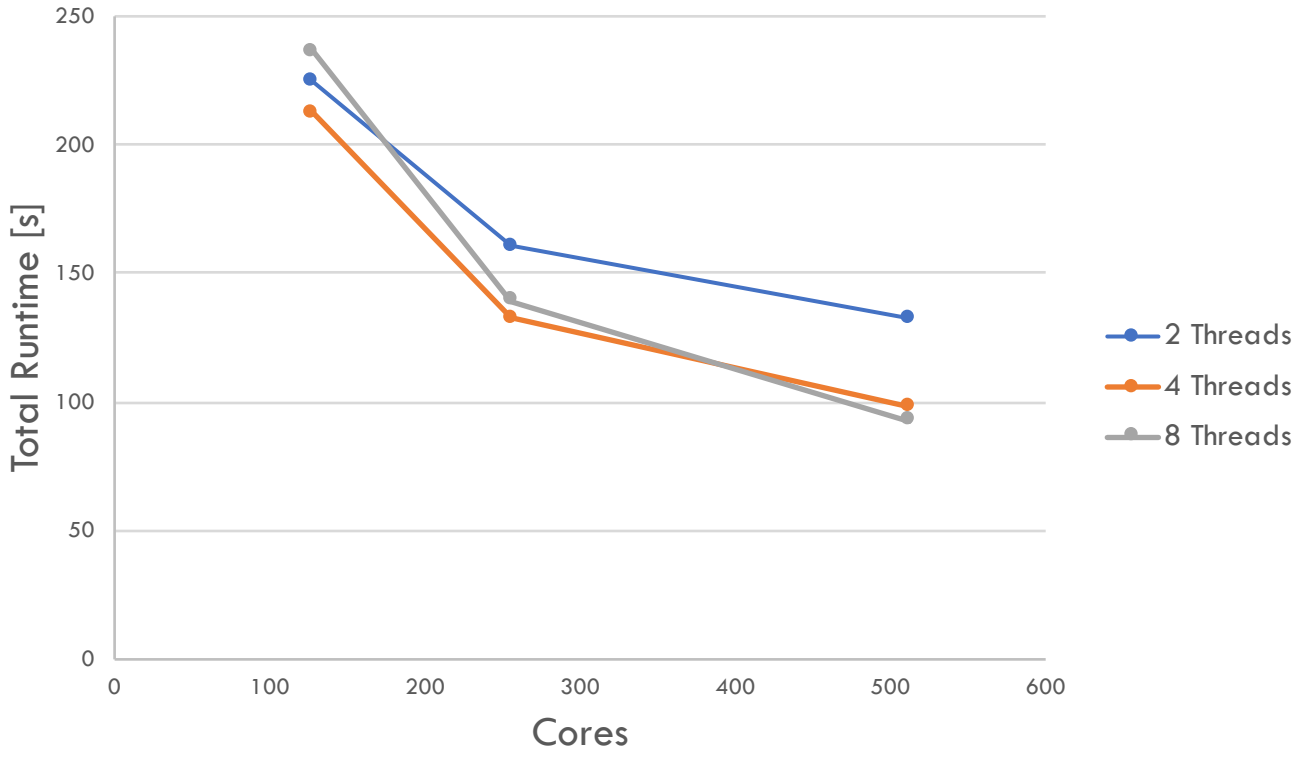
Results – Single node

- Using 1 thread results in out of memory error
  - There is not enough memory per process (128 here)
  - 2 threads allow double the memory per process, remember threads share memory so they do not need their own copy of data
- 4 threads has better performance.
- 32 threads very slow. The threads span 2 memory regions on the node which means data is slow to access

Threads	Time CP2K (s) **T_total**	Time mp_alltoall_z22v (s)	Time mp_sum_dm3 (s)	Time mp_waitany (s)	Total time mp_ routines (s) **T_mp**
1	n/a	n/a	n/a	n/a	n/a
2	224.49	18.377	14.758	4.767	37.902
4	212.471	15.599	8.821		24.42
8	236.41	20.451	5.373	7.207	33.031
16	279.018	24.936		24.819	49.755
32	379.674	35.208		26.258	61.466

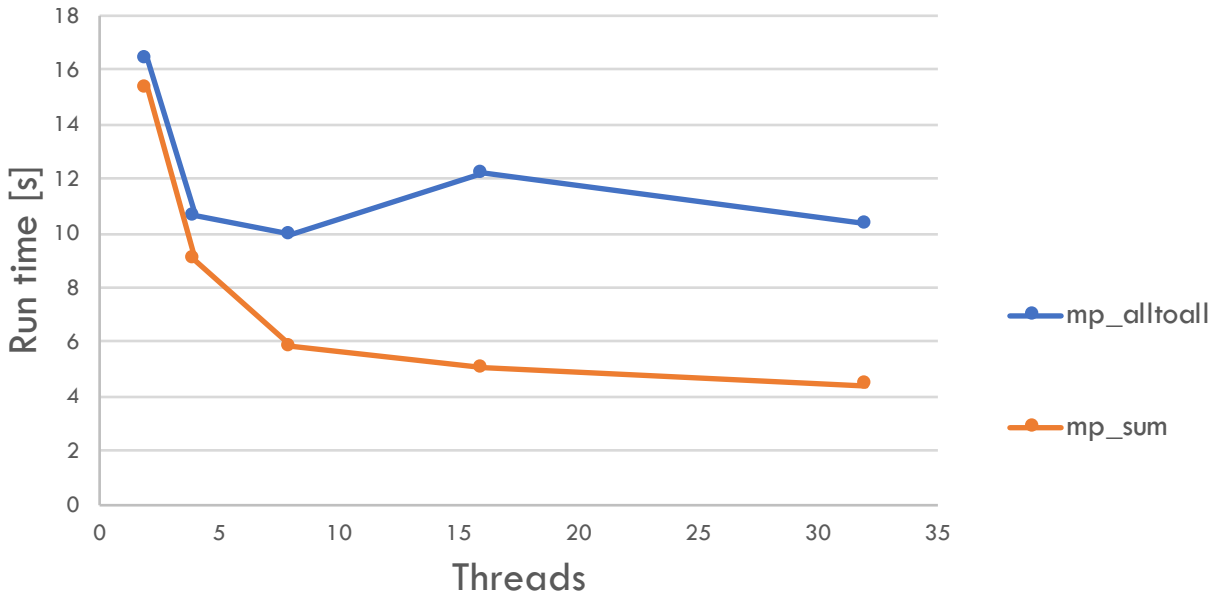
Results – Threading on multiple nodes

- Using 4 threads has the best performance on 1 and 2 nodes
  - 8 threads better on 4 nodes
  - More threads means less processes
  - Potential for less communication overhead
  - Also may exploit threading in areas of code that are not suitable for MPI
- Always a good idea to run tests to see what value is best



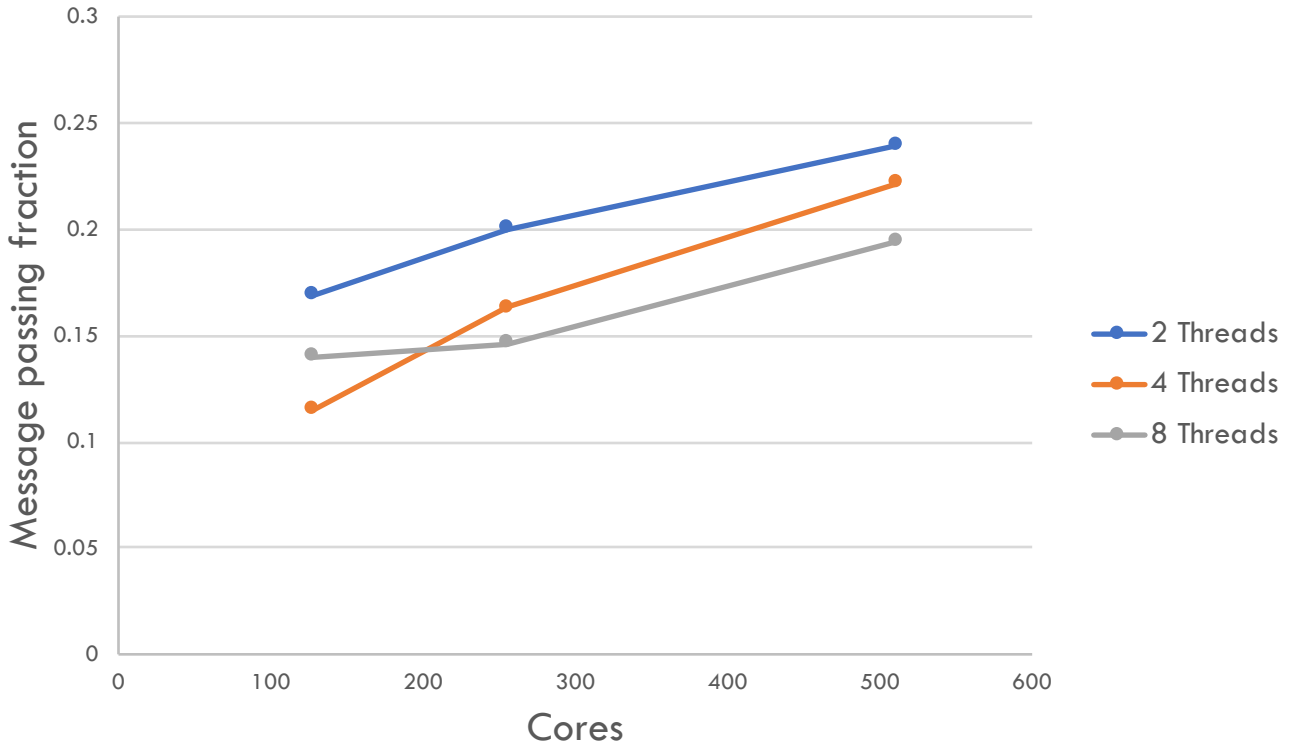
Results – Communications

- Run time of mp_sum decreases with the number of threads
  - Possibly due to less messages being sent
- Run time of mp_alltoall less clear
  - Some improvement with using threads
  - 2-4 threads look to optimal
- Hard to predict how routines and the code as a whole will be affected by changing the number of threads
  - Really have to test this



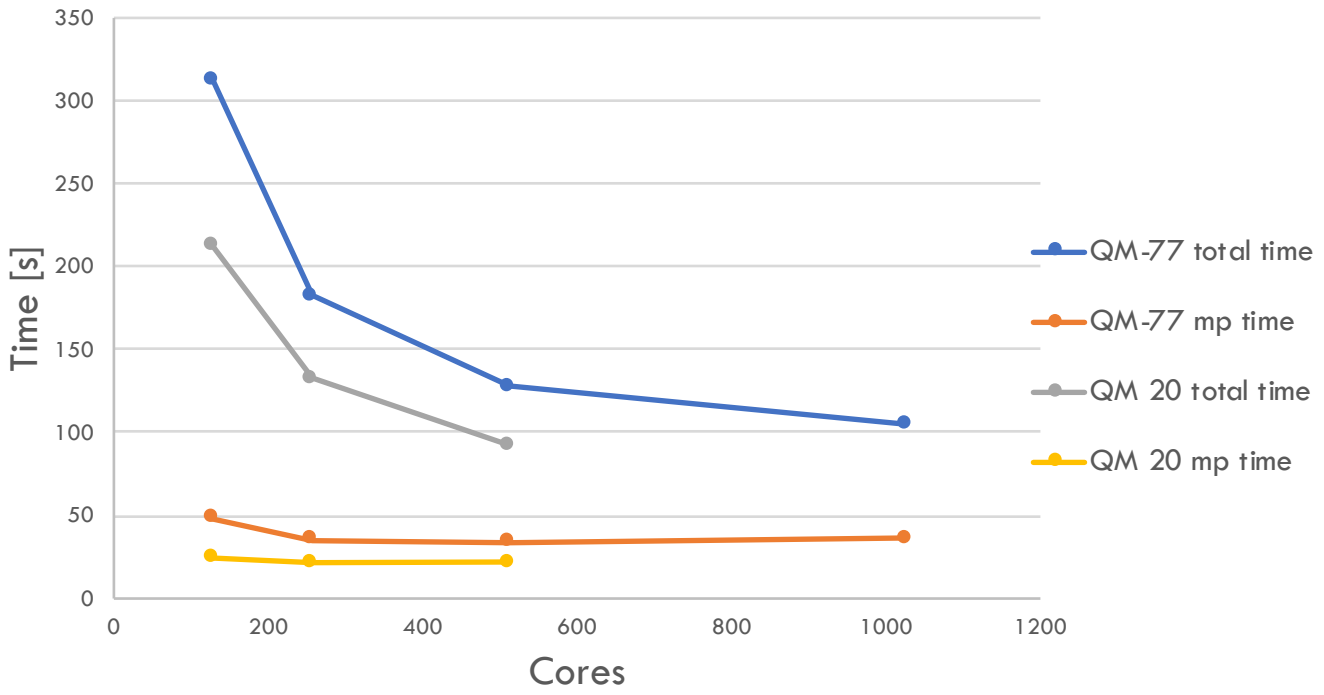
Results – Communications

- Fraction of time spent in MP routines increases with nodes
  - Other areas of the program are improving performance at a greater rate, i.e.. they have greater potential for parallel speed up.
- Changing the number of threads has the potential to reduce this fraction



Results – QM system size

- The large QM system scales better, though perhaps not as well as you would think.
- The system with the large QM region spends a greater fraction of its time doing communications.



BioExcel Partners

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Utrecht University

UNIVERSITY OF JYVÄSKYLA

Horizon 2020
European Union Funding
for Research & Innovation

BioExcel is funded by the European Union Horizon 2020 program under grant agreements 675728 and 823830.