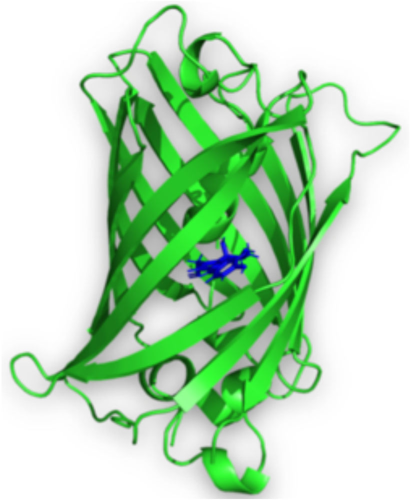
CP2K QM/MM Practical Instructions

The aim of this exercise is to quantify the communication overheads in a CP2K QM/MM simulation. A simulation of an equilibrated Green Fluorescent Protein in water will be run on ARCHER on multiple nodes. This system has 28264 atoms with 20 of those being QM atoms (qmmm-1.inp).



Download the required files

The first part of this exercise will be to run a QM/MM simulation on 1,2, and 4 ARCHER2 nodes (128, 256, 512 cores). Download the relevant files:

Wget

```
qmmm-1.inp
gfp_new1.prmtop
NPT-1.restart
```

qmmm-1.inp is the CP2K input file for the QM/MM calculation. The gfp_new1.prmtop file contains the AMBER forcefield/topology and the NPT-1.restart contains restart information such as the atomic coordinates and velocities. This allows us to start with an equilibrated system. These files should be in the same directory as the CP2K input file when you run the job. You can see these filenames specified in the qmmm-1.inp file.

Next you will have to create a job script to run CP2K. See https://docs.archer2.ac.uk/research-software/cp2k/cp2k/ for example job scripts. There are MPI-only and MPI+OpenMP examples. Please take the MPI+OpenMP example for this exercise. You will need to change the name of the input file to qmmm-1.inp and set the account to the budget code (ta017).

```
#!/bin/bash

# Request 4 nodes with 16 MPI tasks per node each using 8 threads;
# note this means 128 MPI tasks in total.
# Remember to replace [budget code] below with your account code,
# e.g. '--account=t01'.

#SBATCH --job-name=CP2K_test
#SBATCH --nodes=2
#SBATCH --tasks-per-node=16
#SBATCH --cpus-per-task=8
#SBATCH --time=00:20:00
#SBATCH

#SBATCH --account=[budget code]
#SBATCH --partition=standard
#SBATCH --qos=standard

# Load the relevant CP2K module
# Ensure OMP_NUM_THREADS is consistent with cpus-per-task above
# Launch the executable

module load epcc-job-env
module load cp2k

export OMP_NUM_THREADS=8
export OMP_PLACES=cores

srun --hint=nomultithread --distribution=block:block cp2k.psmp -i MYINPUT.inp
```

MPI+OpenMP on a single node

We are going to explore multithreading on a single node of ARCHER2 and have a look at some of the communication overheads. Set `--nodes=1` in the job script.

First try and run with a single thread, you will need to change `—tasks-per-node, --cpus-per-task` and `OMP_NUM_THREADS` in the job script (remember `--tasks-per-node` x `--cpus-per-task` should be 128). You should find that this calculation does not complete. What is the error message?

Now increase the number of threads to 2 (again editing `—tasks-per-node, --cpus-per-task` and `OMP_NUM_THREADS`). The calculation should complete this time (it will take around 2.5 minutes).

When the computation completes the TIMING report is printed at the end of the output file. The TOTAL TIME for CP2K represents the time for the entire run. Subroutine times are printed below in order of the total time spent in them.

```
 -------------------------------------------------------------------------------
 -                                                                             -
 -                                T I M I N G                                  -
 -                                                                             -
 -------------------------------------------------------------------------------
 SUBROUTINE                       CALLS  ASD         SELF TIME        TOTAL TIME
                                MAXIMUM       AVERAGE  MAXIMUM  AVERAGE  MAXIMUM
 CP2K                                 1  1.0    0.346    0.355  224.489  224.490
 qs_mol_dyn_low                       1  2.0    0.289    0.290  214.681  214.945
 qs_forces                            2  3.5    0.028    0.028  136.364  136.365
 qs_energies                          2  4.5    0.057    0.058  132.411  132.411
 scf_env_do_scf                       2  5.5    0.002    0.003  130.218  130.219
 scf_env_do_scf_inner_loop           54  6.3    0.006    0.008  124.300  124.301
 rebuild_ks_matrix                   56  8.2    0.001    0.001  103.475  103.478
 qs_ks_build_kohn_sham_matrix        56  9.2    0.029    0.036  103.474  103.476
 qs_ks_update_qs_env                 57  7.3    0.001    0.001   99.572   99.574
```

Record the total run time. This can be done easily using the grep command:

```
grep ‘CP2K    ‘ slurm-XXXX.out
```

The message passing routines are named with mp at the start e.g. mp_xxxxxx. The run times for these give an idea of the time required for MPI communications. You can find all of the these in the output with the following grep command.

```
grep ‘mp_' slurm-XXXX.out
```

Which could return timings for:

```
mp_alltoall_z22v, mp_sum_d3, mp_waitany
```

Record the **self time maximum** (5th column in the timing report) for each of these.

Repeat for 2, 4, 8, 16, and 32 threads and fill in the values in the table below.

Nodes	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	2					
	4					
	8					
	16					
	32					

Does using more threads speed up the overall run time? What about the message passing routines?
The performance on 32 threads is much poorer than the others. Can you think why this is?

Multiple Nodes and MPI+OpenMP

Try increasing the number of nodes the job script and repeat the multithreading investigation on 2 and 4 nodes. Your table should now look something like this.

Nodes	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	2					
	4					
	8					
	16					
	32					
2	2					
	4					
	8					
	16					
	32					
4	2					
	4					
	8					
	16					
	32					

Is the best performing number of threads the same for 1, 2 and 4 nodes?
Why might threads speed up parts of the calculation?

Communication Overheads

What happens to the run time of mp_sum as the number of threads is increased?
What happens to the run time of mp_alltoall as the number of threads is increased?
Calculate the fraction of time spent doing communications.

i.e. **T_mp/T_total**

How does this change (roughly) as the number of nodes is increased?

Advanced exercise - qmmm-4.inp

Investigate the performance for a different system - qmmm-4.inp. This has 77 QM atoms, but the same number of total atoms as qmmm-1.inp. Record the run time and message passing times using 4 threads on 1,2,4, and 8 nodes. How do these compare to the small QM system?

Nodes	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	4					
2	4					
4	4					
8	4					