CP2K on the VSC

Andy Van Yperen-De Deyne

April 9, 2013

1. What is CP2K?

CP2K is a program which is able to perform both classical force field and quantum chemical calculations for a wide variety of applications such as solid sates, liquids, (bio-)molecules in solutions or single molecules. It offers a large variety of methodologies such as wave function optimization, property calculations, molecular dynamics simulations and many more.

Since there is a large variety in methodologies available, it is essential to know which parts of the code scale well and which don't. This work tries to summarise the benchmark of the most often used parts of the CP2K code, such as

- GPW-DFT calculations[1, 2]
- GAPW-DFT calculations[3]
- QM/MM calculations[4]
- Linear Response calculations[5]
- Vibrational analysis
- TD-DFPT (future?)

1.1. Available versions on Tier-1 and Tier-2

An overview of the many versions of CP2K, compiled on Tier-1 and Tier-2, is given in table 1.

Version	gengar	gastly	haunter	dugtrio	gulpin	raichu	muk
20090728	✔						
20090909	✔						
20100310	✔	✔	✔				
20100317	✔						
20100615	✔						
20100706	✔						
20110124	✔						
20110823	✔				✔		
2.2.425						✔	
20111205	✔	✔	✔		✔		
20130228					✔		✔

Table 1: CP2K Versions on clusters

Remarks

- The line between version 2.2.425 and 20111205 indicates an change is the default settings for `REL_CUTOFF` is changed from 30 to 40 and hence it is possible to find discrepancies between results of both versions.
- I will open a ticket to compile the 2013 version on all Tier-2 clusters, once some library dependencies are cleared out (*i.e.* libint, libxc, libsmm)

2. Tier-1

2.1. Detailed information Tier-1

Muk is contains 528 nodes with following specifications

- each node has two 8-core Intel E5-2670 (2.60 GHz)
- 4 GB RAM per core (64 GB/node), 500 GB local disk/node
- Mellanox Connect-X FDR Infiniband network
- 4 login nodes
- `$VSC_DATA` shared with the Tier-2 clusters
- 400 TB of local scratch

2.2. CP2K version on Tier-1

The benchmarked CP2K compilation is `CP2K/20130228-ictce-4.1.13`, also available on gulpin. It uses following libraries.

- ictce 4.1.13
- FFTW/3.3.3-ictce-4.1.13
- impi/4.1.0.027
- Libint/1.1.4-ictce-4.1.13

This is obviously a parallel compilation using impi and also is compiled with the libint library, hence exact exchange can be calculated.

The *libxc* library is not compiled however, causing failed jobs in the regression tests. Other failed tests use the *Fist* pressure routines, which give rise to segmentation faults and condition failing errors. For details, see the appendix. For users in the CMM this is not a real issue, since this method is not used.

A final remark is that the *libsmm* library is not linked, which has shown to increase the performance[6].

Below the results of the regression tests

```
-------------------------------- summary --------------------------------
number of FAILED  tests 32
number of WRONG   tests 0
number of CORRECT tests 0
number of NEW     tests 2336
number of         tests 2368
-------------------------------------------------------------------------
number of memory  leaks 0
GREPME 32 0 0 2336 2368 0
-------------------------------------------------------------------------
Sun Mar  3 18:36:54 CET 2013
************************** testing ended ******************************
```

3. Testing

3.1. GPW Molecular Dynamics

3.1.1. Summary

The majority of calculations using CP2K, will most probably be GPW Molecular Dynamics simulations or GPW geometry optimizations (which is computationally equiva-

lent to GPW-MD). Therefore extensive benchmarking of these simulations is desirable.

The test systems contains of a number of water molecules (32, 63, 256 and 1024). GTH-Pseudopotentials were used, accompanied by a extensive TZV2P-GTH basis set (40 basis functions/molecule) and a cutoff of 280 Rydberg. An NVE ensemble is used with a timestep of 0.5 fs, during 5 fs (10 MD steps) for all experiments with one exception: the 64-molecule experiment has propagated during 25 fs.

# molecules	# basis sets
32	1280
64	2560
256	10240
1024	40960
4096	163840

Table 2: Number of basis sets for each experiment

3.1.2. Results

The performance of all experiments are plotted in figure 1. The performance is defined as the computation time of a one-core job (if available) divided by the computation time of the job itself. For the 1024 molecule job, at least the memory of 4 nodes was needed, and therefore this time was set as the reference value for that job. The dotted line indicates perfect scaling w.r.t. the number of cores.

The increase in system size, produced an increase in calculation time, as shown in figure 2. For this experiment, the scaling towards system size is only slightly worse then $O(N^2)$, which is remarkably good since $O(N^3)$ would be expected. This is however a preliminary conclusion.

Remarks

- I tried using up to 4096 cores for the 1024-molecule job, but this job lacked memory which is very odd.
- For 2048 cores, the 1024-molecule job still ran, but less efficient compared to the 1024-core job.
- Scaling up to 4096 molecules was impossible, due to limits in memory resources.

3.1.3. Comparison to Tier-2 clusters

The job with 64 H_2O molecules was re-used to compare the performance of *Muk* with those of the available Tier-2 clusters. The timings are shown in figure 3, normalized on

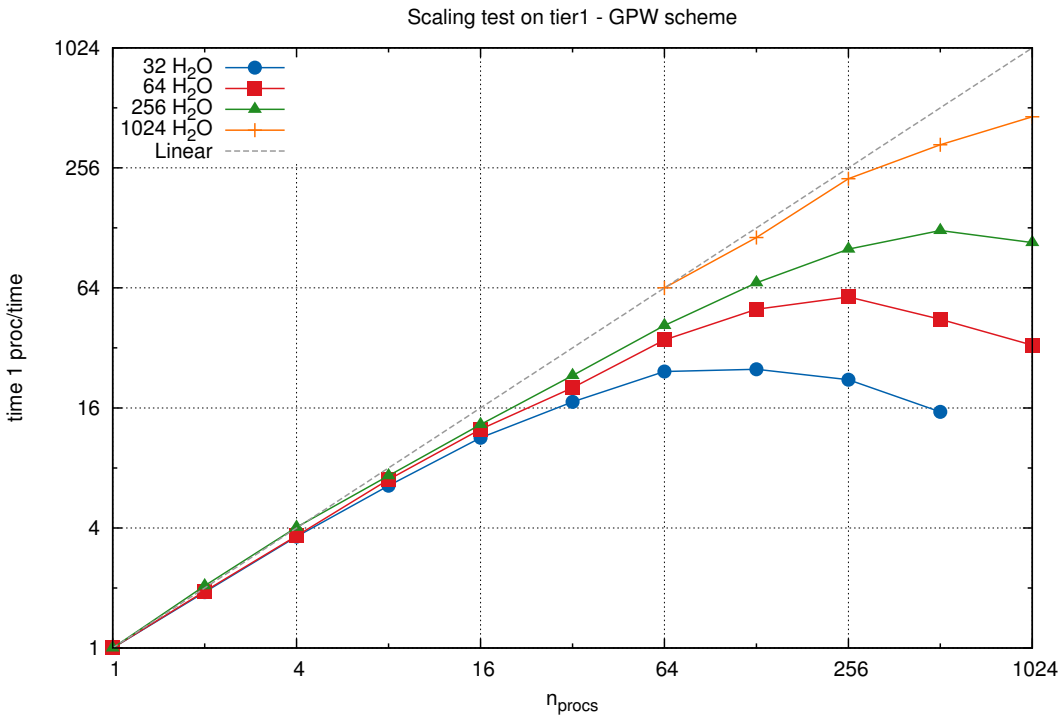


Figure 1: Scaling of the GPW code of CP2K. On the x-axis, the number of cores are plotted in a logarithmic scale. On the y-axis, the total time is given in units of time needed for the one core job (except for the 1024 H_2O experiment, for which the results are scaled on the 64 core job).

the time needed on *Muk*.

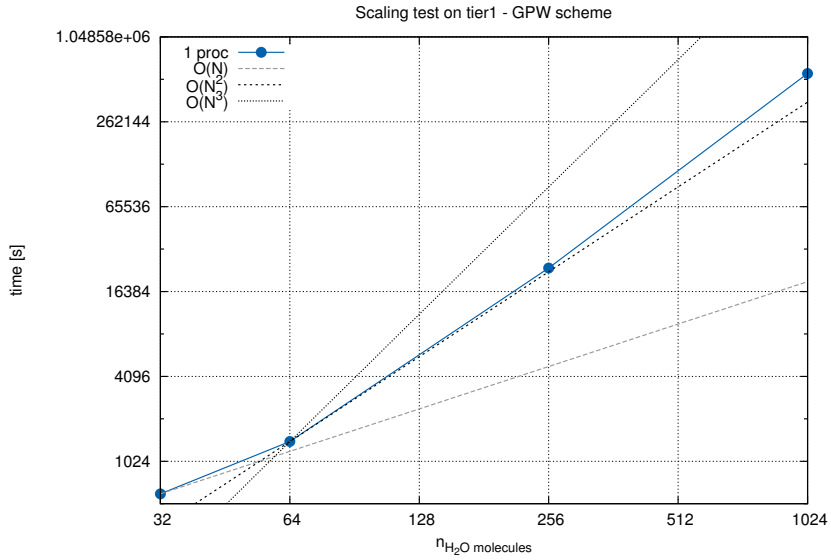


Figure 2: Scaling of the GPW code of CP2K with respect to number of molecules. On the x-axis, the number of molecules are plotted in a logarithmic scale. On the y-axis, the total time is given in seconds for the one core job (except for the 1024 H_2O experiment, for which the results of the 64 core job are multiplied by 64).

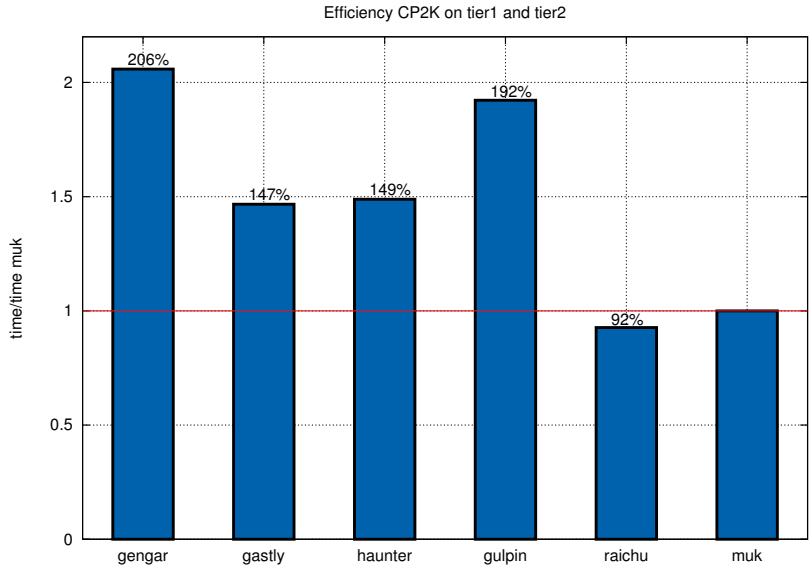


Figure 3: The efficiency of a 8-core job on 64 H_2O molecules on all clusters with the most recent version available (see table 1)

.

The same conclusions hold as for VASP: the clusters with Infiniband perform less good for one-node jobs, compared to those without. *Muk* has the advantage of having both fast nodes and fast inter-nodal communication. For one-node jobs, only *raichu* performs better.

3.2. Linear Scaling SCF

Very recently a linear scaling module is introduced in CP2K[7]. It makes calculations possible up to few ten thousands of atoms. The method uses the sparsity of matrices and available routines which scale linearly for the matrix sign function. The details can be found in VandeVondele et al. [7].

It must however be remarked that results should be benchmarked before the use of this module for scientific work. For scaling purposes however, this validation is not necessary. I leave it to the users of this module to test the accuracy for the system they would like to study and to use this method with care!

3.2.1. First tests

First I tested the "amorph" system, available in the test files. It's a system with 13846 atoms, calculated with a pseudopotential and basis set of double ζ quality, which took about 6.5 hours for a single point energy calculation. Geometry convergence was not reached within the walltime of 3 days. The ability to calculate the energy of such a huge system opens perspectives for large applications in the future.

3.2.2. Water Molecules

To test the scalability on system size, a test has been performed with $32 \cdot N^3$ water molecules in a box. This leads to systems with 32, 256, 864, 2048, 4000, 6912, 10976 and 16384 molecules. The results are given in figure 4. All calculations were performed on 1024 cores and performed 20 SCF cycles, to be able to compare timings. For the smallest system, this number of cores is too much to be efficient (see figure 2), and therefore this data point is not really reliable. Even larger ($N \leq 9$) calculations crash due to memory problems (using 64 nodes).

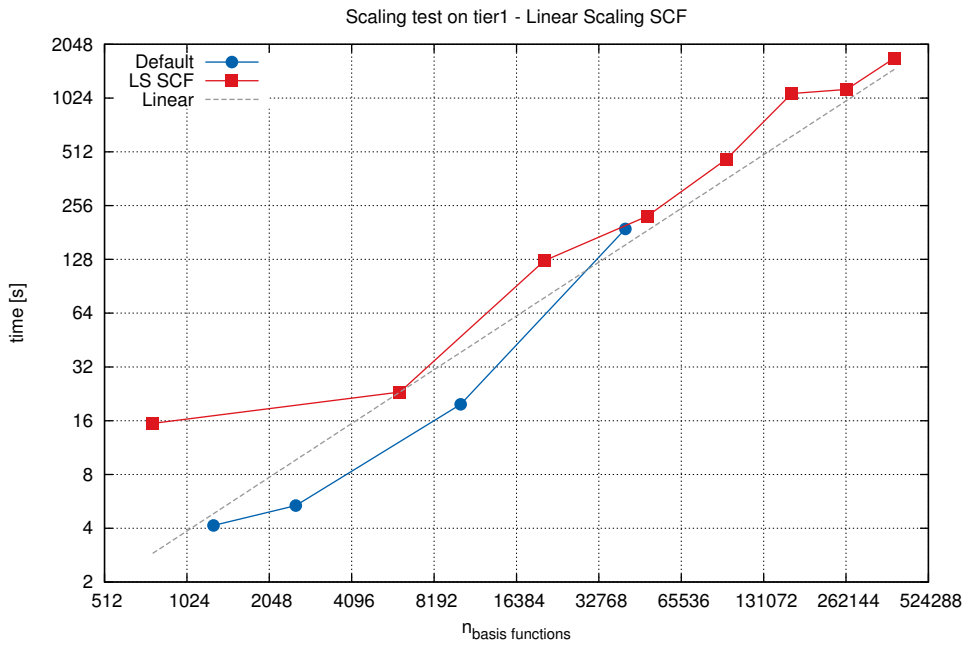


Figure 4: The scaling of CP2K with respect to system size. On the x-axis the number of basis functions is plotted, on the y-axis the corresponding computing time for 20 SCF cycles on 1024 cores

In figure 4 the previous results with the default method are plotted against the linear scaling method in blue. From this figure it is clear that the default method is faster for those systems with less then approximately 40 000 basis functions, while the linear scaling method is more efficient for larger systems. For the system under investigation, this coincides with the memory limits of *Muk* for the default method.

3.3. GAPW method

The same experiment as in section 3.1.1 is performed with an all-electron method (GAPW). This method scales not very well when increasing the number of CPU's. Only 32 and 64 water molecules were tested, but with little improvement for the larger system.

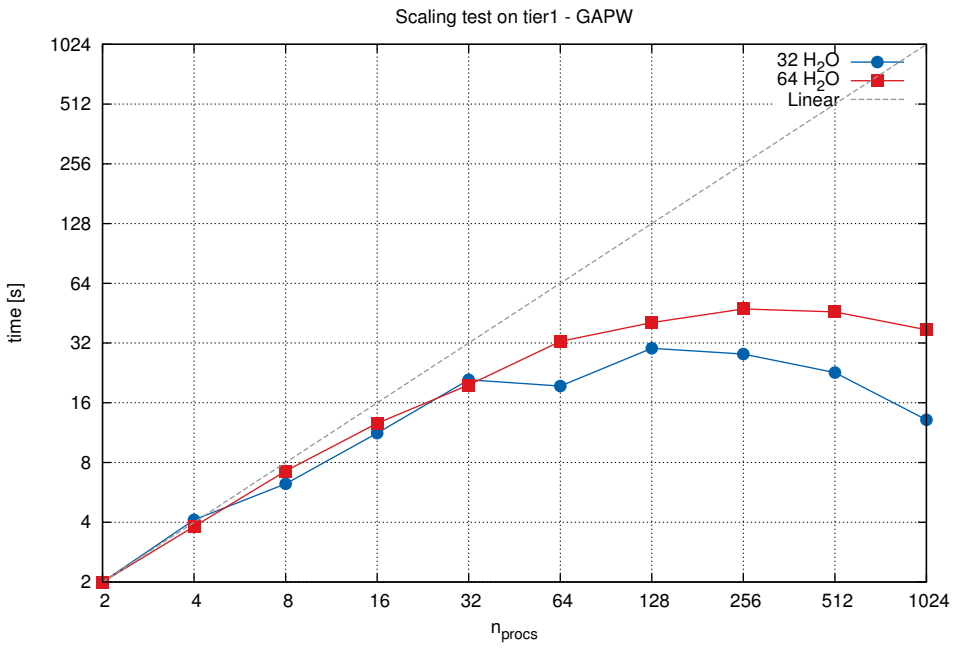


Figure 5: Scaling experiment for the GAPW code in CP2K on both 32 and 64 molecules.

3.4. Hybrid Functionals

Since some time it is possible to use hybrid functionals in the CP2K code. This is a much more expensive method, however, for energy evaluations or geometry optimizations with a good starting point, it is shown to be manageable even at gulpin. The key is to use the auxiliary density matrix method (`&FORCE_EVAL/DFT/AUXILIARY_DENSITY_MATRIX_METHOD`) as described by Guidon et al. [8].

For a system with 1414 basis functions (110 atoms), the main problem seems to be memory, since the computation efficiency improves when using more nodes, which is not to be expected (see figure 6). Only when going from 32 to 64 nodes, the expected scaling is found, indicating there is enough memory from that point on.

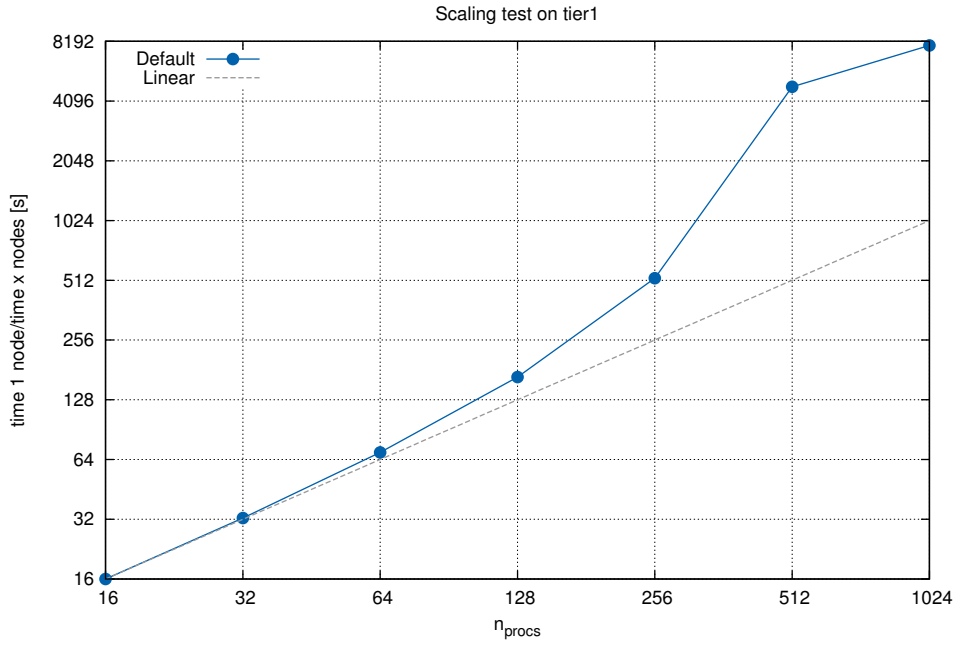


Figure 6: Performance of a hybrid functional calculation (B3LYP) on a SSZ-13 unit cell.

3.5. Frequency Calculations

For frequency calculations, each atom is displaced twice along each direction, forming new *replica*. The energy and force calculations of these $6N$ replica (N the number of atoms) are then distributed over the available cores. The number of cores to be used for each calculation can be set by

```
&VIBRATIONAL_ANALYSIS
  NPROC_REP 1
&END VIBRATIONAL_ANALYSIS
```

The default value is 1, which might be insufficient for huge systems.

A general remark is that for vibrational analysis it is *very* important to use a non-default value for `EXTRAPOLATION` (Alias names: `INTERPOLATION`, `WF_INTERPOLATION`):

```
&QS
  EXTRAPOLATION USE_GUESS
&END QS
```

In this way the initial density for each replica is set equally, for any kind of distribution over the available cores. If the default value for `EXTRAPOLATION` is used, this is not the case, causing different results for different number of cores.

3.6. TD-DFPT

This is still a very experimental part of CP2K. It's scaling is tested (figure 7) and is, as expected, very poor. The test system is ethylorange (41 atoms, GPW method, 389 basis functions, cutoff 300 Rydberg). The limited system size might influence the large-core jobs, but even at 2 or 4 cores, the scaling is very bad.

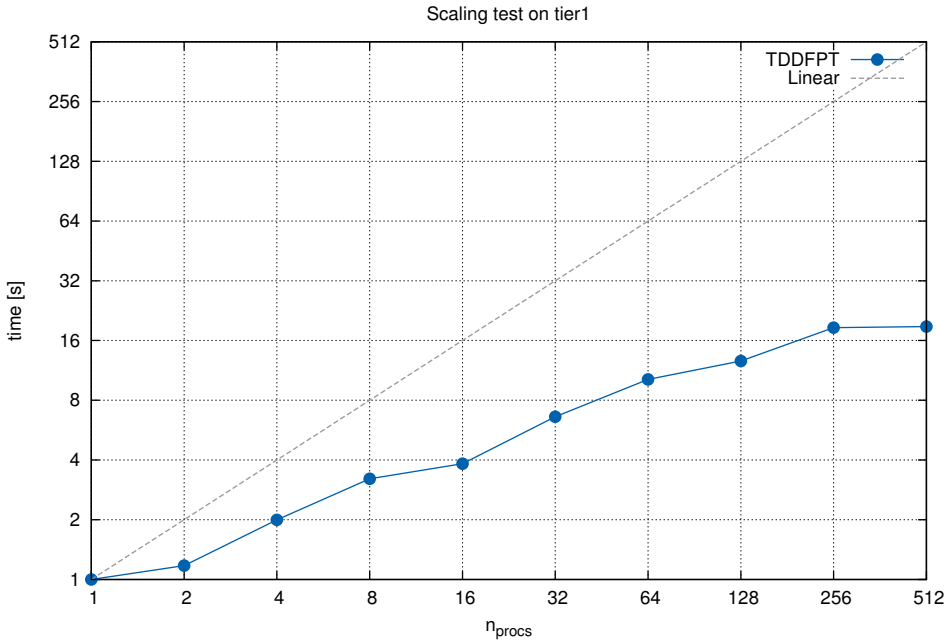


Figure 7: Scaling of TD-DFPT simulation on ethylorange

3.7. Linear Response

Different number of cores leads to different results. The differences are not extreme, but I think it's relevant to look more into detail. On the plus side, the directions are more or less the same.

```
==> nprocs_16/GeoOpt-GTENSOR-1.data <&=&
 129.245 X=   0.4605056380 Y=   0.8864158896 Z=   0.0469193781
3088.809 X=  -0.7247630693 Y=   0.4059905294 Z=  -0.5566778094
5138.657 X=  -0.5124968788 Y=   0.2223478373 Z=   0.8294024286

==> nprocs_32/GeoOpt-GTENSOR-1.data <&=&
 124.966 X=   0.4623380318 Y=   0.8854684764 Z=   0.0467880512
3071.486 X=  -0.7233784378 Y=   0.4071688645 Z=  -0.5576173881
5149.342 X=  -0.5128032567 Y=   0.2239622583 Z=   0.8287784546

==> nprocs_64/GeoOpt-GTENSOR-1.data <&=&
 137.817 X=   0.4605088842 Y=   0.8863711346 Z=   0.0477260857
3105.423 X=  -0.7222025252 Y=   0.4053920021 Z=  -0.5604291545
5150.375 X=  -0.5160959990 Y=   0.2236147050 Z=   0.8268260902

==> nprocs_128b/GeoOpt-GTENSOR-1.data <&=&
 383.239 X=   0.4001792740 Y=   0.9163693457 Z=   0.0111252385
3079.325 X=  -0.8032630976 Y=   0.3565778926 Z=  -0.4770960099
4911.810 X=  -0.4411631725 Y=   0.1819874413 Z=   0.8787807613

==> nprocs_256/GeoOpt-GTENSOR-1.data <&=&
 373.930 X=   0.4034910863 Y=   0.9149014306 Z=   0.0122603204
3119.435 X=  -0.8024443142 Y=   0.3602697840 Z=  -0.4756982293
4896.371 X=  -0.4396340135 Y=   0.1821017709 Z=   0.8795230976
```

4. Long simulations

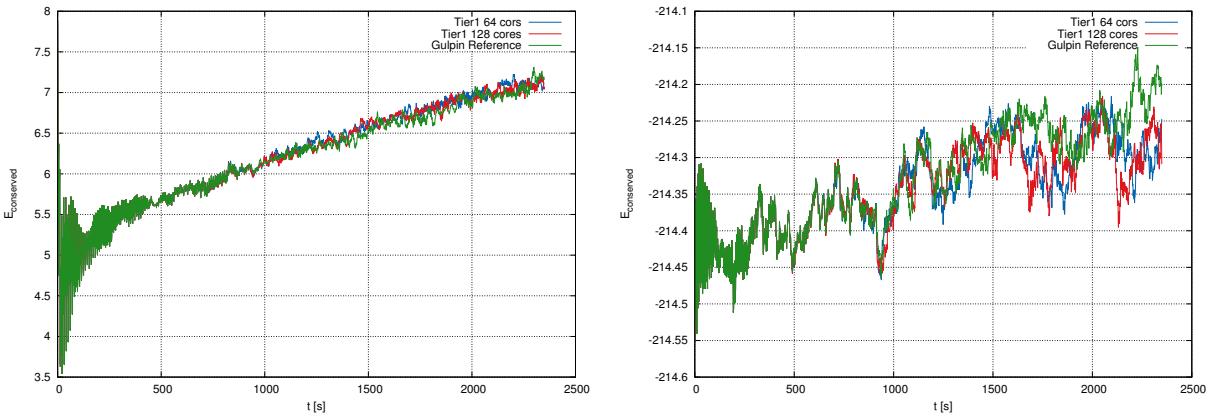
4.1. NPT on silicate

- 2304 atoms / 29952 basis sets
- Optimal performance
  Muk: 1 MD step takes 50 s (1024 cores)
  Gulpin: 1 MD step takes 200 s (384 cores)

- all quantities in correspondence with gulpin results

4.2. QM/MM calculations

- 5543 atoms (41 QM)
- V=22 A^3



(Kinetic Energy) (Conserved Quantity)

Figure 8: Comparison of different cores on a QM/MM calculation

From the results in figure 8 we can see that for th first 500 or 1000 steps, the kinetic energy and conserved quantity are almost identical. Afterwards, the different run diverge, which is expected since small differences propagate during the run. From this figure it is also clear that the system is not completely equilibrated.

References

[1] Lippert, G.; Hutter, J.; Parinello, M. *Molecular Physics* **1997**, *92*, 477–488.

[2] VandeVondele, J.; Krack, M.; Mohamed, F.; Parrinello, M.; Chassaing, T.; Hutter, J. *Computer Physics Communications* **2005**, *167*, 103 – 128.

[3] Lippert, G.; Hutter, J.; Parrinello, M. *Theoretical Chemistry Accounts* **1999**, *103*, 124–140.

[4] Laino, T.; Mohamed, F.; Laio, A.; Parrinello, M. *Journal of Chemical Theory and Computation* **2005**, *1*, 1176–1184.

[5] Weber, V.; Iannuzzi, M.; Giani, S.; Hutter, J.; Declerck, R.; Waroquier, M. *Journal of Chemical Physics* **2009**, *131*, year.

[6] Bethune, I. *CP2K - Sparse Linear Algebra on 1000s of cores, a DCSE Project*, 2012.

[7] VandeVondele, J.; Borstnik, U.; Hutter, J. *Journal of Chemical Theory and Computation* **2012**, *8*, 3565–3573.

[8] Guidon, M.; Hutter, J.; VandeVondele, J. *Journal of Chemical Theory and Computation* **2010**, *6*, 2348–2364.

A. Job file

```
#!/bin/bash
#
#PBS -N jobname
#PBS -o jobname.log
#PBS -e jobname.err
#PBS -q batch
#PBS -l walltime=71:59:00
#PBS -l nodes=64:ppn=16
#PBS -m ae

ulimit -s unlimited
module load cluster
module load VSC-tools
module load CP2K/20130228-ictce-4.1.13

Naam=[INPUTFILE BASENAME]

ORIGDIR=$PBS_O_WORKDIR
WORKDIR=$VSC_SCRATCH/$PBS_JOBID

echo Hostname: $(hostname)
echo ORIGDIR: $ORIGDIR
echo WORKDIR: $WORKDIR

mkdir -p $WORKDIR
cd $WORKDIR

cp $ORIGDIR/* $WORKDIR/

time mympirun -h 16 --branchcount=4 cp2k.popt -i ${Naam}.inp -o ${Naam}.out

mkdir $ORIGDIR/
cp * $ORIGDIR/

rm -Rf $WORKDIR
```

Remarks

- hybrid mode of mympirun: number of cores/node to be used

- branchcount=4 is an option which might be useful if you use a lot of cores. If you don't, all the cores will be occupied without any output being produced.

B. Errors in regression test

B.1. Pressure calculations using Fist

```
*****************************************************************************
*** 18:33:33 ERRORL2 in force_env_methods:force_env_calc_num_pressure ***
*** processor 0  \dots err=-300 condition FAILED at line 622          ***
*****************************************************************************


==== Routine Calling Stack ====

          1 CP2K
 CP2K| condition FAILED at line 622
 CP2K| Abnormal program termination, stopped by process number 0
       application called MPI_Abort(MPI_COMM_WORLD, 1) - process 0
 CP2K| condition FAILED at line 622
 CP2K| Abnormal program termination, stopped by process number 1
       application called MPI_Abort(MPI_COMM_WORLD, 1) - process 1
       rank 0 in job 1  localhost_57604   caused collective abort of all ranks
       exit status of rank 0: return code 1
```

B.2. libxc missing

```
 ****************************************************************
 *** ERROR in xc_functional_get_info (MODULE xc_derivatives) ***
 ****************************************************************

 *** In order to use libxc you need to download and install it ***

 *** Program stopped at line number 239 of MODULE xc_derivatives ***

 ===== Routine Calling Stack =====

            3 qs_init_subsys
            2 quickstep_create_force_env
```

```
           1 CP2K
 CP2K| Abnormal program termination, stopped by process number 1
application called MPI_Abort(MPI_COMM_WORLD, 1) - process 1


 ***************************************************************
 *** ERROR in xc_functional_get_info (MODULE xc_derivatives) ***
 ***************************************************************

 *** In order to use libxc you need to download and install it ***

 *** Program stopped at line number 239 of MODULE xc_derivatives ***

 ===== Routine Calling Stack =====

           3 qs_init_subsys
           2 quickstep_create_force_env
           1 CP2K
 CP2K| Abnormal program termination, stopped by process number 0
application called MPI_Abort(MPI_COMM_WORLD, 1) - process 0
rank 0 in job 1  localhost_37315   caused collective abort of all ranks
  exit status of rank 0: return code 1
```