Report: Comparison of Performance of Quantum Espresso Simulations for NFS and Scratch Space

Prepared: *Alex Dementsov*
Date: June 11, 2010

Motivation

Performance of Quantum Espresso simulations depends on what file system is used for storing temporary files. The purpose of this report is to give some estimates of to what extent the NFS file system and scratch space affect the performance.

Experiment Description

In this report I did some preliminary performance measurements of self-consistent calculation (PW) for two systems: metallic (Al) and isolator (MgB2), running on several number of cores (2x12 = 24 and 16x12 = 192) by varying a single simulation parameter: K_POINTS. The simulations were performed on Foxtrot (foxtrot.danse.us) cluster which has 34 nodes x 12 cores = 408 cores. This cluster is provided by Penguin Computing, Inc. (www.penguincomputing.com) with Scyld ClusterWare. Every node has its own scratch space of about 229G, plenty of space for these types of simulations. Nodes are connected by InfiniBand connection. During the time the simulations were run on the cluster with almost no other jobs running. This should give a more accurate results in case if other jobs are I/O bound on NFS. I considered two file systems where the temporary files are stores: NFS (Case A) and Scratch space (Case B). To submit jobs I used Torque (http://www.clusterresources.com/products/torque-resource-manager.php) scheduler. The sample session for job submission looks like the following:

```
$ ./run.sh
10357.foxtrot.danse.us
```

where `run.sh` sample script is given in Appendix B. The run time for simulation was taken from `resources_used.walltime` parameter. You can get this parameter by executing the command similar to this one:

```
$ qstat -f 10357 | grep resources_used.walltime
    resources_used.walltime = 00:03:45
```

Results

The results of these performance measurements are provided in the tables below both for Al and MgB2. Also the information is provided about size of output file because with more K-points the file size is increasing to about several Mb. Size of temp directory was about 1 to 2 Gb. The output file (pw.out) was consistent both for NFS and scratch space (as it should, the

only difference was in routines execution times). To verify that results are repeatable and consistent I measured them several times and the error did not exceed 5%.

Case A: *Running simulation on NFS*

[Al]

K points	FS	# cores	Time, min:sec	Output Size, Mb
60x60x60	NFS	2x12	01:31	1.04
60x60x60	NFS	16x12	02:34	1.04
100x100x100	NFS	2x12	10:48	2.76
100x100x100	NFS	16x12	07:07	2.76

[MgB2]

K points	FS	# cores	Time, min:sec	Output Size, Mb
32x32x32	NFS	2x12	02:48	0.45
32x32x32	NFS	16x12	01:21	0.45
60x60x60	NFS	2x12	25:33	1.84
60x60x60	NFS	16x12	16:15	1.84

Case B: *Running simulation on scratch space*

[Al]

K points	FS	# cores	Time, min:sec	Output Size, Mb
60x60x60	Scratch	2x12	00:40	1.04
60x60x60	Scratch	16x12	00:54	1.04
100x100x100	Scratch	2x12	02:46	2.76
100x100x100	Scratch	16x12	3:13, 3:19, 3:13	2.76

[MgB2]

K points	FS	# cores	Time, min:sec	Output Size, Mb
32x32x32	Scratch	2x12	1:42	0.45
32x32x32	Scratch	16x12	0:46	0.45

60x60x60	Scratch	2x12	09:27	1.84
60x60x60	Scratch	16x12	02:45, 2:49	1.84

Conclusions

*1. Increase of number of cores on which a simulation is run **not necessarily** improves performance.*

For example, for Al with 60x60x60 K-points and NFS file system increase of nodes by x8 (from 2 to 16) decreased performance by x1.7. Though for 100x100x100 K-points x8 increase of nodes the performance is improved by x1.5 only, instead of x8, as expected. Similar tendency is observed for the scratch space. My guess will be that the parallelization mechanism in Quantum Espresso is flawed and most of the performance loss is due to the network communication between computational nodes.

2. Scratch space can give a significant performance boost compared to NFS

In my measurements the performance improvements varied from 2 to 8 and depended on material, symmetry of the system and number of cores.

3. Simulation with more atoms in the unit cell generally takes more time to run

Solutions

Obtained results can give some idea of how to implement job submission for Quantum Espresso simulations to achieve high performance.

1. Do not reuse temp directory unless explicitly specified

Simulation results need not to be reused for other simulation tasks (as I thought to be useful and efficient in the first place) because a slight change of a parameter can make these results useless. This means that the scratch space can be cleaned up before the simulation starts, except cases when `restart_mode` is not set to `'from_scratch'`.

2. Replicate temp directory to the nodes on which the simulation is about to start.

The advantage of the approach is simplicity and universality. You don't need to know what specific temporary data are stored in these directories. The disadvantage is that temporary data can be large (several Gb). Before implementing this idea one need to measure the performance degradation due to temp data replication.

Future Analysis

It is interesting to compare how the convergence parameters (e.g. `conv_thr`) effect the run time.

Appendix A: Working with computing nodes

To work with computing nodes use `bpsh` (specific for Pinguin systems) command, which is analogous to `rsh` (remote shell, or remote command execution). Below some useful commands are provided.

Create directory (/scratch/dexity/test) on the scratch space of each node

```
$ bpsh -a mkdir -p /scratch/dexity/test
```

Check how large is the scratch space on all of the nodes

```
$ bpsh -a -d -s df -h /scratch
0    --------------------------------------------------------------------
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda2             229G  2.3G  215G   2% /scratch
1    --------------------------------------------------------------------
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda2             229G  1.9G  215G   1% /scratch
2    --------------------------------------------------------------------
[ . . .]
32   --------------------------------------------------------------------
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda2             229G  5.9G  211G   3% /scratch
33   --------------------------------------------------------------------
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda2             229G  9.5G  208G   5% /scratch
```

Appendix B: Example of PW files for Al and MgB2 for scratch space

Sample run script

[run.sh]

 # Sample script for running the PW simulation on 2 nodes x 12 cores taking input from directory specified by "-d". Redirects stdout to STDOUT and stderr to STDERR files.

```
#!/usr/bin/env bash

module add openmpi
module add acml/4.3.0_gfortran64_int32
module add espresso

echo "mpirun pw.x -npool 900 -inp pw.in > pw.out" | qsub -d
/home/dexity/tests/testscratch/scratch -o STDOUT -e STDERR -V -N Al -l
nodes=2:ppn=12 -
```

Sample input files for PW simulation.

[pw.in]

 # PW file for Al

```
&CONTROL
    calculation = 'scf',
    restart_mode = 'from_scratch',
    tprnfor = .true.,
    outdir = '/scratch/dexity/test',
    pseudo_dir = '../pseudo/',
/

&SYSTEM
    ibrav = 2,
    degauss = 0.02,
    smearing = 'gauss',
    occupations = 'smearing',
    ecutwfc = 27.0,
    ecutrho = 300.0,
    celldm(1) = 7.63449299556,
    celldm(2) = 1.0,
    celldm(3) = 1.0,
    celldm(4) = 0.0,
    ntyp = 1,
    nat = 1,
/

&ELECTRONS
    conv_thr = 1.0d-8,
    mixing_beta = 0.7,
/

ATOMIC_SPECIES
```

```
 Al  26.9815 Al.bp-n-van_ak.UPF

ATOMIC_POSITIONS (crystal)
 Al      0.00000000  0.00000000  0.00000000

K_POINTS (automatic)
 100 100 100 0 0 0
```

[pw.in]

```
# PW file for MgB2
&CONTROL
    calculation = 'scf',
    pseudo_dir = '../pseudo/',
    tprnfor = .true.,
    prefix = 'mgb2',
    outdir = '/scratch/dexity/test',
    tstress = .true.,
/

&SYSTEM
    nbnd = 12,
    nspin = 1,
    ecutwfc = 64,
    celldm(4) = -0.5,
    occupations = 'smearing',
    celldm(1) = 5.81347171,
    ibrav = 4,
    celldm(3) = 1.141104624,
    degauss = 0.025,
    smearing = 'methfessel-paxton',
    celldm(2) = 1.0,
    nat = 3,
    ecutrho = 256,
    ntyp = 2,
    la2f = .false.,
/

&ELECTRONS
    conv_thr = 1.0d-12,
/

ATOMIC_SPECIES
 Mg 24.305 mg.ncpp
 B 11.0 B.pbe-n-van_ak.UPF

ATOMIC_POSITIONS (alat)
 Mg    0.0  0.0         0.0
 B     0.5  0.28867513  0.570552312
 B     0.0  0.57735027  0.570552312

K_POINTS (automatic)
 60 60 60 0 0 0
```