Quantum Espresso

Submitting Batch Jobs

The are the commands for ASE/python mode and the "native" (no ASE/python) mode:

```
esp-ver-bsub <version> myscript.py
esp-ver-bsub-native <version> -q suncat-test -o my.log -n 8 pw.x -in pw.inp
```

Dealing With Memory Issues (e.g. Swapping Jobs)

For a "typical" espresso job (default planewave parallelization):

- if a job swaps on suncat (24GB nodes), run it on suncat2 (48GB nodes)
- if a job swaps on suncat2 (48GB nodes), run it on suncat3 (64GB nodes)
- if a job swaps on suncat3, use 2 suncat3 nodes. suncat3 (only!) has a fast interconnect that should help them run reasonably well on multiple nodes

In the longer term we would like to have a memory estimator that will allow you to choose the best queue in advance, although posts on the espresso mailing list suggest this may be difficult.

k-point Parallelization

- NOTE: typically one does NOT do k-point parallelization for large systems. Only the gamma-point is necessary.
- k-point parallelization across nodes will not be as cpu-efficient as planewave parallelization within one node, so use it judiciously
- k-point parallelization is not as memory efficient as planewave parallelization, but it is supposed to scale better to more nodes (ask cpo if you want a better explanation). In particular, my understanding is that k-point parallelization will not reduce the memory usage per node.
- vossj and cpo have not yet seen good scaling behavior for the k-point parallelization for small systems (2x2x3 system). lausche has reported good k-point scaling for 3x3x4 systems. there have been some not-understood hangs with npool=3 or 4 (see below).
- to turn on k-point parallelization:
    - for ase mode: add parameter "parflags='-npool 2'" to the espresso object. This is a general-purpose **string** for passing run-time options to espresso executables.
    - for native mode: add something like "-npool 2" at the **end** of the line
- an example for 16 cores (2 nodes) and npool=2: each of the 2 pools of 8 cores would parallelize over planewaves, but the 2 pools would process pairs of k-points in parallel. If one had 9 k-points, they would get processed in pairs, but the last one would only be processed on one node, leaving the other idle, which is not ideal.
- if you have done it correctly, you should see a line about "K-points division" in your espresso log file (the planewave parallelization produces a line like "R & G space division")
- there is a chicken-and-egg problem: to run your job one needs to know the number of reduced k-points (to determine npool) however one has to run the job to learn what this number is. a workaround for this would be to run it first in the test queue to learn the reduced number of k-points.

Reducing Memory Usage for Large Systems

From http://www.democritos.it/pipermail/pw_forum/2008-January/008101.html

Excerpt (relevant for the "native" (non-ASE) mode):

- consider reducing the planewave cutoff, **IF** it won't affect your results too much
- is your system an isolated system? then use this keyword:

  ```
  K_POINTS gamma
  ```

  it will use k=0 only (which is all you need for an isolated system) and exploit various tricks to reduce memory usage
- setting option "diago_david_ndim" to the minimum (2) and "mixing_ndim" to a smaller value (4) reduces memory usage, but may increase CPU time
- using diagonalization='cg' will also reduce memory usage, but it will increase CPU time by a sizable amount
- do not calculate stress if you do not need to: it is expensive

In ASE-mode we are currently unable to set the "K_POINTS gamma" field and "diago_david_ndim". The other two can be specified with the convergence keyword. "mixing_ndim" is "mix", and "diagonalization" is "diag". For example:

```
convergence = {'energy':1e-6,
               'mixing':0.7,
               'maxsteps':100,
               'mix':4,
               'diag':'cg'},
```

Example Scripts

A simple optimization: esp.py
Calculate density-of-states: espdos.py
Plot density-of-states: espdosplot.py
NEB: espneb.py

Running in "native" mode (see `esp-ver-bsub-native` command above): pw.inp

Versions

Version	Date	Comment
1	12/3/2012	initial version
2	12/5/2012	use mkl fftw
3	12/7/2012	UNSTABLE version: developers allowed to change espresso.py. Users can overwride espresso.py by putting their own espresso.py in directory $HOME/espresso
4,4a	12/10/2012	update to the latest svn espresso-src and espresso python
5	2/14/2013	Entropy corrections added and default parameters changed (smearing type and width)
6,6a	3/7/2013	Many changes: move to combination of dacapo/espresso pseudo potentials (previously just dacapo), add spin polarized BEEF
7,7a	4/5/2013	Update the python interface for bug fixes. Numbers shouldn't change from v6
8,8a	4/5/2013	Important bug fixes: no need for calc.stop(), support for kpoint parallelization with ASE, fix for rhel5 nfs auto mount problem. Numbers shouldn't change from v6/v7.
9,9a	5/28/2013	Add PDOS/NEB calculations. new libbeef interface allows for adding additional beef functionals in future.
10,10a	6/19/2013	Fix problem with pipe buffering that crashed NEB. Dump more information about python/fortran executables to output.
11	7/2/2013	Bug fix for end of job race condition giving "broken pipe" error
12	7/11/2013	Chuan adds new 'diskio' option to allow get_work_function to succeed (was incompatible with avoidio=True).
13,13a	9/4/2013	**BROKEN** (setupenv renamed). band structure calculations and ase constraints being passed down to espresso's internal relaxation routines. calculation results should not change
14,14a	9/10/2013	fix bug in espresso.py forces, fixed bug with LDA+U PDOS in espresso fortran
15,15a	9/10/2013	fix bug where espresso.py would crash if it was given a not-understand ASE constraint, even using an ASE optimization

SUNCAT Quantum Espresso Talks

Introduction/Usage (Johannes Voss): jvexternal.pdf

Accuracy (Jewe Wellendorff, Keld Lundgaard, **NOTE: password protected because it contains VASP benchmark data**): kelu.pdf

Speed/Convergence (AJ Medford): aj.pptx

Scaling behavior (Christopher O'Grady): espscaling.pptx

Private Espresso Builds

Copy this script, and then edit the appropriate lines at the top:

```
/afs/slac/g/suncat/share/scripts/privesp.csh
```

Avoid syntax error in Python run scripts

There is a simple trick to catch syntax errors even before submitting them to the test queue. Just wrap the ase submit command in the following function (e. g. via your bashrc) and a faulty python script will be exited with an error message without waiting in any queue.

```
function esp(){
    python -m py_compile ${1}
if [ $? -ne 0 ]
then
    echo "Not compiling"
    return 1
fi
chmod -v 755 $1
esp-ver-bsub 18 $@
}
```

Espresso ASE To-Do List

- merge branch with espresso trunk
- become part of ASE svn (need to follow new ASE guidelines)
- dry-run mode to get memory estimate
- understand failing espresso tests
- record uspp and executable directory in output (and/or svn version, somehow?)
- neb (done)
- constraints interface to pass ASE constraints to espresso
- dos (done)
- bandgaps (done)
- separation of site-specific code from ASE code (including site-specific "scratch") (done)
- make beef errors accessible from ASE
- beef self-tests integrated with espresso self-tests
- support kpoint parallelization (done)
- look into other parallelization (openmp, scalapack)
- documentation/examples (including on ASE website)
- fix popen warnings on suncat3
- eliminate difference between batch/non-batch running
- how to get automatic python recompilation with setup.py build approach?
- can we eliminate os-dependent stuff, like grep/egrep/sed?
- eliminate need for calc.stop() with multiple calculations (done)
- get work function without dumping out the electrostatic cube file? (chuan has tools for this)
- dipole correction goes in the middle of unit cell by default (in python, chuan makes sure it goes in the biggest gap) (done)