GPAW

Most Recent GPAW Versions

Farm	New Setups (0.9.9672)	Old Setups (0.8.7929)
suncat /suncat2	gpawv40	gpawv41
suncat3	gpawv40a	gpawv41a

Submitting Batch Jobs

```
gpaw-bsub myscript.py ("old/stable" version)
gpaw-ver-bsub <version> myscript.py (user-selected version)
```

GPAW Convergence Behavior

5/1/2013: Jun Yan reports better convergence behavior with a new version of gpaw (v40/v41) that has an improved eigensolver. Use code like this:

```
from gpaw.eigensolvers.rmm_diis_new import RMM_DIIS_new
calc = GPAW(?...,eigensolver=RMM_DIIS_new
(niter=6),...)
```

A talk given by Ansgar Schaefer studying convergence behaviour for rutiles is here (pdf).

General suggestions for helping GPAW convergence are here.

A discussion and suggestions for converging some simple systems can be found here.

Other convergence experience:

System	Who	Action
Graphene with vacancy	Felix Studt	Increase Fermi Temp from 0.1 to 0.2, use cg
Graphene with vacancy	Chris O'Grady	change nbands from -10 to -20, MixerDif(beta=0.03, nmaxold=5, weight=50.0)
Nitrogenase FeVCo for CO2 reduction	Lars Grabow	use Davidson solver (faster as well?), although later jvarley said MixerSum
Several surfaces	Andy Peterson	Broyden mixer with Beta=0.5
TiO2	Monica Garcia-Mota	MixerSum(0.05,6, 50.)
MnxOy	Monica Garcia-Mota	Broyden MixerSum
Co3O4	Monica Garcia-Mota	Davidson eigensolver, MixerSum(beta=0.05, nmaxold=5, weight=50) or MixerSum(beta=0.05, nmaxold=6, weight=100)
MnO2 with DFT+U U=+2eV	Monica Garcia-Mota	Marcin suggests we disable the DipoleCorrectionPoissonSolver (not yet tested)
MnO2 with DFT+U U=+2eV	Monica Garcia-Mota	Henrik Kristofferson suggests: convergence is easier with high U (U=4eV) and then one can shift to preferred value
MnO2 with DFT+U U=+2eV	Monica Garcia-Mota	(from Heine) increase U in steps of say 0.1 (or smaller) and reuse the density and/or wave functions from the previous calculation? This tends to reduce the problem of being trapped in meta-stable electronic states, and it also makes convergence easier. Monica later reported that this helped.

Cu	Ask Hjorth Larsen	first mixer parameter should probably be 0.1 for faster convergence, because it has a low DOS at the Fermi level. (Other transition metals may require lower values.)
N on Co/Ni (with BEEF)	Tuhin	rmm-diis and MixerSum(beta=0.1, nmaxold=5, weight=50)

Other Tricks:

- To speed up the poisson solver, use something like gpts=h2gpts(h=0.18, atoms.get_cell(), idiv=8) to get a nicely divisible-by-large-power-of-2 grid. This helps the "multi grid" poisson solver.
- experiment with Fermi smearing to help improve convergence
- experiment with number of empty bands to help improve convergence
- be sure to specify nbands, otherwise GPAW will add "plenty" of bands which is very expensive in FD calculations. nbands=6*[number of atoms] should be more than enough.

GPAW Planewave Mode

Jens Jurgen has a post here that discusses how to select plane wave mode in your script.

It looks like we have to manually turn off the real-space parallelization with the keyword:

```
parallel={'domain': 1}
```

In planewave mode I believe we also can only parallelize over reduced k-points, spins, and bands. We have to manually set the right numbers for these to match the numbers of CPUs.

- To get parallelization over bands, we can at the moment only use the rmm-diis eigensolver (cg and davidson don't work).
- The number of bands must be divisible by the number of CPUs, according to Jens Jurgen.
- At the moment there is no dipole correction in planewave mode.
- Density mixing is still done in real-space

GPAW Geometry Optimizations

Thoughts from Lin and Thomas: With GPAW one can do geometry optimizations a factor of 10 faster in LCAO mode (with smaller memory requirements). Then it's necessary to "tweak" the optimization with a little bit of running in FD mode.

Plus, LCAO mode has the added feature that convergence is typically easier, according to Heine.

I think it's difficult to automate the above process in one script, since the number of cores required for LCAO is typically lower than FD (because of the lower memory usage).

But if you're limited by CPU time when doing GPAW optimizations it might be worth keeping the above in mind.

AJ adds: I would also warn against using LCAO as an initial guess for NEB calculations. I have tried this for 2 different systems and it turned out to be a tremendous waste of time. The NEB did not converge much faster with LCAO, and when I used the LCAO images as an initial guess for finite difference mode it still took several restarts to converge. I have had better luck using decreased k-point/grid spacing and relaxed convergence criteria as an initial optimization for NEBs.

GPAW Memory Estimation

The get a guess for the right number of cores to run on for GPAW, run the following line interactively:

```
gpaw-python <yourjob>.py --dry-run=<numberofcores>
(e.g. gpaw-python graphene.py --dry-run=16)
```

Number of cores should be a multiple of the number-of-cores-per-node here. The above will run quickly (because it doesn't do the calculation). Then check that the following number fits within the memory-per-core listed here :

```
Memory estimate
---------------
Calculator  574.32 MiB
```

Tips for Running with BEEF

If you use the BEEF functional:

- use xc='BEEF-vdW'
- the code parallelizes over 20 cores (because of the way the VDW contribution is calculated)
- you typically need a lot of memory, even for small calculations. For larger calculations you sometimes have to use more than 20 cores, just to get enough memory (you don't get additional parallelization).
- the GPAW memory estimates are incorrect
- it is typically best to run on the suncat2 (more memory per core) or, even better, the suncat3 farm (more memory per core, plus a much faster infiniband node-node interconnect)
- use the suncat3 farm, which has a fast (4GB/s) infiniband interconnect.

Building a Private Version of GPAW

- Use svn to check out the version of GPAW that you want to use (described here).
- copy /afs/slac/g/suncat/share/scripts/privgpaw.csh into whatever directory you like
- edit the two variables GPAW_BASE ("base" GPAW release that you want to re-use for numpy, mpi etc.) and GPAW_HOME (directory where you checked out GPAW)
- Use the commands:

```
./privgpaw.csh build (build)
./privgpaw.csh build-rh6 (build for the suncat3 RedHat6 farm. can use same source directory as build for
old farms)
./privgpaw.csh test (run gpaw self-tests)
./privgpaw.csh gpaw-bsub <arguments> (submit batch job)
./privgpaw.csh gpaw-bsub-rh6 <arguments> (submit batch job to suncat3 RedHat6 farm)
./privgpaw.csh <cmd> (run <cmd> interactively, using privgpaw.csh environment.  e.g. "gpaw-python junk.
py")
```

Some notes:

- The syntax for the "bsub" option is identical to the gpaw-bsub command described here.
- The above assumes that the base release is compatible with your checked out GPAW version. Talk to cpo if you have questions about this.
- The above doesn't include support for a private version of ASE. When that becomes important we will add it.

Versions

Version	Date	Comment
gpaw v1	9/14/10	gpaw 0.7.6383, first attempt at SUNCAT gpaw installation
gpaw v2	9/21/10	move from standard lapack libraries to acml,blacs,scalapack to improve performance
gpaw v3	9/30/10	move from ase 3.4.0 to 3.4.1
gpaw v4	10/15/10	check out a gpaw svn branch for hannesj/aleksandra
gpaw v5	10/15/10	copy of v3, add scipy
gpaw v6	10/22/10	copy of v5, move GPAW to head for venkat (gpaw 0.8.0)
gpaw v7	11/9/10	gpaw 0.7.2.6974, switch from ifort v9 to gcc 4.1.2, simplify setupenv
gpaw v8	11/16/10	use goto2 blas and open64 to improve performance
gpaw v9	12/9/10	copy of gpawv8, but HEAD of gpaw/ase to fix non-orthorhombic problem for strabo/pgmoses also change gotoblas flags to try to improve performance
gpaw v10	12/9/10	copy of gpawv9, but put in numpy site.cfg to try to use acml for numpy
gpaw v11	1/10/11	copy of gpawv9, a special version for hildur to use
gpaw v12	6/27/11	copy of gpawv9, a special version for jewe to use (preliminary BEEF version)
gpaw v13	7/8/11	softlink to gpawv9, change setupenv to use torque/maui aware openmpi

gpaw v14	8/10 /11	copy of gpawv9, update to HEAD of gpaw svn, and HEAD of ase svn (for wulff construction) requested by Lin
gpaw v15	10/5 /11	test version built with icc/mkl
gpaw v16	10 /12 /11	version like gpawv14, but with python2.7 so we can include hdf5 support for Lin. Only serial version functional with /opt/TWWfsw/python27/bin/python
gpaw v17	11 /16 /11	copy gpaw/ase from v9, but use icc/ifort/mkl to improve performance
gpaw v18	1/3 /12	copy gpaw/ase from v17, but mkl 10.3 to fix LD_PRELOAD hack
gpaw v19	1/20 /12	like gpawv17, but GPAW from head of SVN to get first BEEF release, also most recent setups, and more recent ASE
gpaw v20	1/21 /12	mkl 10.3 with scalapack, gpaw from head of SVN, newer ASE, new setups. **serial version broken** because removed static linking?
gpaw v21	2/1 /12	uses v17 executables, but adds head of ASE and version 0.3.2 of CMR to PYTHONPATH (requested by AJ)
gpaw v22	2/6 /12	like v19 but uses GPAW/ASE from head of SVN to pickup BEEF error estimation. also enables scalapack, CMR
gpaw v23	3/26 /12	like v22 but uses GPAW v0.9, required ASE 3.6.0, required CMR 0.3.3.580
gpaw v24	4/27 /12	like v23 but uses head of GPAW and new ASE to pick up fix for vdW parallel NEB
	6/22 /12	**NOTE THAT GPAW VERSIONS OLDER THAN 25 WILL NO LONGER WORK BECAUSE OF PYTHON-LIBS SECURITY PATCH PROBLEM**
gpaw v25	6/20 /12	copy of v22 but rebuilt to fix python-libs patch problem
gpaw v26	6/20 /12	copy of v17 but rebuilt to fix python-libs patch problem
gpaw v26a	11 /16 /12	(suncat3 farm only) copy of v26 ("old/stable") gpaw, but some tests fail still
gpaw v27	6/20 /12	copy of v24 but rebuilt to fix python-libs patch problem
gpaw v27a	11 /16 /12	v27 rebuilt for suncat3 farm (softlink to v35)
gpaw v28	6/22 /12	copy of v24 but try python2.7 with updated numpy/scipy/matplotlib. gtk graphics broken
gpaw v29	7/10 /12	copy of v16 but rebuilt gpaw because of the python-libs problem. only serial version functional with /opt/TWWfsw/python27/bin/python
gpaw v30	7/23 /12	head of GPAW svn to get FixedOccupations class in occupations.py for Joel. But turned out v27 already had that. Also newer ASE. **DISABLED** this because wasn't optimally built.
gpaw v31	10 /12 /12	same as V30, but with latest official setups. **DISABLED** this because need to update to latest GPAW for latest setups.
gpaw v32	11/7 /12	head of GPAW svn but with latest official setups
gpaw v32a	11 /14 /12	v32 rebuilt for suncat3 farm (softlink to v33)
gpaw v33	11 /14 /12	(suncat3 farm only) like v32, but rebuilt for Sandy Bridge and RHEL6
gpaw v34	11 /14 /12	(suncat3 farm only) like v33, but uses older setups **DISABLED** because saw strange results
gpaw v35	11 /16 /12	(suncat3 farm only) copy of v27 gpaw, so we can use older setups
gpaw v36b	11 /16 /12	(gpu farm only) checkout of Jun's branch rpa-gpu-expt built for the new gpu farm. old setups.

gpaw v37b	1/28 /13	(gpu farm only) checkout of Samuli's cuda branch built for the new gpu farm. old setups.
gpaw v38	2/1 /13	build of libxc 2.0.0 branch for doylead/alevoj so they can use HSE.
gpaw v39	2/25 /13	copy of v27 with newer ASE 3.6.1.2981
gpaw v40	3/24 /13	latest gpaw from trunk to fix get_all_electron_density problem. new setups. two tests fail running with -j 8, but work running tests without -j 8
gpaw v41	3/24 /13	like v40, but old setups.
gpaw v42	4/18 /13	copy of v29 environment to pick up hdf5 for Lin. parallel version broken. latest gpaw from trunk to fix get_all_electron_density problem. old setups (so self-tests don't work). still need to use /opt/TWWfsw/python27/bin/python, I believe
gpaw v43	6/28 /13	first version built with new libxc
gpaw v44	10/7 /13	like v43, but with newer ASE for chank to pick up new minimahopping.py