DFT on GPUs

Long Term Goals

- think about what the field will need "8 years from now" and what scientists want:
    - energies, derivatives of energies (most important)
    - density of states
- dirac-kohn-sham
- all-electron with ability to "understandably" switch to PAW
- ability to handle degenerate states
- use multigrid for eigensolver (O(N^2)) so we do less orthogonalization (O(N^3)). do orthogonalization on the coarse grid, and don't lose it moving to finer grids. (mid-90's codes avoided this)
- multi-GPU
- ability to scale out errors (like early GPAW). limited eventually by machine accuracy. e.g. romberg integration
- real-space
- gpu data exchange with GPU-Direct (within node and over IB ... no data exchange through MPI)
- avoid orthogonalization, use multi-grid instead
- adaptive grid (treated locally, only interacts with valence electrons)
- periodic boundary conditions
- find some small but interesting problems "along the way"

Goals for "May"

- port matlab code to GPU
- demonstrate GPU speed for simple H solid
- start looking for interesting problems, e.g.
    - how to avoid orthogonalization and handle degenerate states
    - how to extrapolate out errors
    - GPU efficiency
- real space
- periodic boundary conditions
- 1 GPU
- kohn-sham (dirac-kohn-sham too much work, but keep in mind)
- adaptive grid, all electron
- multigrid approaches, eigenvalue solver
- understand todd approach better?
- make more modular
- don't build hamiltonian like gpaw to save memory?