GPAW!

Massively parallel electronic structure calculations with Python software

Jussi Enkovaara

Software Engineering

CSC – the finnish IT center for science

CSC

GPAW

- **Software package for electronic structure calculations within the density-functional theory**
- **Python + C programming languages**
- **Massively parallelized**
- **Open source software licensed under GPL**

www.csc.fi/gpaw

wiki.fysik.dtu.dk/gpaw

Collaboration

Tech. Univ. of Denmark
- J. J. Mortensen
- M. Dulak
- C. Rostgaard
- A. Larsen
- K. Jacobsen

Helsinki Univ. of Tech.
- L. Lehtovaara
- M. Puska
- R. Nieminen
- T. Eirola

Tampere Univ. of Tech.
- J. Ojanen
- M. Kuisma
- T. Rantala

Jyväskylä University
- M. Walter
- O. Lopez
- H. Häkkinen

Åbo Akademi
- J. Stenlund
- J. Westerholm

- Funding from TEKES

Outline

- ➢**Density functional theory**
- ➢**Uniform real-space grids**
- ➢**Parallelization**
  - domain decomposition
  - Python + C implementation
  - results about parallel scaling
  - future prospects
- ➢**Summary**

Density functional theory

- Calculation of material properties from basic quantum mechanics
- First-principles calculations, atomic numbers are the only input parameters
- Currently, maximum system sizes are typically few hundred atoms or 2-3 nm
- Numerically intensive simulations, large consumer of supercomputing resources

Density functional theory

- **Kohn-Sham equations in the projector augmented wave method**

$$(-\frac{\nabla^2}{2} + V_{eff}(r))\psi_i(r) = e_i\psi_i(r)$$

$$n(r) = \sum_i |\psi_i(r)|^2$$

$$V_{eff}(r) = V_{ext}(r) + V_{Coul}(r) + V_{xc}(n(r))$$

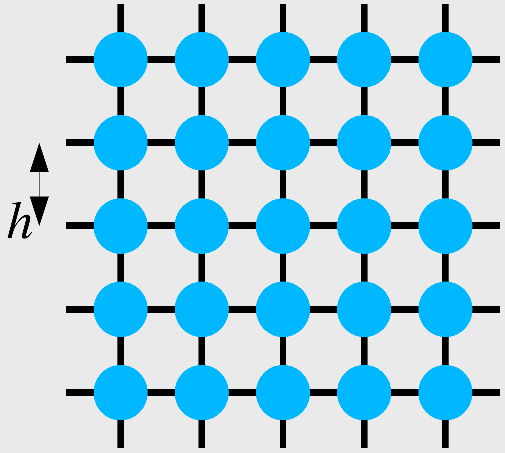
$$\nabla^2 V_{Coul}(r) = 4\pi n(r)$$

$$V_{eff}(r) = \tilde{V}_{eff}(r) + \sum_a V_{nl}^a(r, r')$$

- **Self-consistent set of single particle equations for N electrons**
- **Non-locality of effective potential is limited to atom-centered augmentation spheres**

Real space grids

- Wave functions, electron densities, and potentials are represented on uniform grids.
- Single parameter, grid spacing h



- Accuracy of calculation can be improved systematically by decreasing the grid spacing

Finite differences

- Both the Poisson equation and kinetic energy contain the Laplacian which can be approximated with finite differences

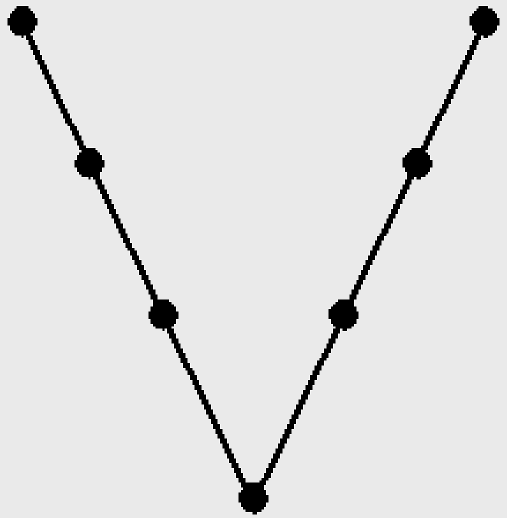
$$\frac{\partial^2 \psi(x_i)}{\partial x^2} = \sum_{n=-N}^{N} C_n \psi(x_i + nh) + O(h^{2N+2})$$

- Accuracy depends on the order of the stencil N
- Sparse matrix, storage not needed
- Cost in operating to wave function is proportional to the number of grid points

Multigrid method

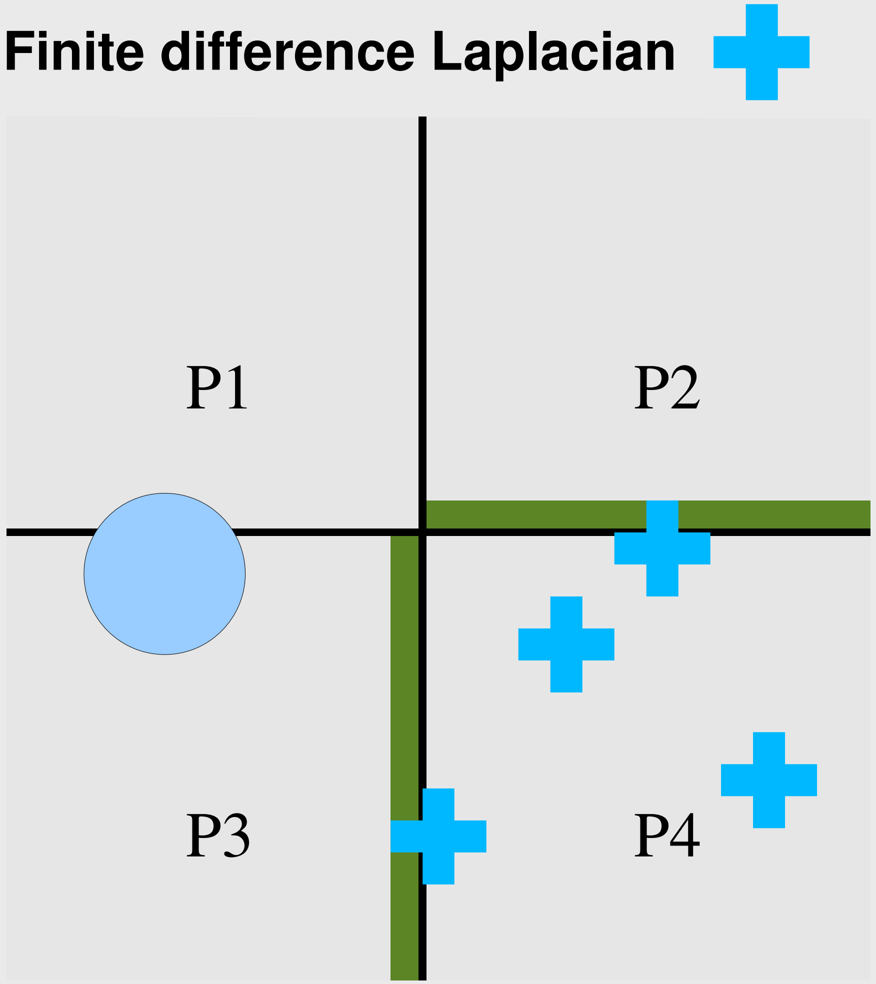
- General framework for solving differential equations using a hierarchy of discretizations

- **Recursive V-cycle**
- **Transform the original equation to a coarser discretization**
  - restriction operation
- **Correct the solution with results from coarser level**
  - interpolation operation



Domain decomposition

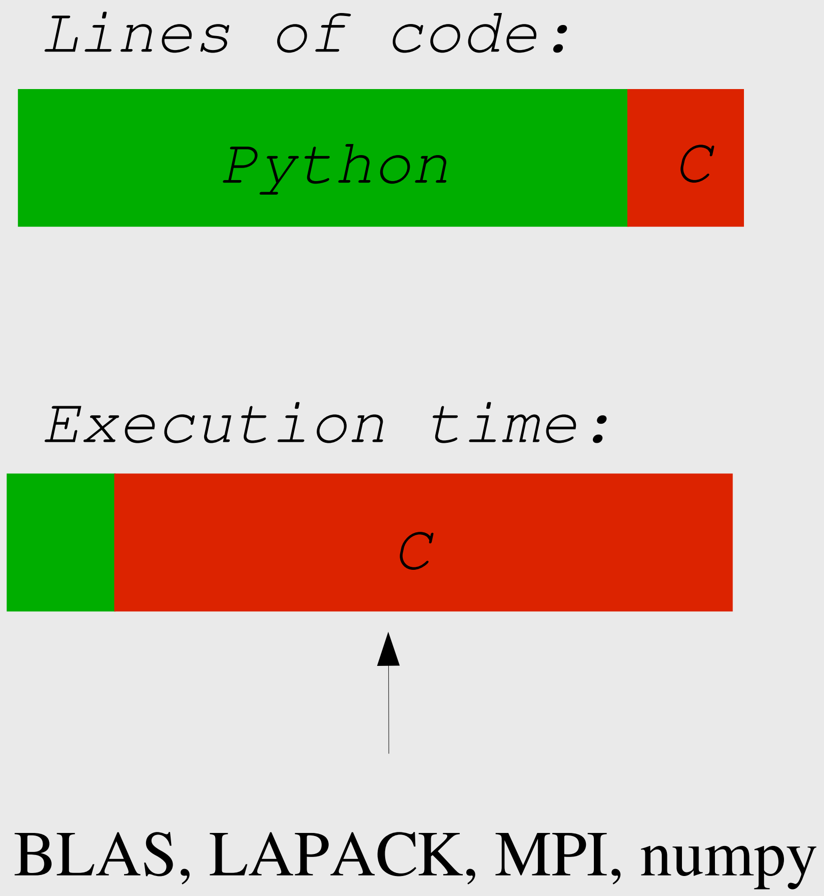
Finite difference Laplacian



- **Domain decomposition: real-space grid is divided to different processors**
- **Communication is needed:**
  - Laplacian, nearly local
  - restriction and interpolation in multigrid, nearly local
  - integrations over augmentation spheres
- **Total amount of communication is small**

Python

- **Modern, object-oriented general purpose programming language**
- **Rapid development**
- **Interpreted language**
  - possible to combine with C and Fortran subroutines for time critical parts
- **Installation can be be intricate in special operating systems**
  - Catamount, CLE
  - BlueGene
- **Debugging and profiling tools are often only for C or Fortran programs**



Python overhead

- **Execution profile of serial calculation (bulk Si with 8 atoms)**

```
   ncalls   tottime   percall   cumtime   percall filename:lineno(function)
    76446    28.000     0.000    28.000     0.000 :0(add)
    75440    26.450     0.000    26.450     0.000 :0(integrate)
    26561    13.316     0.001    13.316     0.001 :0(apply)
     1556    12.419     0.008    12.419     0.008 :0(gemm)
       80    11.842     0.148    11.842     0.148 :0(r2k)
       84     4.470     0.053     4.470     0.053 :0(rk)
     3040     2.957     0.001    10.920     0.004 preconditioner.py:29(__call__)
    14130     2.815     0.000     2.815     0.000 :0(calculate_spinpaired)
       76     2.553     0.034   104.253     1.372 rmm_diis.py:39(iterate_one_k_point)
   103142     2.390     0.000     2.390     0.000 :0(matrixproduct)
```

- **88 % of total time is spent in C-routines**
- **69 % of total time is spent in BLAS**
- **In parallel calculations also Python parts are run on parallel**

Implementation details

- **Message passing interface (MPI)**
- **Finite difference Laplacian, restriction and interpolation operators are implemented in C**
  - MPI-calls directly from C
- **Higher level algorithms are implented in Python**
  - Python interfaces to BLAS and LAPACK
  - Python interfaces to MPI functions

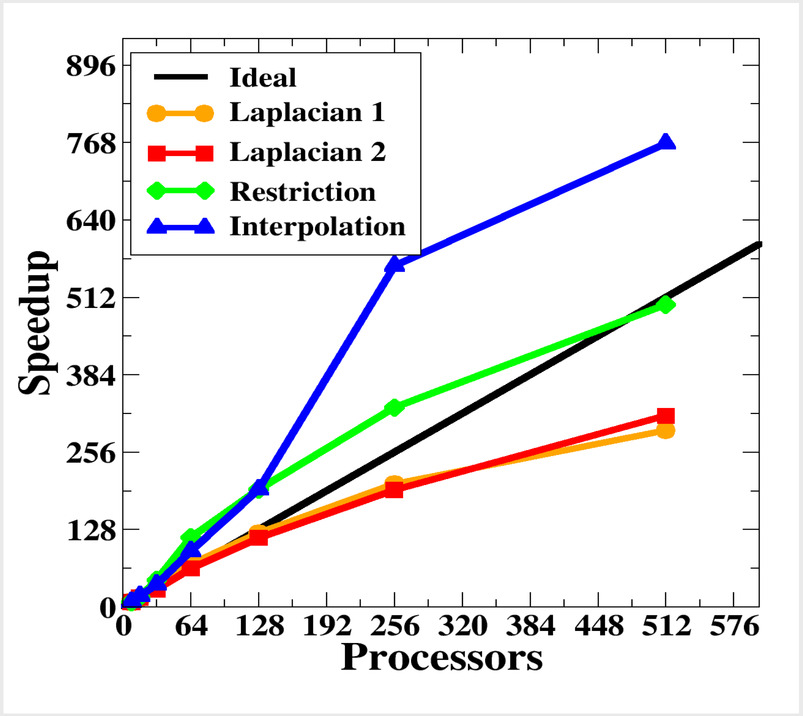
```
# Calculate the residual of pR_G, dR_G = (H - e S) pR_G
hamiltonian.apply(pR_G, dR_G, kpt)
overlap.apply(pR_G, self.work[1], kpt)
axpy(-kpt.eps_n[n], self.work[1], dR_G)

RdR = self.comm.sum(real(npy.vdot(R_G, dR_G)))
```

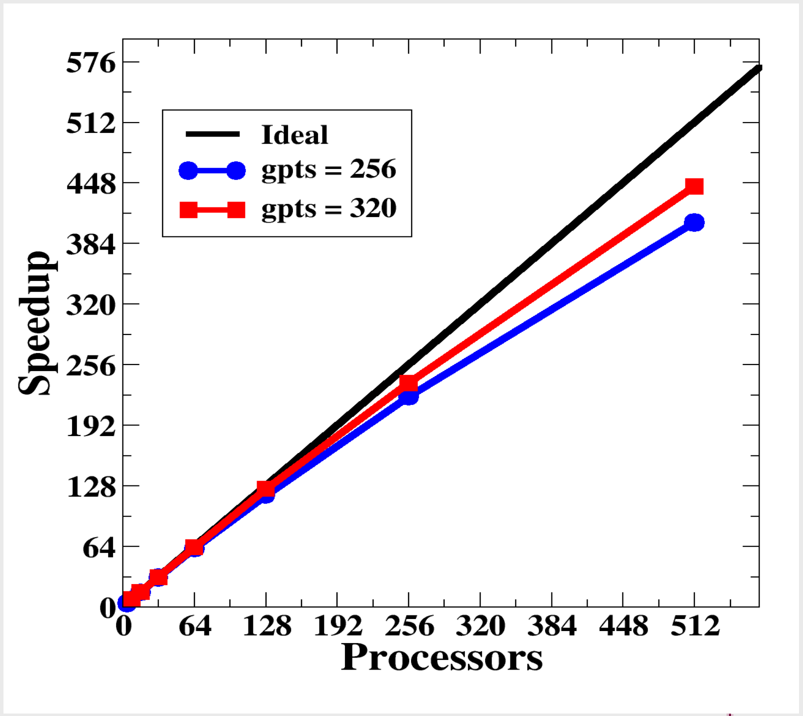
Python code sniplet from iterative eigensolver

Parallel scaling of multigrid operations

- Finite difference Laplacian, restriction and interpolation applied to wave functions
- Poisson equation (double grid)



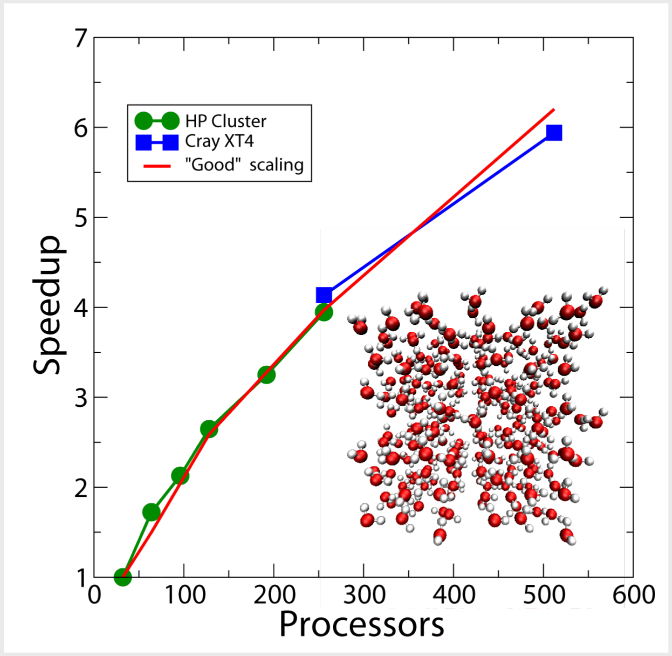
Multigrid operations in 128^3 grid

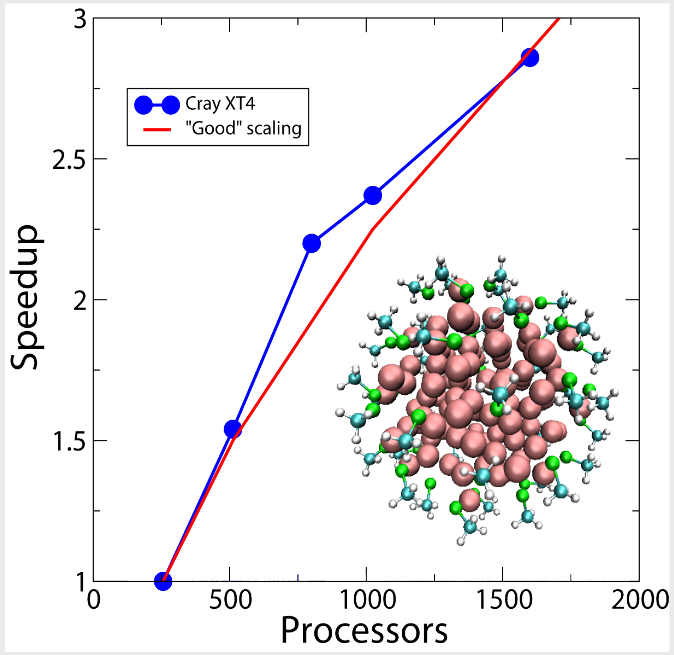


Poisson solver

Parallel scaling of whole calculation

- **Realistic test and production systems**





256 water molecules

768 atoms, 1024 bands, 96 x 96 x 96 grid

Au-(SMe) cluster

327 atoms, 850 bands, 160 x 160 x 160 grid

Bottlenecks in parallel scalability

- **Load balancing**
  - atomic spheres are not necessarily divided evenly to domains
- **Latency**
  - Currently, only single wave function is communicated at time, many small messages
  - minimum domain dimension 10-20
- **Serial $O(N^3)$ parts**
  - insignificant in small to medium size systems
  - starts to dominate in large systems

Additional parallelization levels

- **In (small) periodic systems parallelization over k-points**
  - almost trivial parallelization
  - number of k-points decreases with increasing system size
- **Parallelization over spin in magnetic systems**
  - trivial parallelization
  - generally, doubles the scalability
- **Parallelization over electronic states (work in progress)**
  - In ground state calculations, orthogonalization requires all-to-all communication of wave functions
  - In time-dependent calculations orthogonalization is not needed, almost trivial parallelization

Summary

- GPAW is a program package for electronic structure calculations within density-functional theory
- Implementation in Python and C
- Domain decomposition scales to over 1000 cores
- Additional parallelization levels could extend the scalability to > 10 000 cores

www.csc.fi/gpaw

wiki.fysik.dtu.dk/gpaw