Python in High performance computing

Jussi Enkovaara and Martti Louhivuori

CSC – Tieteen tietotekniikan keskus Oy
CSC – IT Center for Science Ltd.

Outline

- High performance computing
- Python and high performance
- Python challenges

High performance computing

- Advanced computational problems
- Scientific computing
- Large scale numerical problems
- Massively parallel supercomputers
- “Every FLOP matters”
- Today's HPC systems: 10^{12}-10^{15} FLOP/s

High performance computing

- Traditionally, HPC focuses on the computer aspect of performance
- Compiled programming languages
  - Fortran, C, C++
- Parallel computing
  - Message passing paradigm (MPI)
  - Hybrid (threads + message passing)
- Emerging technologies

Why Python?

- High performance for programmer
  - Fast program development
  - Simple syntax
  - Easy to write well readable code
  - Large standard library
  - Lots of third party libraries
- High computer performance with Python?
  - NumPy, C-extensions, optimized libraries

GPAW

- Software package for quantum mechanical simulations of nanostructures
- Implemented in combination of Python and C
- Massively parallelized
- Open source under GPL
- 20-30 developers in Denmark, Finland, Sweden, Germany, UK, US

J. Enkovaara *et al*.**,** J. Phys. Condens. Matter **22,** 253202 (2010)

wiki.fysik.dtu.dk/gpaw

Python benefits

- Lists and dictionaries provide powerful data structures for set-up data
- Convenient file and text manipulation
- Dynamic typing:
  - Depending on input, computations are performed either with real or complex numbers

```
def apply_local_potential(self, psit_nG, Htpsit_nG, s):
    ...
    vt_G = self.vt_sG[s]
    for psit_G, Htpsit_G in zip(psit_nG, Htpsit_nG):
        Htpsit_G += psit_G * vt_G
    ...
```

Numpy – fast array interface

- Standard Python is not well suitable for numerical computations
  - lists are very flexible but also slow to process in numerical computations
- Numpy adds a new **array** data type
  - static, multidimensional
  - fast processing of arrays
  - some linear algebra, random numbers

Array operations

- Most operations for NumPy arrays are done element-wise
- Operations are carried out in compiled code
  - e.g. loops in C-level
  - NumPy code should be "vectorized"
- Numpy has special functions which can work with array arguments
  - sin, cos, exp, sqrt, log, ...
- Performance closer to C than "pure" Python

Linear algebra

- Numpy has routines for basic linear algebra
  - Numpy can be linked to optimized linear algebra libraries (BLAS/LAPACK)
- Performance in matrix multiplication
  - C = A * B
  - matrix dimension 200
  - pure python: 5.30 s
  - naive C: 0.09 s
  - numpy.dot: 0.01 s
- (GPAW uses custom BLAS interface)

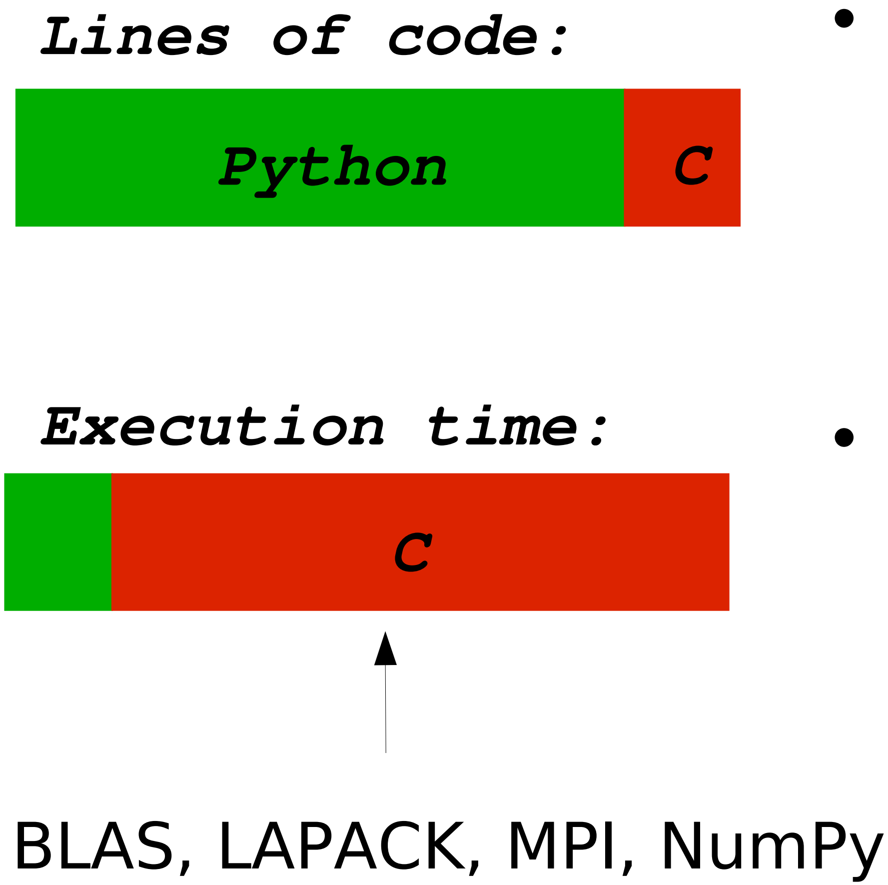
C - extensions

- Some times there are time critical parts of code which would benefit from compiled language
- It is relatively straightforward to create a Python interface to C-functions
- Some tools can simplify the interfacing
  - SWIG
  - Cython, pyrex

C-extensions

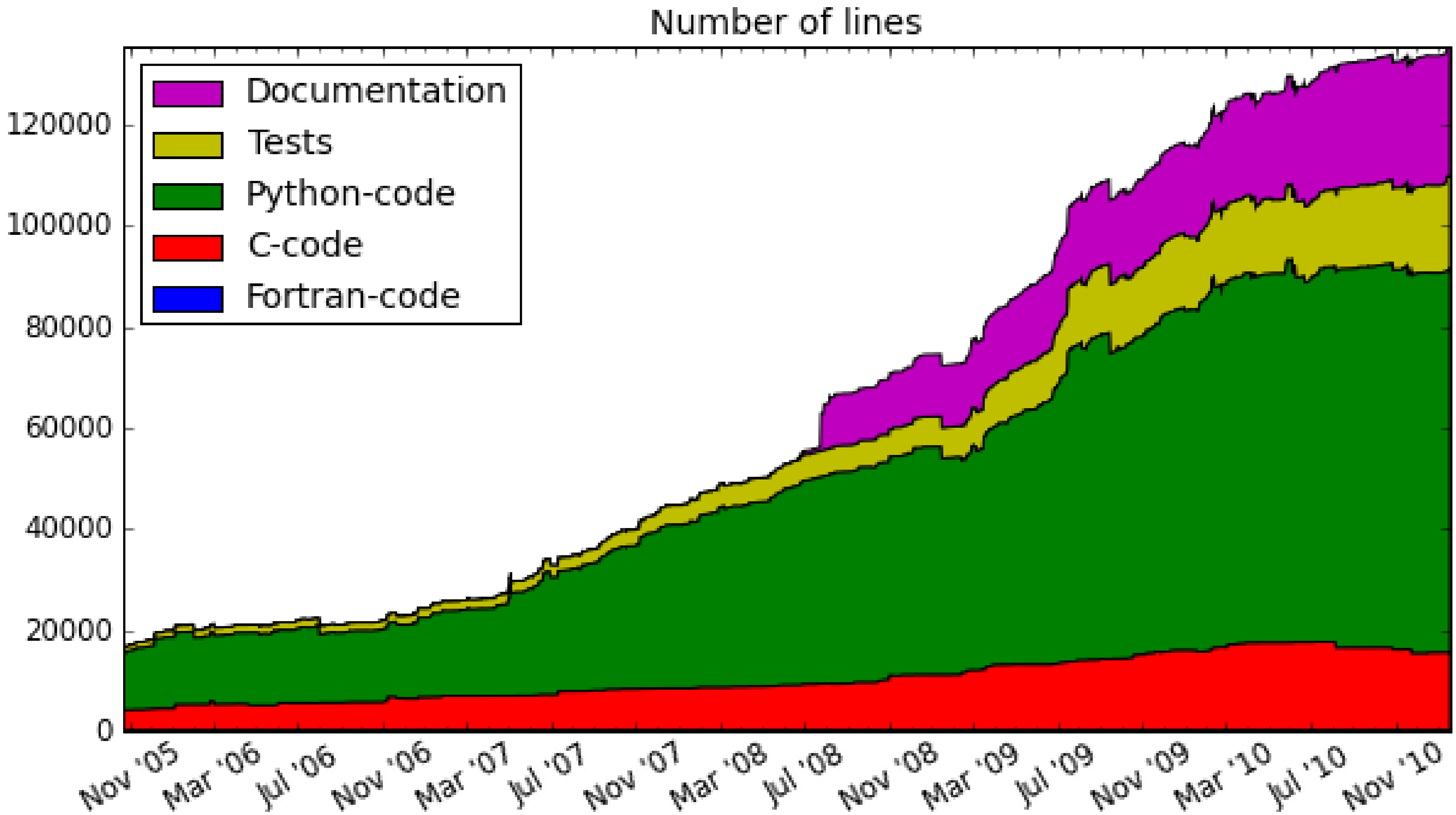
- GPAW uses custom C-extensions for time-critical numerical kernels
- Custom interfaces also to optimized libraries
- NumPy arrays used as data containers

Python + C implementation



- Python (+ NumPy)
  - Fast development
  - Slow execution
  - High level algorithms
- C
  - Fast execution
  - Slow development
  - Main numerical kernels

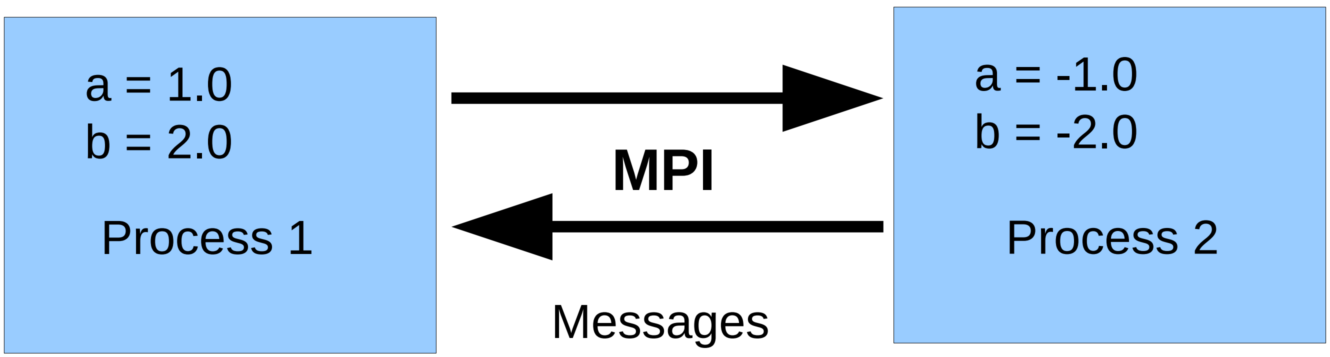
Python + C implementation



Time line of GPAW's codebase

Parallelization in GPAW

- Message passing paradigm
  - Independent processes
  - Memory is local to processes
  - Processes exchange data by sending and receiving messages
  - MPI : standard, API, and library for message passing applications



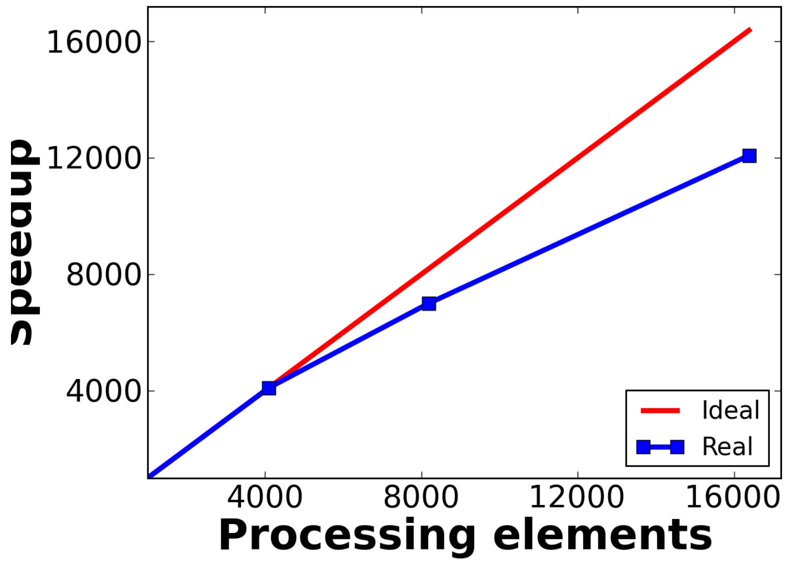
Parallelization in GPAW

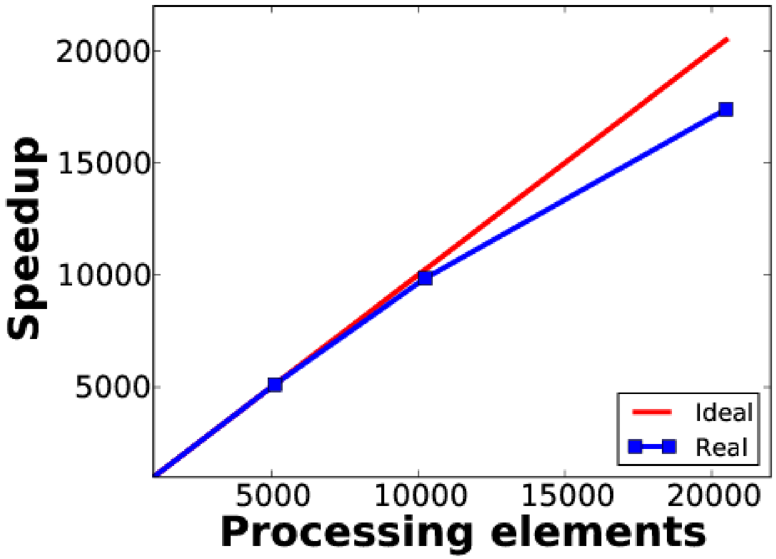
- N independent Python interpreters are launched
  - Interpreter processes use MPI for data exchange
- Custom Python interface to MPI
- MPI calls both from Python and from C

```
# MPI calls within the apply C-function
hamiltonian.apply(psi, hpsi)
# Python interface to MPI_Reduce
norm = gd.comm.sum(np.vdot(psi,psi))
```

- All the normal parallel programming concerns

Parallel scalability





- Ground state DFT
  - Blue Gene P, Argonne
- TD-DFT
  - Cray XT5 Jaguar, Oak Ridge

Python challenges

Minor challenges

- Special operating systems in supercomputers
  – No dynamic linking
- Debugging and profiling
  – Python contains debugger and profiler for serial applications
- Most parallel development tools support only Fortran and C

Python initialization

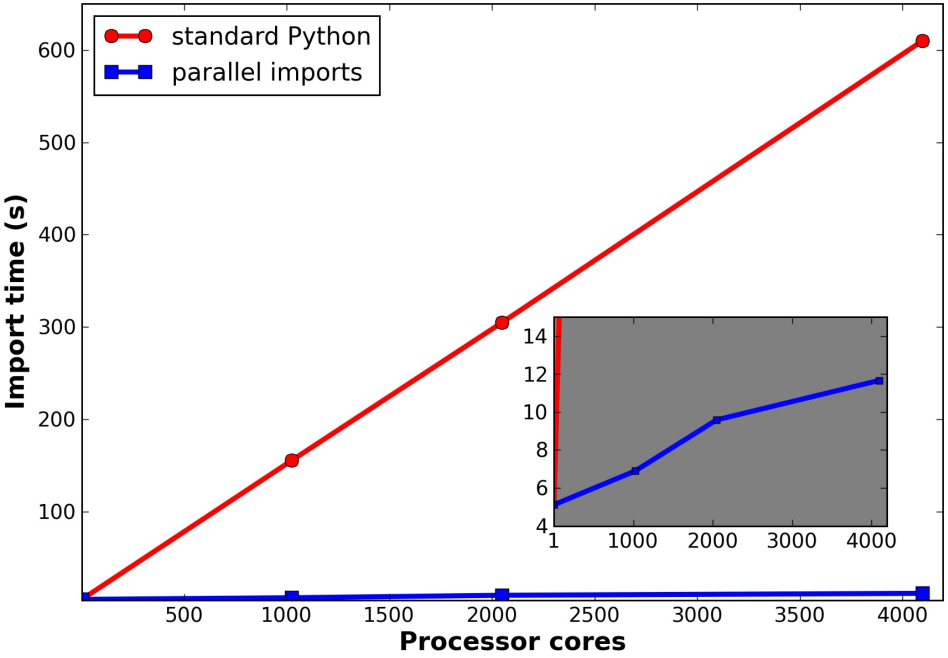
- **import** statements in Python trigger lots of small-file I/O
- **Import foo** triggers fopen/stat system calls:
  - Directory foo, foo.so, foomodule.so, foo.py, foo.pyc
  - In working directory, default directories, PYTHONPATH
- GPAW initialization:
  ~350 imports, ~3400 fopen/stat calls

Python initialization

- In parallel calculations all processes perform the same I/O
- Introduces severe bottleneck with large number (> 512) of processes
- In Blue Gene P, importing NumPy + GPAW specific modules with ~32 000 processes can take 45 minutes!

Python initialization

- Create special Python interpreter
  - Single process does I/O in imports, data broadcast to others with MPI



Global interpreter lock

- There is threading support in Python level
- Global interpreter lock in (CPython) interpreter:
  - Only single thread is executed at time
- Threading has to be implemented in C-extensions
  - Higher granularity than algorithmically necessary

Summary

- Python can be used in massively parallel high performance computing
- Combining Python with C one gets best of both worlds
  – High performance for programmer
  – High performance execution
- GPAW: ~25 % of peak performance with 2048 cores

Acknowledgements

- Partnership for Advanced Computing in Europe (PRACE), EU FP7 programme
- Finnish Technology Agency, MASI program
- Argonne Leadership Computing Facility, US DoE
- Whole GPAW development team
  - Marcin, Ask, Christian, Carsten, Lauri, Michael, Lara, Mikael, Olga, Mikkel, Jun, Thomas, Jess, Mathias, Carsten, Kristian, Jacob, Hannu, Tapio, Martti, Risto, ...

Acknowledgements

Questions ?