CP2K: parallel algorithms

Joost VandeVondele
University of Zurich

Joost.VandeVondele@pci.uzh.ch

CP2K: the swiss army knife of molecular simulation

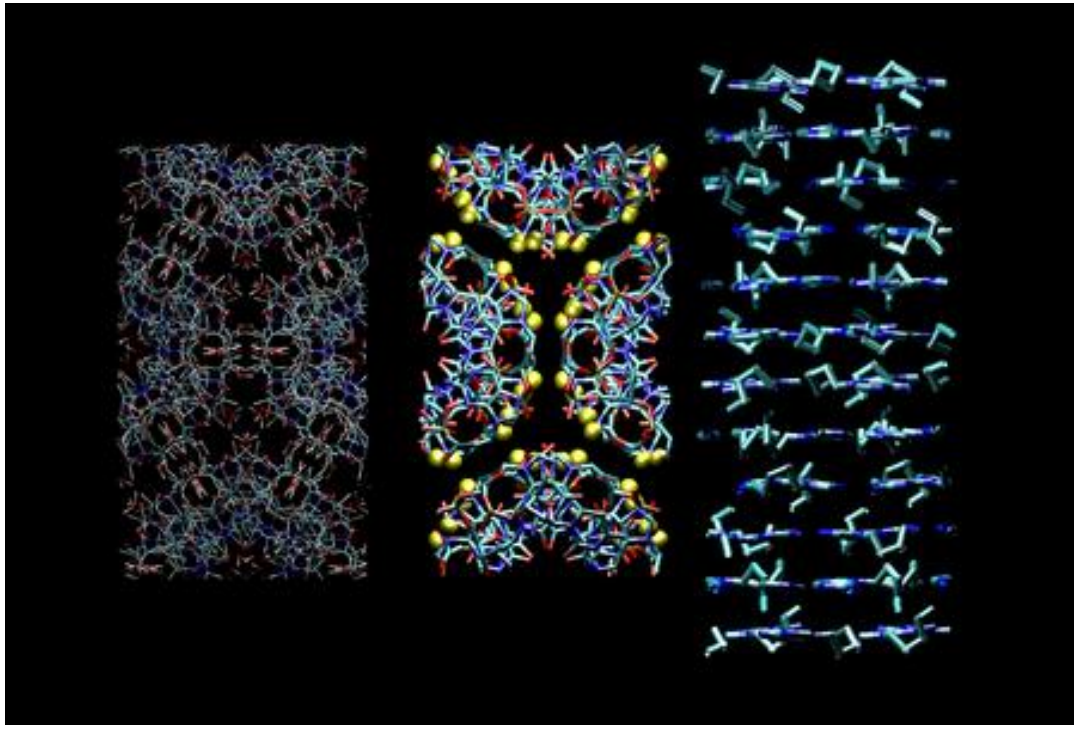
- A wide variety of models Hamiltonians
  - classical
  - semi-empirical
  - local and non-local DFT
  - Combinations (e.g. QM/MM)
- Various algorithms
  - Molecular dynamics & Monte Carlo
    - NVE, NVT, NPT
  - Free energy and PES tools
  - Ehrenfest MD
- Properties
  - Vibrational
  - NMR, EPR, XAS, TDDFT
- Open source & rapid development
  - 600.000 lines of code

CP2K: the swiss army knife of molecular simulation

- A wide variety of models Hamiltonians
  - classical
  - semi-empirical
  - local and non-local DFT
  - Combinations (e.g. QM/MM)
- Various algorithms
  - Molecular dynamics & Monte Carlo
    - NVE, NVT, NPT
  - Free energy and PES tools
  - Ehrenfest MD
- Properties
  - Vibrational
  - NMR, EPR, XAS, TDDFT
- Open source & rapid development
  - 600.000 lines of code

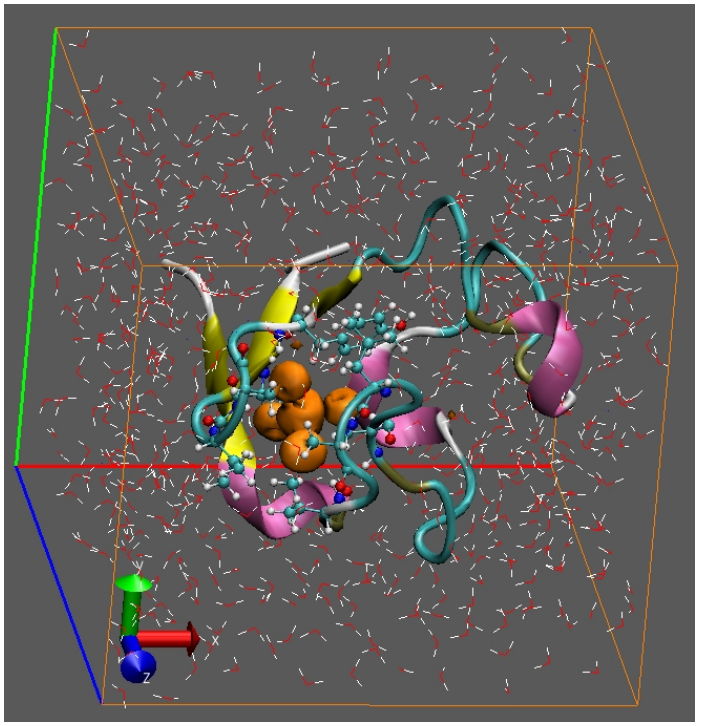
Accurate DFT for large systems

DNA crystal[1)]



2388 atoms

Solvated metallo-protein[2)]



2825 atoms

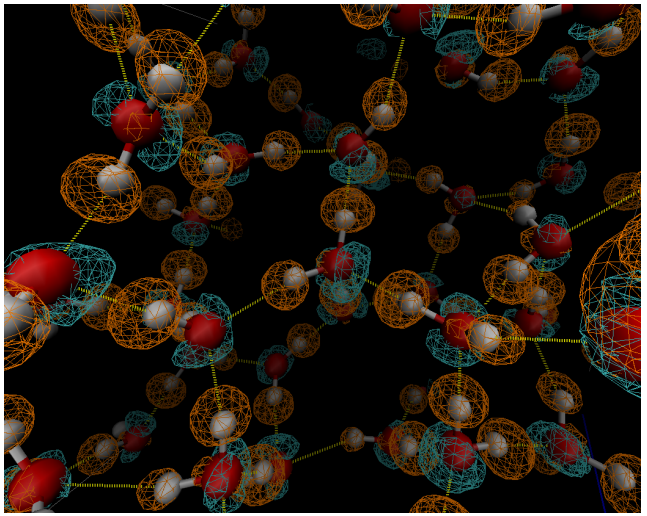
Linear scaling construction of the Kohn-Sham matrix,
robust and efficient electronic minimization

1) J. VandeVondele, J. Hutter, 2003, JCP 118, 4365-4369

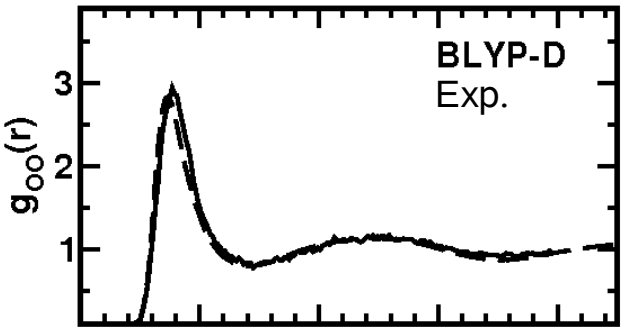
2) Sulpizi, M.; Raugei, S.; VandeVondele, J.; Carloni, P.; Sprik, M. 2007. JPCB 111, 3969.

Robust Ab initio MD

'Simple' liquids

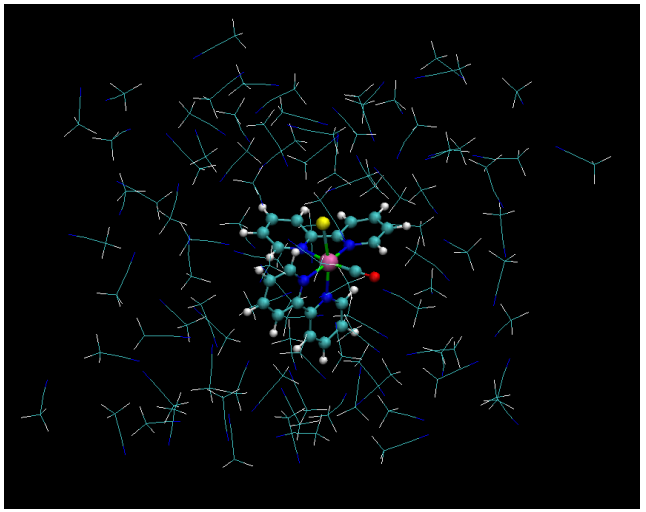


Water electronic structure[1]



Water liquid structure[2]

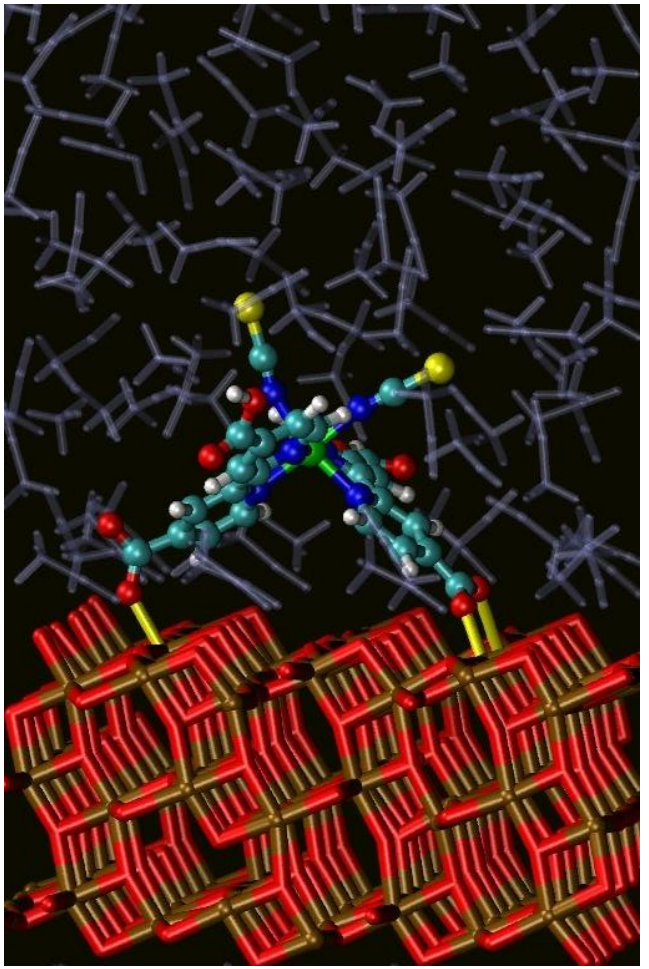
Solutes in explicit solvent



$Ru(bpy)_2COCl$ in acetonitrile, $[21.43Å]^3$ or 620 Atoms

e.g. Redox properties

Complex Interfaces

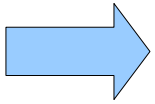


In situ IR spectroscopy (1300 atoms)

1) M. Guidon, F. Schiffmann, J. Hutter, J. VandeVondele, JCP 128, 214104
2) J. Schmidt, J. VandeVondele, W. Kuo, D. Sebastiani, J. I. Siepmann, J. Hutter, and C.J. Mundy

Intrinsically parallel methods in molecular simulation

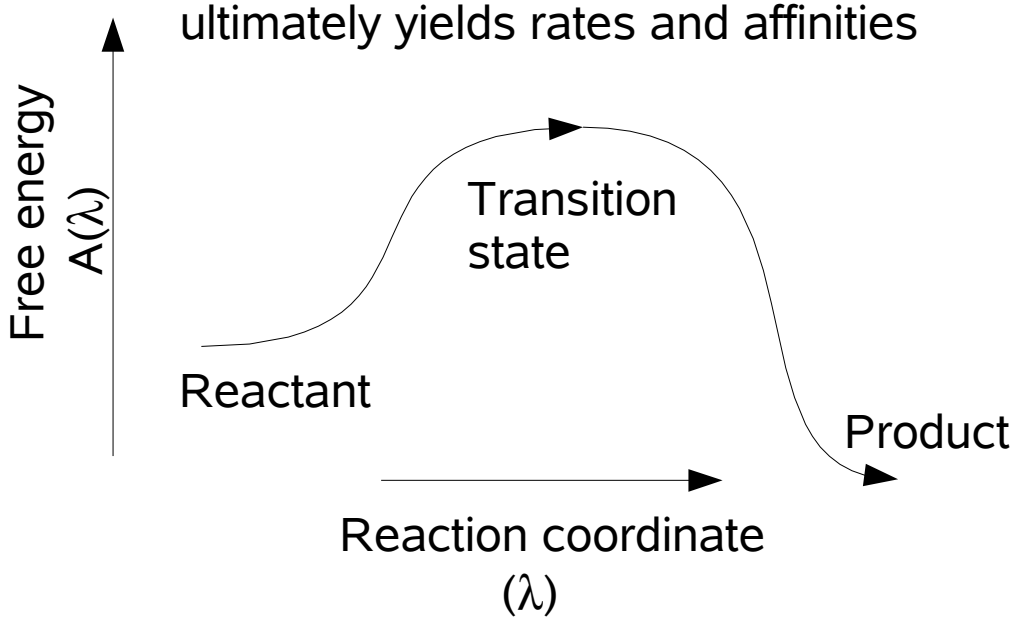
- Good science sometimes requires embarrassingly parallel methods
  - Good statistics from independent simulations
  - Free energy profiles
  - Properties sampled over a large number of configurations
  - Parameter scans / sensitivity analysis
  - Global optimization



Becomes increasingly important:
from anecdotal to systematic and quantitative ab initio molecular dynamics results

Example: free energy profiles

Essential quantitative information for chemistry, biology, material science....
'free energy' replaces 'total energy' in complex systems at finite temperature,
ultimately yields rates and affinities



$$\Delta A_{a\to b} = A(\lambda_b) - A(\lambda_a) = \int_{\lambda_a}^{\lambda_b} \frac{dA(\lambda)}{d\lambda} d\lambda$$

$$\Delta A_{a\to b} = \int_{\lambda_a}^{\lambda_b} \left\langle \frac{\partial H(x, p_x; \lambda)}{\partial \lambda} \right\rangle_{\lambda} d\lambda$$

For each value of λ, the integrand can be obtained
from a (long) ab initio molecular dynamics / Monte Carlo simulation.

The integral needs to be discretized over e.g. ~16 λ-values.

Do the obvious: parallellize integration as well as integrand evaluation

Basic Computational local DFT

$$n(r)=\sum_{\mu\nu} P^{\mu\nu}\varphi_\mu(r)\varphi_\nu(r)$$

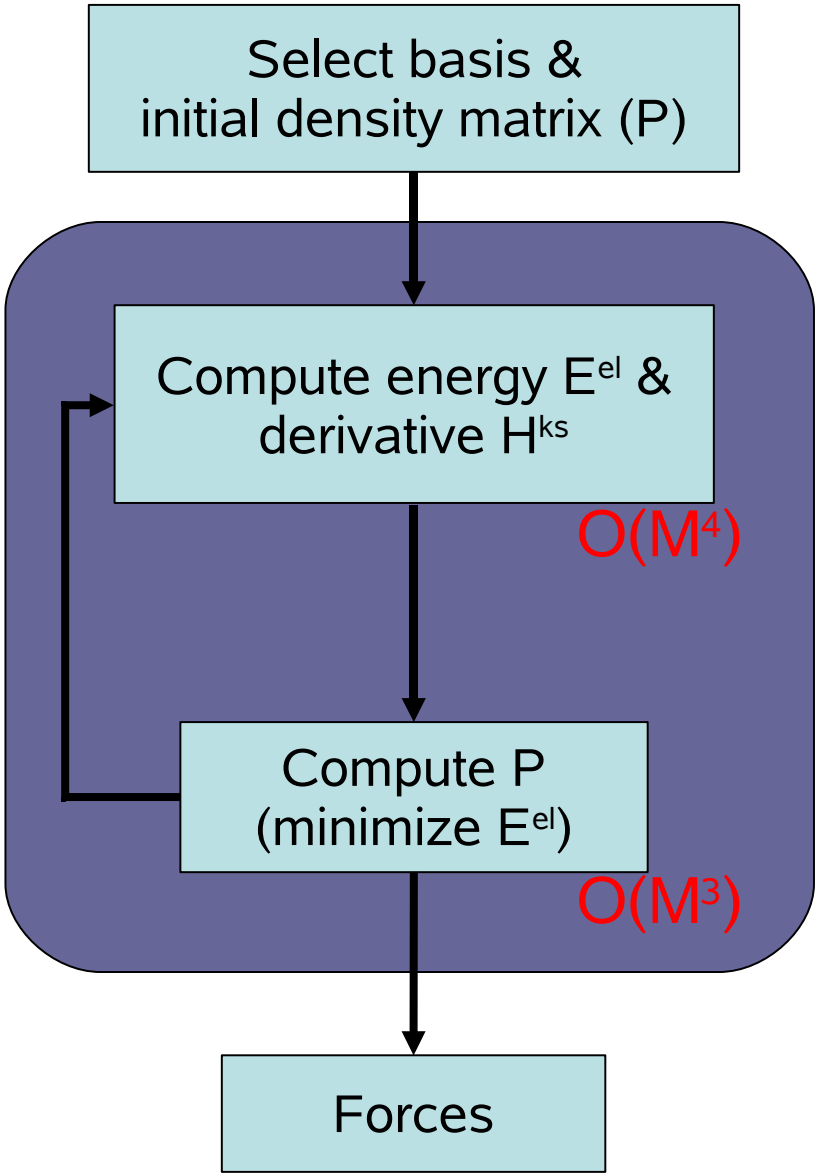
$$E^{el}[P^{\mu\nu}]=\sum_{\mu\nu} P^{\mu\nu}\int \varphi_\mu(r)(-\frac{\Delta}{2})\varphi_\nu(r)$$

$$+\sum_{\mu\nu} P^{\mu\nu}\int\int \varphi_\mu(r)V^{PP}_{sep}(r,r')\varphi_\nu(r')$$

$$+\frac{1}{2}\int\int \frac{n(r)n(r')}{|r-r'|}$$

$$+\int n(r)\varepsilon_{xc}[n](r)$$

Formally O(M⁴)



DFT in CP2K: Quickstep

Combining the best of two worlds:

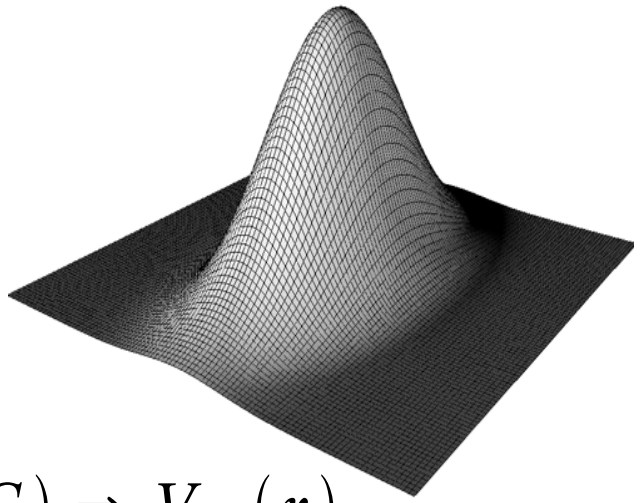
- Basis functions : Gaussians
  - compact
  - sparse H^{ks} (and P)
- Density : Plane waves
  - Auxiliary basis / Grid
  - FFT for electrostatics

J. VandeVondele, M. Krack, F. Mohamed, M. Parrinello, T. Chassaing and J. Hutter, Comp. Phys. Comm. 167, 103 (2005).

Coulomb energy: A linear scaling algorithm (GPW)

Real space (rs) density mapping and integration
Fourier space (FFT) for the coulomb problem

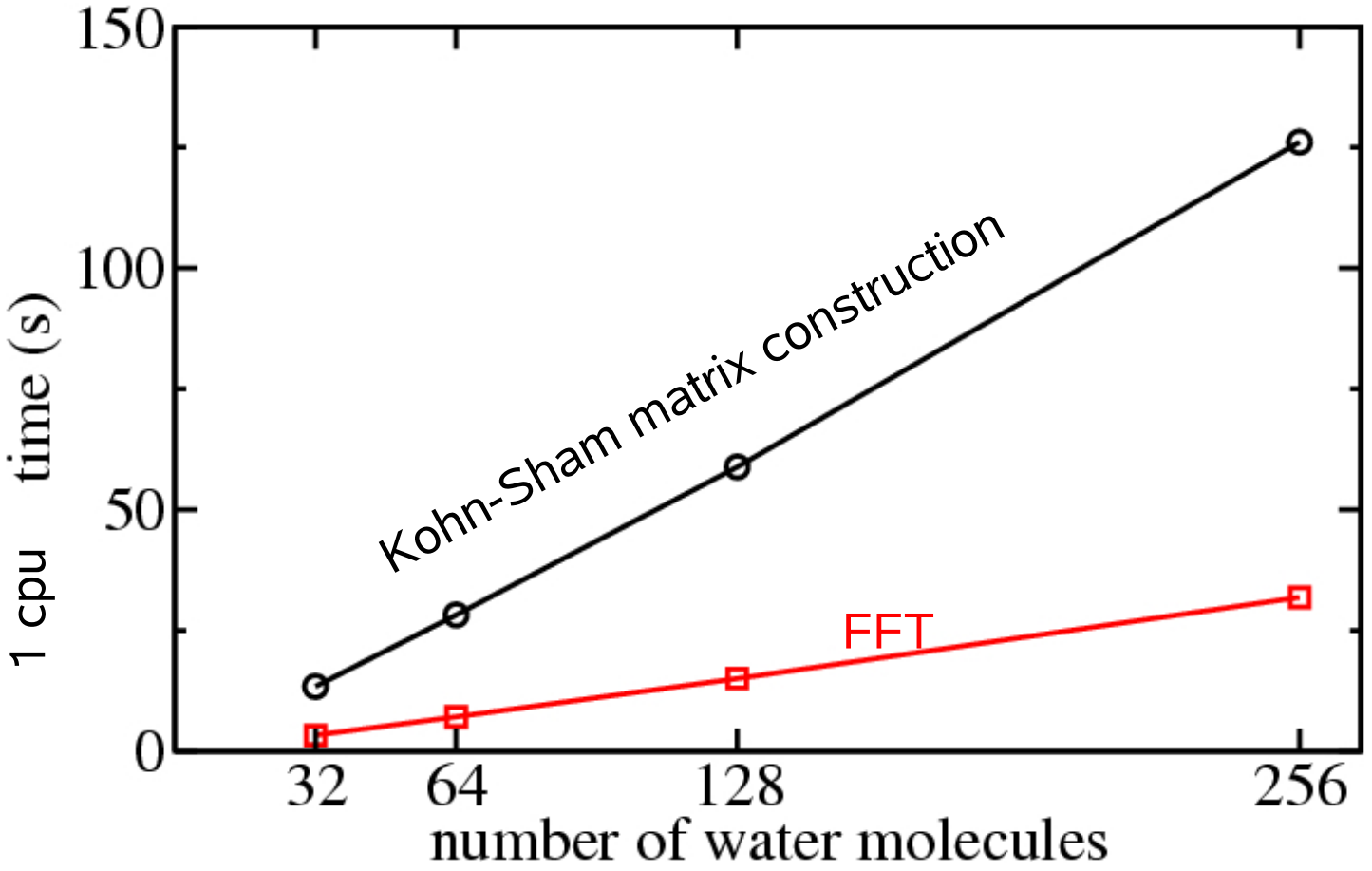
$$\sum_{\mu\nu} P^{\mu\nu}\varphi_\mu(r)\varphi_\nu(r) \underbrace{\Rightarrow}_{RS} n(r) \underbrace{\Rightarrow}_{FFT} n(G)$$



$$\Rightarrow V_H(G)=\frac{4\pi n(G)}{G^2}, \quad E_H=\Omega\sum_G n^{¿}(G)V_H(G) \underbrace{\Rightarrow}_{FFT} V_H(r)$$

$$\underbrace{\Rightarrow}_{RS} V_{\mu\nu}=\int V_H(r)\varphi_\mu(r)\varphi_\nu(r)$$

System size : linear scaling



DZV(d,p) = 736 @ 32

linear scaling also for 3D systems !

Basic Computational local DFT

$$n(r)=\sum_{\mu\nu} P^{\mu\nu}\varphi_\mu(r)\varphi_\nu(r)$$

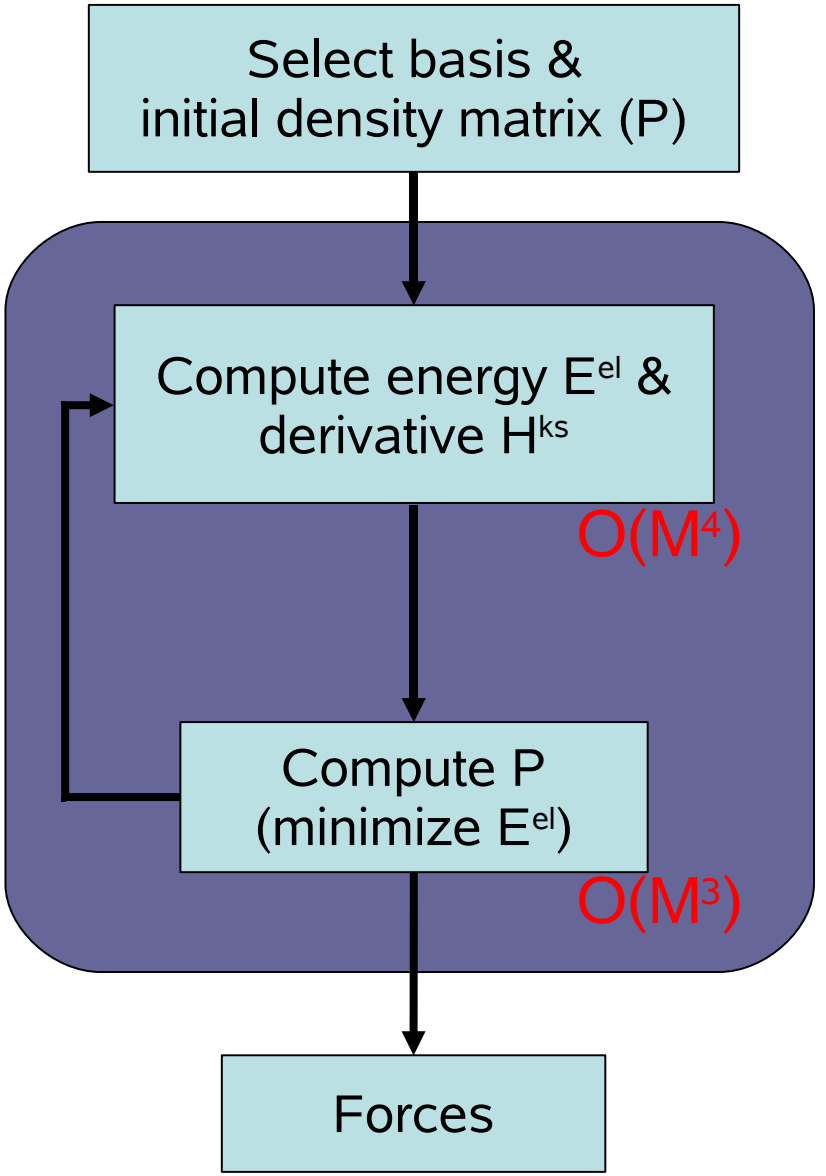
$$E^{el}[P^{\mu\nu}]=\sum_{\mu\nu} P^{\mu\nu}\int \varphi_\mu(r)(-\frac{\Delta}{2})\varphi_\nu(r)$$

$$+\sum_{\mu\nu} P^{\mu\nu}\int\int \varphi_\mu(r)V^{PP}_{sep}(r,r')\varphi_\nu(r')$$

$$+\frac{1}{2}\int\int \frac{n(r)n(r')}{|r-r'|}$$

$$+\int n(r)\varepsilon_{xc}[n](r)$$

Formally O(M⁴)



Orbital transformations (OT)

A cubic, very robust algorithm avoiding the of traditional diagonalization

- New variables

$$C(X) = C_0 \cos\left(\sqrt{X^T S X}\right) + X \frac{\sin\left(\sqrt{X^T S X}\right)}{\sqrt{X^T S X}}$$

$$X^T S C_0 = 0 \qquad C(X)^T S C(X) = 1 \;\; \forall X$$

- Direct minimization of $E_{ks}[\{X\}]$
- Linear constraint -> guaranteed convergence!

J. VandeVondele, J. Hutter, *J. Chem. Phys., 2003,* Vol. 118 No. 10, 4365-4369

Orbital transformations (OT)

- Large systems
  - CPU : MN^2 Memory : MN (M/N ~ 3 – 10)
  - Mostly (p)dgemm & dsyevd (N^3)
- Good preconditioners
  - M^2N (for non-sparse preconditioners)

favourable approach in parallel and for larger basis sets

	DZVP	TZVP	TZV2P	QZV2P	QZV3P
OT	0.50	0.60	0.77	0.87	1.06
SCALAPACK	6.02	8.40	13.80	17.34	24.59

J. VandeVondele, J. Hutter, *J. Chem. Phys., 2003,* Vol. 118 No. 10, 4365-4369

Intermediate Summary

CP2K has significant functionality

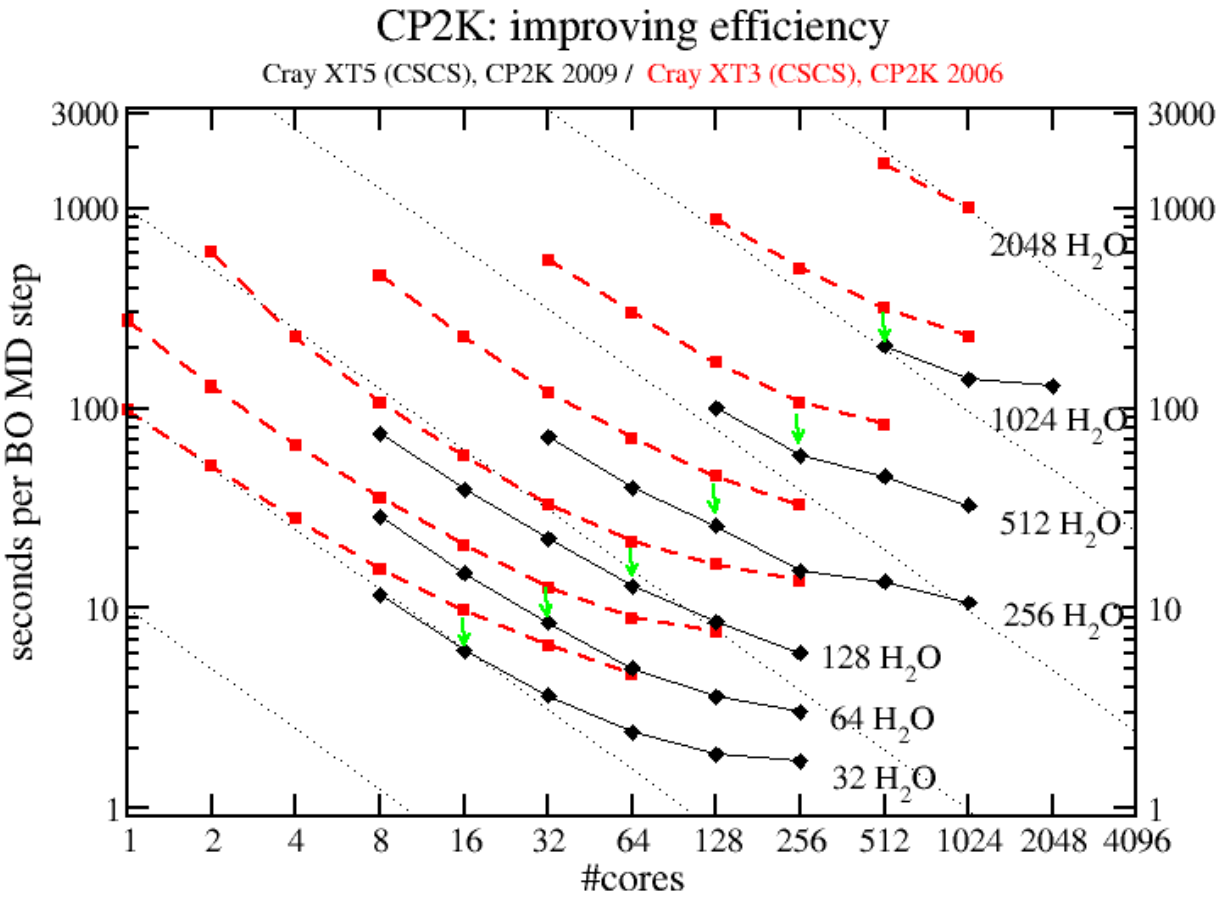
Is very efficient for large and inhomogenous systems

Smaller systems are dominated by GPW, larger systems by OT

?

Can the underlying method be parallellized well?

Parallel CP2K based on local DFT ?



GPW and OT can be parallelized relatively well, and progress is steady.

Weak scaling illustration

N water molecules on N cores of an XT5

System	CP2K efficiency	PDGEMM	M	N
32 H_2O	77.30	25.35	1280	128
64 H_2O	65.84	33.42	2560	256
128 H_2O	50.07	29.94	5120	512
256 H_2O	52.67	30.13	10240	1024
512 H_2O	50.78	38.30	20480	2048
1024 H_2O	70.78	61.52	40960	4096

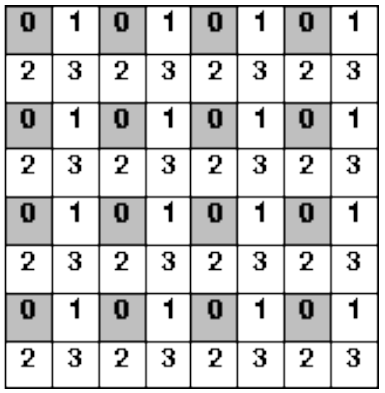
Weak scaling OK. Intermediate sizes in worst regime

nowadays the intermediate sizes (128-512) are interesting for AIMD

Data layouts in CP2K: scalapack

Dense (full) matrices are being used for wavefunction coefficients

2D block cyclic distribution



2D processor grid

Ncpu= P x Q

→ is the reason why CP2K performance is usually better with $2^{(2N)}$ tasks

Communication goes as M^2/P, computation as $M^3/Ncpu$

For increasingly large M, good performance is obtained

For CP2K, the limit of 1 block per CPU is can be reached, blocks are being scaled in size to avoid CPUs without load.

256 waters example (I)

The 'ultimate' operation of DFT based programs is the calculation of the overlap matrix between two set of Molecular Orbitals*, this is required to fullfil the wavefunction orthonormality criterium

C=Transpose(A)*B

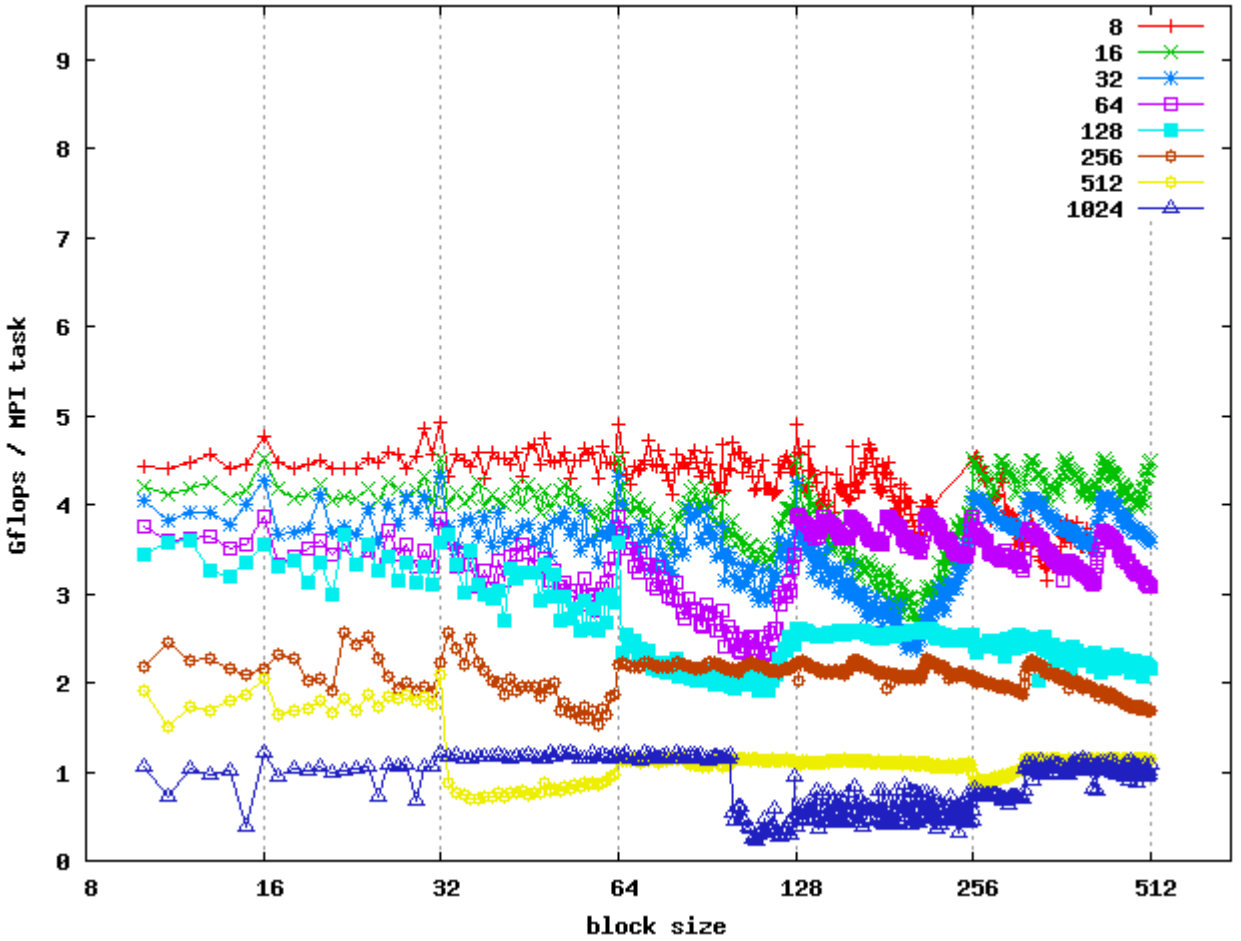
For CP2K/QS, we have A, B ~10240	x	1024
For PW codes we have A,B ~102400	x	1024

Good performance for this operation is essential

* assuming a non linear scaling wavefunction solver

PDGEMM Measurements

Performance vs block size for various number of MPI tasks



Best pdgemm performance
is about 50% of peak

Block size does matter,
for this system 32 is best
for the core counts considered

However, not always the case,
for larger and non-power of 2
matrices larger blocks
are found to be better (64/92).

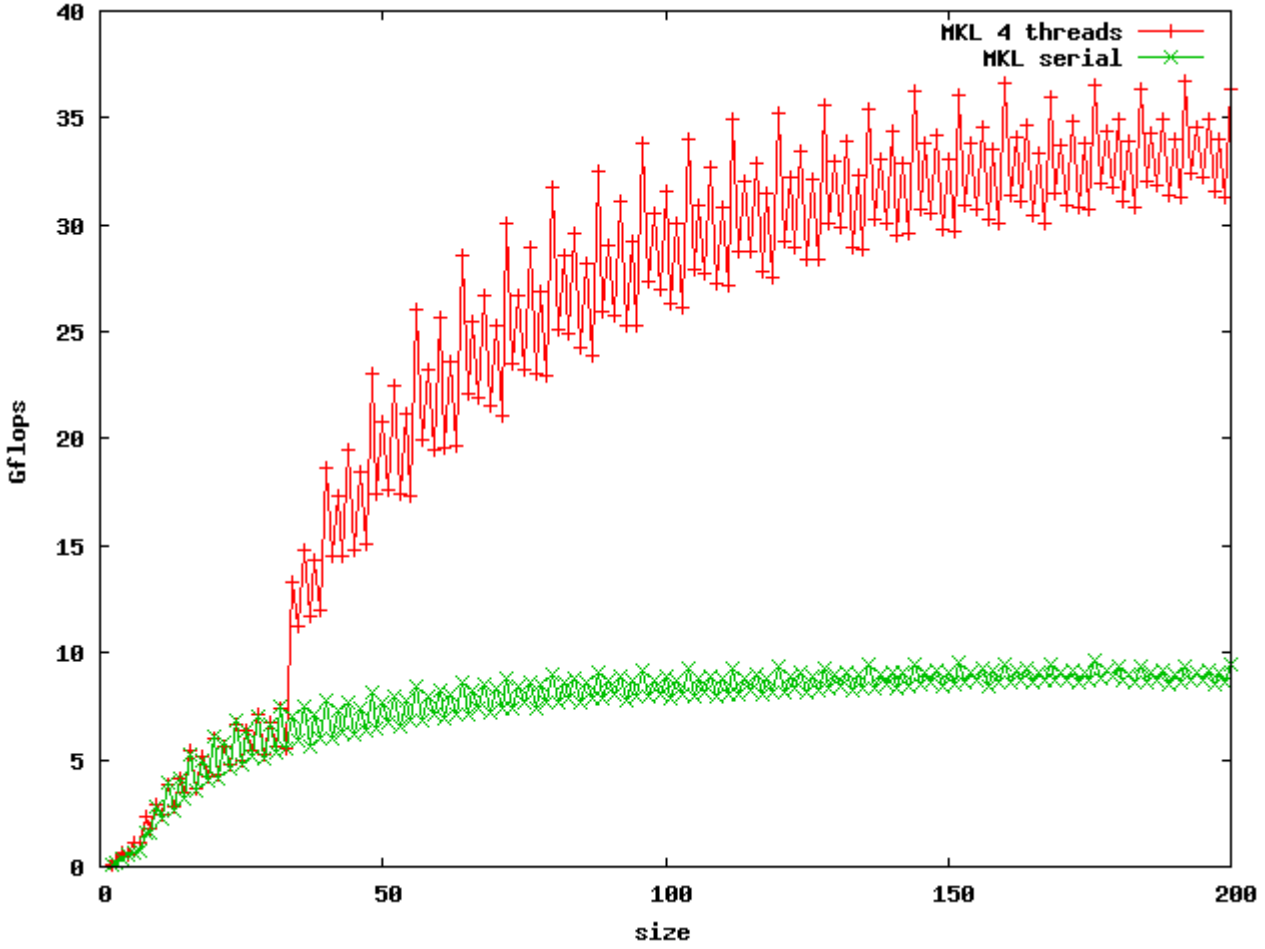
Rosa, libsci

Flops 2 x 1024 x 1024 x 10240

Help from vendors needed!
pdgemm should perform near peak and scale!

MKL: DGEMM threaded vs serial

our scalapack block size is 32x32



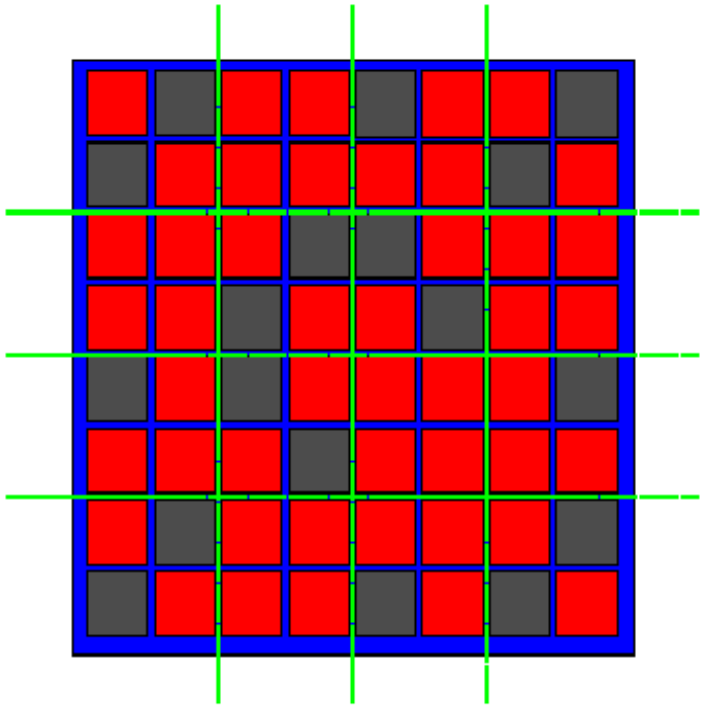
Threading kicks in from N=32

multiples of (8,4,2) is best

OMP libs (?) from gcc 4.4.1
MUCH better than 4.3.2
for small matrix sizes (5x for 32,
and mind .so libraries...)

Testing on Nehalem, matrices
are hot in cache.

Data layout in CP2K: sparse matrices



Overlap, KS, and density matrix are stored as sparse matrices in an 'atomic-block' way

Sparse means typically 20%-100% occupied

Symmetry is being considered

Uses a block cyclic layout to obtain scalapack like efficiency for the full case.

Becomes faster as blocks are zero, but doesn't exploit sparsity for communication

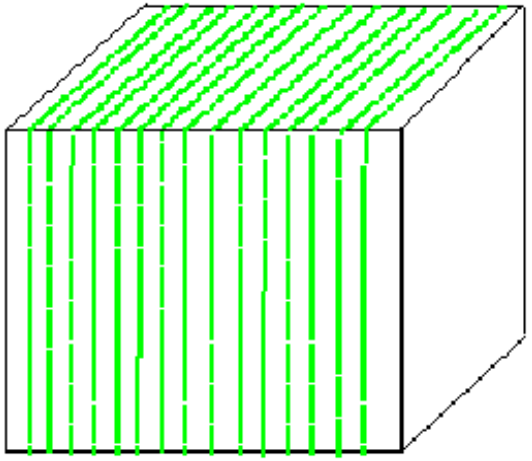
Specialized multiplication routine for atomic-sparse-matrix times scalapack dense matrix (sizeable fraction of total time spent here)

A new sparse matrix library is on its way...

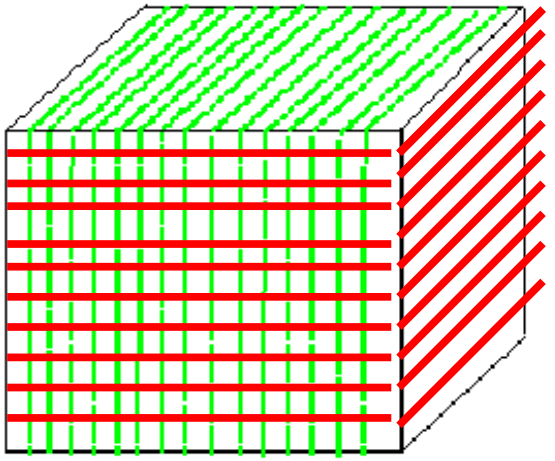
Data layout in CP2K: Fourier grids (I)

Fourier grids are either 1D or 2D decomposition of space

1D



2D



more efficient for < 256 cores

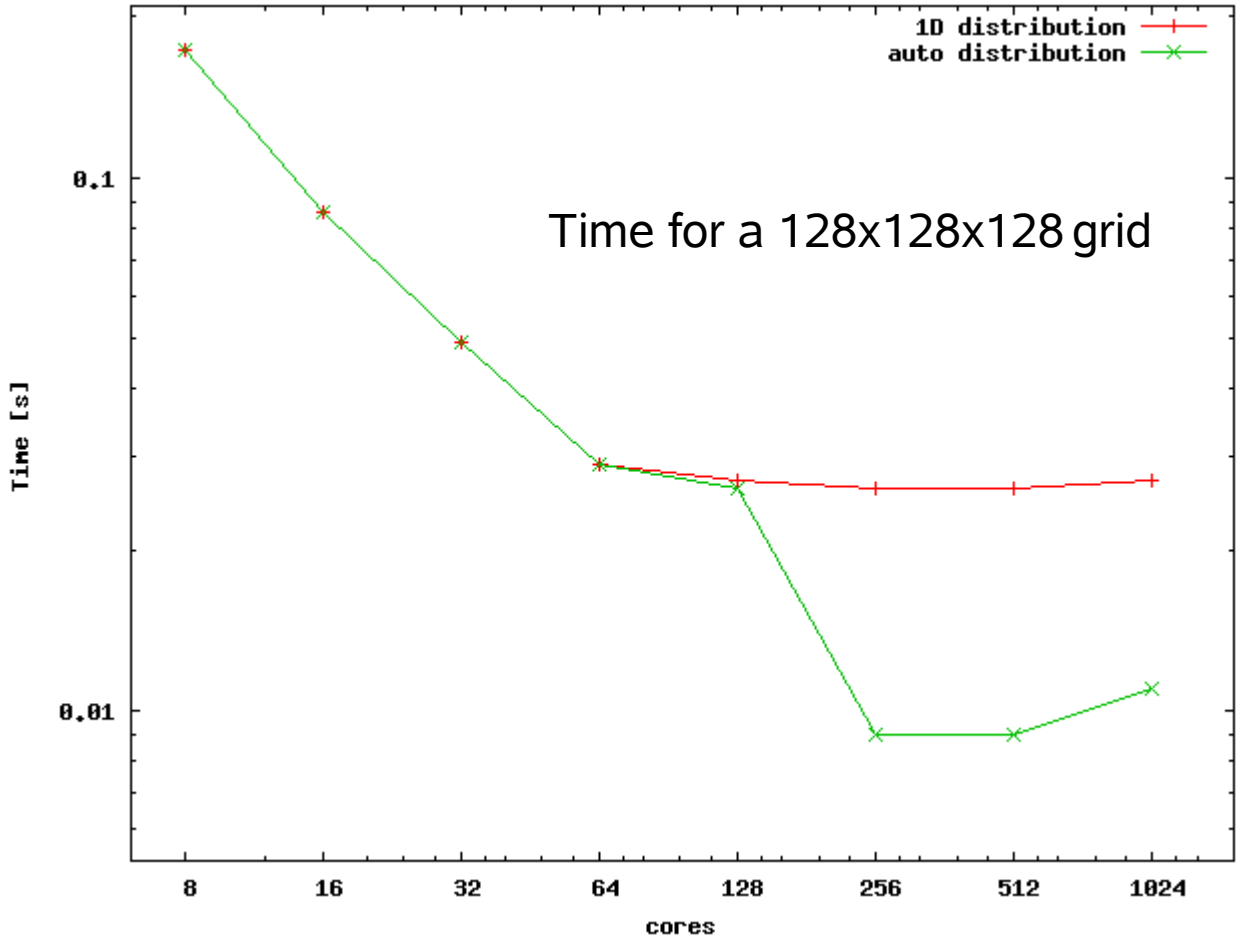
more efficient for > 256 cores

typical grids are 128^3 -- 256^3

Our 3D FFT tries to employ the optimal layout depending on core count

effect can be huge
e.g. on a 240x216x405 grid : 1D = 592s , 2D = 18s @4096 cores
1D = 32s @1024 cores

Data layout in CP2K: Fourier grids (II)



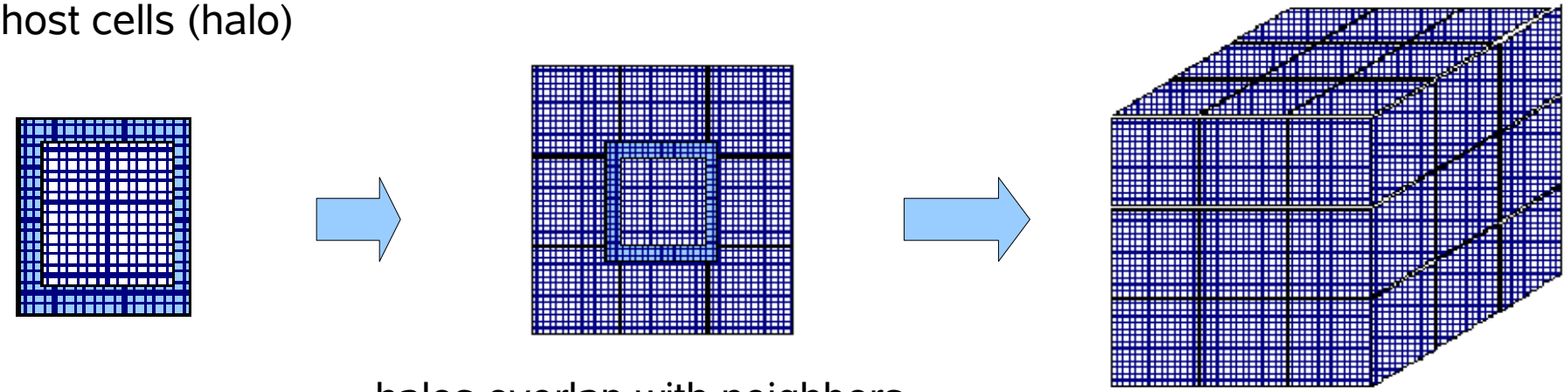
the effect can be very large
e.g. on a 240x216x405 grid : 1D = 592s , 2D = 18s @4096 cores
1D = 32s @1024 cores

Data layout in CP2K: realspace grids (I)

We want to compute the values of Gaussian products fully local:

and domains can be fully 3D

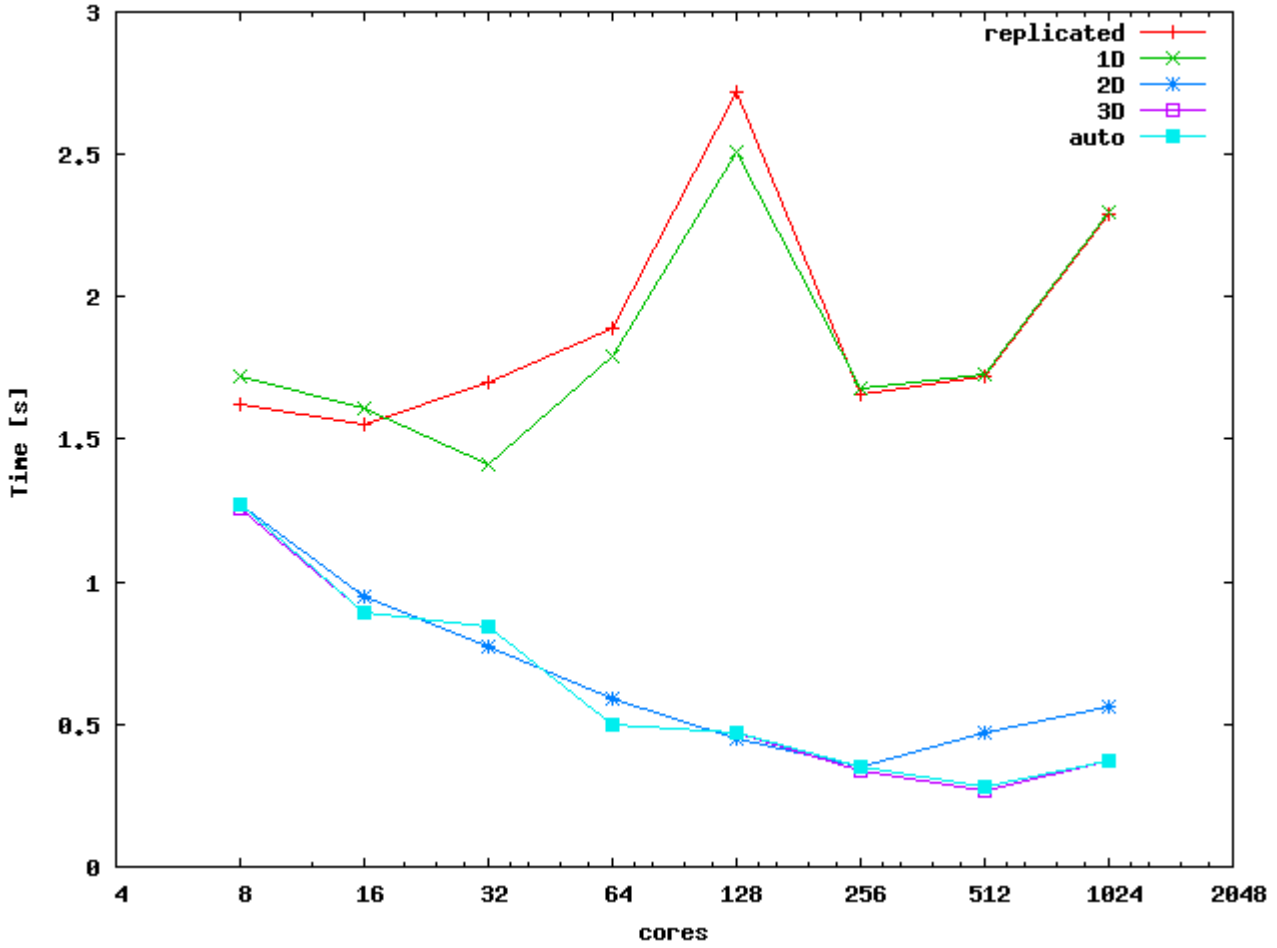
The local domain of a CPU while ghost cells (halo)



halos overlap with neighbors

The actual decomposition depends on the number of cores
replicated / 1D / 2D / 3D -> minimize the volume of the halo points

Data layout in CP2K: realspace grids (II)



128x128x128 grid
17 points border
30 repetitions

Halo exchange and redistribution become faster with core count, but do not scale well.

Data layout in CP2K: realspace grids (II)

We have very tight Gaussian functions and very smooth Gaussian functions:

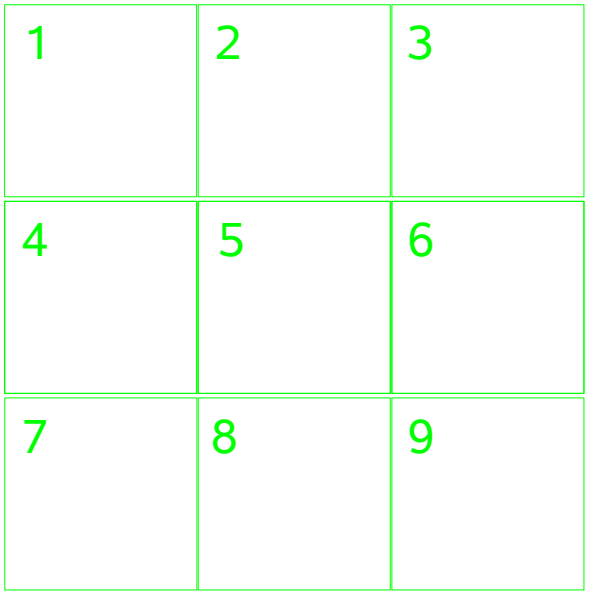
Use a multigrid representation (fine and coarse meshes)

fine grids are distributed, coarse grids might be replicated

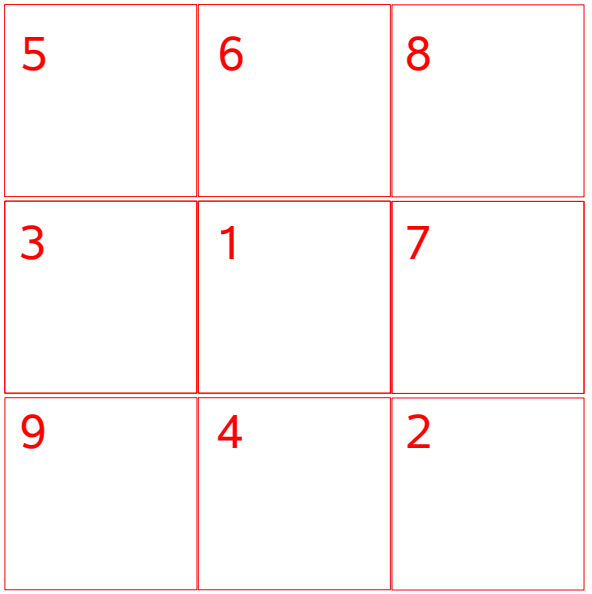
do injection/prolongation using FFTs (all commensurate in G-space)

Data layout in CP2K: realspace grids (III)

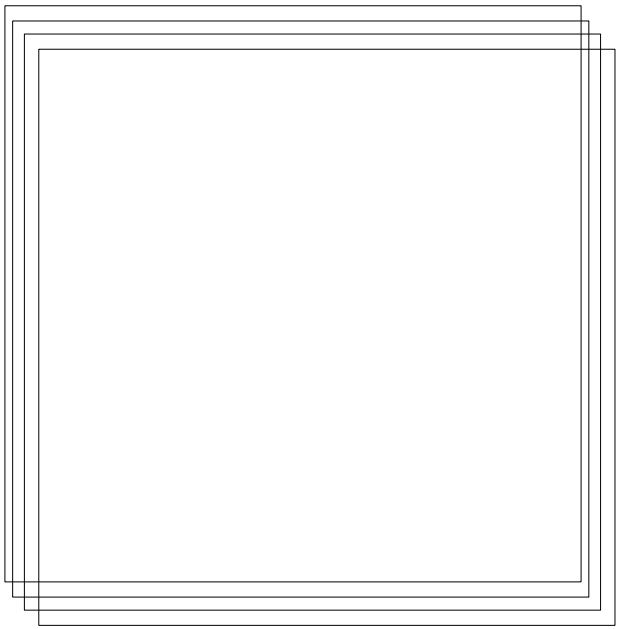
Level 1, fine grid, distributed



Level 2, medium grid, dist



Level 3, coarse grid, replicated

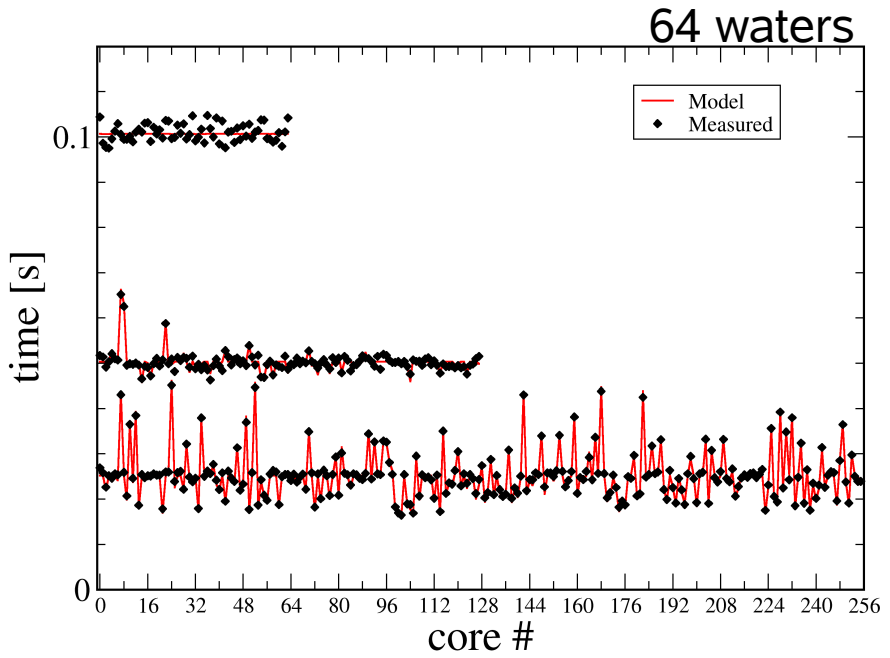


Load balance work done on these grids!
Assign different regions of space at each level to the same MPI rank, further balance on replicated grids

Data layout in CP2K: realspace grids (IV)

Load balancing a water cluster (W216) in a big box, i.e. worst case scenario

	No balancing	Balancing	balancing on optimized multigrids
Maximum load:	1738978	1165637	625139
Average load:	176232	475032	561845
Minimum load:	0	317590	542017

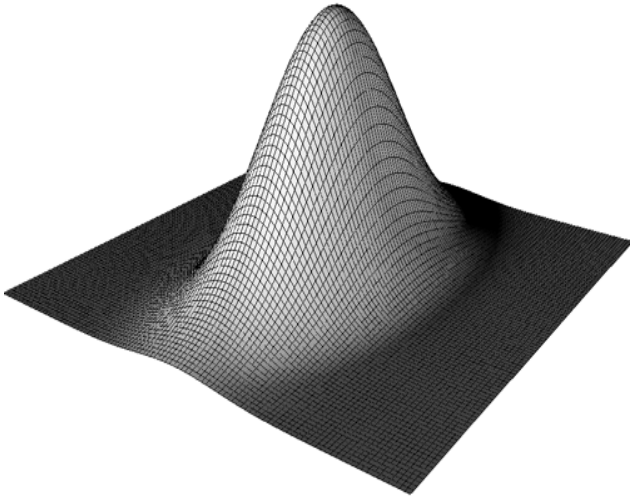


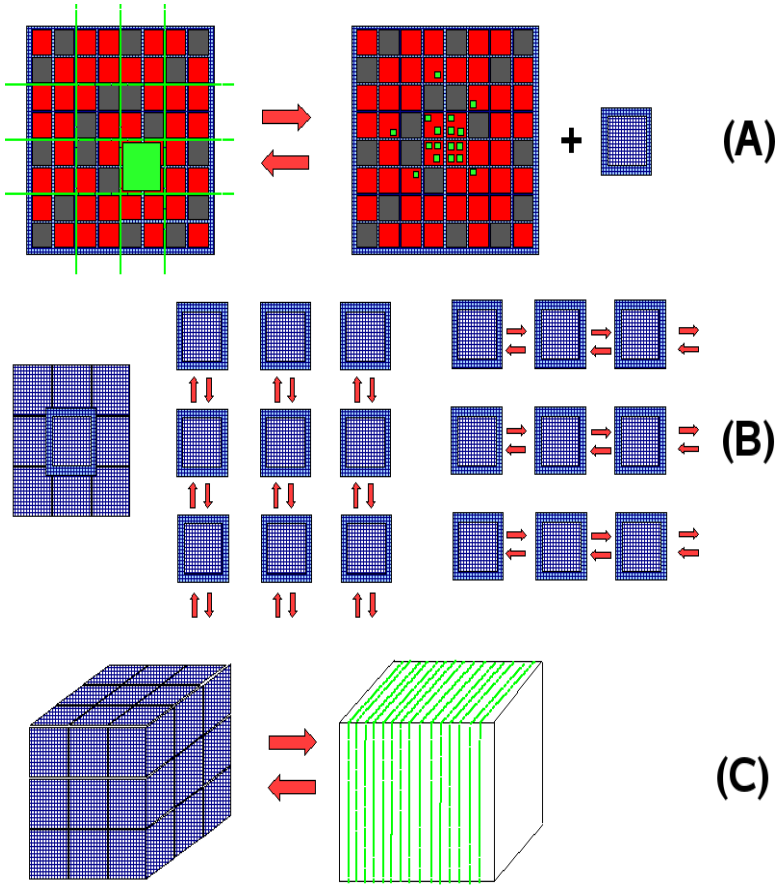
We have a very accurate model to predict the cost of the local tasks!

All spikes properly predicted, having a load balanced model is good enough

From formula to implementation...

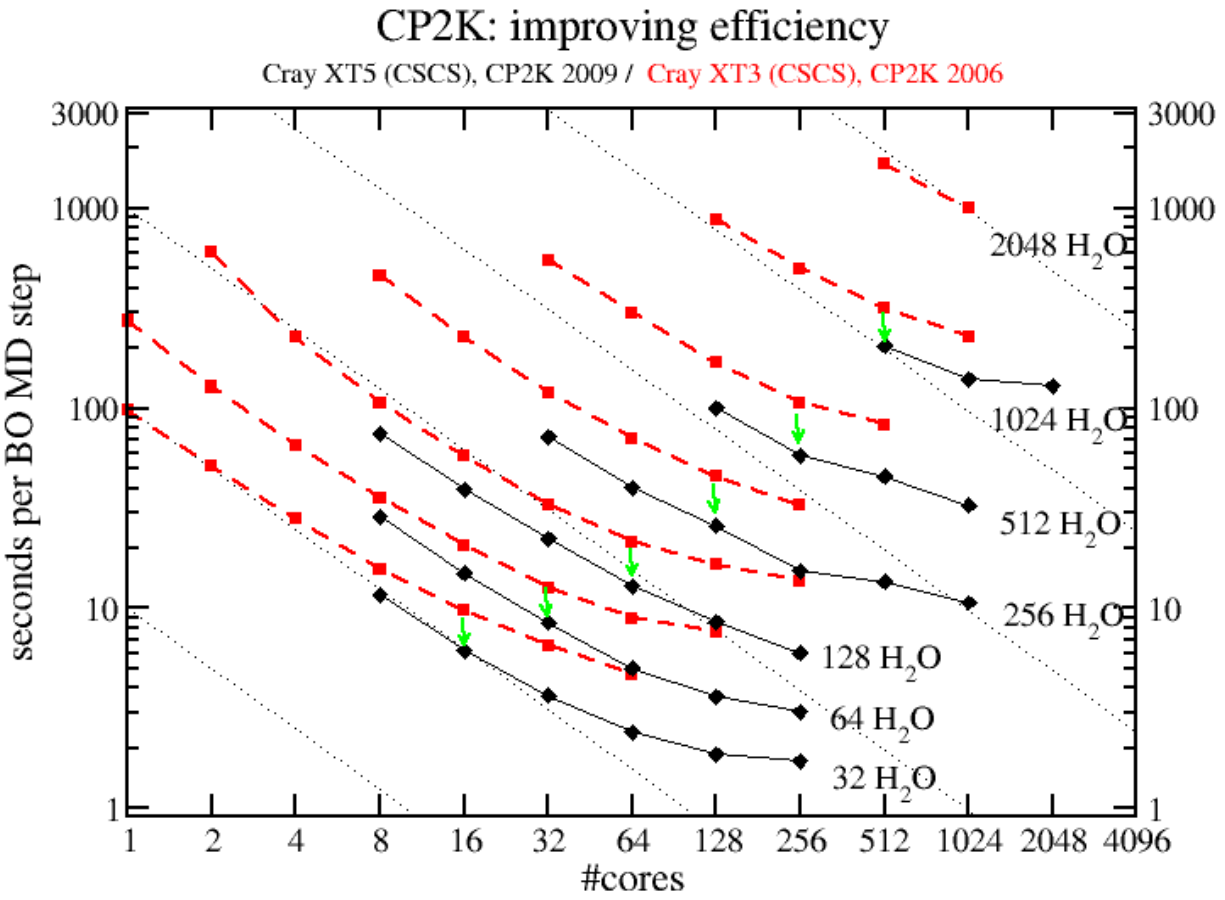
$$\sum_{\mu\nu} P^{\mu\nu}\,\varphi_\mu(r)\varphi_\nu(r) \underbrace{\Rightarrow}_{RS} n(r) \underbrace{\Rightarrow}_{FFT} n(G)$$





Redistribution of P (get elements local)
Map on multigrids
Halo exchange
Redistribute to FFT grid
Perform 3D FFT
Sum Multigrids

Parallel CP2K based on local DFT ?



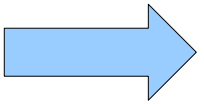
GPW and OT can be parallelized relatively well, and progress is steady.

[reminder]

Is the KS matrix part done ?

The canonical benchmark...

H2O	cores	full time	KS
32	128	41.84	21
64	256	120.10	77
128	512	125.6	36
256	1024	217.8	43
512	2048	788.7	76



Pushed to the limits, linear algebra now dominates for the interesting systems
(time to revisit OT & linear algebra ?)

Hybrid functionals
the return of the KS matrix

local GGAs are efficient, but of limited efficiency.
Virtually all modern functionals are based on Hartree-Fock Exchange (HFX).

-better chemistry (reactivity)
-better level diagrams (band gaps).

Add a simple looking term to the KS energy / KS matrix:

$$E_x^{\mathrm{HF}} = -\frac{1}{2}\sum_{\alpha\beta\gamma\delta} P_{\alpha\beta}P_{\gamma\delta}(\phi_\alpha\phi_\gamma|\phi_\beta\phi_\delta)$$

where

$$(\phi_\alpha\phi_\gamma|\phi_\beta\phi_\delta) = \int d\mathbf{r}d\mathbf{r}' \frac{\phi_\alpha(\mathbf{r})\phi_\gamma(\mathbf{r})\phi_\beta(\mathbf{r}')\phi_\delta(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}$$

In-core SCF and screening

$$(\phi_\alpha \phi_\gamma | \phi_\beta \phi_\delta) = \int d\mathbf{r} d\mathbf{r}' \frac{\phi_\alpha(\mathbf{r})\phi_\gamma(\mathbf{r})\phi_\beta(\mathbf{r}')\phi_\delta(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}$$

Can be stored and reused many times ... but is a large amount of data.

Our largest calculation so far: 1977539062500000000 spherical contracted integrals (2000 peta-integrals, 2E18).

Many are small and can be screened away .... needs to be efficient when screening, and highly efficient (we compute all of them in 2E17 flops, or 1h on 8000 cores)

We are actually doing a sparse matrix times vector product : $E=-0.5\ v^T M v$

where M is a $1.4 \cdot 10^9$ x $1.4 \cdot 10^9$ matrix, and can be used 10-20 times with different v's

An example

trick	number of integrals	memory [MB]
none	365'216'351'984	2'786'380
symmetry	45'652'043'998	345'297
$\epsilon = 10^{-6}$	1'300'799'772	9'924
$P-$ screening	532'091'877	4'060
compression	239'211'929	152(13)

Using (large) amounts of memory speeds up calculations (10x)

64 H2O cluster 6-31G** basis (i.e. small basis)

The matrix of integrals is really sparse (e.g. 0.001-1% occupied)

Largest in-core calculations so far used 11Tb of storage for integrals...
but 29Tb total memory usage (we have other stuff to store as well).

Parallel algorithm

Step 1: Replicate density-matrix (ring-topology)

Step 2: Construct list of all integrals that pass the screening

Step 3: Divide integrals into bins and assign cost (integrals, time)

Step 4: Load balance cost and assign work

Step 5: Let all cores assemble its local Kohn-Sham matrix

Step 6: Re-distribute Kohn-Sham matrix (ring-topology)

+ Easy communication pattern
+ Easy load balancing
+ Easily allows for exploiting all symmetries

- Requires a lot of memory
- Requires a lot of communication

Combine OMP/MPI

-Largest benefit in this case : reduced memory & communication per node
- share the replicated P and KS matrix between all threads
- P is read-only, modify KS with atomic updates

-Also helps: reduces the impact of badly scaling algorithms
- e.g. halo exchange or diagonalization is slower at 16000 MPI tasks than at 2000

-Relies on the fact that the non-OMP parallel part of CP2K is actually not very costly for these systems.

LiH benchmark system

Goal: compute the basis set limit of the HF cohesive energy of LiH
Added value: No symmetry / No k-points -> can do real materials with defects as well
Difficulty: Hard !

Fully periodic, up to 37500 basis functions, 1000 atoms, 65536 cores

Memory really an issue... several 'useless' Gbs removed (from GAPW and HFX)

Overflows lurking (# cores square overflows default int, number of BF**2 as well)

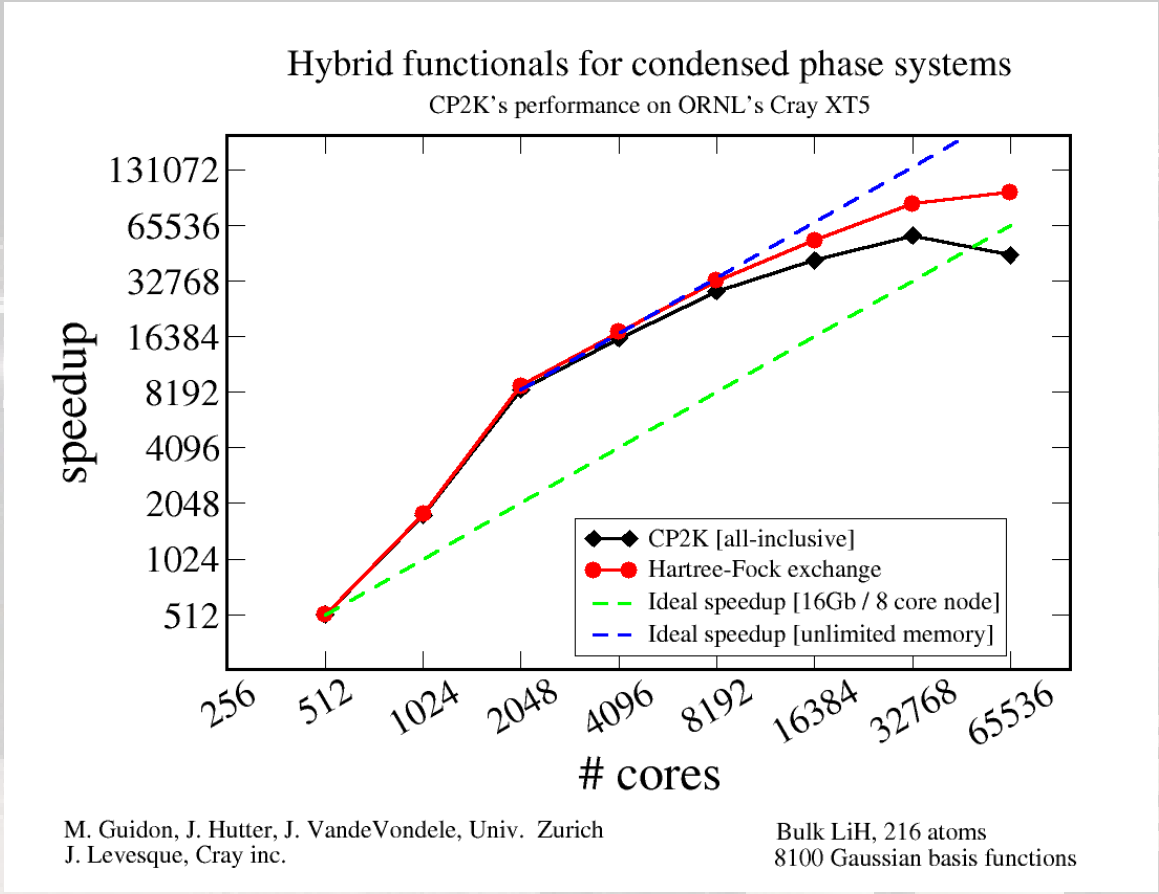
Protecting against 0 local work needs to be done carefully (e.g. load balance should not put all zero-sized tasks on least loaded cpu).

Some of the MPI turned out to perform better after putting sync's before and after comm.

Improved performance by doing blocking on the level of the 4 center loop (cache, atomic updates)

Worked very hard on load balancing. Needs to be 'perfect' and very fast (i.e. very parallel)

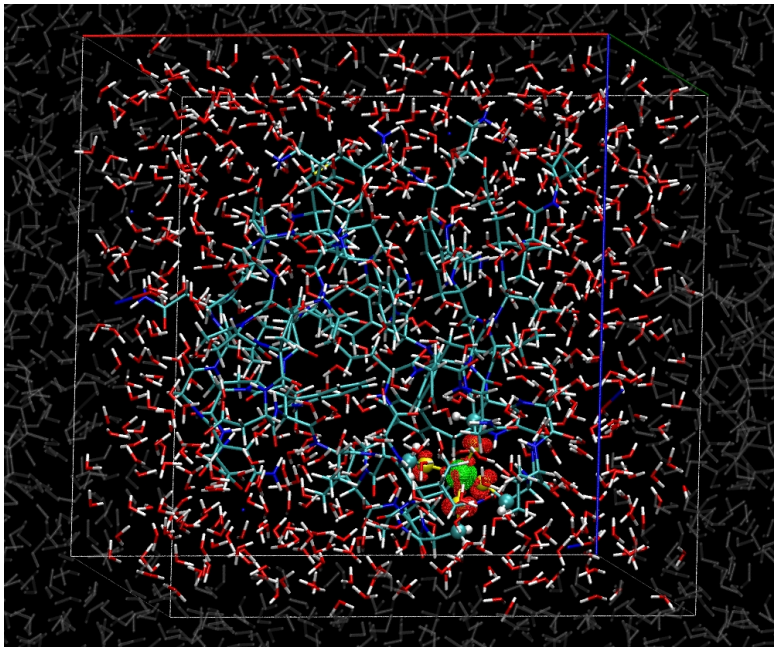
LiH Results

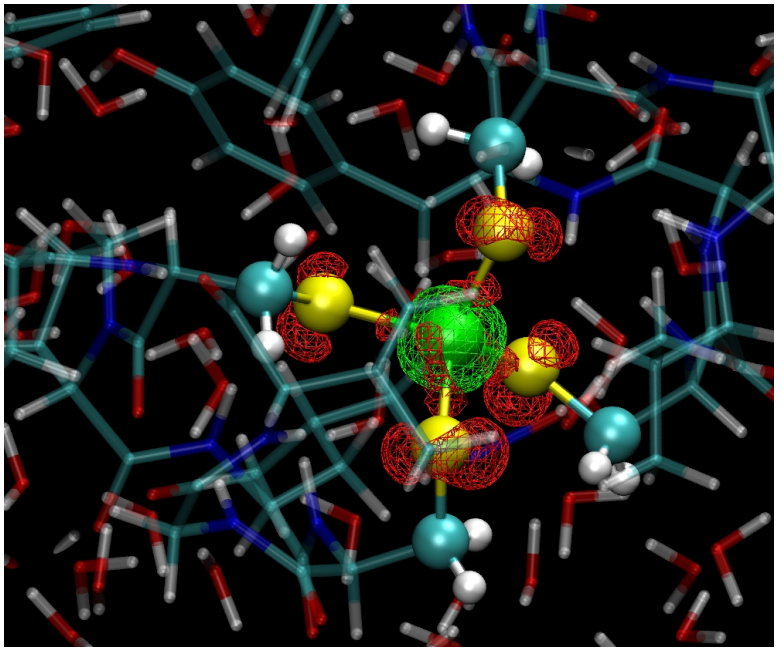


Ecoh=131.948mHt ref. 131.95+-0.1mHt

HFX scales superlinearly from 512-2048 cores (as more memory becomes available)
HFX scales well from 2048-65536 cores (but the GGA part of CP2K starts to dominate)

Rubredoxin in solution

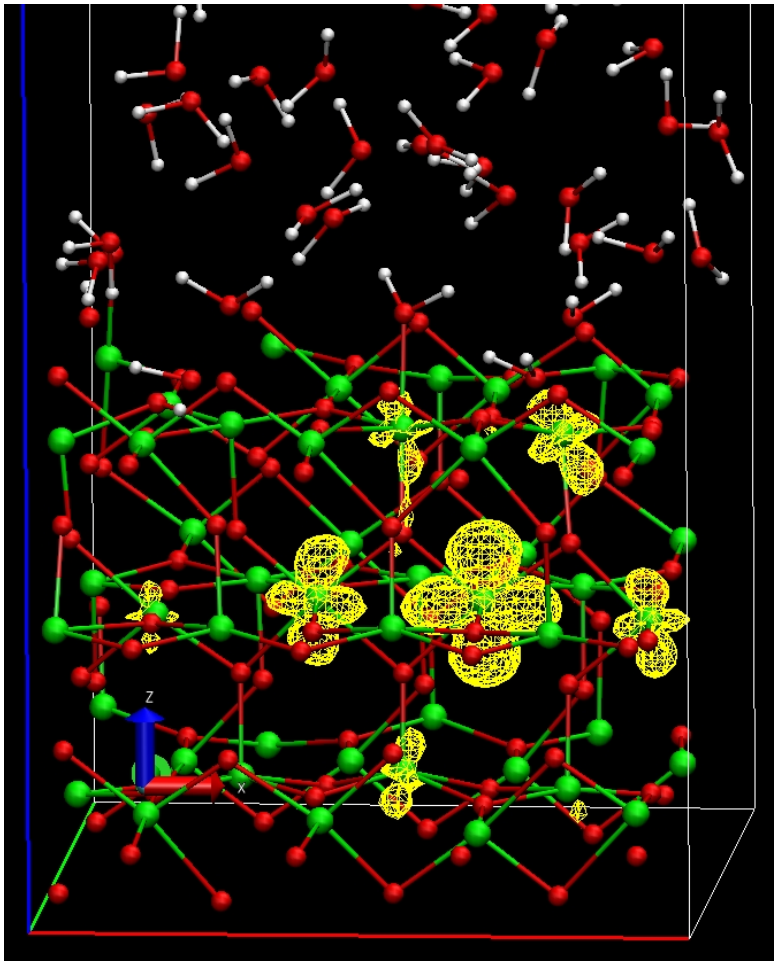




Difference in spin density between BLYP and B3LYP
shows spurious spin delocalization on the sulfur atoms for the local functional

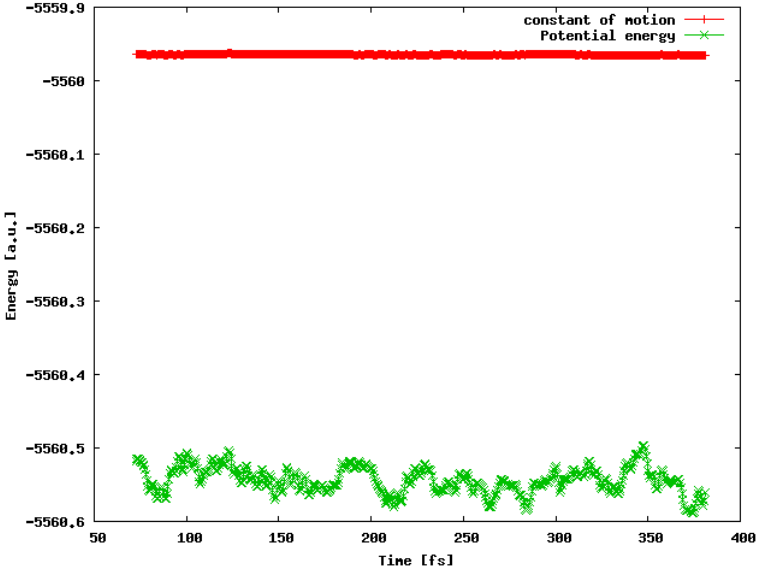
Fully solvated protein described as a bulk (periodic) system
2825 atoms in the unit cell, 5022+5017 electrons
with a Ahlrich's pVTZ basis (31247 BF)
B3LYP vs BLYP calculation
3h on 8192 cores // 11Tb memory usage

Hybrid MD of 'real' systems



Look at excess electron and hole in TiO2 / water interface (356 atoms, 5200 BF).

250s / MD step on 4096 cores, 80% parallel efficiency
0.1 Tb ERI memory with screening 1.0E-7
1.0 Tb ERI memory with screening 1.0E-9



Interface model by Jun Cheng and Michiel Sprik

Acknowledgements

Manuel Guidon & Juerg Hutter, Univ. Zurich

John Levesque & Jason Beech-Brandt, Cray Inc

Iain Bethune, EPCC

Matt Watkins & Ben Slater, UCL

CSCS

Pekka Manninen, PRACE, CSC