HPC view at Quantum Chemistry Software

Valera Veryazov

Valera.Veryazov@teokem.lu.se

Department of Theoretical Chemistry
Lund University
Sweden

Overview: HPC and QChS

- What a HPC person should know about QChS?
- Why a HPC person should aware of QChS?
- How HPC experience can be used in QChS?

Quantum Chemistry in 2-3 slides

$$\hat{H}\Psi = E\Psi$$

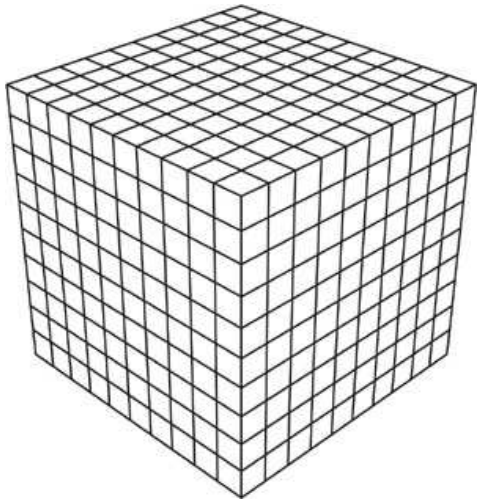
Looks rather simple...

Hamiltonian describes interactions between particles,
so we can solve Schrödinger equation,
and use wavefunction Ψ to compute various properties:

$$< \Psi | \hat{X} | \Psi >$$

A simple example

Let's have an example: 26 particles,
in a cubic box 10x10x10



Classical mechanics: particles are independent
wavefuction can be described by 26*10*10*10 'combinations'

.. is not so simple

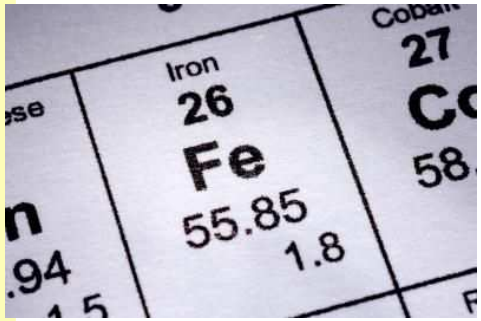
In Quantum mechanics: particles are not independent

$$\Psi(x_1, y_1, z_1, x_2, y_2, z_2, ...x_n, y_n, z_n)$$

So, instead of 26000, we have $10*10*...10=10^{78}$

The Sun contains [only] 10^{58} *protons*

And 26 particles it is :

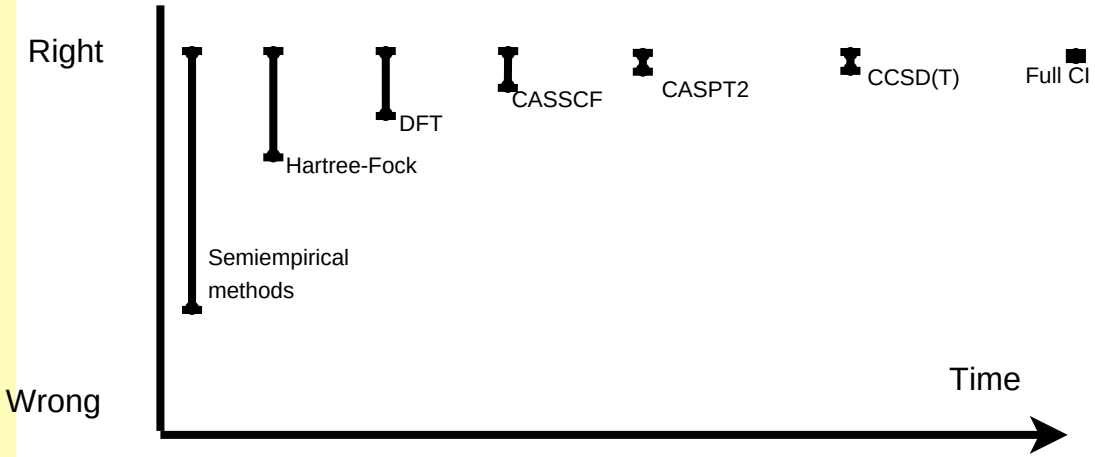


Approximations are mandatory

Quality vs. time

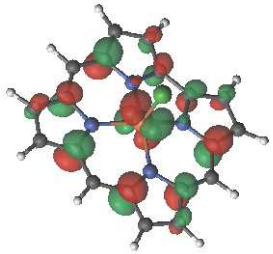
Basis set is used, so the problem is converted into matrix problem
Larger basis sets means larger size of matrices, and longer time

Hamiltonian quality:

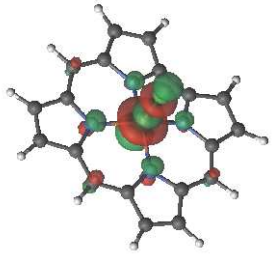


Precise methods need more time!

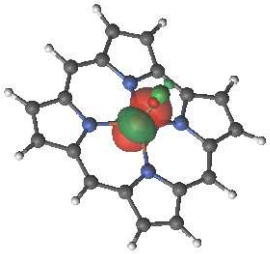
Just some nice pictures



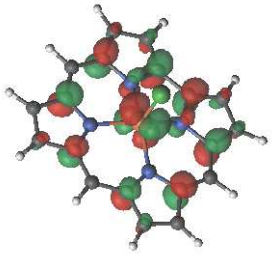
40a” (1.14)



58a’ (1.01)

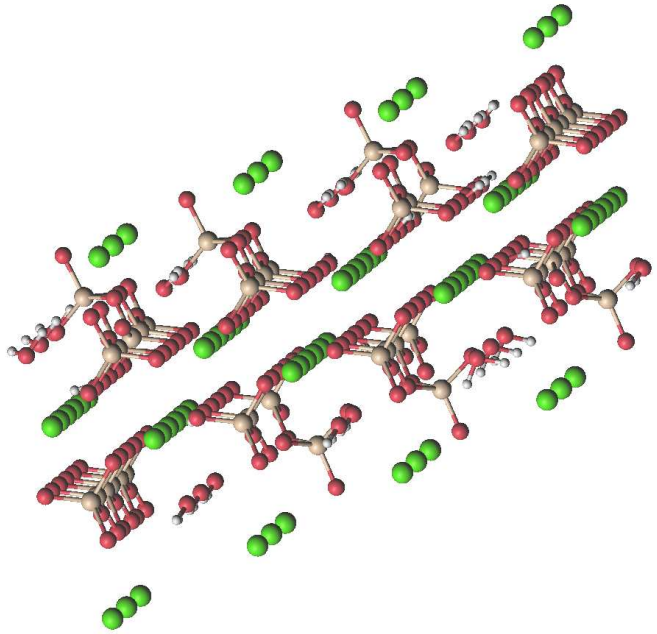


55a’ (0.99)



42a” (0.85)

File:model3.xyz



Quantum chemical software

- Established software packages (10-20 years of development)
- 'no experience is required'
- Substitution for 'wet chemistry'
  - ♦ prediction of new materials
  - ♦ understanding of chemical reactions
- occupies near 30% of computational resources

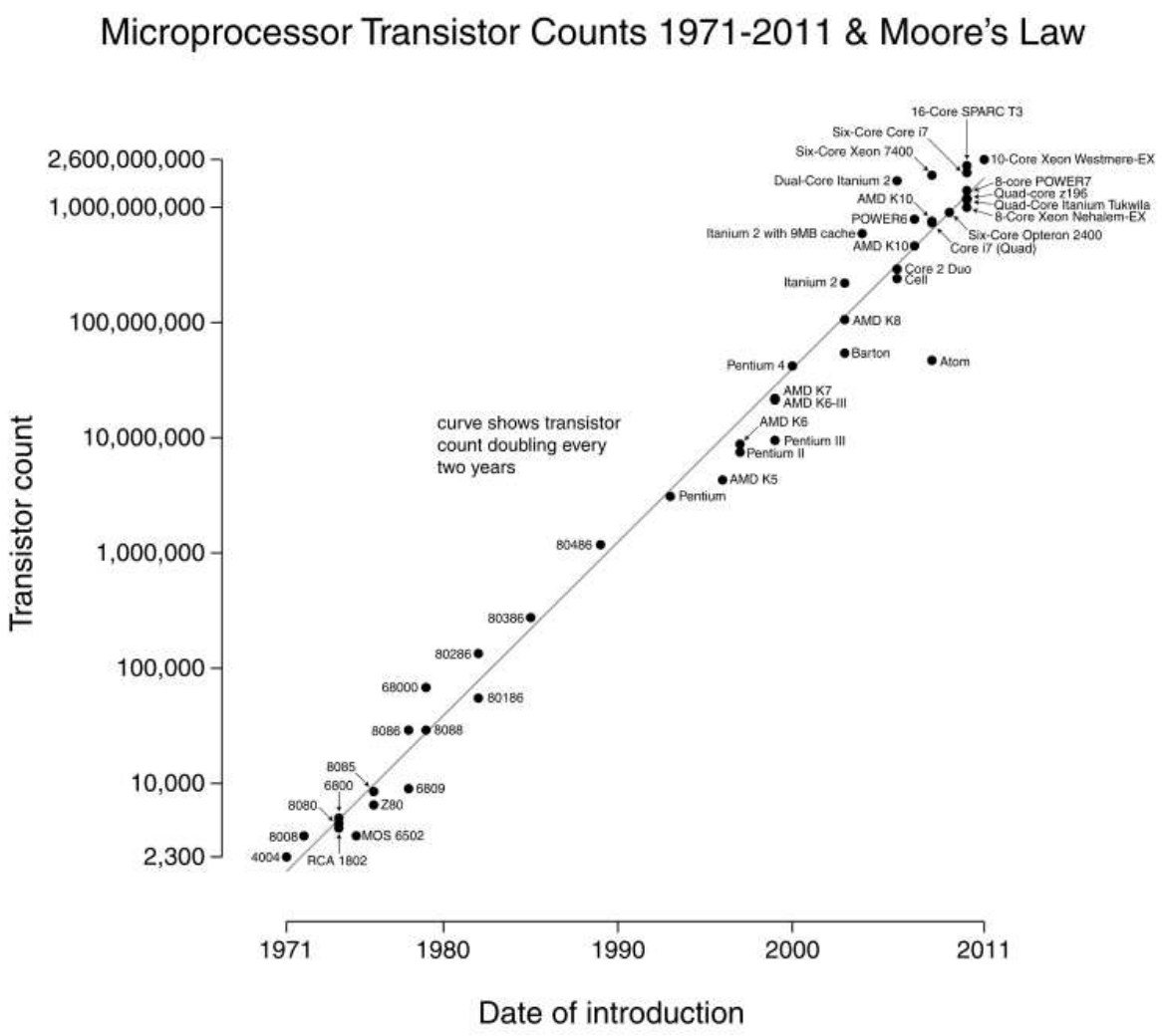
MOLCAS

- developed in Lund since 90-es
- with emphasis to multiconfigurational approach, and precise calculations
- non-profitable University based project
- 33 Mb of the source code (Fortran 77 + C)
- runs on all platforms
- best use: PC with a lot of memory, Linux clusters
- www.molcas.org

MOLCAS modules

- computational tasks are very different:
  - Computing of integrals (with possible packing)
    - CPU, I/O (writing)
  - Optimizing wavefunction
    - CPU (BLAS), memory (large matrices), I/O
  - Optimizing geometry (numerical)
    - parallelization by task (low communication)

"Theoretical" progress



So, why it "doesn't work" for QCh?

- very long development cycle
- code demands not only CPU power, but memory and I/O
- not obvious parallelization

Some sad stories..

...told by Molcas users and SysAdmins.

- multicore CPUs
  Parallel run can be slower (!) than serial
- advanced network file system
  code uses CPU only by $3-5\%$
- GPUs and CUDA BLAS
  the code runs slower

Back to the drawing board....

I/O from historical perspectives

- conventional integral code
  - integrals are reused, let's keep them on disk
- direct integral code
  - disks are slow, let's recompute all integrals
- semidirect integral code
  - a hybrid: keep only some integrals on the disk
- Resolution of Identity / Cholesky decomposition
  - keep the data, which can be used to reconstruct integrals

I/O

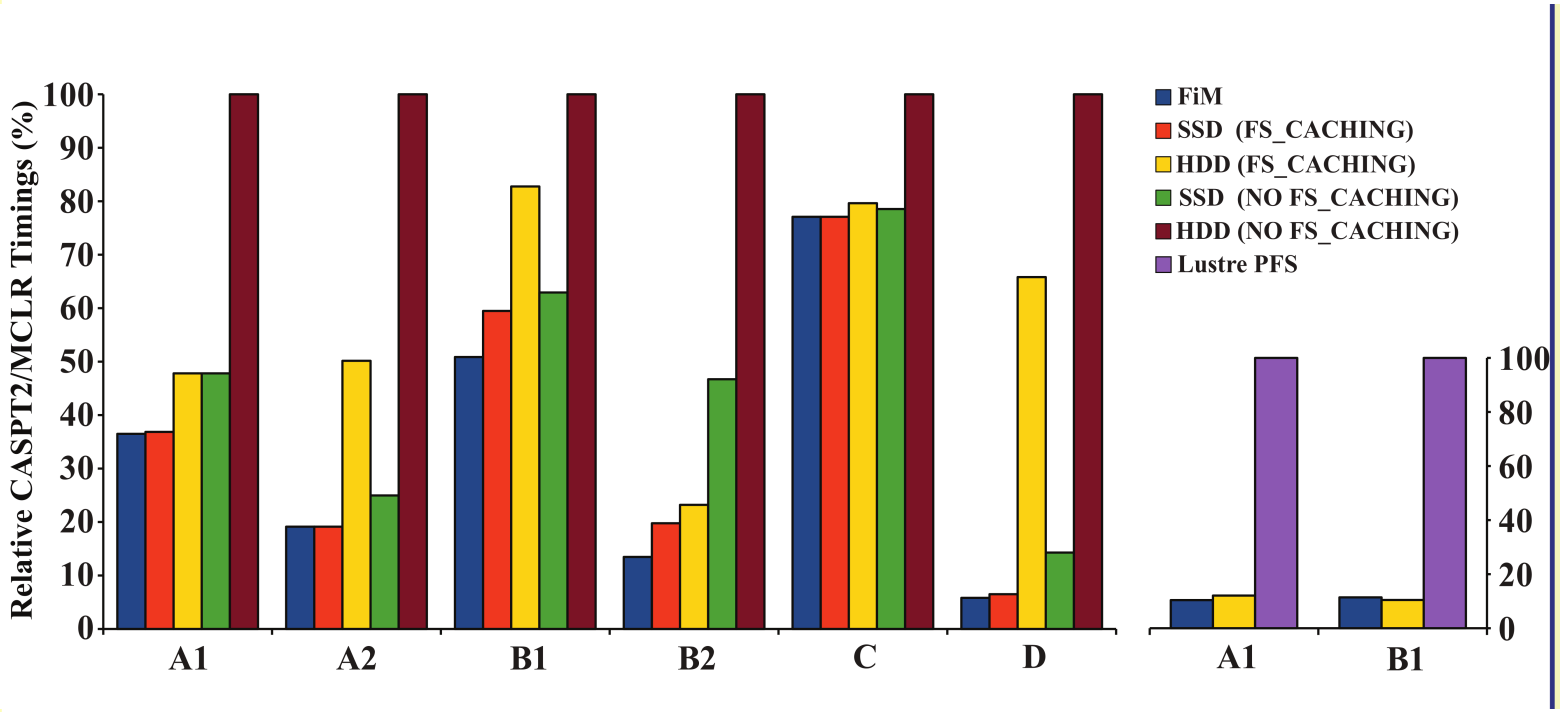
Problem:

- The amount of integrals and intermediate data can easily be measured in Gb, or even hundreds of Gb
- Read access is random, or spread
  (a result of 'writing by columns and reading by rows)

Solutions:

- Local disks
- Solid State Devices
- Files in Memory

I/O benchmarks



Files in Memory

Why FiMs are better than system I/O caching?

- Developer (or user) can choose which files to keep in memory
- Writing to disk (at the end) can be done by large blocks
- No need to save temporary files
- Prepared for future parallelization

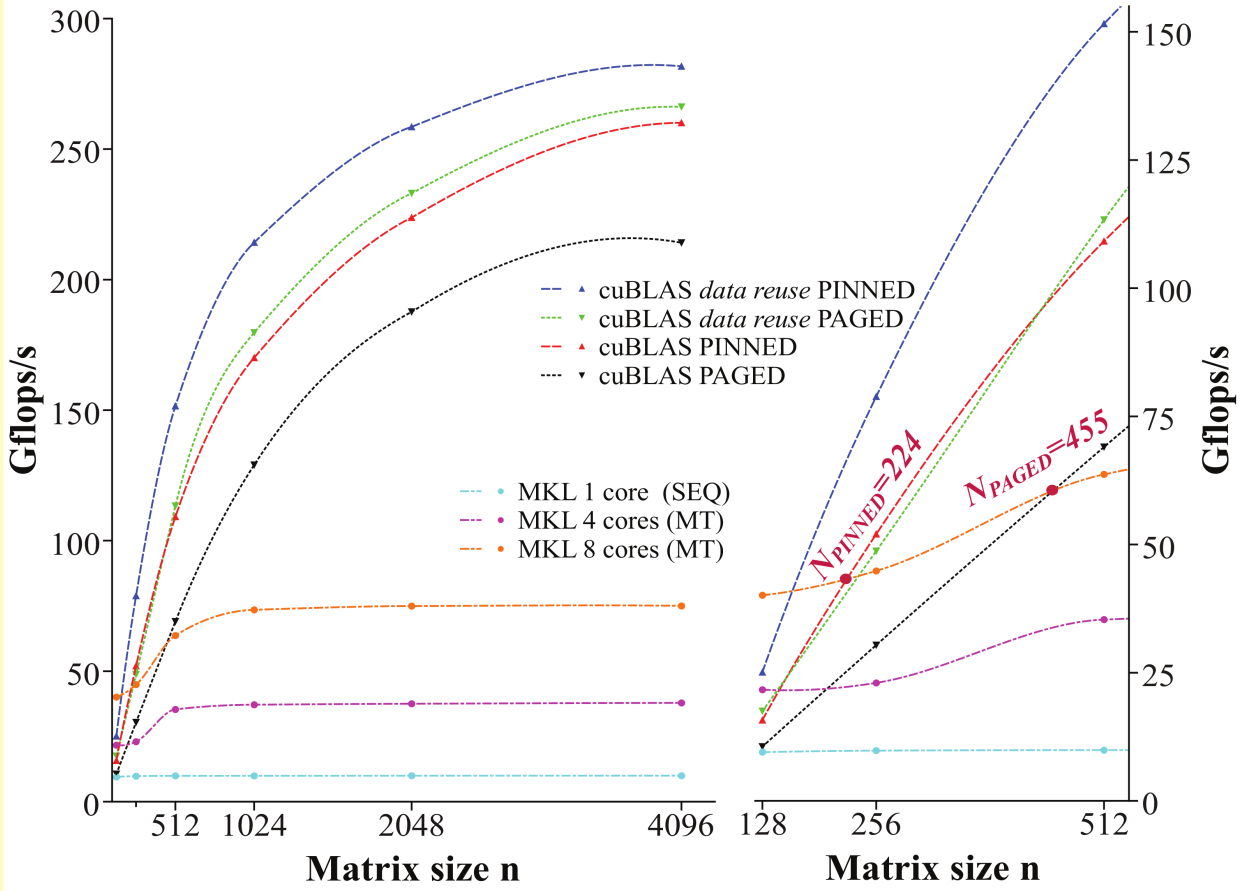
Will be a part of Molcas 8.

GPU and BLAS

A success story: Todd J. Martinez
Hartree-Fock code completely written for GPU.
(With "some" restrictions: everything is in memory,
so basis sets are tiny)

up to 80% of calculations in Molcas is BLAS (or LAPACK) calls
so, can we just
- move data to GPU,
- process it in parallel on GPU,
- return the result to 'CPU' ?

What is the proper size?



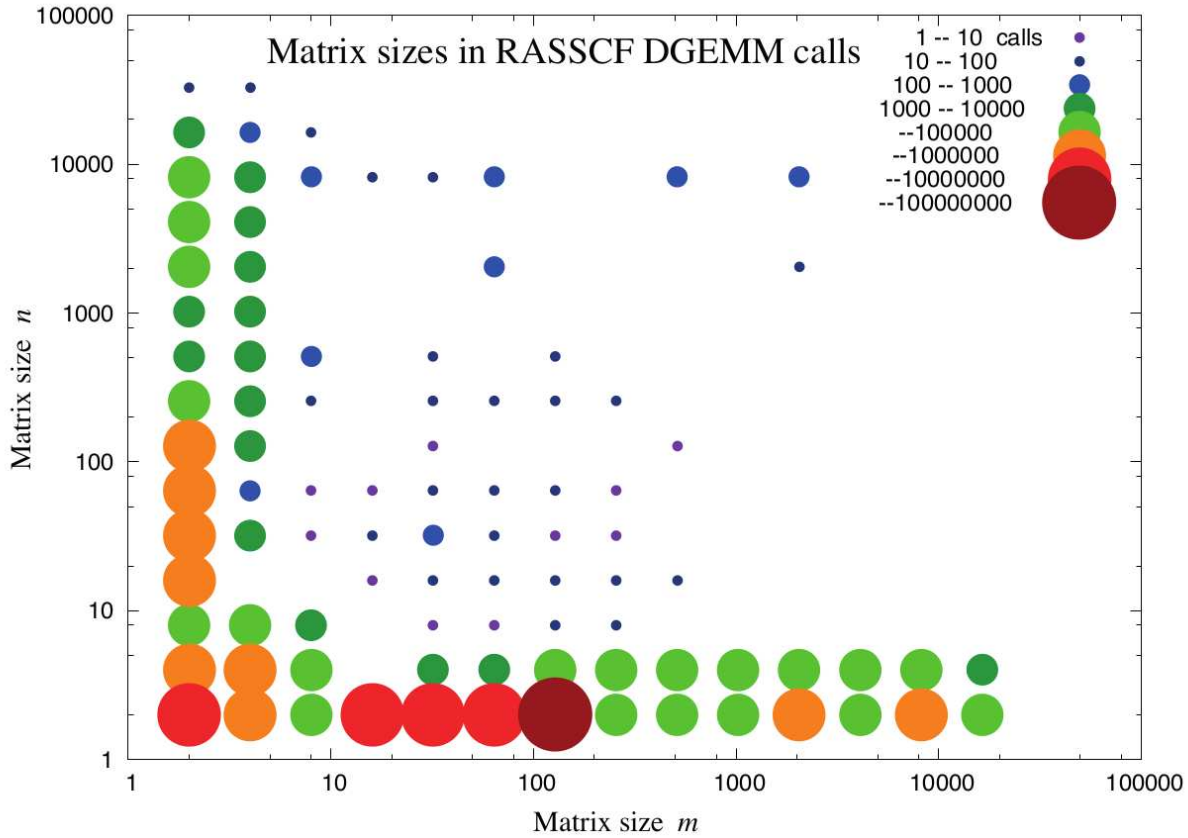
Multiplication of small matrices is faster on CPU!

CPU/GPU balance

The critical size (for modern GPUs) is about 128*128 (and it was about 500*500 two years ago).

```
PARAMETER (NCUDA=128*128)
IF (SIZE_N*SIZE_M .gt. NCUDA) then
    Call CuBLASS_DGEMM(..)
ELSE
    Call DGEMM(..)
ENDIF
```

But let's profile the code...



Only few calls are large enough!

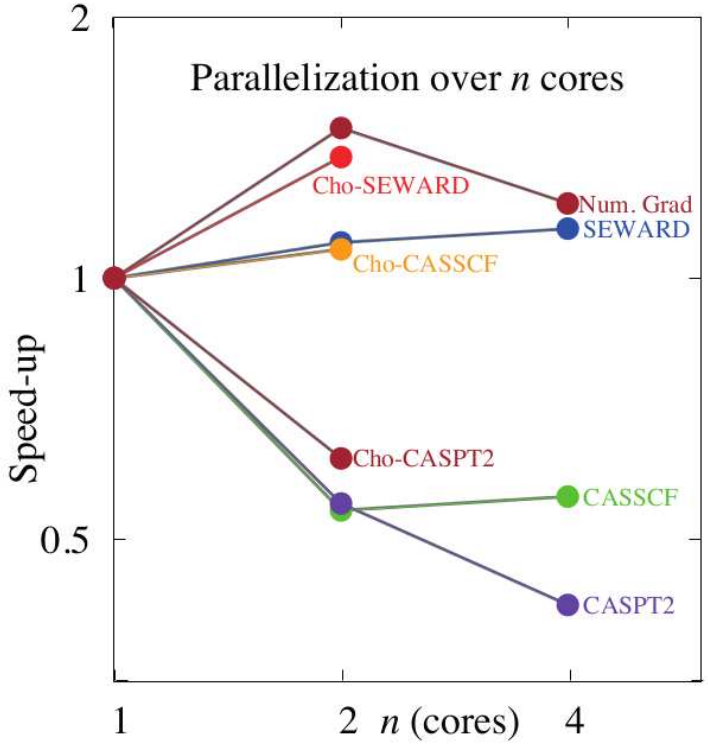
Amdahl's law exercise

- 80% of CPU time: calls to DGEMM
- 10% of these calls are executed on GPU (due to the size)
- expected speed up e.g. 16 times

compute the difference in timing.

Parallelization

for large calculations: Memory and I/O
are more important than CPU power



BlueGene technology is [now] useless for Molcas.

How to utilize multicore technology?

- control the usage of resources
- separate computations and I/O
- data packing
- decrease memory consumption (e.g. reuse shared memory)
- use it only for average-sized systems

Profiling

- developers of QCh software does not profile their code but they know the cases which works unusually slow
- profiling software does exist and it can find bottlenecks in computations, in I/O and in memory consumption
- HPC personal should be able to help in profiling

Software optimization

Compiler optimization can improve the performance,
but it might lead to overoptimization, and
to wrong results.
Tools to handle overoptimization:

- Verification
  - ’trusted’ version generates reference values
  - large number of tests
  - thresholds for each checked value
  - verification with various optimizations, compilers
  - can work in an automatic way
- divide and conquer search for overoptimized routines

Some conclusions

- Real applications are different from 'small test cases'
- HPC approach can contribute to 'algorithmic improvements'
  - Make profiling of the code
    - note that the results will be different for different modules
    - note that the results will depend on the system
  - "Properly" use hardware

Acknowledgements

- Molcas team
- Victor Vysotskiy and Steven Vancoillie
- CINECA