How does hardware development influence quantum chemistry?

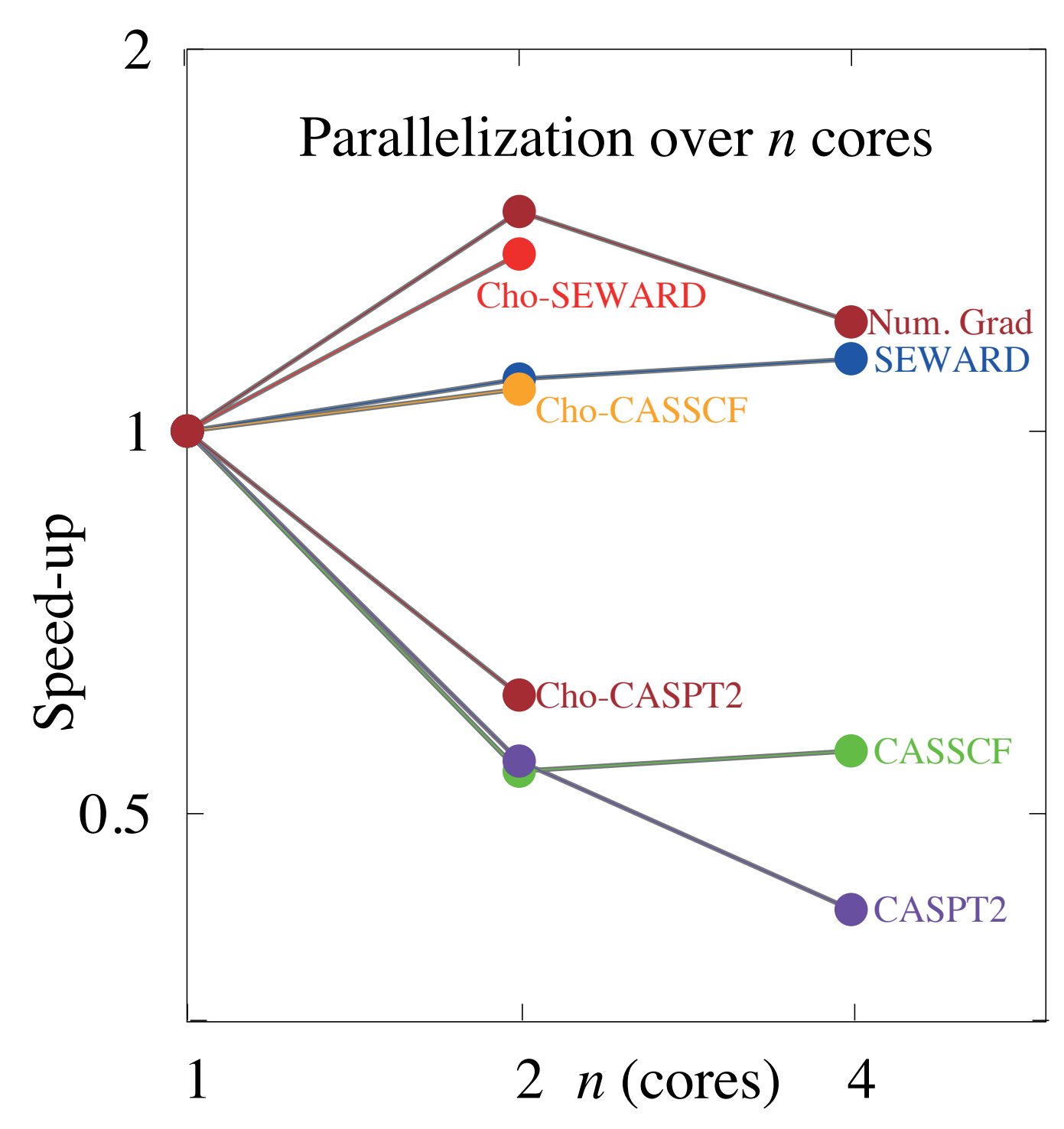
Mickaël Delcey, Per-Åke Malmqvist, Steven Vancoillie, Valera Veryazov

Theoretical Chemistry, Chemical Center, P.O.B. 124, Lund 22100, Sweden, E-mail: Valera.Veryazov@teokem.lu.se

Parallel computing

Processors capacities have increased exponentially, following the famous Moore's law. The development of computational units go now to a new dimension: instead of increasing the number of transistors on the same chip, various parallelization techniques are used: communication between CPUs via network, via bus on the same mainboard and finally units can be combined as computational cores on the same CPU.

• Multicores



These hardware implementations have different transfer speed of data between computational units, different transfer speed between CPUs and memory and different cache size per CPU. So different computational algorithms can benefit differently from these hardware solutions.

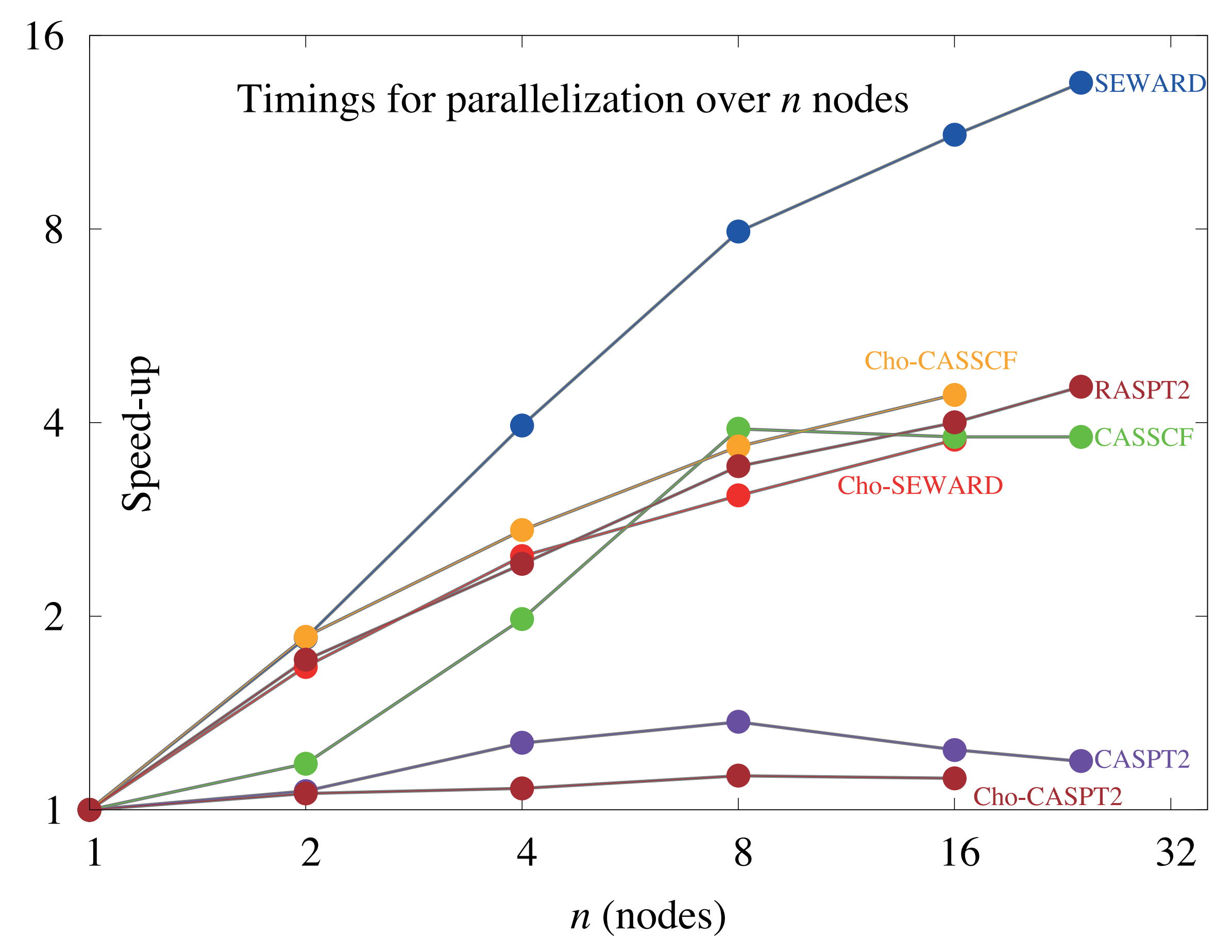
MOLCAS code[2] uses three kinds of parallelization algorithms. Independent calculations (Numerical Gradients), computing of parts of integrals (SEWARD+wave function calculation) and fine grain parallelization (multi-particle density matrices in CASSCF and CASPT2). Except for the last kind of parallel algorithms, the amount of data transferred between CPUs is fairly small.We selected benzene dimer as a test case, with 528 basis functions, which corresponds to an average calculation with MOLCAS code.

Differences in scaling of different modules for parallelization over cores are shown on the left. Clearly, parallelization over cores is not efficient, although the different modules behave differently. Thus, Numerical Gradients and SEWARD see a small improvement by using more cores on the same CPU, while CASSCF and CASPT2, code are even slower in parallel. The situation is better with Cholesky decomposition technique, where I/O is less important.

• Multinodes

On the right are shown results of parallelization over nodes up to 24 nodes. The tests were made on LUNARC cluster with relatively slow Gigabit connection. Numerical gradient shows a perfect scaling until the number of nodes is bigger than the number of displacements. Computing and usage of integrals needs small data transfer and thus shows an almost perfect scaling. The result of RASPT2 for a polyyne system is also shown, to be representative of small molecules with a big active space. The scaling is clearly better for this system since only the many-particle density matrices generation in the RASPT2/CASPT2 code is parallelized yet.

Leading CPU manufacturers claim that in a near future, CPUs with 96 and more cores will be available. Quantum chemical codes should be adapted in order to use efficiently these technologies.



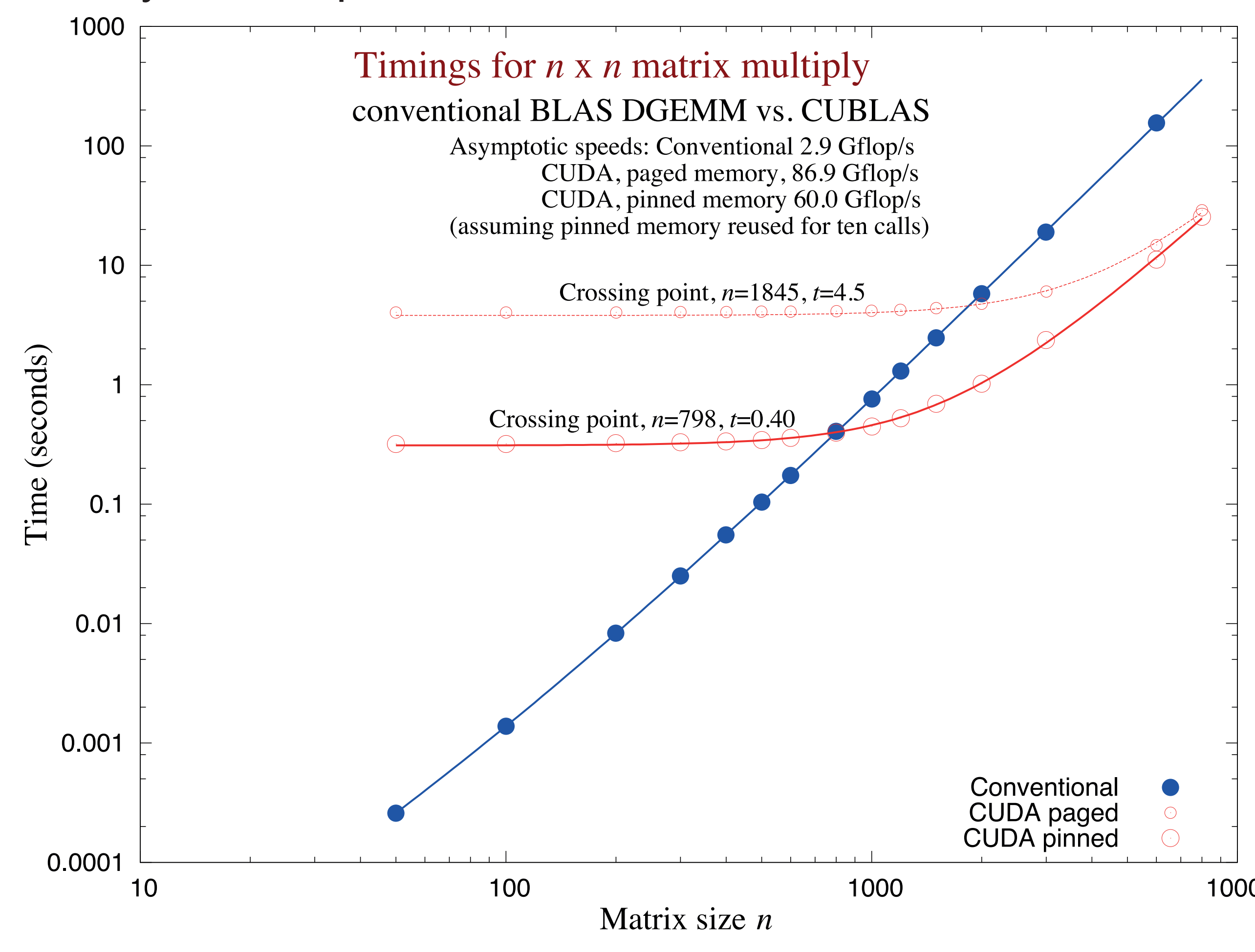
SDD

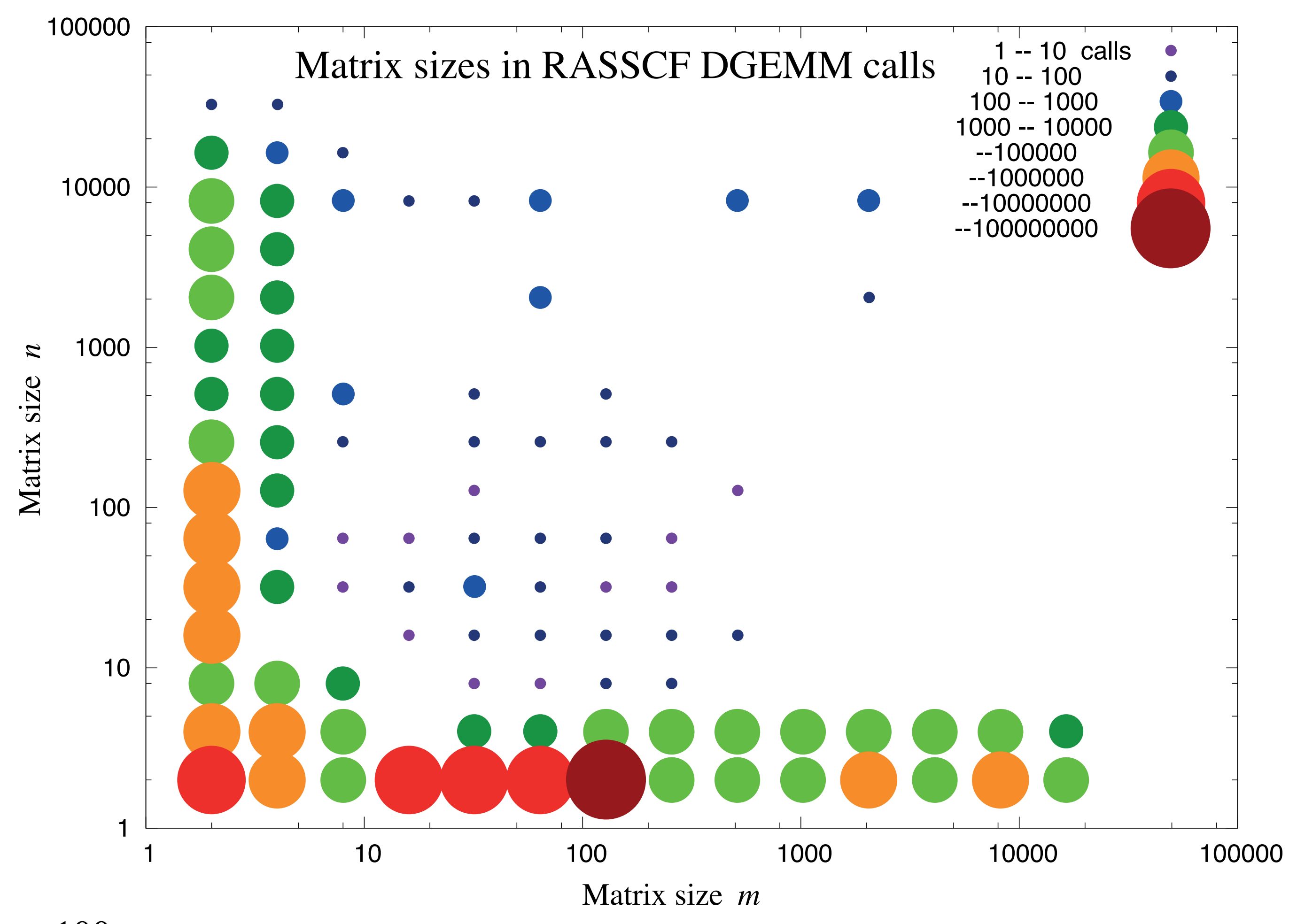
Solid State Drives and cheap memory cards used e.g. in a camera or as USB memory stick have quite different read and write timings compared to conventional disks. A test MOLCAS job running on an external memory drive (connected via the USB 2.0 interface) showed a surprising result: RASSCF was running slightly faster and CASPT2 was only 17% slower, than using the conventional HDD.

Graphical Processing Units (GPUs)

A modern GPU is a highly parallel and multi-threaded processor with very high flop performance and internal memory bandwidth. Nvidia has made these features useful for non-graphical tasks, creating an environment which includes code development tools, and the high performance libraries CuBLAS and CuFFT. A quantum chemistry software is typically rich in BLAS calls.

This suggests a simple approach to GPU acceleration of QCh code: can we simply use a BLAS library that implements the BLAS calls by performing the corresponding CuBLAS ones? This idea is simple enough to be tested at once: we run MOLCAS code with the only modification that the matrix multiplication code was replaced by a wrapper to CuBLAS_DGEMM.Test calculations were done on a Tesla GPU, provided by STFC Daresbury Lab. However, the transfer of data between GPU card and CPU requires transfer via temporary allocated memory, and takes time. It is fastest if a portion of memory can be 'pinned' and reused for several calls



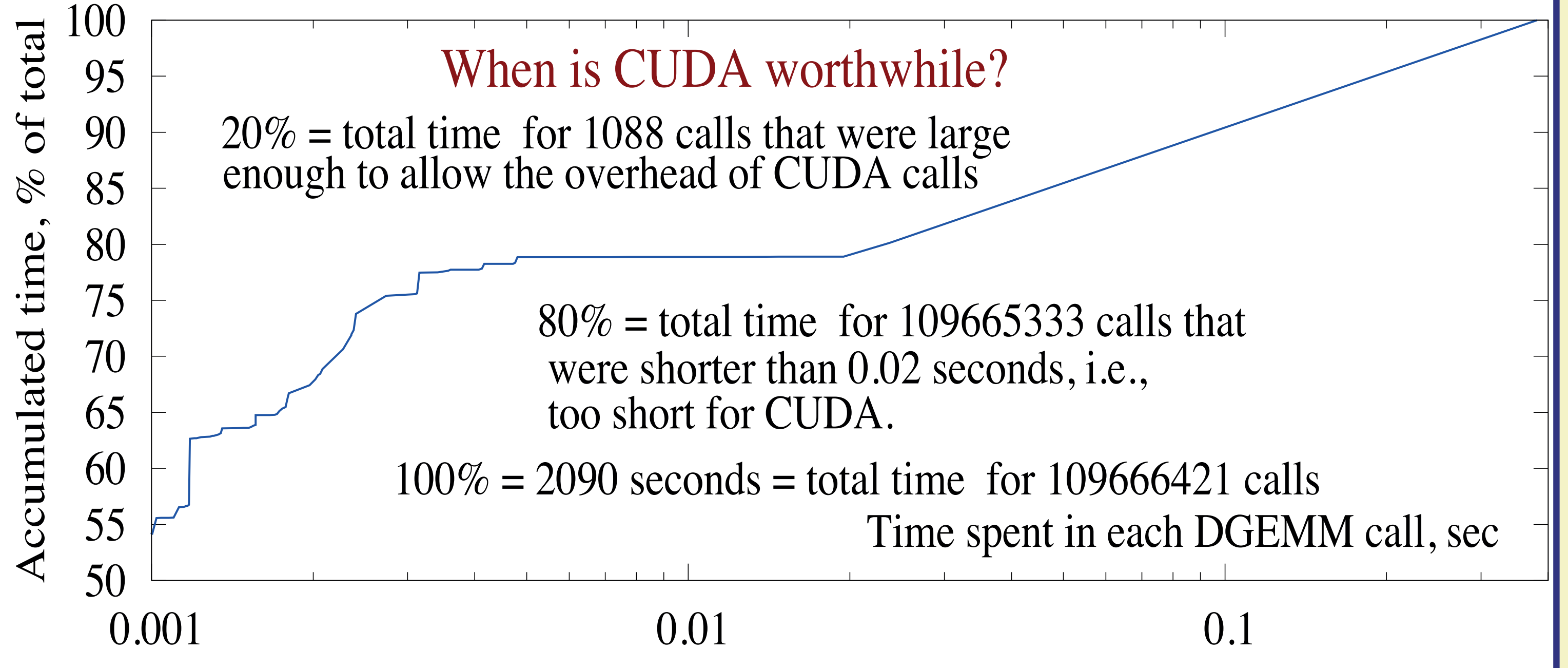


• CuBLAS

Using of CuBLAS_DGEMM is clearly faster for huge matrices. RASSCF and CASPT2 codes spend up to 80% of the time in DGEMM. But do we have such large matrices in a typical calculation? Polyyne was used as a representative RASSCF calculation with 234 basis function and a 4/8/4 active space.

The statistics of DGEMM calls for this test case shows that the percentage of big matrices is very small. Also, these sizes are asymmetric, while CUBLAS_DGEMM is more efficient for square matrix.

Efficiently batched algorithms such as the one in MOLCAS have less advantages from GPU technology. Increasing of connection speed between CPU and GPU might change the conclusion completely.



Conclusions

Hardware development is not driven by the needs of computational chemistry. At the same time, quite often these new technologies can not be used directly, and require some modifications of the application software. Substantial time can be needed to adjust computational codes and take advantage of new computer architectures.

References

[1] W. A. de Jong, E. Bylaska, N.Govind, C. L. Janssen, K. Kowalski, T. Müller, I. M. B. Nielsen, H. J. J. van Dam, V. Veryazov and R. Lindh, Utilizing High Performance Computing for Chemistry: Parallel Computational Chemistry, *PCCP* 12, 6896-6920, (2010).

[2] .F. Aquilante, L. De Vico, N. Ferré, G. Ghigo, P.-Å Malmqvist, P. Neogrády, T. B. Pedersen, M. Pitoňak, M. Reiher, B. O. Roos, L. Serrano-Andrés, M. Urban, V. Veryazov, and R. Lindh. MOLCAS 7: The Next Generation. *J. Comput. Chem.*, 31, 224-247, (2010)