CASPT2! ... but can it fly?

Victor P. Vysotskiy, Valera Veryazov

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

• CASPT2 I/O

Multiconfigurational second-order perturbation method CASPT2 is known as a reliable computational tool for the electronic structure calculations. The original CASPT2 code in MOLCAS has been developed in the beginning of 90s.The main development of the code focused on algorithmic improvements, for example, recent development allows to use RASSCF reference wavefunction. The change of hardware architecture was addressed much less. It is well know fact that a speed of a typical **CASPT2** calculation is limited by storing and reading data, i.e. it is **I/O-bound problem**. Among CASPT2 scratch files, only the two-electron integrals or Cholesky vectors files are read sequentially for several times, while the rest files are accessed constantly and randomly. In other words, the CASPT2 I/O workload is dominated by **random write and read** operations, which is the worst case scenario for conventional HDD, due to the excessively high latency of spinning hard disks. One may expect that a caching mechanism of a underlying filesystem (FS) should improve the overall I/O performance as long as there is no large files and available memory is sufficient for buffering all needed data. However, the caching mechanism is not selective in a sense that it tries to buffer all opened/accessed files simultaneously/uniformly, regardless their sizes and I/O access patterns. Generally speaking, without any assumptions about certain FS and its caching mechanism, the best possible performance of the CASPT2 module can be obtained only by using an electronic data storage device with the lowest available latency and the best random I/O performance like, e.g.,Random Access Memory (RAM), or Solid State Device (SSD).

Although nowadays most of the computers are equipped with large amount of memory, neither CASPT2 code itself, or operating system by caching I/O, can't use this memory in efficient way. In order to utilize RAM directly for I/O we have developed a new framework called as "*Files in Memory*" (*FiM*). The key idea of ***FiM*** is to ***keep*** a ***scratch file in RAM entirely*** instead of using a HDD/SSD disk. In sharp contrast to FS caching, within *FiM* one has an explicit and transparent control on a housing data in RAM. The beauty of *FiM* that it is easy to use for both MOLCAS end user and developer: there is no need to change source code, one just needs to edit an external resource file!

• Benchmarks

•Tests

✓ For I/O benchmarking were selected several typical CASPT2/RASPT2 jobs. In addition, the benchmark set was extended by adding one MCLR test.

✓ The ext3 FS was installed on all storage devices and their was mounted with the "noatime" option. The Lustre FS was tuned within "lfs -c 1 -s 1m". command.

Test	N_{ATOMS}	N_{AO}	N_{CSFs}	ERIs	Method	Filesizes (Mb) ERIs	Filesizes (Mb) CASPT2/MCLR	Time*) (h)
A	21	518	226512	acCD-4	MS-CASPT2(12,12)	4053	9920	5.5
B1	28	279	19404	CD-10	MS-CASPT2(10,10)	558	3108	2.2
B2	28	279	19404	Conventional	MS-CASPT2(10,10)	5821	7336	2.4
C	30	312	1126404	CD-4	MS-RASPT2(4/8/4)	516	4938	5.8
D	54	394	226512	Conventional	MCLR/CASSCF(12,12)	17419	33257	19.2

*)The reported time corresponds to the slowest cacheless computation on HDD, so-called "HDD (NO FS_CACHING)"

•Results

✓ *FiM* provides the best performance;

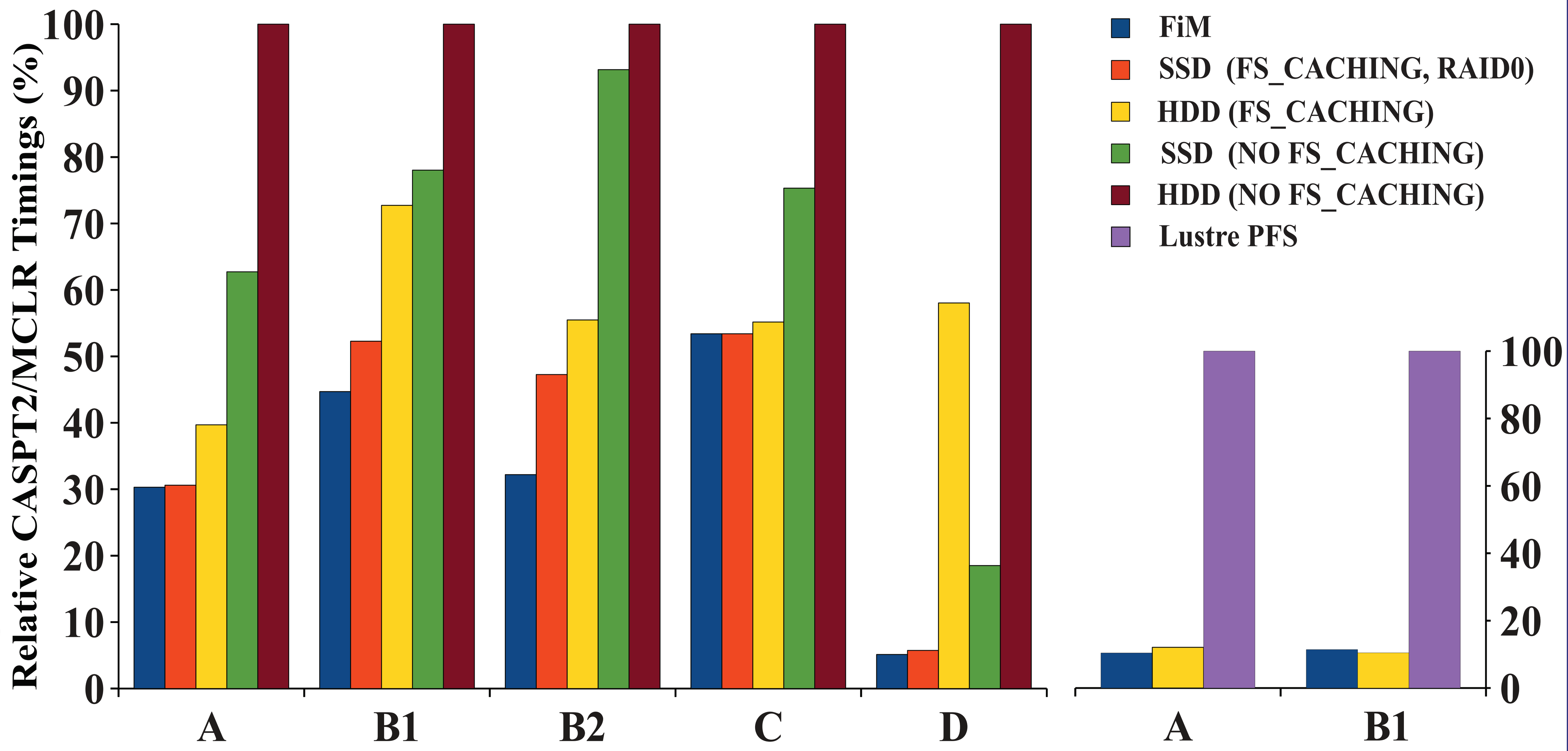
✓ CASPT2: SSD outperforms HDD ~1.1-1.6**x**;
MCLR: SSD outperforms HDD >10**x**;

✓ FS Caching remarkably improves I/O throughput speed by factor of 2;

✓ *FiM* over Lustre FS provides virtually the same performance as a local HDD;

✓ Within *FiM* it is now possible to run MOLCAS on a diskless HPC node/workstation without performance penalty;

✓ *FiM* is useful and powerful tool for data analysis, debugging.



• CASPT2 BLAS3

•DGEMM: GPU vs CPU

CASPT2 code spends up to **80%** of the computational time in **DGEMM.**

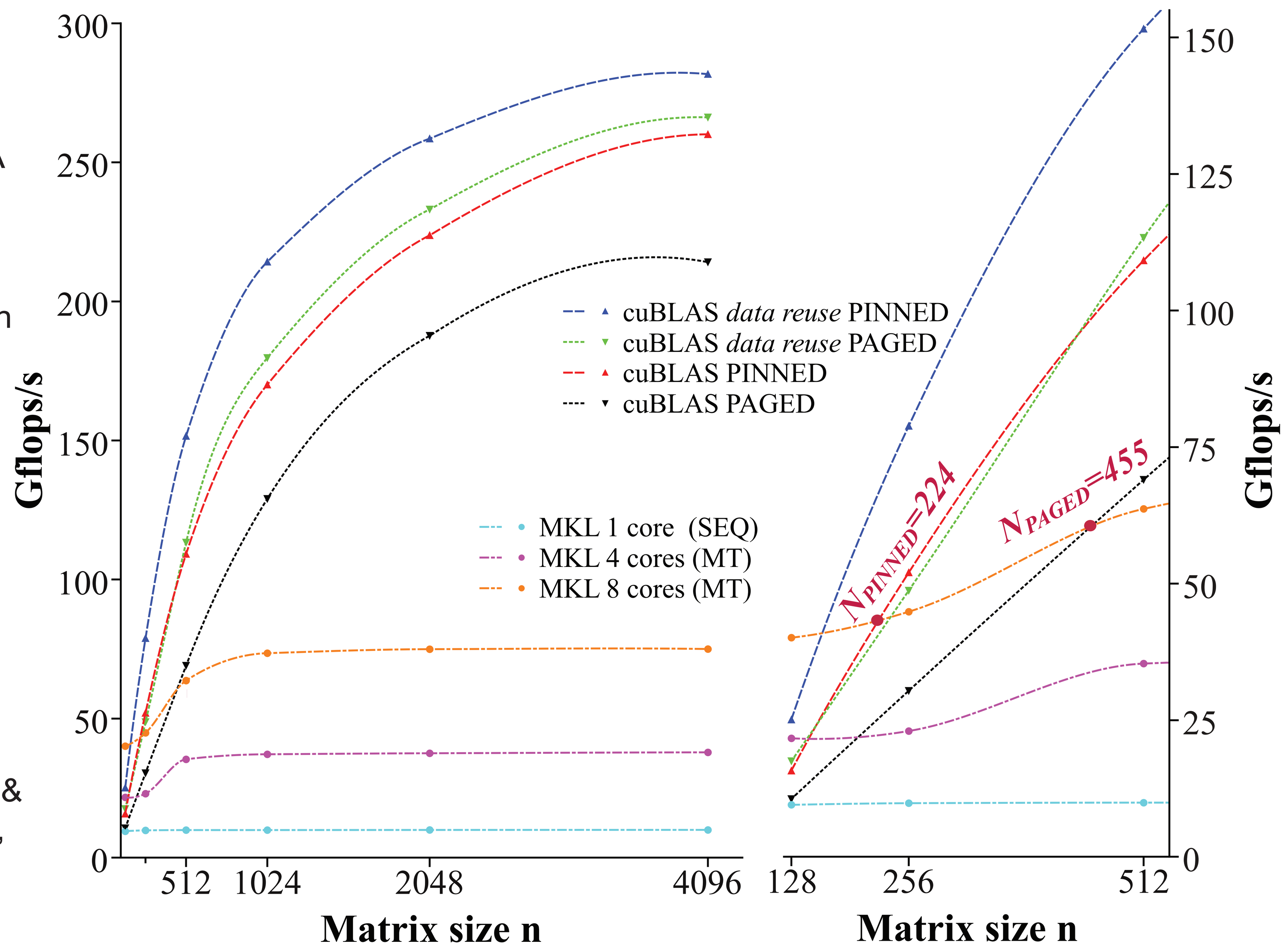
✓ Intel MKL v10.2 on Intel Xeon E5520 (2.27 GHz)

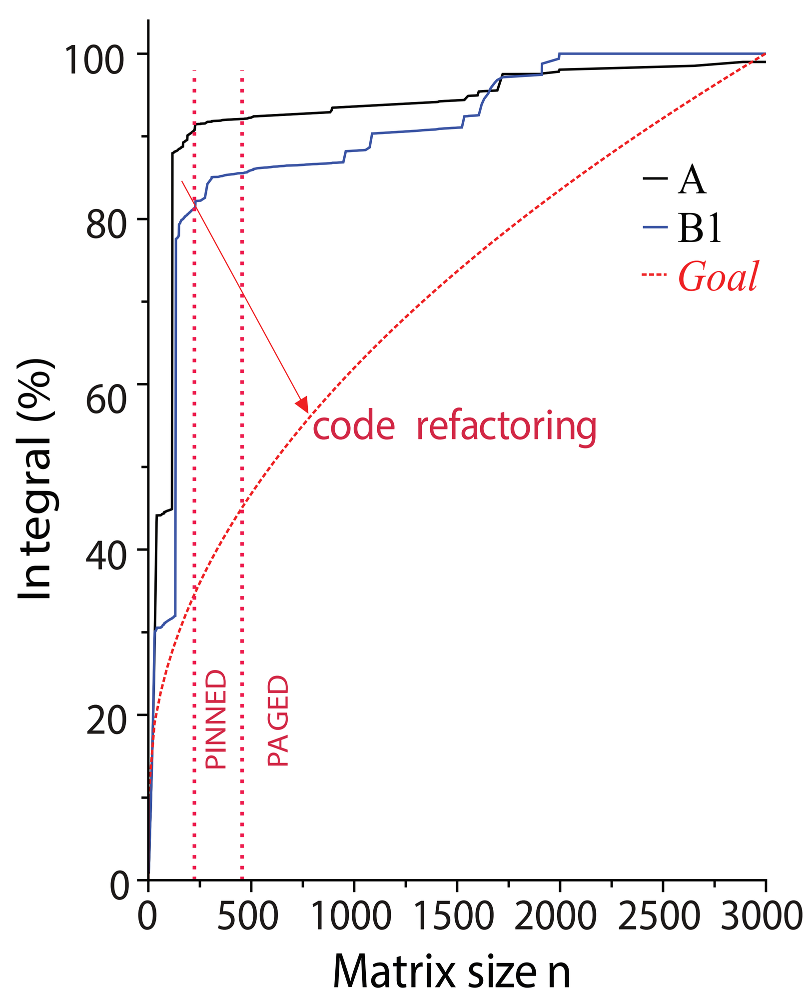
✓ CUDA BLAS v4.1 on NVIDIA Tesla M2050.

✓ *Pinned memory* means that allocated memory pages remain in real RAM all the time.

✓ In *Data reuse* scenario the C matrix was resided on the GPU device

✓ *The previous GPU* hardware generation and corresponding *CUDA* libraries was ***4x*** *times slower* than the current one. In particular, the N_{pinned} and N_{paged} crossing points for Tesla S1070& &cuBLAS v2 are ***796*** and ***1845***, respectively.





• Molcas v8.0

Planning features and changes:

✓ *FiM* with data compression (in progress);
✓ Global Arrays free (MPI-2 and new Memory Allocator);
✓ Better support of many-core architectures, especially regarding the BLAS3 matrix operations.

http://www.molcas.org