Files In Memory: New RAM based I/O Layer in MOLCAS

Victor P. Vysotskiy, Valera Veryazov

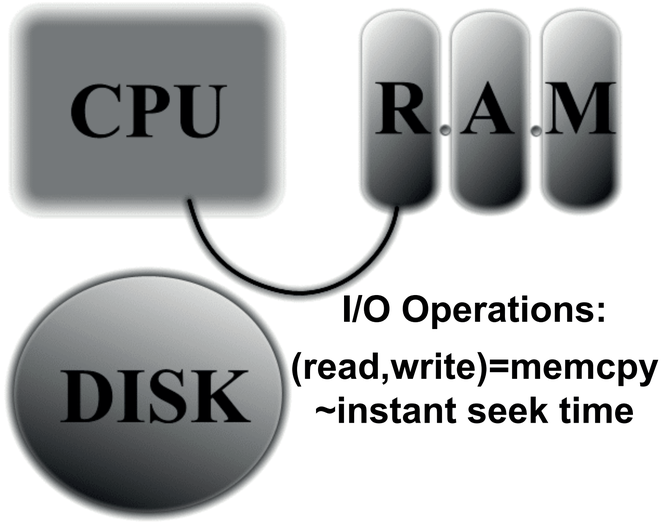
Theoretical Chemistry, Chemical Center, P.O.B. 124, Lund 22100, Sweden, E-mail: Victor.Vysotskiy@teokem.lu.se

• 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 [1].The main development of the code focused on algorithmic improvements, for example, recent development allows to use RASSCF reference wavefunction [2], and the Cholesky Decomposition method [3]. 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 performanceas 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).

• Files In Memory

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 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. By design, **FiM is capable to place data in Sys V shared memory** segments and thus can be shared between several different **MPI** processes running on the same node at **no extra message passing cost**.



Unlike to the memory-resident I/O layer of CRAY FFIO [4], **FiM is a general framework** and **can be used on any POSIX compliant operation system** such as Linux, AIX, Windows, Solaris.

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! In addition, **FiM** provides **environment variables** that control the execution of MOLCAS and automatic (dynamical) switching between I/O layers at **runtime**.

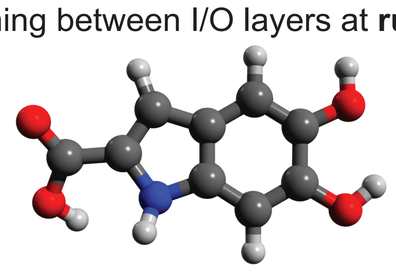
• Benchmarks

✓ Hardware configuration:
- 2-way Intel Xeon CPU E5630 (2.53GHz);
- 48 Gb of DDR3 (1066MHz) RAM;
- 2 Intel SSDSC2MH250A2 250GB are attached to the RocketRaid 62x SATA RAID 6Gb/s Controller (RAID1);
- 1 HDD WDC WD10EURS-630AB1 SATA II 1000 Gb.

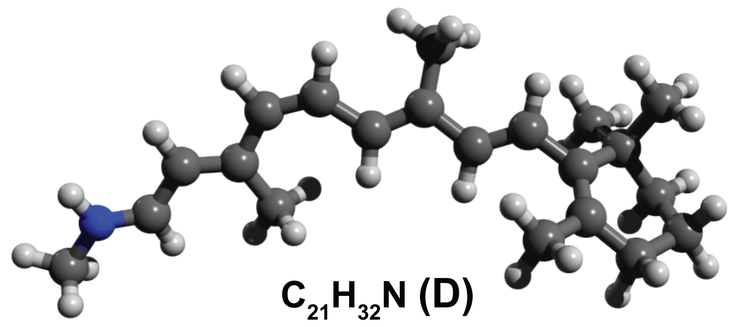
✓ The ext3 FS was installed on all storage devices and disks were mounted with the "noatime" option. The Lustre FS was tuned within "lfs -c 1 -s 1m" command.

✓ The "NO FS_CACHING" results were obtained by using only 4Gb of RAM (the rest 44 Gb of RAM were physically removed prior benchmarking).

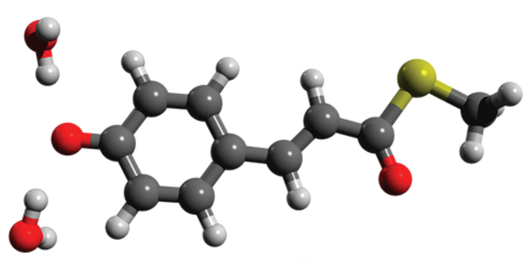
✓ For I/O benchmarking were selected several typical CASPT2/RASPT2 jobs. In addition, the benchmark set was extended by adding one MCLR test.



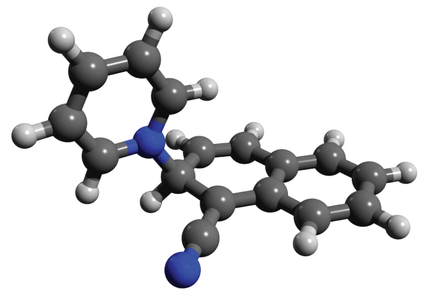
$C_9H_7NO_4$ **(A)**



$C_{21}H_{32}N$ **(D)**



$C_{10}H_{14}O_4S$ **(B)**



$C_{16}H_{12}N_2$ **(C)**

Test	N_{ATOMS}	N_{AO}	N_{CSFs}	ERIs	Method	Filesizes (Mb)		Time*) (h)
						ERIs	CASPT2/MCLR	
A1	21	518	226512	acCD-4	MS-CASPT2(12,12)	4053	9920	5.5
A2	21	518	226512	Conventional	MS-CASPT2(12,12)	102751	40652	15.7
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 I/O performance;

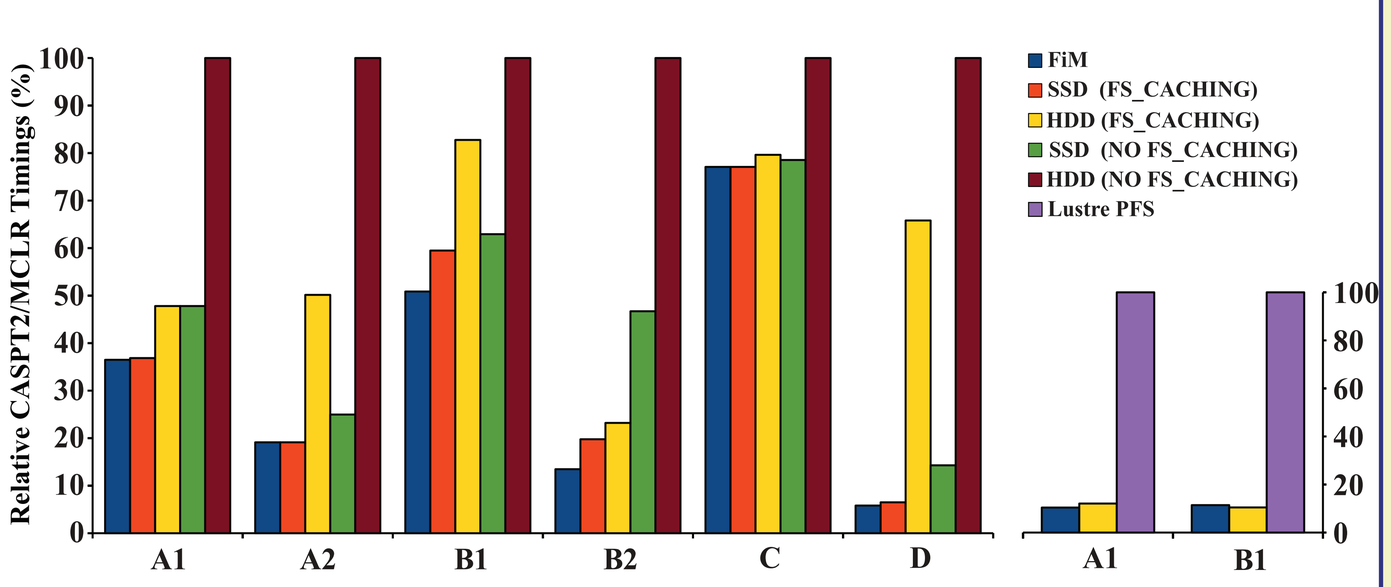
✓ CASPT2: SSD outperforms HDD ~1.1-1.6**x**;
MCLR: SSD outperforms HDD >10**x**;

✓ In the case of HDD, FS Caching remarkably improves I/O throughput speed;

✓ 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 can help make more efficient use of the shared memory on SMP nodes, thus mitigating the need for explicit intra-node communication.



• References

[1] K. Andersson, P.-Å. Malmqvist, B. O. Roos, A. J. Sadlej, K. Wolinski, *J. Phys. Chem.* 94, 5483-5488 (1990).
[2] P.-Å. Malmqvist, K. Pierloot, A. R. Moughal Shahi, C. J. Cramer, L. Gagliardi, *J.Chem. Phys.* 128, 204109(1-10) (2008).
[3] F. Aquilante, L. D. Vico, N. Ferré, G. Ghigo, P.-Å. Malmqvist, P. Neogrády, T. B. Pedersen, M. Pitoňák, M. Reiher, B. O. Roos, L. Serrano-Andrés, M. Urban, V. Veryazov, R. Lindh, *J. Comput. Chem.* 31, 224-247 (2010).
[4] Cray T3E™ Fortran Optimization Guide - 004-2518-002, Chapter 5. Input/Output.

http://www.molcas.org