Received: 2 March 2017 | Revised: 2 April 2017 | Accepted: 6 April 2017

DOI: 10.1002/qua.25392

FULL PAPER

WILEY International Journal of QUANTUM CHEMISTRY

Performance analysis of open-source distributed file systems for practical large-scale molecular *ab initio*, density functional theory, and GW + BSE calculations

Loïc M. Roch[1,2] | Tyanko Aleksiev[3] | Riccardo Murri[3] | Kim K. Baldridge[1,2]

[1]Department of Chemistry, University of Zürich, Winterthurerstrasse 190, Zürich 8057, Switzerland

[2]School of Pharmaceutical Science and Technology, University of Tianjin, 92 Weijin Road, Nankai District, Tianjin 3000072, People's Republic of China

[3]Service and Support for Science IT, University of Zürich, Winterthurerstrasse 190, Zürich 8057, Switzerland

Correspondence
Dr. Kim K. Baldridge, School of Pharmaceutical Science and Technology, University of Tianjin, 92 Weijin Road, Nankai District, Tianjin 3000072, People's Republic of China.
Email: kimb@tju.edu.cn

Funding information
The authors acknowledge the National Basic Research Program of China (2015CB856500), the Qian Ren Scholar Program of China, and the Synergistic Innovation Center of chemical Science and Engineering (Tianjin) for support of this work.

Abstract

Application of conventional post-Hartree–Fock theory, density functional theory, and GW + BSE theory, on chemical and biological problems of growing molecular scale, has created a need for innovative designs in high-performance computing infrastructures and associated middleware technologies. The growing trend toward petascale computing creates an enormous demand for highly efficient storage solutions. Notably, in reaching limitations of the *scale-up* model, technological progress, and continual growth in demand has evolved into solutions that alternatively *scale-out*. This investigation involves an analysis of commonly used open source storage solutions for high demand *ab initio* calculations. Comparisons are shown for the most competitive solutions for a broad range of workloads, following disk usage of key *ab initio* algorithms in the widely used open source electronic structure theory software, GAMESS. Assessment is based on results of read/write speeds and ratios, scaling ability, and impact of block-size on overall performance of the distributed storage. GlusterFS is shown to generally outperform CEPH and local disk file systems, providing the most reasonable alternative to local storage, and is therefore recommended for large-scale molecular calculations.

KEYWORDS

ab initio calculations, CEPH, distributed file system, GlusterFS, HPC

1 | INTRODUCTION

The constant need for greater computational capacity and capability in computational and theoretical quantum chemistry[1] contributes significantly to the development of high-performance computing (HPC) platforms, and efficient methods to exploit them.[1–3] Massively parallel architectures and efficient parallel software solutions have been critical in the transition of sophisticated *ab initio* algorithms for more routine application.[4] There are now several highly regarded and widely used quantum chemistry software suites for which significant efforts have been extended to optimize time to solution for large-scale computations.[5–11] In these efforts, highly parallelized algorithms have been established for many of the fundamental capabilities, such as computation of two-electron integrals,[12–14] second-order Møller–Plesset energies,[15] parallel multireference singles and doubles configuration interaction (MRCISD),[16] and Coupled Cluster calculations,[17–20] to name but a few. Seminal works of many prominent research groups have lead to availability of highly parallel electronic structure theory software[21–28] as well as cost-effective capabilities for multiscale modeling,[29–33] all of which have had important impact on basic understanding of molecular processes.

In the reach toward petascale computing, while the performance of HPC systems has drastically improved with advancements in software and hardware,[34] a *fit-for-all* programming language to deal with widely differing types of parallelism, has yet to be found. This has imposed limitations for ideal scaling for certain algorithms and associated applications.[35] Over and above efficient parallel programming languages and sophisticated performance monitoring, efficient scaling is additionally related to tuning and profiling details of the HPC architecture and associated systems tools.

A major limitation in scaling performance concerns the sheer amount of data generated by large-scale computations, which has drastically increased with the study of real-world chemistry applications. Many highly parallel scientific applications require a file system that is shared across

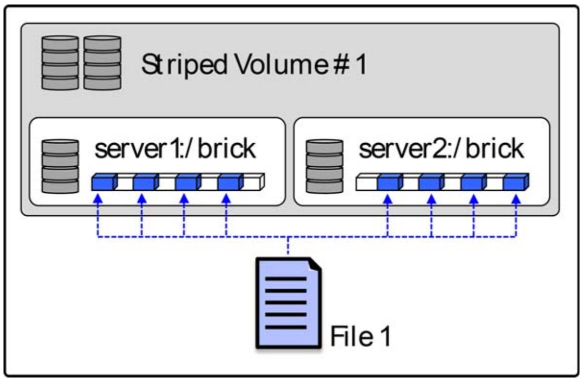


FIGURE 1 Simplified layout of a striped trusted pool consisting of two bricks

compute nodes. Such a file system allows access to files for processes running on different servers, and additionally enables high capacity for jobs with significant storage requirements. In this aspect, application of conventional electronic structure theory on chemical and biological problems of growing scale, and thereby, of growing storage demand, has created a need for innovative design modifications in HPC infrastructure.[36] Notably, in reaching limitations of the *scale-up* model,[37] technological progress and continual growth in disk storage requirements evolved into solutions that alternatively *scale-out* (In the *scale-up* approach disk space is increased by adding resources to a physical system, while in the *scale-out* approach, disk space is increased by connecting multiple entities to work as a single logical unit).[37] Although performance assessment of various scale-out solutions has been reported in the context of computer science, climatography and engineering in the literature,[38–44] benchmarking studies for practical large-scale molecular *ab initio* calculations are sorely missing, making unavailable any quantitative assessment of expected performance. In the current work, utilization of a distributed file system is shown to enable processing of very large datasets across a HPC system, thereby facilitating computations that would otherwise be extremely difficult if possible at all. In particular, several commonly used open source storage solutions are discussed, and the most viable alternatives assessed against local file storage, as to their performance across a broad range of *ab initio* theory workloads for large-scale molecular applications.

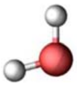
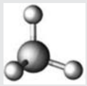
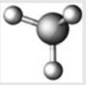
1.1 | Distributed file systems

Distributed file systems[45] (DFS) provide a solution to the limitations of the scale-up model for file storage. Such file systems scale-out on multiple servers, expand transparently from the user and applications perspective, and appear as a single high-capacity disk storage. The architectures of DFS used for accessing storage over a network are similar to network attached storage (NAS). That is, they provide standard POSIX file system functionality.[46–48] Commonly used DFSs include BeeGFS,[49] Lustre,[50] CEPH,[51] and GlusterFS.[52]

In the current work, the open source software project GlusterFS has been investigated and compared to traditional local file system and to CEPH. Both GlusterFS and CEPH enable distributed processing of large-scale datasets across a clustered environment. However, due to its simplicity in all operational aspects, most notably, for initial setup and configuration, GlusterFS is a good candidate for quick setup of a scale-out storage, and requires minimum knowledge in Linux systems administration.[53] The fact that GlusterFS is supported by the most popular Linux distribution along with the fact that it does not require additional hardware to create a DFS particularly simplifies the overall configuration. Additionally, GlusterFS does not separate meta-data[‡] ([‡]Readers are referred to the supporting information for additional definitions of computer-specific terminology) from data and therefore does not rely on a separate meta-data server, neither centralized nor distributed. For example, both CEPH and Lustre require additional hardware components for their operation. On one hand, a storage infrastructure configured with CEPH needs at least three more computational resources where the so-called monitors are executed. Lustre, on the other hand, requires at least two additional nodes and a disk array to store the objects meta-data of the whole cluster. These additional components contribute to the complexity in both the initial setup and operation of Lustre and CEPH, requiring dedicated support to properly configure and operate the latter DFSs. Bootstrapping time from *off-the-shelf* hardware deployment to creating, formatting, and mounting the volume to form a large GlusterFS DFS is reduced to a few days (A user with little knowledge in Linux can create, format, and mount a GlusterFS DFS within two days). From a practical point of view, everyday maintenance tasks have been proven to be straightforward. Additionally, GlusterFS is easily tunable based on the specific application needs. Last, thanks to its ability to aggregate the storage capacity of multiple servers, combined to fault-tolerance and data-striping features, GlusterFS provides the ideal DFS for quantum chemistry programs such as GAMESS, since they create large temporary working files saving temporarily results while the computation progresses. The benefits of the aforementioned characteristics of a file system are attractive for academic use, enabling the build of a simple scalable storage cluster meeting specific application requirements.

Figure 1 illustrates the notion of a striped trusted pool consisting of two logical volumes, or bricks, which could be either a single HD or a virtual Redundant Array of Independent Disks[‡] (RAID) storage. As depicted, striping allows the splitting of files into smaller blocks and distribution of those files over multiple bricks. This mechanism enables storage of large files, which could extend beyond the capacity of a single brick.

TABLE 1 Disk usage (expressed in write percentage), number of basis functions, and file size (in MB), generated by DFT calculations for water and methane dimers

System	Basis set	Basis Functions	File size (MB)	Write %
	CCD	50	25	99.78
	ACCD	86	82	99.84
	CCT	130	258	99.90
	ACCT	210	2150	99.97
	CCQ	280	5939	99.99
	ACCQ	430	45 056	98.85
	CC5	532	67 584	95.56
	ACC5	784	436 224	29.12
	CC6	924	544 768	15.58
	CCD	70	44	99.77
	ACCD	122	287	99.75
	CCT	190	1126	99.97
	ACCT	310	14 336	99.98
	CCQ	420	28 672	99.99
	ACCQ	650	227 328	34.56
	CC5	812	329 728	35.13
	ACC5	1204	2 411 725	14.59
	CC6	1428	2 726 298	15.58

CCx refers to cc-pVxZ and ACCx to aug-cc-pVxZ Dunning-style basis sets.[62–67]

GlusterFS provides a native redundancy mechanism based on the replication of files across a logical collection of bricks[‡] forming the trusted storage pool.[‡] The notion of a replicated volume is highly desirable in environments where high-availability and high-reliability are critical, such as those found in large-scale *ab initio* computations. Combining RAID redundant virtualization technologies to the GlusterFS replication mechanism provides additional redundancy. In the present study, the impact of replicated volumes on read/write performance is not the primary focus, as the choice of replication mechanism to provide redundancy is case specific.

Herein, a detailed performance study of GlusterFS is carried out on a broad range of workloads following disk usage of the GAMESS software,[5] a widely used open source quantum chemistry software. As such, read/write speeds are measured on a broad range of workloads, and the scaling ability and impact of block-size on the general performance is assessed and discussed. Performances are further compared to traditional local file system and to CEPH. Because recent technological advances in HPC hardware and software are not always readily available to the computational chemistry community, focus is on the performance of *off-the-shelf* systems.

The remainder of this article is organized as follows. Section 2.1 briefly describes the HPC infrastructure used to assess the performance of GlusterFS. Section 2.2 highlights the methodology used in evaluating the read/write ratio, which leads to the design of the test suites (section 2.3). The configuration of the nodes is detailed in section 2.4. Performance results of GlusterFS, local file system and CEPH are analyzed and discussed in section 3. Before drawing conclusions in section 5, an illustration of the impact of GlusterFS for carrying out two large-scale *ab initio* research calculations is provided in section 4.

2 | METHODS

2.1 | High performance compute platform

The nodes used in the tests described here make up the 6352-hyperthreaded core high-performance computing infrastructure, *Arran*, designed and built by this group of authors. The nodes run the Linux distribution Ubuntu 14.04 LTS server and GlusterFS 3.4.2. The network architecture follows a star topology[54] (see Figure S2 Supporting Information). The inter-node communication reaches 20 Gbps via direct-attached copper cables, through port-channeling over two 10 Gbps interfaces per node. Each compute-node contains two hyper-threaded Intel Xeon E5-2690 v3 at

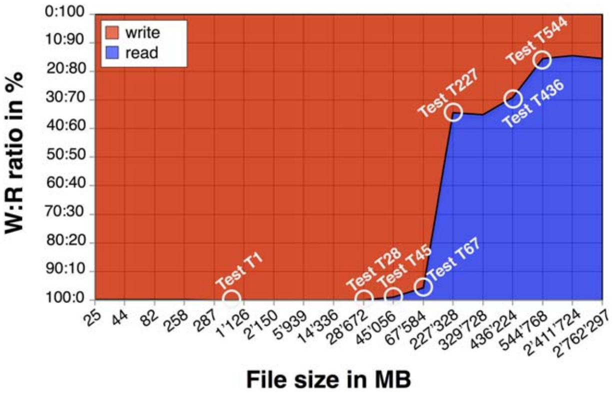


FIGURE 2 Representation of the GAMESS read/write ratio as a function of the file size (in MB). The reads are represented in blue, and the writes in red. File sizes are as reported in **Table 1**. Note that the scale of the horizontal axis is not linear

2.6 GHz, for a total of 48 cores. The total memory per node reaches 256 GB, as 8x 32 GB LR-DIMM, at 2133MT/s. Each node has ten 1.2 TB SAS 6 Gbps HDDs, at 10 kRPM, and a 1GB NV cache RAID controller.

2.2 | I/O benchmarks

The disk usage of GAMESS was established by evaluating the read/write ratio as a function of the total job size using statistics provided by the "kernel-meter" *nfsstat*, which displays the workload of the NFS[55] client and server. The usage of *nfsstat* for collecting, measuring, and analyzing NFS activities is well discussed in the literature,[56–58] and, therefore, will not be repeated here.

Different wave function types and molecular compositions were considered in the evaluation process of the disk usage. Wave function types include density functional theory (DFT),[59,60] Møller–Plesset second-order perturbation theory (MP2),[15] and Coupled Cluster[17] CCSD(T).[61] In all cases, single point calculations with conventional storage of integrals on disk were considered on the water dimer and methane dimer at a variety of basis set levels, as summarized in Table 1. Note that, although these tests systems involve very small molecules, the number of basis functions ranges from 50 to 1428, corresponding to a disk space usage of up to 2.6 TB. Although the exact read/write sequence of DFT, MP2, and CCSD(T) can differ here, the overall read/write ratios for the three different *ansatz* are within a fraction of percent, and, thus, only the DFT results are shown in Table 1 (full tabulation of results can be found in the Supporting Information, Table S1). It is important to emphasize that large-scale high-demand molecular calculations are typically not carried out with direct methods since integral storage in RAM is extremely difficult due to physical or financial reasons. As such, only conventional methods with storage on disk are considered here.

2.3 | Test suites

The performance characteristics of GlusterFS, local file system, and CEPH, are evaluated using FIO 2.1.3,[68] a versatile I/O generator typically used for benchmark and stress/hardware verification. FIO writes and/or reads a job file matching a specific I/O pattern defined by the user. Herein, the latter job file follows the pattern provided by nfsstat on a variety of GAMESS workloads, as depicted in Figure 2 and tabulated in Table 1.

Figure 2 shows a schematic representation of the values tabulated in Table 1, and depicts read/write ratio as a function of the file size. One can see that a plateau is reached at a read/write ratio of about 85/15, which spans file sizes ranging from 544 768 MB (H_2O dimer, at the cc-pV6Z level) to 2 726 298 MB (CH_4 dimer at the cc-pV6Z level). Note that the scale of the horizontal axis is not linear.

Based on the behavior of GAMESS from the above data analysis, seven tests were designed as summarized in Table 2 and depicted in Figure 2. The first two tests (tests T1 and T28) consist of 100% writing, representing the small file size from the I/O benchmark. It is worth mentioning that test T28 is at the upper limit of this 100% writing regime, as evidenced from Table 1. Tests T45 to T544 cover all file sizes reported in Table 1, from 45 056 MB (water, at the aug-cc-pVQZ level) up to 544 768 MB (water, at the cc-pV6Z level), where the aforementioned plateau at a read/write ratio of about 85/15 is reached. Since the two largest files, 2 411 725 and 2 726 298 MB, give read/write ratios similar to the smaller 544 768 MB file size, it was not necessary to include all three in the test suite as the speed performance obtained for large file-size, for example, 26 TB and 34 TB as illustrated in section 4, is expected not to differ from the results obtained at a file size of 544 768 MB.

All seven tests administered follow a mixed random read and write I/O pattern issued with the Linux native asynchronous I/O, known as *libaio*. The rationale for random reads and writes lies in the architecture of spinning disks, where access patterns are rarely continuously sequential.[69] Because block-sizes change from one disk file used by GAMESS to another (e.g., external basis library, atomic coordinates, kinetic integrals, overlap integrals, etc.), spanning a few kB to several MB, the lower bound scenario with block-sizes of 4 kB were used for the tests. Typically, read and write

TABLE 2 Read and write percentages and file size for the designed seven tests

Tests	Test size (MB)	Reads (%)	Writes (%)
T1	1024	0	100
T28	28 672	0	100
T45	45 056	1	99
T67	67 584	5	95
T227	227 328	65	35
T436	436 224	71	29
T544	544 768	85	15

system calls are intercepted and recorded using strace,[70] a standard userspace utility for Linux typically used as diagnostic, instructional and debugging tool. The theoretical network usage N_{use}, is given by

$$N_{use} = HD_{sustained} \times N_{HD} \tag{1}$$

where $HD_{sustained}$ is the sustained read/write speed of the disk, and N_{HD} is the number of disks. In the present study, N_{use} reaches 16 Gbps in the best-case scenario when all the 10 disks sequentially write at 204 MB/s,[71] corresponding to about 80% of the maximum network speed. The network bandwidth as such is guaranteed not to be a limiting factor on the speed measurement.

2.4 | Node configuration

The performance of different GlusterFS scale-out storage configurations is investigated by computation of each of the tests described in section 2.3 (Table 2) on different node combinations. In this regard, eight groups were setup, as described in Table 3. The first set investigated, as represented by groups L-1.2T, L-12T, G-1.2T, and G-12T, enables comparison of GlusterFS as a user space file system (FUSE)[72] with a native in-kernel file system on one single server, and, thus, with a storage size from 1.2 to 12 TB. Additionally, the granularity in this particular set of groups shows the impact of an underlying bare disk versus RAID0 virtualization technology. The second set investigated, groups G-24T, G-48T, G-96T, highlights the effect of the striping‡ mechanism implemented in GlusterFS. In this particular case, striping size was set to the number of nodes representing the distributed volume: that is, 1, 2, 4, and 8, respectively. In either case, the *ext4* file system was created using the default values as provided by the front-end Linux file system builder, *mkfs*.

The RAID0 virtualization for groups L-12T, G-12T, G-24T, G-48T, and G-96TB was configured with *No Read Ahead*, and *Write Back* policies. The stripe size was set to 64 kB. The GlusterFS volumes were created via the following command line: gluster volume create scratch stripe ***N*** transport tcp ***N***x[compute:/data/brick1/vg0] where N is the number of compute-nodes used in the group (e.g., $N = 8$ for Group G-96T).

Performance results were further compared to CEPH, version 0.94.8 (group C-96T, Table 3). The architecture used consisted of three monitor nodes and three storage nodes. While the monitor nodes are hosted on blade servers, the storage nodes are comprised of one head node and two directly attached storage enclosures. The latter have 12x 4TB 7.2K RPM Near-Line SAS disks for a total of 24 disks per storage node. CEPH has its own way to provide high availability at the data level: the stored files are split into multiple smaller objects files forming the building blocks of the CEPH cluster. In this particular case, these object files are replicated three times. In addition, CEPH defines failure domains, which confine the distribution of the replica. In the environment used for the tests the failure domain is down to a storage node preventing the situation where two

TABLE 3 Description of the eight groups of node used to assess the performance of Local file system (L), GlusterFS (G), and CEPH (C)

Groups	Descriptions
L-1.2T	1× bare 1.2 TB disk
L-12T	1× 12 TB RAID0 virtual disk
G-1.2T	1× 1.2 TB brick
G-12T	1× 12 TB RAID0 virtual distributed disk
G-24T	2× [1× 12 TB RAID0 virtual] distributed bricks
G-48T	4× [1× 12 TB RAID0 virtual] distributed bricks
G-96T	8× [1× 12 TB RAID0 virtual] distributed bricks
C-96T	2× 12× 4 TB OSDs

TABLE 4 Average read speed in (kB/s) of the eight groups of compute-nodes for the five tests with non-zero reads

	Tests				
Groups	T45	T67	T227	T436	T544
L-1.2T	50	154	643	635	934
L-12T	388	601	885	843	1289
G-1.2T	337	1823	2220	1524	916
G-12T	422	1920	3930	1185	1119
G-24T	488	1894	6635	1083	992
G-48T	431	2006	6324	6118	6318
G-96T	457	1972	6318	6346	6841
C-96T	294	619	542	–	–

replicated objects are saved in the same storage node. Hence, an overall disk space of 96TB is provided to the clients. The seven tests (Table 2) were executed on a CEPH client using a RBD volume mapped to a physical 1TB block device formatted with the default options of *mkfs.ext4*.

3 | SPEED PERFORMANCE: ANALYSIS AND DISCUSSION

Results for the seven tests described in Table 2 and administrated on the eight groups of nodes listed in Table 3 have been collected over three runs, ensuring, thus, the robustness of the results. Detailed data regarding read and write speeds are tabulated in Tables S2–S7 in the Supporting Information. Performances are averaged and summarized in Tables 4 and 5 for the read and write performance, respectively. For clarity, the analysis is carried out separately for read performance and write performance results hereafter.

3.1 | Read performance

Figure 3 and Table 4 summarize the read results obtained with FIO. By comparing the random read operations performed on the groups of nodes consisting of a single server only (i.e., groups L-1.2T, L-12T, G-1.2T, and G-12T) with the GlusterFS nodes (groups G-1.2T and G-12T), one can see that the latter groups of node outperform the native file system, for example, tests T45 and T67 in particular. This observation is explained by introducing the notion of page caching. The GlusterFS distributed file system was mounted without the *direct io* option, which enables the usage of page caching. In Linux, the cache memory buffers memory access and provides a faster channel to the requested data. The *read()* method, used by the applications, will initially try to read from the page cache and fetch what has been required. If pages are not preset, the kernel will put the application in sleep mode and read from the disk, which will increase waiting time for the read() request to complete. When the *write()* method is called, pages are written in cache and the kernel will eventually flush changes to the disk. As a consequence, a higher rate of cached page hits for files that were recently written to and for which the entire content has greater probability to still reside in the cache, justifying therefore the slightly higher performance of GlusterFS over to the native file systems.[72] Tests T436 and T544 confirm the latter assumption that, with significantly bigger file sizes ($>$ 400 GB) that are above the eventual cache capabilities, a performance drop is observed such that almost no difference between the FUSE

TABLE 5 Average write speeds in (kB/s) for the eight groups of compute-nodes for the seven tests

	Tests						
Groups	T1	T28	T45	T67	T227	T436	T544
L-1.2T	72 567	72 695	5039	2933	346	259	164
L-12T	386 251	427 820	38 390	11 433	476	344	227
G-1.2T	45 829	43 338	41 682	34 658	1196	622	161
G-12T	48 021	46 395	41 737	36 513	2117	484	197
G-24T	45 189	52 053	48 309	36 013	3573	442	175
G-48T	48 251	40 152	42 620	38 148	3406	2499	1115
G-96T	52 527	49 772	45 193	37 495	3402	2592	1207
C-96T	838 190	157 892	294	619	542	–	–

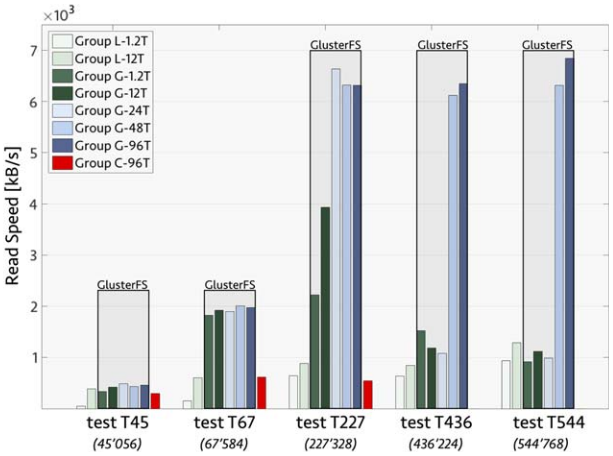


FIGURE 3 Performance results for reads. Green-scale bars show the results on a single host (groups L-1.2T, L-12T, G-1.2T, and G-12T), and blue-scale bars the results on multiple nodes, that is, 2, 4, and 8 nodes for Groups G-24T, G-48T, and G-96T, respectively. While the results framed in a black box (groups G-1.2T, G-12T, G-24T, G-48T, and G-96T) are the GlusterFS results, the unframed bars are the *ext4* results and the CEPH results in red (group C-96T)

and the native file systems is noticed. Additional tests composed by 100% reads further confirm this observation (see Table S8 in the Supporting Information).

Groups G-12T, G-24T, G-48T, and G-96T revealed clear evidence of cache usage. In particular, a linear performance growth, with no difference in speed, is noticed until test T67. Test T227 is the breaking point for group G-12T, where more page cache misses implies direct reads from the disk, yielding speed degradation. For group G-24T, the combined cache capacity of two nodes yields an increased read speed up to test T227. However, as depicted in Figure 3, a performance drop is measured starting from test T436. In the case of groups G-48T and G-96T, a constant performance growth is observed up to test T544 due to the combined capacity of the cache across the nodes (4 and 8 nodes, respectively), which can still page-cache the designed file sizes.

By comparing group L-1.2T and G-1.2T, the performance of GlusterFS is revealed: from test T45 to test T436 the scale-out storage is between 2.4 and 11.9 times faster than the scale-up storage. For test T544, the read speed is slightly above 900 kB/s in both cases. Indeed, reading of large files does not benefit from a scale-out storage setup. Such poor performance on small files has already been reported in the literature.[8]

With a total storage of 12 TB (group L-12T and G-12T), GlusterFS results in faster performance for tests T67, T227, and T436, showing a clear benefit from the synergy between RAID virtualization and GlusterFS. Indeed, RAID systems allow data striping, making it possible for data segmentation over different physical disks. Such a benefit can be better evaluated in a comparison of groups L-1.2T and G-1.2T.

The ability of GlusterFS to scale out is evaluated with groups G-12T, G-24T, G-48T, and G-96T. On test T45, the read speed is roughly similar for all scenarios. Group G-24T is twice as fast as group G-1.2T on test 5 and displays similar speed on tests T436 and T544. Groups G-48T and G-96T, however, have greater speed with larger file sizes.

Further performance analysis of GlusterFS is carried out by a comparison of the read speed as obtained with CEPH. Because the aim is investigation of large DFS for large-scale molecular calculations, the CEPH volume is setup to compare with the largest GlusterFS pool, that is, 96 TB.

From Table 4 and Figure 3, it is shown that the performances of group C-96T are much lower than that obtained for group G-96T: a ratio ranging from 1.6 (test T45) to 11.7 (test T227) is noted, which makes CEPH comparable to the read speed of the local file system group L-1.2T, and L-12T.

3.2 | Write performance

The FIO results for write performance are summarized in Table 5 and depicted in Figure 4. Before discussing the actual write performance capabilities, it is relevant to briefly summarize several concepts regarding the data flow between application and disk. When using a native file system, data needs to be copied only once from the kernel page cache to the application, or from the application to the kernel page cache, and subsequently if required, synced to the disk. When using a FUSE file system, data additionally flows from page cache through a special /dev /fuse device until it reaches the kernel page cache and then, if required, it is synced to the disk.

In the case of write performance, one can partially cluster two sets of similarly performing groups. In particular, groups L-1.2T and L-12T up to test T67 have a performance pattern that differs from the rest of the groups (G-1.2T, G-12T, G-24T, G-48T, and G-96T). More

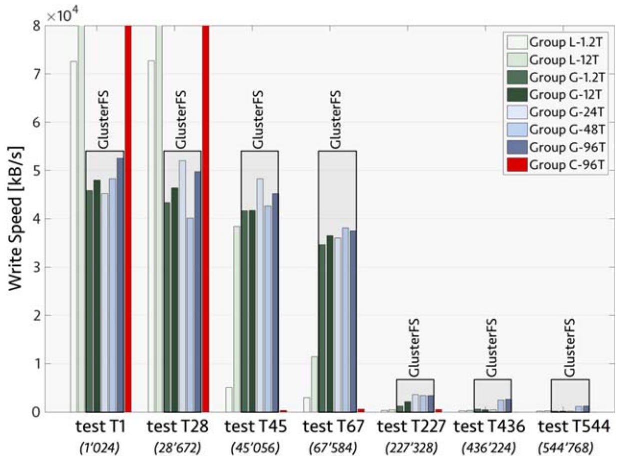


FIGURE 4 Performance results for writes. Green-scale bars show the results on a single host (groups L-1.2T, L-12T, G-1.2T, and G-12T), and blue-scale bars the results on multiple nodes, that is, 2, 4, and 8 nodes for groups G-24T, G-48T, and G-96T, respectively. While the results framed in a black box (groups G-1.2T, G-12T, G-24T, G-48T, and G-96T) are the GlusterFS results, the unframed bars are the *ext4* results and the CEPH results in red (group C-96T)

precisely, the benefit from the direct access between application (i.e., FIO) and kernel page cache is evident for the smaller files (test T1 and T28), where a significant part of the write operations most probably takes place directly in the cache. Group L-12T outperforms for those two tests with an average speed of 349 MB/s due to the underlying RAID0, allowing the files to stripe over the 10 disks. With a write speed of 838 190 kB/s, CEPH is twice as fast as group L-12T on test T1. While similar performances were observed for groups L-1.2T and L-12T on both tests T1 and T28 (i.e., ca. 72 000 kB/s, and 400 000 kB/s, respectively), a performance drop is observed with group C-96T from test T1 to test T28. As a matter of fact, a drop of a factor of 5.5 is observed. GlusterFS, however, gives similar performance across tests T1 to T67. The additional layer of caching implemented on the FUSE file systems is the most plausible explanation to the latter initial difference.

The performance speed drastically drops in all the groups for tests T227 to T544, but drops already at test T45 for group C-96T to reach only a few hundreds of kB/s. Test T227 is the breaking point for the write operation where the benefit of caching is eliminated. More specifically, due to the larger file size, the cache will be promptly filled limiting the speed to that required for flushing or syncing the data to disk. The correlation between the two levels of caching within a FUSE file system and the single level of caching presented in the native file systems is also very well evidenced in Figure 2. In particular, FUSE caching mechanism clearly depends on native file system caching. Last, it is important to point out that any benefit from striping through GlusterFS is rather limited.

As previously mentioned, the 4 kB block-size choice ensures a partial use of the network capacity while assessing the performance of GlusterFS. However, the reported speeds reflect only the lower bound scenario, which is far from the maximum read/write speed of spinning disks. The usage of *strace* enables block-size probing of GAMESS at run time. Depending on the file accessed (e.g., integrals, gradient vector, Hessian matrix, etc.), the block-sizes span a few hundreds of kB to tens of MB. As a result, it is of interest to take a look at the impact of the block-size choice on the read/write speed of group G-96T, a summary of which is given in Table 6.

TABLE 6 Impact of block-size (4 kB and 8 MB) on the read (R) and write (W) speed for group G-96T

G-96T	R, 4 kB	R, 8 MB	W, 4 kB	W, 8MB
T1	7649[a]	763 562[a]	52 527	989 971
T28	7389[a]	812 373[a]	49 772	1 004 859
T45	457	8988	45 193	911 386
T67	1972	41 582	37 495	764 123
T227	6318	324 970	3402	174 211
T436	6346	189 176	2592	77 585
T544	6841	195 092	1207	35 138

[a] 100% read test of 1024 MB (test T1) and 28 672 MB (test T28).

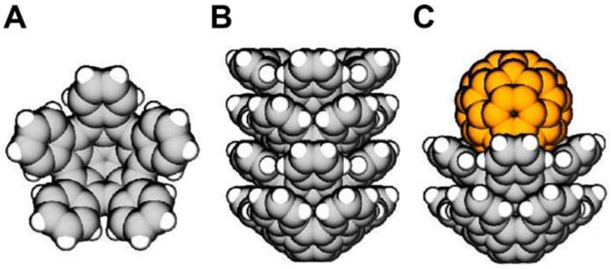


FIGURE 5 Space filling representation of (A) 1, crystal structure of (B) 1_4, and (C) $C_{60}@1_2$ calculated at the B97-D[73]/Def2-TZVP[74] level of theory

As observed from the results in Table 6 the choice of the block-size has a non-negligible impact on the performance. Every application uses a different I/O request size, which can have significant influence on the storage performance. The latter is measured in IOPS and/or throughput according to the following relation,

$$\text{Throughput} = \text{IOPS} \times \text{I/O} \tag{2}$$

Any change in the I/O size impacts both the throughput and the IOPS. When the I/O size is higher, the number of IOPS tends to decrease while throughput generally increases. In the present case, as shown in Table 6, GlusterFS reveals a very similar trend. The DFS outperforms by one or two orders of magnitudes when the I/O ranges from 4 kB to 8 MB. Since the block-sizes of GAMESS runs can vary from a few hundreds of kBs to tens of MBs, the application fully benefits from the higher rate of GlusterFS while using larger block-sizes.

4 | ILLUSTRATIVE RESEARCH APPLICATION

With a robust HPC infrastructure at hand, including a transparent and adaptable file system, one can readily expand research endeavors to include very large-scale chemical systems. This might be in terms of structural analysis at a very high level of theory or wave function type, analysis of any number of properties (NMR, UV-Vis., etc.) with high basis set extent, or, investigation of materials supported in complex environments, to name but a few.

As a case in point, recent work is this research group required us to exploit the technology discussed in this article to carry out a large-scale nanomaterials investigation involving the complexation and aggregation of pentaindenocorannulene (**1**, $C_{50}H_{20}$), Figure 5A, a deep bowl polynuclear aromatic hydrocarbon. In a recent communication,[75] our investigations revealed an interesting new polymorph of **1** that shows several promising materials property characteristics. In the solid state, **1** crystallizes in columnar bowl-in-bowl assemblies (Figure 5B), and additionally is observed experimentally to form a nested $C_{60}@\mathbf{1}_2$ complex (Figure 5C)

The complex interactions across a variety of aggregates from dimers to trimers to tetramers of **1**, as well as complexes of $C_{60}@\mathbf{1}$ and $C_{60}@\mathbf{1}_2$ were investigated in a set of collaborative experimental and theoretical investigations. The study of such large-scale molecular complexes, reaching up to 7680 Cartesian Gaussian basis functions in the case of $\mathbf{1}_4$ and 6000 for the case of $C_{60}@\mathbf{1}_2$ was only made possible through the use of a GlusterFS extended file system capability. Indeed, in terms of structural optimization using the GAMESS software, the d-orbitals symmetry transformation generates 34 TB for the tetramer, $\mathbf{1}_4$, and 26 TB for the $C_{60}@\mathbf{1}_2$ complex. This exceeds the limit of the typical local storage (10 TB in the present system).

In addition, the exciton properties in the crystal were investigated. In terms of materials properties, the nature of the molecular system can modify solid-state packing structure and thereby the optoelectronic properties. Quasiparticle excitation energies were calculated using GW theory, in this case using the Berkeley GW package.[8] This provides a first-order correction to the DFT starting point within the generalized gradient approximation, and optical excitations by explicit inclusion of electron-hole interactions, via the BSE approach.[76]

Figure 6 displays the organic crystal setup for 1. The pentaindenocorannulene crystals contain 4 molecules per unit cell with lattice vectors: $a = 8.498$ Å, $b = 17.885$ Å, $c = 21.953$ Å, $\alpha = 90°$, $\beta = 90°$, $\gamma = 90°$, for a total of 280 atoms (880 electrons). To build the dielectric function and self-energy requires a total of 1025 bands of states. The BSE sum was computed with 20 occupied and 20 unoccupied bands and expanded through a fine k grid in a supercell of 8 × 4 × 4 primitive cells. One of the heaviest parts of the calculation concerns the computation of the polarizability of the system. The *epsilon* executable computes the static RPA polarizability using the following expression[8]:

$$\chi_{GG'}(\mathbf{q};0)=\sum_{n}^{\text{occ}}\sum_{n'}^{\text{emp}}\sum_{\mathrm{k}} M_{nn'}(\mathrm{k},\mathrm{q},\mathrm{G})M^{*}_{nn'}(\mathrm{k},\mathrm{q},\mathrm{G'})\frac{1}{E_{n\mathrm{k}+\mathrm{q}}-E_{n'\mathrm{k}}}. \tag{3}$$

Equation 3 involves a summation over transitions between the occupied (occ) states and the empty (emp) states, and is dependent on the number of k points (coarse grid for epsilon calculation: 4 × 2 × 2 primitive cells (16 k points; fine grid for optical calculation: 8 × 4 × 4 primitive cells,

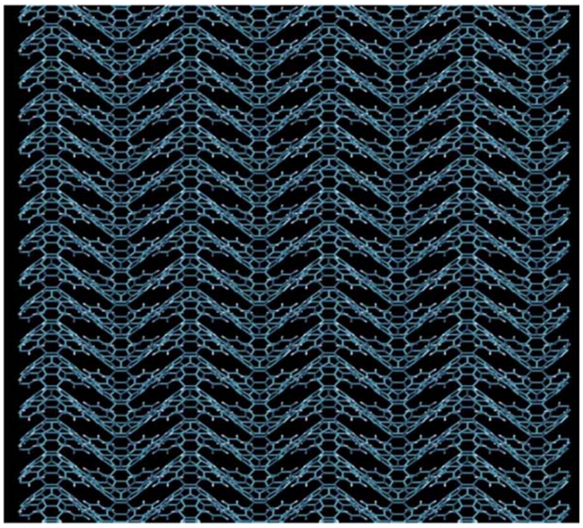


FIGURE 6 Depiction of the crystalline arrangement for **1**

that is, 128 *k* points). The matrix elements are computed using Fast Fourier Transforms (FFTs), scaling as $N^3\log N$, where *N* is the total number of atoms.[7] The calculation is also very demanding in terms of CPU time for determination of the self-energy (sigma), the convergence of which is quite critical with respect to the dielectric energy cutoff, and the corresponding number of empty states.

Such a calculation would not be affordable, and typically not even possible, using traditional computational resources, and required a GlusterFS setup as described in this work. Due to memory requirement (RAM), such a calculation running on, for example, 256 cores of Arran, requires 11 nodes for the epsilon part, all accessing a common temporary storage (specific to BerkleyGW). The calculation of sigma for 121 states requires about 7 days.

5 | CONCLUSIONS

Many efforts are ongoing to move computational chemistry toward the petascale to tackle large-scale challenges of real world problems, for example, multithousand atom biological systems, or, various complex environments involved in investigations for large nanomaterials design. It is of great interest to find facilitating solutions that enable the general user to proceed with such studies without having to consider specialty platforms. This work has elaborated on one particular bottleneck as presented by large-scale data storage and manipulation. In this regard, the performance of GlusterFS on various workloads was investigated, following the disk usage of the widely used electronic structure theory software, GAMESS, for illustration. Not only assessment of the read/write speed, scaling ability, and impact of block-size were reported, but also, further comparison with CEPH as well as local file storage, was carried out. The in-house designed test suite spans 1 GB to 532 GB of disk usage with variation in read/write ratios. Furthermore, the impact of the block-size was measured on the performance of GlusterFS, assessing the ability of the latter to scale-out.

Overall, GlusterFS generally outperforms CEPH and file systems on local disk when large sustained I/O is performed, and therefore, is a good alternative to local storage. Such behavior has already been reported in the literature.[38] For read operations, a double caching layer generally increases the number of memory hits, and, thus, translates into higher performance rates for larger files. Although the write operations sustainably outperform the scale-up speed by cache aggregation, in the write operations are limited by the cache paging of the kernel. As discussed, GlusterFS does not benefit from striping. Finally, challenging research examples provide illustration of extremely large capacity data loads, which are only enabled through the use of the GlusterFS-enabled setup. In this application, glusterd (GlusterFS deamon) required up to 300% of the CPU, meaning that four threads per compute-node are to be allocated to the distributed file system. Consequently, in a production environment, it is strongly recommended to allocate 5-10% of the CPU resources per node to managing the distributed file system.

As mentioned in the discussion regarding the DFS, an additional strong benefit of GlusterFS is its simplicity. As a matter of fact, CEPH requires additional hardware components for its operation: three more computational resources are required to execute the so-called monitors. These additional components contribute to the complexity in both the initial setup and operation of CEPH, requiring dedicated support to properly configure and operate the latter DFS. In the case of GlusterFS, however, bootstrapping processes and ordinary maintenance procedures can be achieved with little Linux system administration knowledge. These aspects are of significant importance within the scientific community, which often require commodity storage to be set up for higher common scratch capacity.

Based on available works comparing GlusterFS to various DFS such as CEPH, HDFS, PVFS, and NFS, GlusterFS performs the best on a broad type of systems, although it can present instabilities in some cases.[38,40,42] Furthermore, with the ability to export local storage along with fault-tolerance and data-striping features, GlusterFS provides an ideal DFS for quantum chemistry programs such as GAMESS, since these applications create large temporary working files, saving partial results while the computation progresses. Lustre is considered as a high-performance DFS, appropriate for large-scale HPC infrastructures with up to tens of thousands of clients for multipetabytes storage. Consequently, it could be a possible alternative to GlusterFS. However, as mentioned, Lustre requires at least two additional nodes and a disk array to store the objects meta-data of the whole cluster. These extra components contribute to the complexity in both the initial setup and operation of Lustre, requiring dedicated support for proper configuration and operation. Due to this higher complexity, Lustre is beyond the scope of this study, although its performance is expected to be at least as good as that reported for GlusterFS.

All in all, GlusterFS appears as an excellent choice for quantum chemistry HPC clusters specialized in computational chemistry and is, therefore, recommended for large-scale quantum chemistry calculations.

ACKNOWLEDGMENT

The authors acknowledge the National Basic Research Program of China (2015CB856500), the Qian Ren Scholar Program of China, and the Synergistic Innovation Center of chemical Science and Engineering (Tianjin) for support of this work.

REFERENCES

[1] C. R. Bertozzi, C. J. Chang, B. G. Davis, M. O. de la Cruz, D. A. Tirrell, D. Zhao, *ACS Cent. Sci.* **2016**, *2*, 1.

[2] C. Engelmann, S. L. Scott, C. Leangsuksun, X. He, *J. Comput.* **2006**, *1*, 43.

[3] E. Lusk, T. Sterling, in Beowulf Cluster Computing with Linux, The MIT Press Cambridge, Massachusetts, London, England **2003**.

[4] R. Breslow, M. V. Tirrell, J. K. Barton, M. A. Barteau, C. R. Bertozzi, R. A. B. P. Gast, I. E. Grossmann, J. M. Meyer, R. W. Murray, P. J. Reider, W. R. Roush, M. L. Shuler, J. J. Siirola, G. M. Whitesides, P. G. Wolynes, R. N. Zare, in *Beyond the Molecular Frontier: Challenges for Chemistry and Chemical Engineering*, National Academies Press: Washington, D.C., USA **2001**.

[5] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, T. L. Windus, M. Dupuis, J. A. Montgomery, *J. Comput. Chem.* **1993**, *14*, 1347.

[6] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. J. A. Montgomery, J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Ö. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski, D. J. Fox, *Gaussian 09 Revision E.01*, Gaussian, Inc., Wallingford CT, **2016**.

[7] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo. A. Dal Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, Ch. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, R. M. Wentzcovich, *J. Phys. Condens. Matter* **2009**, *21*, 395502.

[8] J. Deslippe, G. Samsonidze, D. A. Strubbe, M. Jain, M. L. Cohen, S. G. Louie, *Comput. Phys. Commun.* **2012**, *183*, 1269.

[9] G. Kresse, J. Hafner, *Phys. Rev. B* **1993**, *47*, 558.

[10] J. VandeVondele, M. Krack, F. Mohamed, M. Parrinello, T. Chassaing, J. Hutter, *Comput. Phys. Commun.* **2005**, *167*, 103.

[11] M. Häser, R. Ahlrichs, *J. Comput. Chem.* **1989**, *10*, 104.

[12] J. A. Pople, W. J. Hehre, *J. Comput. Chem.* **1978**, *27*, 161.

[13] H. F. King, M. Dupuis, *J. Comput. Phys.* **1976**, *21*, 144.

[14] M. Dupuis, J. Rys, H. F. King, *J. Chem. Phys.* **1976**, *65*, 111.

[15] C. Møller, M. S. Plesset, *Phys. Rev.* **1934**, *46*, 618.

[16] M. W. Schmidt, M. S. Gordon, *Annu. Rev. Phys. Chem.* **1998**, *49*, 233.

[17] J. Čížek, *J. Chem. Phys.* **1966**, *45*, 4256.

[18] N. Flocke, R. J. Bartlett, *J. Chem. Phys.* **2004**, *121*, 10935.

[19] G. E. Scuseria, P. Y. AyalaView, *J. Chem. Phys.* **1999**, *111*, 8330.

[20] C. Riplinger, F. Neese, *J. Chem. Phys.* **2013**, *138*, 034106.

[21] R. M. Olson, J. L. Bentz, R. A. Kendall, M. W. Schmidt, M. S. Gordon, *J. Chem. Theory Comput.* **2007**, *3*, 1312.

[22] J. L. Bentz, R. M. Olson, M. S. Gordon, M. W. Schmidt, R. A. Kendall, *Comput. Phys. Commun.* **2007**, *176*, 589.

[23] J. Dieterich, D. Krisiloff, A. Gaenko, F. Libisch, T. Windus, M. Gordon, E. Carter, *Comput. Phys. Commun.* **2014**, *185*, 3175.

[24] Z. Gan, Y. Alexeev, M. S. Gordon, R. A. Kendall, *J. Chem. Phys.* **2003**, *119*, 47.

[25] T. L. Windus, M. W. Schmidt, M. S. Gordon, *Theor. Chim. Acta* **1994**, *89*, 77.

[26] T. Ramdas, G. K. Egan, D. Abramson, K. K. Baldridge, *Comput. Phys. Commun.* **2008**, *178*, 817.

[27] T. Ramdas, G. K. Egan, D. Abramson, K. K. Baldridge, *Theor. Chem. Acc.* **2008**, *120*, 133.

[28] B. Bolding, K. K. Baldridge, *Comput. Phys. Commun.* **2000**, *128*, 55.

[29] M. S. Gordon, D. G. Fedorov, S. R. Pruitt, L. V. Slipchenko, *Chem. Rev.* **2012**, *112*, 632.

[30] Q. A. Smith, K. Ruedenberg, M. S. Gordon, L. V. Slipchenko, *J. Chem. Phys.* **2012**, *136*, 244107.

[31] K. R. Brorsen, N. Minezawa, F. Xu, T. L. Windus, M. S. Gordon, *J. Chem. Theory Comput.* **2012**, *8*, 5008.

[32] L. V. Slipchenko, M. S. Gordon, *J. Comput. Chem.* **2007**, *28*, 276.

[33] S. R. Pruitt, M. A. Addicoat, M. A. Collins, M. S. Gordon, *Phys. Chem. Chem. Phys.* **2012**, *14*, 7752.

[34] M. Feldman, *Petaflop Club Closes in on 100 Members*. **2016**. https://www.top500.org/news/petaflop-club-closes-in-on-100-members/. Accessed September 9, 2016.

[35] R. Ho, J. Bradbury, K. Barton, *10th Workshop on Challenges for Parallel Computing*, IBM Corp., Markham, Canada **2015**, pp. 307–309.

[36] W. A. de Jong, E. Bylaska, N. Govind, C. L. Janssen, K. Kowalski, T. Muller, I. M. B. Nielsen, H. J. J. van Dam, V. Veryazov, R. Lindh, *Phys. Chem. Chem. Phys.* **2010**, *12*, 6896.

[37] H. El-Rewini, M. Abd-El-Barr, Advanced Computer Architecture and Parallel Processing, Wiley: Hoboken, New Jersey, USA, **2005**.

[38] D. Giacinto, M. Giovanni, D. J. Domenico, *Phys. Conf. Series* **2014**, *513*, 042014.

[39] E. B. Boyer, M. C. Broomfield, T. A. Perrotti, in *Computing and Information Technology Student Mini Showcase*, SciTech Connect: Los Alamos, New Mexico. Accessed August **2012**.

[40] G. Juve, E. Deelman, G. B. Berriman, B. P. Berman, P. Maechling, *J. Grid. Comput.* **2012**, *10*, 5.

[41] Z. Liu, B. Wang, T. Wang, Y. Tian, C. Xu, Y. Wang, W. Yu, C. A. Cruz, S. Zhou, T. Clune, S. Klasky, *22nd Int. Conf. on Computer Communication and Networks. (ICCCN)*, (Ed: IEEE Spectrum), July–August, **2013**, New York.

[42] S. Saini, J. Rappleye, J. Chang, D. Barker, P. Mehrotra, R. Biswas, *19th Int. Conf. on High Performance Computing* (Ed: IEEE Spectrum), December 2012, New York, 1.

[43] H. Luu, M. Winslett, W. Gropp, R. Ross, P. Carns, K. Harms, M. Prabhat, S. Byna, Y. Yao, A Multiplatform Study of I/O Behavior on Petascale Supercomputers, ACM: Portland, Oregon **2015**, pp. 33–44.

[44] S. A. Brandt, D. D. E. Long, C. Maltzahn, E. L. Miller, S. A. Weil, *7th Conference on Operating Systems Design and Implementation (OSDI'06)*, University of California, Santa Cruz **2006**, pp 307–320.

[45] E. Levy, A. Silberschatz, *ACM Comput. Surv.* **1990**, *22*, 321.

[46] S. Shirinbab, L. Lundberg, D. Erman, *Int. J. Comput. Appl.* **2013**, *20*, 195.

[47] R. Noronha, L. Chai, T. Talpey, D. K. Panda, Designing NFS with RDMA for Security, Performance and Scalability, The Ohio State University: Xi'an, China **2007**.

[48] G. A. Gibson, R. Van Meter, *Commun. ACM* **2000**, *43*, 37.

[49] BeeGFS, http://www.beegfs.com/content/. Accessed October 28, **2016**.

[50] S. Oral, G. Shipman, F. Wang, Understanding Lustre FileSystem Internals, Oak Ridge National, Laboratory (ORNL), Center of Computational Science: Oak Ridge, USA, **2009**.

[51] CEPH, https://ceph.com. Accessed February 22, **2017**.

[52] GlusterFS, https://www.gluster.org. Accessed December 21, **2016**.

[53] D. Alex, O. Alessandro, *Linux J.* **2013**, *2013*, 72.

[54] A. S. Tanenbaum, D. J. Wetherall, Computer Networks, Pearson Education: Boston, USA, **2010**.

[55] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, B. Lyon, UNIX to Berkeley, California: USENIX Association, **1985**, 119.

[56] F. Blusseau, in *Winter Simulation Conf. Proc.* (Ed: ACM), December **1989**, New York, 971–979.

[57] G. L. Schaps, P. Bishop, in 7th System Administration Conf . (Ed: LISA), November **1993**, USENIX Association, Berkeley, CA, 165–170.

[58] D. Ellard, M. Seltzer, in *17th Large Installation System Administration. Conf*. (Ed: LISA), October, **2003**, USENIX Association, Berkeley, CA, 73.

[59] P. Hohenberg, W. Kohn, *Phys Rev B* **1964**, *136*, 864.

[60] W. Kohn, L. J. Sham, *Phys Rev A* **1965**, *140*, 1133.

[61] K. Raghavachari, G. W. Trucks, J. A. Pople, M. Head-Gordon, *Chem. Phys. Lett.* **1989**, *157*, 479.

[62] T. H. Dunning, Jr., *J. Chem. Phys.* **1989**, *90*, 1007.

[63] R. A. Kendall, T. H. Dunning, Jr., R. J. Harrison, *J. Chem. Phys.* **1992**, *96*, 6796.

[64] D. E. Woon, T. H. Dunning, Jr., *J. Chem. Phys.* **1993**, *98*, 1358.

[65] T. van Mourik, A. K. Wilson, T. H. Dunning, Jr., *Mol. Phys.* **1999**, *96*, 529.

[66] A. K. Wilson, D. E. Woon, K. A. Peterson, T. H. Dunning, Jr., *J. Chem. Phys.* **1999**, *110*, 7667.

[67] T. van Mourik, T. H. Dunning, Jr., *Int. J. Quantum Chem.* **2000**, *76*, 205.

[68] J. Axboe, *Freshmeat Project Website* **2011**. Accessed October 24, 2016.

[69] C. Ruemmler, J. Wilkes, *UNIX disk access patterns; Computer Systems Laboratory*, Hewlett-Packard Laboratories, Palo Alto, CA **1993**.

[70] Strace, https://strace.io. Accessed February 5, 2017.

[71] Seagate, *Enterprise Performance 10K HDD* **2013**. Accessed October 26, 2016.

[72] A. Rajgarhia, A. Gehani, Performance and Extension of User Space File Systems, ACM, Sierre, Switzerland **2010**.

[73] S. Grimme, *J. Comput. Chem.* **2006**, *27*, 1787.

[74] F. Weigend, R. Ahlrichs, *Phys. Chem. Chem. Phys.* **2005**, *7*, 3297.

[75] S. Lampart, L. M. Roch, A. K. Dutta, R. Warshamanage, A. D. Finke, A. Linden, K. K. Baldridge, J. S. Siegel, *Angew. Chem. Int. Ed.* **2016**, *128*, 1.

[76] M. Rohlfing, S. G. Louie, *Phys. Rev. B* **2000**, *62*, 4927.

SUPPORTING INFORMATION

Additional Supporting Information may be found online in the supporting information tab for this article.

How to cite this article: Roch LM, Aleksiev T, Murri R, Baldridge KK. Performance analysis of open-source distributed file systems for practical large-scale molecular *ab initio*, density functional theory, and GW + BSE calculations. *Int J Quantum Chem.* 2018;118:e25392. https://doi.org/10.1002/qua.25392