Evaluating Lustre's Metadata Server on a Multi-Socket Platform

Konstantinos Chasapis, Manuel F. Dolz, Michael Kuhn, Thomas Ludwig
Department of Informatics, University of Hamburg
Bundesstraße 45a, 20146 Hamburg, Germany
Email: {firstname.lastname}@informatik.uni-hamburg.de

***Abstract*—With the emergence of multi-core and multi-socket non-uniform memory access (NUMA) platforms in recent years, new software challenges have arisen to use them efficiently. In the field of high performance computing (HPC), parallel programming has always been the key factor to improve applications performance. However, the implications of parallel architectures in the system software has been overlooked until recently. In this work, we examine the implications of such platforms in the performance scalability of the Lustre parallel distributed file system's metadata server (MDS). We run our experiments on a four socket NUMA platform that has 48 cores. We leverage the `mdtest` benchmark to generate appropriate metadata workloads and include configurations with varying numbers of active cores and mount points. Additionally, we compare Lustre's metadata scalability with the local file systems ext4 and XFS. The results demonstrate that Lustre's metadata performance is limited to a single socket and decreases when more sockets are used. We also observe that the MDS's back-end device is not a limiting factor regarding the performance.**

I. Introduction

Servers used in current high performance computing (HPC) systems tend to use multiple sockets with non-uniform memory access (NUMA) architectures. Software developers have to efficiently exploit the increasing number of cores per socket to make full use of these systems. Furthermore, it is necessary to take special care when accessing memory because non-optimal accesses can negatively impact performance [1]. Although many applications have already adapted to such architectures system software is still in the early stage. Researchers are working towards this direction proposing new operating system designs suitable for multi-core architectures [2], [3], [4].

Today, large HPC systems are equipped with dedicated storage systems that have capacities in the range of tens of petabytes. According to [5], [6] the number of files in such systems is in the range of billions, where single directories can host more than millions of files. Scientific applications often create per-process or per-iteration files, resulting in large amounts of metadata operations within the parallel file system. Due to this, as the number of cores – and consequently, processes – increases, high metadata performance becomes important for overall file system performance.

In contrast to data servers, metadata server requests are mainly small random accesses, making latency an important factor. Multi-core and multi-socket platforms can be used to serve many metadata requests in parallel. Apart from that, using multi-socket systems allows increasing the maximum amount of main memory since it is limited per socket. Moreover, the amount of storage devices and the maximum storage bandwidth are also limited per socket; contemporary storage systems support upwards of 80 hard disk drives (HDDs) per storage server [7].

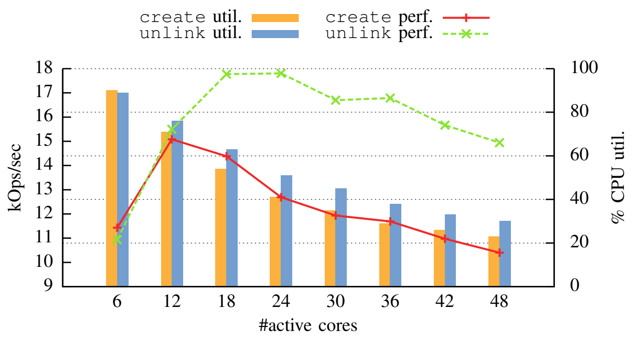

Figure 1: Lustre performance of the `create` and `unlink` operations with respect to the number of active cores and CPU utilization during experiment runs.

In this paper, we focus our research on the parallel distributed file system Lustre [8]. Figure 1 shows the scaling behavior of Lustre's metadata server and operations (in this case, `create` and `unlink`) when increasing the number of active cores on a four socket machine. Each socket consists of 12 cores and the number of processes is twice the number of active cores in this configuration. For the `create` operation, we see that even though the CPU utilization is high when using 6 cores, the number of operations per second only increases as long as only one socket is used; as soon as more sockets are used, the performance steadily decreases and Lustre can not make use of the additional cores. The `unlink` operation behaves in a similar fashion but the performance increases up to 18 cores. A more detailed explanation of these effects is provided in Section III.

The main contributions of this paper are: *i)* an extensive performance evaluation and analysis of the `create` and `unlink` operations in Lustre, *ii)* a comparison of Lustre's metadata performance with the local file systems ext4 and XFS, and *iii)* the identification of hardware best suited for Lustre's metadata server.

The rest of this paper is organized as follows: Section II gives a short overview of Lustre. Next, Section III presents experimental results of our evaluation. Afterwards, Section IV discusses related work in the field. Finally, Section V summarizes our findings.

II. LUSTRE FILE SYSTEM

Lustre is an open-source, POSIX-compliant, parallel distributed file system that is widely used in HPC systems [8], [9] and is implemented in the Linux kernel. The left side of Figure 2 illustrates the generic Lustre architecture. Lustre separates metadata and data servers. The *metadata servers* (MDSs) are responsible for maintaining the file system tree, permissions, etc. File I/O is handled by the *object storage servers* (OSSs). Metadata and data are stored on the *metadata targets* (MDTs) and the *object storage targets* (OSTs) that are attached to the MDSs and the OSSs, respectively. Lustre uses a back-end file system for data storage. Currently, `ldiskfs` and ZFS [10] are supported. In our evaluation we leverage `ldiskfs`, which is based on `ext4` [11].

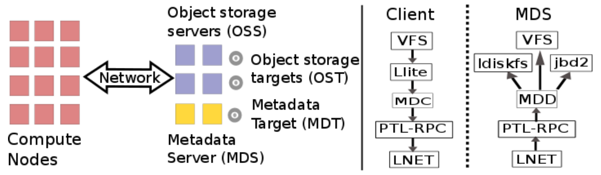

Figure 2: Lustre's general architecture (left side) and metadata operation path (right side).

On the right side of Figure 2, we illustrate the metadata operation path. On the client side, the operations are initiated from the user space and propagated to the virtual file system (VFS). Then, the `llite` Lustre module forwards the metadata requests to the `mdc` module. `ptlrpc` allows to reorder the handling of remote procedure calls (RPCs) on the servers. The communication between the servers and clients is handled by the `lnet` (Lustre networking) module. On the server side, `lnet` accepts the request and forwards it to `ptlrpc`. Afterwards, the request is propagated to the `mdd` module, which uses the back-end file system modules to serve the request. A more detailed explanation of Lustre internals can be found in [12].

The following Lustre modules consumed more than 0.1 % of CPU time during our experiments: `osd_ldiskfs` connects `ldiskfs` with Lustre. `obdclass` code is a generic Lustre configuration and device handling module. `libcfs` provides an API to support fundamental primitives and subsystems such as process management and debugging. `lvfs` is a permission interface. `mdd` and `mdt` implement metadata handling. The journaling block device (`jbd2`) provides a file-system-independent interface for file system journaling and is used by `ext4`.

In recent releases of Lustre there have been several improvements in metadata operation handling and performance. Symmetric multiprocessing (SMP) improvments have been added in version 2.0 and 2.1. From version 2.3, the performance in shared directories has been optimized using fine grained directory locking. In version 2.4 the Distributed Name spacE (DNE) was introduced, allowing the metadata to be distributed across multiple MDTs [13]. The second phase of the DNE, launched in version 2.6, allows to stripe a single directory across multiple MDTs. A design for SMP improvements in the Lustre's MDS is illustrated in [14].

III. EVALUATION

In this section we present the experimental results from Lustre's MDS evaluation. Moreover, in order to identify potential bottlenecks imposed by the Linux kernel we also evaluate the local file system metadata performance.

A. Testbed specifications

To conduct our evaluation we use a four socket server, called MAGNY, consisting of: *i)* a Supermicro motherboard model H8QG6 [15], *ii)* 4× AMD Opteron 6168 Magny-Cours 12-core processors running at 1.9 GHz [16], *iii)* 128 GB of DDR3 main memory running at 1,333 MHz, *iv)* a Western Digital Caviar Green 2 TB SATA2 HDD and *v)* 2× Samsung 840 Pro Series 128GB SATA3 SSDs.

Each socket of MAGNY has 2 NUMA nodes and supports quad-channel DDR3 [17] memory. The sockets are interconnected using 16-bit AMD HyperTransport [18] (version 3.0) that has a maximum unidirectional bandwidth of 10.4 GB/s. Figure 3 illustrates the block diagram of the server components. The actual memory layout is important to know because accessing remote memory is slower than accessing local memory; consequently, NUMA-unaware applications often suffer from latency and network bandwidth penalties. The measured memory throughput on MAGNY is ≈ 8.7 GB/s for local accesses, while remote accesses decrease performance by about 55 % to ≈ 4.0 GB/s.

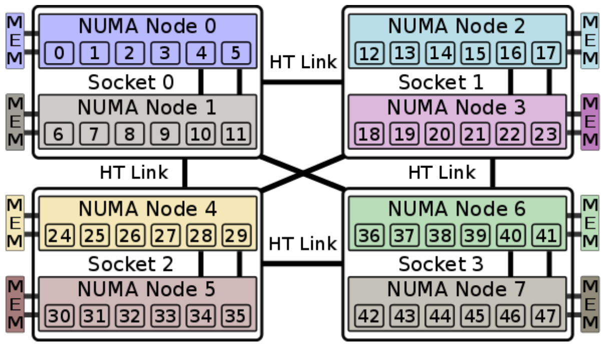

Figure 3: MAGNY motherboard block diagram.

MAGNY runs CentOS 6.4 with the patched Lustre kernel version 2.6.32-358.6.2.el6. For all the experiments we set the Linux governor to `performance`, which operates all the cores at the maximum frequency and gets the maximum memory bandwidth [19].

B. Lustre MDS measurements

We use `mdtest` [20] to generate metadata workloads for the Lustre MDS evaluation. This MPI-parallelized benchmark runs in phases, where in each phase a single type of POSIX metadata operation (`create`, `stat` or `unlink`) is issued to the underlying file system. To avoid interferences between phases, we unmount the file system and drop the kernel caches after each phase. For the evaluation, we only focus on the file creation (`create`) and removal (`unlink`) operations; we do not consider the `stat` operation since its performance also depends on the OSSs. Also, in order to minimize the contention among the `mdtest` processes, we assign a private directory per process. Our working set consists of 500,000 files, which is sufficient to obtain reproducible results. Using more files causes performance drops because the directory grows too large; the scaling curves still remain the same, however. We use the Lustre 2.4 RPMs as provided in [21]. To the best of our knowledge, no work has been done in newer versions that would significantly alter our results and they should therefore still represent Lustre's current state regarding metadata scalability concerning multi-core and multi-socket platforms.

To identify inefficiencies caused by the Linux kernel or the back-end file system we also evaluate local file systems. We choose ext4 (that is the base of ldiskfs) and XFS. Although ZFS is also supported as back-end file system for Lustre, we did not consider it in our evaluation since previous studies revealed serious performance limitations in metadata operations [22]. Since metadata performance in local file systems is much higher than Lustre's, we increase the file set to 3,000,000 to obtain reproducible results.

Our evaluation consists of four different configurations: First, we present results mounting the Lustre file system in multiple points. After that, using the best parameter for the number of mount points, we evaluate the performance of Lustre's MDS while increasing the number of active cores. Next, we perform experiments binding the `mdtest` processes to the sockets. Finally, we explore the impact of the underlying MDT storage device.

Although Lustre is a parallel distributed file system, we run all the Lustre components on the same server for several reasons. First, the OSS is idle since we do not execute any data I/O. Also, preliminary results verified that the performance of the `create` and `unlink` operations is not influenced by the OSS. Moreover, this has the advantage of eliminating network delays between Lustre's clients and MDS, taking into account that this can lead to a higher contention between them.

Moreover, in order to ensure that the I/O activity generated during the experiments is initiated only from the benchmark, we format the MDT device without *lazy journal initialization*. In our experiments, we use an SSD as the OST and the `cfq` device scheduler for the MDT. In the cases of ext4 and XFS, we also use an SSD. We also collect CPU utilization and storage device statistics from the appropriate `proc` files during the experiment runs. Additionally, we employ `oprofile` [23] to identify the modules consuming the most CPU cycles. We were not able to collect lock statistics using the `CONFIG_LOCK_STAT` kernel option because the resulting kernel crashed during the experiments.

1) Use of multiple mount points: Figure 4 shows the results for `create` and `unlink` operations per second while mounting the file system using multiple mount points (MPs). We run 24 `mdtest` processes on 12 cores (one socket). The `mdtest` processes are equally distributed and access the file system through different mount points. We observe that we can gain up to 4× more operations per second when 6 MPs are used. As can be seen, the CPU utilization follows the same trend as the performance curve. The MDT storage device utilization starts from an average of less than 5 %, reaching up to 10 %, for the `create` operation. For the `unlink` operation it increases from 10 % to 20 %.

	`create`		`unlink`		`create`		`unlink`	
Module	**1 M**	**12 M**	**1 M**	**12 M**	**12 C**	**24 C**	**12 C**	**24 C**
`ptl-rcp`	2.38	11.38	3.04	11.67	<0.1	<0.1	<0.1	<0.1
`obdclass`	1.56	7.53	3.03	12.86	6.38	3.58	12.96	8.57
`mdtest`	<0.1	<0.1	1.46	3.66	4.22	3.59	<0.1	<0.1
`ldiskfs`	1.30	6.53	0.62	3.12	5.30	2.26	3.01	1.70
`lnet`	0.82	3.75	<0.1	<0.1	<0.1	<0.1	3.24	2.17
`libcfs`	0.68	30.6	<0.1	<0.1	2.61	1.28	3.30	2.04
`osd_ldiskfs`	0.47	2.61	0.43	2.10	2.16	0.86	2.06	1.11
`jbd2`	0.45	2.15	0.38	1.26	1.76	0.97	1.23	0.78
`mdt`	0.36	2.11	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1
`lvfs`	0.30	1.74	0.44	1.97	1.41	0.98	1.93	1.60
`lustre`	0.29	1.70	0.42	1.84	1.32	0.56	1.81	0.88
`lod`	0.18	1.06	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1
`lov`	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1	1.10	0.55
`mdd`	0.18	1.03	<0.1	<0.1	1.65	0.30	0.9	0.42
`mdc`	0.13	0.71	<0.1	<0.1	<0.1	<0.1	<0.1	<0.1
`mdt`	<0.1	<0.1	0.42	1.87	1.65	0.73	1.84	0.97

Table I: For the Lustre experiments, CPU utilization (in %) of Linux modules during `create` and `unlink` operations obtained with the `oprofile` tool. The first and last four columns stand for the configurations with different numbers of mount points and active cores, respectively.

We execute the same experiment using the ext4 and XFS file systems (Figure 5 reports the results). We see that with ext4 the performance also increases as we add more MPs, whereas for XFS it remains the same regardless of the number of MPs. Thus we can conclude that this behaviour of Lustre is inherited by ext4. Apart from the results presented in Figures 4 and 5 we have also measured the impact of the number of MPs when more cores are used. Our observation is that even with more cores the metadata performance of Lustre and ext4 does not increase when more than 12 mount points are used.

Exclusively for the Lustre case, the modules' usage for the configurations with 1 MP and 12 MPs is presented in the first four columns of Table I. We observe that the usage of `ptl-rcp`, `obdclass`, `ldiskfs`, `osd_ldiskfs`, `lvfs`, `lustre`, `mdd` and `mdc` is increased by approximately a factor of 5× from 1 MP to 12 MPs.

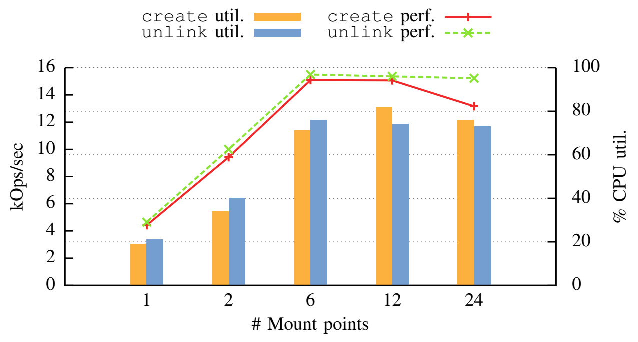

Figure 4: Lustre performance of the `create` and `unlink` operations with respect to the number of mount points and CPU utilization during the experiments run.

2) Scaling with the number of cores: Figure 1 presents the performance scalability of the `create` and `unlink` operations with respect to the number of active cores. We restrict the amount of active cores with the kernel's boot parameter `maxcpus`. In this configuration, the number of mount points is constant for all cases and is set to 6; the number of `mdtest` processes is double the number of cores. Using as many `mdtest` processes as the number of cores produces the same results. As can be seen, Lustre's metadata performance does not improve when more than one socket is

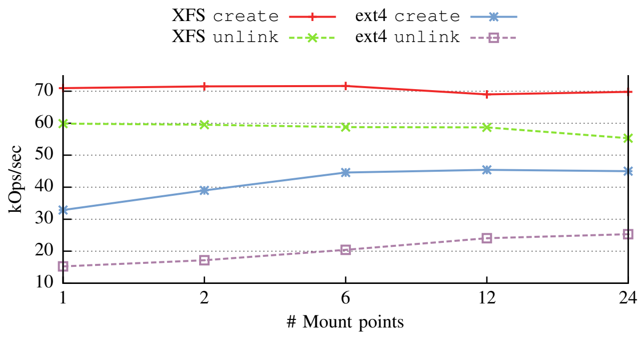

Figure 5: XFS and ext4 file systems' performance of the `create` and `unlink` operations with respect to the number of mount points.

used. The CPU utilization of the Linux modules presented in the last four columns of Table I reveals that for all modules there is a drop by a factor of 2 when going from one to two sockets. Experiments with ext4 and XFS using the same parameters (results reported in Figure 7) reveal that their performance also decreases as we add more cores. Thus we can suspect that the Linux kernel is the limiting factor.

To gain more insights into this behaviour, we include a configuration in which we bind the `mdtest` processes to the sockets (this is not possible for the Lustre threads that run inside the Linux kernel without source modification). We retain the total number of `mdtest` processes and mount points and distribute them equally across the sockets. Keeping the same workload while increasing the number of cores will decrease the potential resource contention. From the results illustrated in Figure 6, we observe that the performance of the `create` operation drops, whereas the `unlink` performance improves. Moreover, the CPU utilization decreases when more sockets are used for both operations. This is due to the increase of processing power for the same workload. The MDT storage device utilization increases for the `unlink` operation (from 21 % to 25 %) and decreases for the `create` operation (from 10 % to 8 %) when more cores are used. One reason that justifies the performance improvement of the `unlink` operation is the use of more kernel threads running in parallel, thus avoiding the competition with `mdtest` processes.

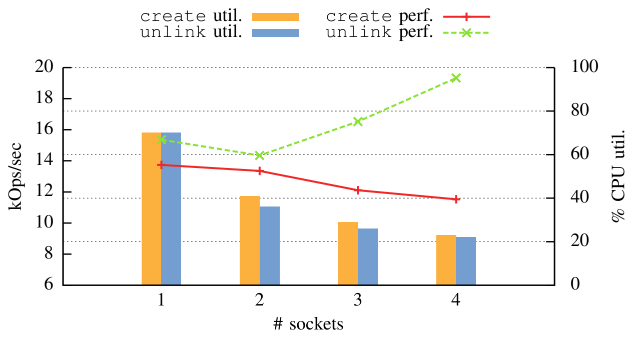

Figure 6: Lustre performance of the `create` and `unlink` operations when `mdtest` processes are bound to the sockets and CPU utilization during the experiment run.

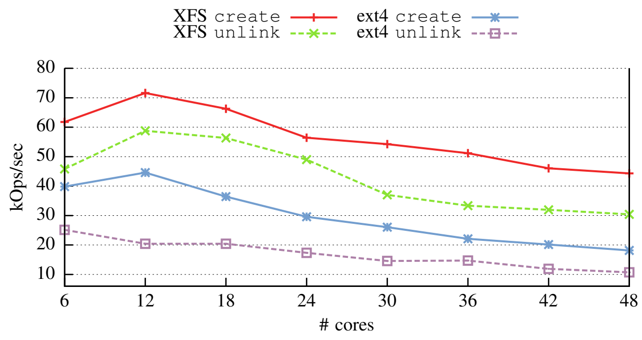

Figure 7: XFS and ext4 file systems' performance of the `create` and `unlink` operations when `mdtest` are we increase the number of cores.

3) Back-end device impact: In order to estimate the practical maximum performance of Lustre's MDS, we run the previous configuration using a RAM disk as MDT storage device. For this purpose, we leverage the RAM disk provided by the Linux kernel. For all cases, we use 6 mount points (which delivers the best performance when using SSDs), and vary the number of active cores. Similar to the previous experiments, the number of `mdtest` processes is twice the number of active cores. Figure 8 displays the operations per second in comparison to the SSDs. As can be seen, the RAM disk delivers a higher performance for both operations. This is due to the elimination of `iowait` time when this device is used. However, there is not a significant difference between the RAM disk and the SSD configurations. From the results we conclude that the MDT storage device is not the limiting factor for the scalability of Lustre's MDS when more than one socket is used.

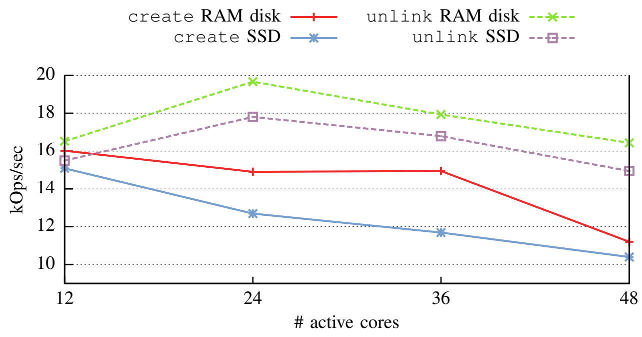

Figure 8: Lustre performance of the `create` and `unlink` operations using different back-end devices.

IV. Related Work

Performance evaluation and optimizations of metadata operations in parallel file systems have been attracting interest from the storage research community. Alam et al. have evaluated the performance of metadata operations in Lustre and GPFS [24], [25]. In their study they have measured the implications of the network overhead on the file systems' scalability. Additionally, they have compared metadata performance when using HDDs and SSDs as storage devices. Shipman et al. have proposed several optimizations that can be used to improve the performance of metadata operations in Lustre [26]. Several improvements in the metadata performance of PVFS [27] are presented in [28].

Furthermore, a substantial amount of research is dealing with multi-core and shared-memory NUMA architectures. The importance of proper locking is discussed in [29] which covers concurrency from the early stage of computers until today's real-world experience. Wait-free synchronization [30] can be used to exploit concurrency. A suite of microbenchmarks to measure scalability to multi-cores is presented in [31]. In the Linux kernel Read-Copy-Update [32] and NUMA support [33] have been added to improve performance scalability in multi-core and NUMA systems. Boyd-Wickizer et al. have performed an analysis of the Linux kernel while running on a 48-core server in [1]. They have used seven applications with different characteristics to perform their evaluation. However, they have not included the full I/O path since they have stored the data in `tmpfs`. Boyd-Wickizer et. al in [2] also report bottlenecks in the Linux kernel.

V. Conclusions

We have analyzed Lustre's metadata performance on a multi-socket NUMA machine with different configurations regarding the number of cores, sockets and mount points. There are still several problems remaining that prevent Lustre from reaching optimal performance. In detail, the main findings of our evaluations are: *i)* Lustre's metadata performance does not scale when increasing the number of sockets and cores due to inefficiencies inside the Linux kernel, *ii)* the MDS's back-end device is not the limiting factor regarding performance, and *iii)* Lustre's parallelism can be increased by using multiple mount points due to inefficiencies in its back-end file system ldiskfs.

Using multiple sockets for Lustre's MDS is currently not feasible because performance is actually reduced in some cases: while `unlink` operations only benefit from up to two sockets (15 % increase), the rate of `create` operations is reduced by 33 % when going from one to four sockets. Even when binding processes to sockets explicitly, this does not change significantly: `unlink` performance can be improved by 20 % when using four sockets instead of one, but `create` performance still decreases by 15 % under the same conditions. We have observed similar performance decreases for both ext4 and XFS, indicating that this is a problem within the Linux kernel instead of a file-system-specific inefficiency.

While it is possible to increase the utilization of a single socket using multiple mount points, this solution is also not feasible in its current form because application developers would have to manually make use of these multiple mount points. Additionally, using more mount points also only increases performance up to a total of six mount points, further limiting the usefulness of this approach. Because we could reproduce the same behavior using ext4 but not with XFS, we conclude that this is due to problems in ldiskfs (and ext4) rather than a general VFS scalability problem.

Our results also show that the MDS back-end device is not the limiting factor since replacing the SSD with a RAM disk only increases performance by 10–15 %.

Based on current and previous results [34], it is advisable to procure single socket machines with a moderate amount of fast processors for Lustre's MDS: As shown previously, increasing the processor clock rate by a factor of two also doubles the number of metadata operations. Using more sockets, however, has – at best – only a negligible positive effect on performance.

Acknowledgment

The research leading to these results has been funded by the German Helmholtz Association's LSDMA project and the EU's SCALUS (FP7-PEOPLE-ITN-2008-238808) project. We are also thankful to the anonymous reviewers for their valuable feedback.

References

[1] S. Boyd-Wickizer, A. T. Clements, Y. Mao, A. Pesterev, M. F. Kaashoek, R. Morris, and N. Zeldovich, "An analysis of Linux scalability to many cores," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–8. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924944

[2] S. Boyd-Wickizer, H. Chen, R. Chen, Y. Mao, F. Kaashoek, R. Morris, A. Pesterev, L. Stein, M. Wu, Y. Dai, Y. Zhang, and Z. Zhang, "Corey: An operating system for many cores," in *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 43–57. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855741.1855745

[3] D. Wentzlaff and A. Agarwal, "Factored operating systems (fos): The case for a scalable operating system for multicores," *SIGOPS Oper. Syst. Rev.*, vol. 43, no. 2, pp. 76–85, Apr. 2009. [Online]. Available: http://doi.acm.org/10.1145/1531793.1531805

[4] A. Baumann, P. Barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, and A. Singhania, "The multikernel: A new os architecture for scalable multicore systems," in *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, ser. SOSP '09. New York, NY, USA: ACM, 2009, pp. 29–44. [Online]. Available: http://doi.acm.org/10.1145/1629575.1629579

[5] R. R. et al., "High End Computing Revitalization Task Force (HECRTF), Inter Agency Working Group (HECIWG) File Systems and I/O Research Guidance Workshop HECIWG FSIO," http://institutes.lanl.gov/hec-fsio/docs/HECIWG-FSIO-FY06-Workshop-Document-FINAL6.pdf, 2006.

[6] B. W. et al., "Optimizing a hybrid SSD/HDD HPC storage system based on file size distributions," in *Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on*, May 2013, pp. 1–12.

[7] Xyratex, "ClusterStor 6000 Datasheet," http://www.xyratex.com/sites/default/files/files/field_inline_files/ClusterStor6000_Datasheet.pdf, 04 2014, last accessed: 2014-08.

[8] P. Schwan, "Lustre: Building a file system for 1000-node clusters," in *Proceedings of the 2003 Linux Symposium*, vol. 2003, 2003.

[9] D. Fellinger, "The State of the Lustre® File System and The Lustre Development Ecosystem," http://www.opensfs.org/wp-content/uploads/2013/04/LUG_2013_vFinal.pdf, 04 2013, last accessed: 2013-10.

[10] Lawrence Livermore National Lab, "Native ZFS on Linux," http://zfsonlinux.org, last accessed: 2014-08.

[11] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier, "The new ext4 filesystem: current status and future plans," in *Proceedings of the Linux Symposium*, vol. 2. Citeseer, 2007, pp. 21–33.

[12] F. Wang, S. Oral, G. Shipman, O. Drokin, T. Wang, and I. Huang, "Understanding Lustre filesystem internals," *Oak Ridge National Lab technical report ORNL/TM-2009/117*, 2009.

[13] D. Wang, "Distributed Name spacE phase I," http://cdn.opensfs.org/wp-content/uploads/2013/04/LUG-2013_DNE.pdf, last accessed: 2014-08.

[14] R. Henwood, "MDS SMP Node Affinity Solution Architecture," https://wiki.hpdd.intel.com/display/PUB/MDS+SMP+Node+Affinity+Solution+Architecture, last modified on May 11, 2012.

[15] Super Micro Computer, Inc., "Supermicro, H8QGi-F Motherboard," http://www.supermicro.com.

[16] Advanced Micro Devices, Inc. , "Introduction to "Magny-Cours"," http://developer.amd.com/resources/documentation-articles/articles-whitepapers/introduction-to-magny-cours/.

[17] JEDEC, "DDR3 sdram standard," http://www.jedec.org/standards-documents/docs/jesd-79-3d.

[18] Advanced Micro Devices, Inc., "AMD hypertransport^{TM} technology-based system architecture."

[19] R. Schöne, D. Hackenberg, and D. Molka, "Memory performance at reduced CPU clock speeds: an analysis of current x86 64 processors," in *Proceedings of the 2012 USENIX conference on Power-Aware Computing and Systems*. USENIX Association, 2012, pp. 9–9.

[20] LLNL, "mdtest, LLNL Metadata Benchmark," http://sourceforge.net/projects/mdtest.

[21] "Lustre RPMS," https://downloads.hpdd.intel.com/public/lustre/lustre-2.4.0/.

[22] K. Chasapis, M. Kuhn, and T. Ludwig, "An Analysis of Lustre Metadata Server Scalability," 06 2013. [Online]. Available: http://www.isc-events.com/isc13_ap/presentationdetails.php?t=contribution&o=2119&a=select&ra=sessiondetails

[23] OProfile, "OProfile," http://oprofile.sourceforge.net/news/, last accessed: 2014-08.

[24] F. Schmuck and R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," in *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, ser. FAST '02. Berkeley, CA, USA: USENIX Association, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=1083323.1083349

[25] S. R. Alam, H. N. El-Harake, K. Howard, N. Stringfellow, and F. Verzelloni, "Parallel I/O and the metadata wall," in *Proceedings of the sixth workshop on Parallel Data Storage*, ser. PDSW '11. New York, NY, USA: ACM, 2011, pp. 13–18. [Online]. Available: http://doi.acm.org/10.1145/2159352.2159356

[26] D. A. Dillow, D. Fuller, F. Wang, H. S. Oral, Z. Zhang, J. J. Hill, and G. M. Shipman, "Lessons Learned in Deploying the World s Largest Scale Lustre File System," Oak Ridge National Laboratory (ORNL); Center for Computational Sciences, Tech. Rep., 2010.

[27] I. F. Haddad, "PVFS: A Parallel Virtual File System for Linux Clusters," *Linux J.*, vol. 2000, no. 80es, Nov. 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=364352.364654

[28] M. Kuhn, J. Kunkel, and T. Ludwig, "Directory-Based Metadata Optimizations for Small Files in PVFS," in *Euro-Par '08: Proceedings of the 14th international Euro-Par conference on Parallel Processing*, University of Las Palmas de Gran Canaria. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 90–99.

[29] B. Cantrill and J. Bonwick, "Real-world concurrency," *Queue*, vol. 6, no. 5, pp. 16–25, Sep. 2008. [Online]. Available: http://doi.acm.org/10.1145/1454456.1454462

[30] M. Herlihy, "Wait-free synchronization," *ACM Trans. Program. Lang. Syst.*, vol. 13, no. 1, pp. 124–149, Jan. 1991. [Online]. Available: http://doi.acm.org/10.1145/114005.102808

[31] Y. Cui, Y. Chen, and Y. Shi, "Osmark: A benchmark suite for understanding parallel scalability of operating systems on large scale multi-cores," *Computer Science and Information Technology, International Conference on*, vol. 0, pp. 313–317, 2009.

[32] P.E. McKenney, et al., "Read-copy update," in *Proceedings of the Linux Symposium 2002*, 2002, p. 338–367.

[33] A. Kleen, "A NUMA API for LINUX," http://developer.amd.com/wordpress/media/2012/10/LibNUMA-WP-fv1.pdf, last accessed: 2014-08.

[34] K. Chasapis, M. Kuhn, and T. Ludwig, "Performance Implications of NUMA and Multi-Core in Lustre's Metadata Server," 06 2014. [Online]. Available: http://www.isc-events.com/isc14_ap/presentationdetails.htm?t=presentation&o=260&a=select