Metadata Workloads for Testing Big Storage Systems

Cristina L. Abad* Huong Luu Yi Lu Roy H. Campbell
University of Illinois at Urbana-Champaign

Abstract

Efficient namespace metadata management is becoming more important as next-generation file systems are designed for the peta and exascale era. A number of new metadata management schemes have been proposed. However, evaluation of these designs has been insufficient, mainly due to a lack of appropriate namespace metadata traces. Specifically, no Big Data storage system metadata trace is publicly available, and existing traces are a poor replacement. We studied publicly available traces and one Big Data trace from Yahoo! and note some of the differences and their implications to metadata management studies. We discuss the insufficiency of existing evaluation approaches and present a first step towards a statistical metadata workload model that can capture the relevant characteristics of a workload and is suitable for synthetic workload generation.

1 Introduction

Large-scale file systems in the Internet services and the high-performance computing (HPC) communities already handle big data: Facebook has 21PB in 200M objects and Jaguar ORNL has 5PB (see Table 1 for other examples). Stored data is growing so fast that exascale storage systems are expected by 2018-2020, by which point there should be over five hundred 10PB deployments [15].

Table 1: Some deployments of petascale storage.

Deployment	Description
Yahoo! (Internet services)	15PB in 4000+ nodes (HDFS); 150 million of objects (files + blocks) [27]
Facebook (Internet services)	21PB in 2000 nodes (HDFS), 200 million of objects (files + blocks) [27]
eBay (Internet services)	16PB in 700-nodes (HDFS) [27]
Jaguar ORNL (HPC)	5PB of storage; Lustre file system [18]
Jugene FZJ (HPC)	4.2PB of storage; GPFS file system [19]
Columbia NASA (HPC)	1PB of storage; CXFS file system [11]

Large-scale storage systems typically implement a separation between data and metadata in order to maintain high performance [2]. With this approach, the management of the metadata is handled by one or more namespace or metadata servers (MDSs), which implement the file system semantics; clients interact with the MDSs to obtain access capabilities and location information for the data [31]. Larger I/O bandwidth can be achieved by adding storage nodes to the cluster, but improvements in metadata performance cannot be achieved simply by deploying more MDSs, as the defining performance characteristic for serving metadata is not bandwidth but rather latency and number of concurrent operations that the MDSs can handle [2, 13]. For this reason, novel and improved distributed metadata management mechanisms are currently being proposed [25, 26, 31]. However, realistic validation and evaluation of these designs is not possible as no adequate traces or workload models are available (see Section 2).

In this position paper, *we argue for the need of Big Data traces and workload models to enable advances in the state-of-the-art in metadata management*. Specifically, we consider the case of namespace metadata traces. We define a **namespace metadata trace** as a storage system trace that contains both a snapshot of the namespace (file and directory hierarchy) as well as a set of events that operate atop that namespace (e.g., open a file, list directory contents, create a file). I/O operations need not be listed, making the trace significantly smaller. Namespace metadata traces can be used to evaluate namespace management systems, including their load balancing, partitioning, and caching components.

Publicly available traces do not meet our definition since they do not contain a snapshot of the namespace. Due to the heavy-tailed nature observed in Big Data workloads, this means that the I/O trace-induced namespace will not contain a huge portion of the namespace (i.e., that portion that was not accessed during the I/O trace[1]). More importantly, existing traces are not representative of Big Data workloads and are frequently scaled up through ad-hoc poorly documented mechanisms to compensate for the fact that they were taken from systems that are orders of magnitude smaller. These limitations are discussed in Section 2.

In Section 3, we present a description of a new (work-in-progress) statistical model for namespace metadata workloads. Preliminary results suggest that our model

*Cristina L. Abad is also an intern at Yahoo!.

[1] We use the term I/O trace to refer to any trace of I/O events that includes metadata operations; for example, a trace of I/O system calls.

Table 3: Traces analyzed in this paper.

Trace	# Files	Mean interarrival time (milliseconds)	File AOA (mean and median, in minutes)
Yahoo	150M	1.04	41994, 267
Home02	> 1M	243.80	99941.5, 4682
EECS2	> 1M	27.20	62520.3, 1228

can be used to generate synthetic workloads that mimic the original metadata workload.

Finally, we discuss opportunities for industry and academia and explain how Big Data traces and workload models can benefit other research (Section 4).

2 Limitations of existing approaches

We surveyed the papers published in the last five years that propose novel file system metadata management schemes and identified their evaluation methodology (see Table 2). The approaches fall into one of three categories: (i) microbenchmarks, that test a specific part of the system and do not attempt to represent a realistic workload; (ii) application benchmarks, which provide real or synthetic application-based workloads; and (iii) trace-based approaches, which aim at looking at the full workload of the system. In this paper, we focus on the limitations of approach (iii), detailed in Section 2.1, but for clarity purposes, we provide a brief discussion on issues (i) and (ii), after our methodology discussion below.

Methodology. We support our arguments through an analysis of three traces: a Big Data *namespace metadata trace* from Yahoo! and two traces used in prior work (Home02 and EECS2 in Tables 3 and 4; which are the most recent public traces used in the papers we surveyed). The Yahoo trace was obtained from the largest Hadoop cluster at Yahoo! (4000+ nodes, using the Hadoop Distributed File System, or HDFS, as the storage layer); this is a production cluster running pipelines of data-intensive jobs like processing advertisement targeting information. The *namespace metadata trace* consists of a snapshot of the namespace on April 30th, 2011, obtained with Hadoop's Offline Image Viewer tool, and a one-month metadata trace (May 2011), obtained by parsing the name node (MDS) auditing logs.

Storage system workloads are, by nature, multi-dimensional [9, 30] and can be defined by several characteristics like namespace size and shape, arrival patterns, and temporal or spatial locality patterns. In this paper, we discuss of some of these dimensions where appropriate. One of these dimensions is the temporal locality present in the workload, which we measure through the distribution of the age of a file at the time it is accessed (*AOA*). For every operation (namespace metadata event), we calculate how old the file is and use this information to build a cumulative distribution function (CDF) that represents the workload with respect to this dimension[2]. We chose this dimension because it is one that is very relevant to namespace metadata management schemes, since the temporal locality of the workload has an incidence in mechanisms like load balancing, dynamic namespace partitioning/distribution, and caching.

It should be noted that (in the context of this paper) an *access* to an *object* (file or directory) refers to an access through a *namespace metadata operation*. In this sense, any metadata operation (like getting the attributes of a file) constitutes a metadata access. We make this generalization since we are interested in how the workload stresses the metadata management system. See Section 3 for a discussion on the role of different metadata operations in the workload.

Metadata-intensive microbenchmarks. While many I/O benchmarks exist, a recent survey [29] found that they do not evaluate the metadata dimension in isolation, making their use—without modifications—inadequate for metadata management studies. There are, however, a few benchmarks that look specifically into metadata-intensive workloads. Two of these benchmarks were used in some of the papers we surveyed: mdtest and metarates. Both of these tools have proven useful as microbenchmarks, but they do not attempt to model real workloads. For example, one can easily use mdtest to perform a high-rate of creates (of zero-sized files) in a random namespace hierarchy, but not to do creates that are modeled after real workloads and work atop a realistic namespace. While a random access pattern test is useful as a microbenchmark, it should not be used to evaluate how well the proposed metadata distribution mechanism performs with respect to features like load balancing, *unless* the random generation of events is based on a statistical model that captures the relevant characteristics of the workload (see Section 3).

Application benchmarks. This type of benchmark is convenient, as synthetic benchmarks exist and real non-interactive applications can be used too. Application benchmarks are good for recreating specific workloads; however, the full workload of a system is typically a combination of many different applications and client usage patterns that, as a whole, can differ significantly from the individual application workloads.

2.1 Limitations of existing traces

We analyzed the traces used by the studies cited in this paper (see Table 4), as well as the I/O traces available for download at the Storage Networking Industry As-

[2]Note that, for traces without a namespace snapshot, it is not possible to know the age of files that are not created during the trace of (namespace metadata) events. To enable comparison between traces that have a snapshot and those that don't, when calculating the AOA we ignored those files that were not created during the trace.

Table 2: Evaluation approaches used in prior metadata management papers.

Evaluation mechanism	Paper(s)	Description
Metadata-intensive microbenchmarks		
mdtest	[2, 8, 10, 24, 26, 32]	Multiple processes create/stat/delete files/directories in shared or separate directories.
metarates	[2, 31, 33]	MPI program that coordinates file system accesses from multiple clients.
self-designed	[3, 4, 6, 8, 13, 20, 25]	Non-standard scripts/programs that perform some sequence of namespace operations.
Application benchmark		
Checkpoint	[4]	Each process writes its system states to an individual checkpoint file periodically. Metadata intensive and commonly found in large scale parallel applications.
SSCA	[3, 4]	Pseudo-real applications that closely mimic high-performance computing workloads.
IMAP build	[28]	Metadata operations recorded during a Linux kernel build and IMAP server.
mpiBLAST	[10]	MPI program; searches multiple DB fragments and query segments in parallel with large DB size.
Trace-based		
Simulation	[7, 14, 16, 17, 22, 23]	Simulator designed by authors; traces may be scaled up.
Replay	[16]	Replay of real traces, scaled up to simulate a larger system.

Table 4: Description of the traces used by one or more of the surveyed papers.

Trace name	Year	Source description	Publicly available?	Was it scaled?
Sprite	1991	One month; multiple servers at UC Berkeley.	Yes	Yes[14]
Coda	1991-3	CMU Coda project, 33 hosts running Mach.	Yes	Yes [7]
AUSPEX	1993	NFS activity of 236 clients, during one week in UC Berkeley.	Yes	No[23]
HP	2000	10-day trace; working group using HP-UX time-sharing server; 500 GB.	Yes	Yes [16, 17, 22]
INS (HP)	2000	HP-UX traces of 20 machines in labs for undergraduate classes.	Yes	Yes [17]
RES (HP)	2000	HP-UX; 13 desktop machines; 94.7 million requests, 0.969 million files.	Yes	Yes [17]
Home02 (Harvard)	2001	From campus general-purpose servers; 48GB.	Yes	Yes [22]
EECS (Harvard)	2001	NFS trace; `home` directory of computer science department; 9.5GB.	Yes	Yes [16]

sociation (SNIA) trace repository (`iotta.snia.org`), and identified the following limitations in the context of namespace metadata management studies: (1) no public petascale traces are available, (2) no traces include both a snapshot of the namespace and a trace of operations on that namespace, and (3) no big data traces are available[3].

No public petascale traces. Sub-petascale traces are often scaled up in some way; the modification of the traces, if done through ad-hoc poorly documented mechanisms, makes the results difficult to reproduce and raises questions about the validity of the results. Working at a smaller scale is not always adequate since inefficiencies in design/implementation may only be evident at scale.

No traces include both a namespace snapshot and a trace of operations on that namespace. If no namespace snapshot is included with trace, the researcher must rely on the trace-induced namespace[4] (i.e., that portion of the namespace that is accessed during the trace). The trace-induced namespace can be significantly smaller–due to the very heavy-tailed access patterns in which some files are rarely, if ever, accessed (see Figure 1)–and may have a different form than the full namespace (see Figure 2). In other words, the trace-induced namespace is a bad predictor of the actual namespace, and the omission of a significant portion of the files and directories in an evaluation could affect its results.

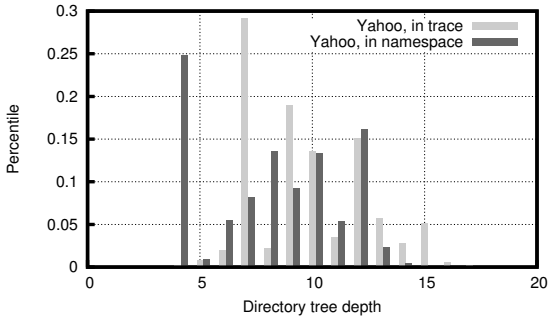
No Big Data traces. Big Data or data-intensive computing is increasingly more popular within the HPC, cloud computing and Internet services communities. Previous studies have noted that these workloads may differ considerably from more traditional workloads. For example, consider the age of a file at the time it is accessed, which is a measure of the temporal locality on a trace based on previous Big Data observations: Figure 3 shows the difference in the workloads between two traditional traces used in previous studies and a Big Data trace. In the Yahoo trace, most of the access of a file occur within a small window of time after the file is created; this is typical of MapReduce pipelines. In contrast, in the traces Home02 and EECS2 files remain popular for a longer period of time.

2.2 Lack of metadata workload models

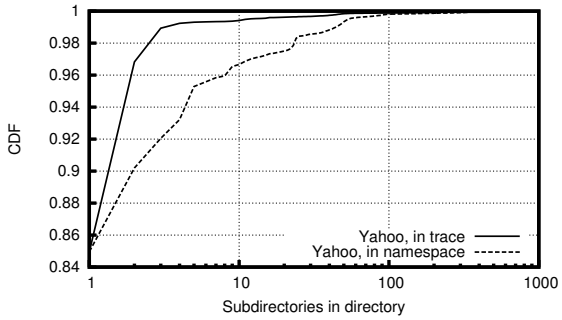
When adequate traces are not available, workload models can be useful by enabling researchers to generate synthetic traces or modify existing traces in a mean-

[3] Application traces and workloads exist, such as Hadoop's Gridmix for MapReduce workloads and the simulation traces from Sandia National Labs. While these traces are useful, they do not represent the full load observed by the *storage* system.

[4] Or, create a namespace from a model [1], if available.



(a) Percentiles of files at each depth in hierarchy; a high percentage of files are stored at depth 4 but rarely accessed.



(b) In trace-induced namespace, non-leaf subdirectories appear to have less children than in snapshot.

Figure 2: Differences in namespace hierarchy shape between snapshot and trace of namespace events; trace-induced namespace is a bad predictor of real namespace.

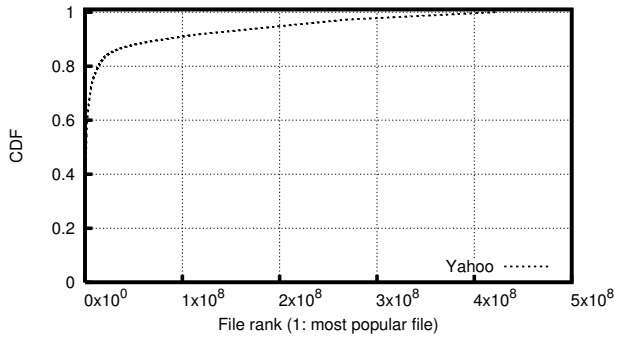


Figure 1: Cumulative distribution function (CDF) of the popularity of files in Yahoo trace, for those that were accessed at least once during trace. We observe that data access patterns are heavy-tailed: 3.28% of files receive 80% of accesses, with a long tail of rarely accessed files.

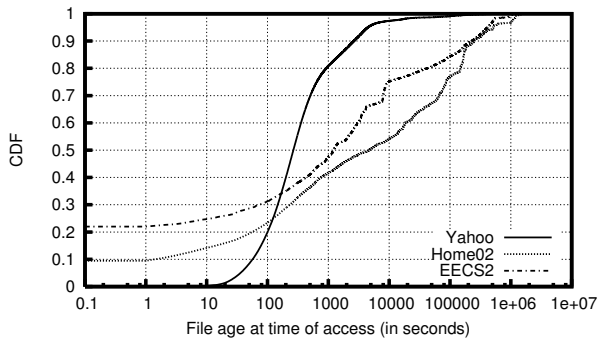


Figure 3: File age at time of access; we observe a significant difference in temporal locality between workloads.

Table 5: Statistical parameters captured by our model.

Namespace characterization
Number of directories *and* number of files
Distribution of files at each depth in namespace hierarchy
Distribution of directories at each depth in namespace hierarchy
Distribution of number of files per directory
Distribution of number of subdirectories per directory
Distribution of file sizes (in MB)
Workload characterization
Percentiles of operation type in trace
Interarrival rate distribution
Percentiles of operations observed at each depth in namespace
Distribution of files per depth in namespace, as observed in trace
Distribution of dirs. per depth in namespace, as observed in trace
Distribution of number of files per directory, as observed in trace
Distribution of number of subdirs. per dir., as observed in trace
Distribution of file age at time of access
Distribution of file age at deletion (i.e., file life span)

ingful way. While many models have been proposed to describe certain storage system properties (e.g., directory structure in namespace snapshots [1]), to the best of our knowledge no work has been published that proposes a model that combines a namespace structure and a (metadata) workload on that trace. This lack of adequate models make it hard for researchers to design their own synthetic workloads or to modify existing traces to better fit access patterns of large scale storage systems.

3 Towards a metadata workload model

We are currently working on a statistical workload model for namespace metadata traces that *captures the relevant statistical characteristics of the workload and is suitable for synthetic workload generation that mimics the original trace with respect to these characteristics*. Our model consists of a set of parameters that together defines the namespace (modeled after the original snapshot) and the workload of metadata operations (modeled after a trace of metadata events); our current implementation contains the parameters listed in Table 5. We plan to

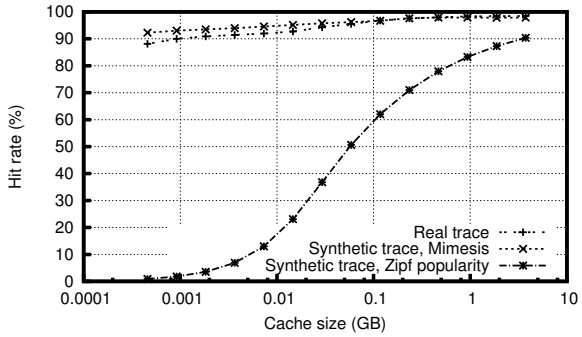


Figure 4: Hit rate of a LRU metadata cache at MDS, using three traces: the Yahoo trace, a synthetic one created with our tool/model, and a naive approach using a Zipfian distribution of file popularities modeled using the CDF in (Figure 1). When temporal locality is not properly modeled (Zipfian), evaluation is sub-optimal.

add other important workload-defining parameters, specially regarding spatial locality.

We have a work-in-progress implementation, called Mimesis, that contains a format in which the model can be stored, processed and shared. We have also implemented: (1) a tool that takes a snapshot of a namespace and a trace of events atop that namespace, and generates a configuration file with these parameters, and (2) a tool that takes the configuration file with the model and generates a synthetic metadata trace that reproduces the statistical characteristics described in the model.

While our preliminary results from a small case study on evaluating a least-recently-used metadata cache for the HDFS MDS are promising (Figure 4), we are working on performing a more complete validation of our model (including comparison with scaled up EECS2 and Home02 traces; set of simulations was not completed before this submission); we plan to make our model extensible, to facilitate adding new statistical parameters.

4 Discussion

Release of petascale traces by industry would open many research opportunities in petascale and exascale systems. The first step to disclosing namespace metadata traces is obtaining those traces. For some systems, like the Hadoop Distributed File System, this may be simple since metadata accesses can be logged for auditing purposes and namespace snapshots can be obtained using existing tools. For other systems, unobtrusive mechanisms to obtain these traces may not be available, and should thus be implemented before the traces can be recorded. This can be done via network snooping, system call interception at the clients, logging mechanisms at the clients, or logging mechanisms at the MDSs. The latter has shown, in practice, to be a viable alternative when the logging is properly implemented (e.g., the name node audit logs in HDFS).

Once obtaining the traces is possible for a specific file system, industry can (a) publicly release anonymized traces, (b) release anonymized traces to academic institutions after signing non-disclosure agreements, or (c) model the workloads of their traces and release these models. The latter has the advantage of being being less burdensome to transfer via a network.

To enable (c), researchers should come up with expressive metadata workload models and tools to process the traces and obtain the models. Workload generators or compilers can be built to take the models as an input and generate realistic synthetic workloads or configuration files in the languages of existing benchmarking tools. While synthetic workloads will, by definition, differ from the original ones, it has been argued that they can be useful if they maintain the characteristics of the original workload that the researcher is interested in and, when used in evaluations, lead to results within some small margin of those that would be obtained with the original workload [30]. Selecting those features that make a workload relevant [9] is crucial to this process. Our model and tools are a first step towards this goal.

In this paper we have focused on distributed metadata management research because we consider it to be an important problem for next-generation file systems and one that is extremely sensitive to the namespace metadata workloads. However, other types of research that need information about realistic namespaces and/or realistic workloads or data access patterns in Big Data systems could benefit significantly from access to these traces and models: job scheduling mechanisms that aim to increase data locality [34], dynamic replication [12], search in large namespaces [21], schemes that want to treat popular data differently than unpopular data [5], among others.

The storage community has long acknowledged the need for workload models, and I/O benchmarking tools typically come prepackaged with typical workloads or personalities, including those of databases and Web servers. We believe Big Data workloads should be added to the list of important workloads to be modeled and studied. We plan on releasing the statistical parameters (described by our model) obtained from three large clusters at Yahoo!, as well as the tools used to process the traces and generate synthetic workloads.

Acknowledgments

We thank Nathan Roberts and Kihwal Lee from Yahoo! for their helfpul insight, and Prof. Marianne Winslett for her help in improving this paper. This paper is based on research sponsored by the Air Force Research Laboratory and the Air Force Office of Scientific Re-

search, under agreement FA8750-11-2-0084. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

[1] AGRAWAL, N., ARPACI-DUSSEAU, A., AND ARPACI-DUSSEAU, R. Generating realistic Impressions for file-system benchmarking. *ACM Transactions on Storage (TOS) 5* (2009).

[2] ALAM, S. R., EL-HARAKE, H. N., HOWARD, K., STRINGFELLOW, N., AND VERZELLONI, F. Parallel I/O and the metadata wall. In *Proc. Petascale Data Storage Wrkshp. (PDSW)* (2011).

[3] ALI, N., DEVULAPALLI, A., DALESSANDRO, D., WYCKOFF, P., AND SADAYAPPAN, P. An OSD-based approach to managing directory operations in parallel file systems. In *Proc. Intl. Conf. Cluster Comp. (CLUSTER)* (2008).

[4] ALI, N., DEVULAPALLI, A., DALESSANDRO, D., WYCKOFF, P., AND SADAYAPPAN, P. Revisiting the metadata architecture of parallel file systems. In *Proc. Petascale Data Storage Wrkshp. (PDSW)* (2008).

[5] ANANTHARANAYANAN, G., GHODSI, A., WANG, A., BORTHAKUR, D., KANDULA, S., SHENKER, S., AND STOICA, I. PACMan: Coordinated memory caching for parallel jobs. In *Proc. USENIX Symp. Networked Sys. Design and Impl. (NSDI)* (2012).

[6] BUI, H., BUI, P., FLYNN, P., AND THAIN, D. Roars: A scalable repository for data intensive scientific computing. In *Proc. ACM Intl. Symp. High Perf. Distrib. Comp. (HPDC)* (2010).

[7] CAI, B., XIE, C., AND ZHU, G. Performance evaluation of a load self-balancing method for heterogeneous metadata server cluster using trace-driven and synthetic workload simulation. In *Proc. IEEE Intl. Parallel and Distrib. Proc. Symp. (IPDPS)* (2007).

[8] CARNS, P., LANG, S., ROSS, R., VILAYANNUR, M., KUNKEL, J., AND LUDWIG, T. Small-file access in parallel file systems. In *Proc. IEEE Intl. Parallel and Distrib. Proc. Symp. (IPDPS)* (2009).

[9] CHEN, Y., SRINIVASAN, K., GOODSON, G., AND KATZ, R. Design implications for enterprise storage systems via multi-dimensional trace analysis. In *Proc. ACM Symp. Operating Sys. Principles (SOSP)* (2011).

[10] CHEN, Z., XIONG, J., AND MENG, D. Replication-based highly available metadata management for cluster file systems. In *Proc. Intl. Conf. Cluster Comp. (CLUSTER)* (2010).

[11] Columbia hardware overview, 2012. Available at: `http://www.nas.nasa.gov/hecc/support/kb/entry/Columbia-Hardware-Overview_82.html`.

[12] CRISTINA ABAD, YI LU, R. C. DARE: Adaptive data replication for efficient cluster scheduling. In *Proc. Intl. Conf. Cluster Comp. (CLUSTER)* (2011).

[13] DEVULAPALLI, A., AND OHIO, P. File creation strategies in a distributed metadata file system. In *Proc. IEEE Intl. Parallel and Distrib. Proc. Symp. (IPDPS)* (2007).

[14] FU, Y., XIAO, N., AND ZHOU, E. A novel dynamic metadata management scheme for large distributed storage systems. In *Proc. Intl. Conf. High Perf. Comp. and Comm. (HPCC)* (2008).

[15] GEIST, A. Paving the road to exascale. *SciDAC Review Special Issue*, 16 (2010).

[16] HUA, Y., JIANG, H., ZHU, Y., FENG, D., AND TIAN, L. SmartStore: A new metadata organization paradigm with semantic-awareness for next-generation file systems. In *Proc. ACM/IEEE Conf. Supercomp. (SC)* (2009).

[17] HUA, Y., ZHU, Y., JIANG, H., FENG, D., AND TIAN, L. Scalable and adaptive metadata management in ultra large-scale file systems. In *Proc. Intl. Conf. Distrib. Comp. Sys. (ICDCS)* (2008).

[18] Jaguar overview. Available at: `http://www.olcf.ornl.gov/computing-resources/jaguar`.

[19] Forschungszentrum Jülich reveals new insights into the human condition with supercomputing solution based on IBM Blue Gene, 2011. Available at: `http://www-01.ibm.com/software/success/cssdb.nsf/CS/STRD-8JBDBN`.

[20] KUHN, M., KUNKEL, J., AND LUDWIG, T. Directory-based metadata optimizations for small files in pvfs. In *Proc. Intl. European Conf. Parallel and Distrib. Comp. (Euro-Par)* (2008).

[21] LEUNG, A. W., SHAO, M., BISSON, T., PASUPATHY, S., AND MILLER, E. L. Spyglass: Fast, scalable metadata search for large-scale storage systems. In *Proc. USENIX Conf. File and Storage Tech. (FAST)* (2009).

[22] LIN, L., LI, X., JIANG, H., ZHU, Y., AND TIAN, L. AMP: An affinity-based metadata prefetching scheme in large-scale distributed storage systems. In *Proc. IEEE Intl. Symp. Cluster Comp. and the Grid (CCGRID)* (2008).

[23] LIN, W., WEI, Q., AND VEERAVALLI, B. Wpar: A weight-based metadata management strategy for petabyte-scale object storage systems. In *Proc. IEEE Intl. Wrkshp. Storage Network Architecture and Parallel I/Os (SNAPI)* (2007).

[24] MESHRAM, V., BESSERON, X., OUYANG, X., RAJACHANDRASEKAR, R., DARBHA, R. P., AND PANDA, D. K. Can a decentralized metadata service layer benefit parallel filesystems? In *Proc. Intl. Conf. Cluster Comp. (CLUSTER)* (2011).

[25] NICOLAE, B., ANTONIU, G., AND BOUGÉ, L. Enabling high data throughput in desktop grids through decentralized data and metadata management: The blobseer approach. In *Proc. Intl. European Conf. Parallel and Distrib. Comp. (Euro-Par)*.

[26] PATIL, S., AND GIBSON, G. Scale and concurrency of GIGA+: File system directories with millions of files. In *Proc. USENIX Conf. File and Storage Tech. (FAST)* (2011).

[27] SHVACHKO, K. V. Apache Hadoop: The scalability update. *USENIX ;login 36*, 3 (2011).

[28] STENDER, J., KOLBECK, B., HÖGQVIST, M., AND HUPFELD, F. BabuDB: Fast and efficient file system metadata storage. In *Proc. IEEE Intl. Wrkshp. Storage Network Architecture and Parallel I/Os (SNAPI)* (2010).

[29] TARASOV, V., BHANAGE, S., ZADOK, E., AND SELTZER, M. Benchmarking file system benchmarking. In *Proc. HotOS* (2011).

[30] TARASOV, V., KUMAR, K., MA, J., HILDEBRAND, D., POVZNER, A., KUENNING, G., AND ZADOK, E. Extracting flexible, replayable models from large block traces. In *FAST12*.

[31] WELCH, B., UNANGST, M., ABBASI, Z., GIBSON, G., MUELLER, B., SMALL, J., ZELENKA, J., AND ZHOU, B. Scalable performance of the panasas parallel file system. In *FAST08*.

[32] XING, J., XIONG, J., SUN, N., AND MA, J. Adaptive and scalable metadata management to support a trillion files. In *Proc. ACM/IEEE Conf. Supercomp. (SC)* (2009).

[33] YANG, S., LIGON, W. B., AND QUARLES, E. C. Scalable distributed directory implementation on orange file system. In *Proc. IEEE Intl. Wrkshp. Storage Network Architecture and Parallel I/Os (SNAPI)* (2011).

[34] ZAHARIA, M., BORTHAKUR, D., SEN SARMA, J., ELMELEEGY, K., SHENKER, S., AND STOICA, I. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In *Proc. European Conf. Comp. Sys. (EuroSys)* (2010).