Towards Realistic Benchmarking for Cloud File Systems: Early Experiences

Zujie Ren[1], Weisong Shi[2] and Jian Wan[1]
[1]School of Computer Science
Hangzhou Dianzi University
{renzj,wanjian}@hdu.edu.cn
[2]Department of Computer Science
Wayne State University
weisong@wayne.edu

***Abstract*—Over the past few years, cloud file systems such as Google File System (GFS) and Hadoop Distributed File System (HDFS) have received a lot of research efforts to optimize their designs and implementations. A common issue for these efforts is performance benchmarking. Unfortunately, many system researchers and engineers face challenges on making a benchmark that reflects real-life workload cases, due to the complexity of cloud file systems and vagueness of I/O workload characteristics. They could easily make incorrect assumptions about their systems and workloads, leading to the benchmark results differing from the fact.**

As the preliminary step for designing a realistic benchmark, we make an effort to explore the characteristics of data and I/O workload in a production environment. We collected a two-week I/O workload trace from a 2,500-node production cluster, which is one of the largest cloud platforms in Asia. This cloud platform provides two public cloud services: data storage service (DSS) and data processing service (DPS). We analyze the commonalities and individualities between both cloud services in multiple perspectives, including the request arrival pattern, request size, data population and so on. Eight key observations are highlighted from the comprehensive study, including the arrival rate of requests follows a Lognormal distribution rather than a Poisson distribution, request arrival presents multiple periodicities, cloud file systems fit partly-open model rather than purely open model or closed model. Based on the comparative analysis results, we derive several interesting implications on guiding system researchers and engineers to build a realistic benchmark on their own systems. Finally, we discuss several open issues and challenges raised on benchmarking cloud file systems.

I. Introduction

The last decade has witnessed a rapid research progress in the field of cloud computing. Many large-scale cloud platforms such as Amazon EC2 [1] and Microsoft Azure [2], have been implemented and widely used in practice. Cloud file systems, as the fundamental parts of cloud platforms, serve to store petabyte-scale data and handle high-speed I/O requests. Typical examples of cloud file systems in the community include GFS [3] and HDFS [4]. During the past few years, these cloud file systems have attracted plenty of research efforts to improving system throughput [5], availability [6], reliability [7], fault-tolerance [8,9] and so on.

As the cloud file systems mature, the demand to evaluate the performance of these file systems rises. However, the research of benchmarking cloud file systems has not been well studied yet. To date, there are no standard benchmarks available for cloud file systems. Traditional file system benchmarks such as Postmark [10] and SPC [11] are unsuitable for benchmarking cloud file systems, as new kind of workloads are continuously emerging in cloud platforms. The heterogeneity and diversity of workloads, as well as the system complexity, pose extra difficulty for benchmarking cloud file systems.

To design a realistic benchmark for cloud file systems, two key issues need to be addressed as a preliminary step. (1) How to synthesize representative I/O request streams? (2) How to pre-populate the data objects that reflect actual cases in cloud platforms? Although some previous work reported the data characteristics and access pattern in large-scale distributed systems [12,13], the understanding of characteristics of data and workload on the file storage layer of cloud platforms is far from complete. A major reason is the insufficiency of a publicly available workload trace. Therefore, system researchers or engineers could easily make incorrect assumptions about their systems and workloads, leading to inaccurate benchmark results.

In this paper, we address these issues by exploring the characteristics of I/O workloads on a production cloud, called ACloud, which is one of the largest cloud platforms in Asia[1]. The workload being studied in this paper comes from a cluster of ACloud containing over 2,500 nodes and hosting two cloud services for the public: Data Storage Service (DSS) and Data Processing Service (DPS). DSS is available for thousands of cloud applications to store/access various files, such as images, videos, logs, Web pages, and so on. DPS is used to build data warehouses and execute offline data analytical jobs. The workload trace was logged from Oct.12-25, 2013. Over 852 million read/write requests on the ACloud file system are sampled during the two-week period. The key contributions of this paper are listed as follows.

- We collect a two-week I/O workload trace from a 2,500-node production cloud platform. The trace covers over 850 million I/O requests, which are generated by two cloud services, data storage service and data processing service.
- We analyze the characteristics of workload and data in terms of multiple properties, including the request arrival pattern, session behaviors, periodicity pattern, request size, and so on. The main observations and their direct implications from this analysis are presented in Table I. These findings can guide other researchers and engineers to implement a realistic benchmark for their own cloud file systems.

[1]Due to confidentiality reasons, we anonymize the real name of the cloud platform and its underlying file system discussed in this paper.

TABLE I: Summary of observations and implications.

Observations	Implications	Sections
The workload fluctuation of DSS and DPS appears with different patterns. The workload peaks of DSS and DPS emerge at different time intervals.	The arrival rate of requests in a benchmark should vary over time and show a regular daily periodicity, rather than with a static or a random mode. The period where workload peak emerges differs notably between different cloud services.	IV-A
The arrival rate of DSS and DPS requests follows a Lognormal distribution, rather than a Poisson distribution.	For designing a workload generator in cloud benchmarks, it is inappropriate to simply assume that the arrival rate of requests follows a Poisson distribution.	IV-A
During the period of daytime, the file system behaves like a partly-open system under the DSS request stream. While in the early morning, the file system behaves like a closed system.	A substantial portion of I/O requests provided by benchmarks should arrive as a sequence of a session. The assumption that cloud file systems behaves like an open or closed model will be not satisfied.	IV-B
Besides a dominated periodicity of one day, the other three major periodicities (12/8/6 hours) also exist among the request arrival stream.	The I/O requests in cloud file systems show multiple periodicities, rather than a single periodicity. For different cloud services, their periodicities would probably differ notably.	IV-C
The size of write requests in ACloud follows a Zipf distribution, while the size of read requests follows a Lognormal distribution.	For designing a workload generator in cloud benchmarks, the size of write requests should follow a Zipf distribution, while the size of read requests should follow a Lognormal distribution.	IV-D
More than 90% of files are smaller than the default size of a single chunk (64MB).	When pre-populating data for cloud benchmarks, those files smaller than 64MB should be the majority in the testbed.	V-A
File directory depths range from 3 to 20, following a Normal distribution with the mean value of 9.	As directory depth affects the performance of meta-data access, the directory depth distribution of pre-populated data should match the real cases.	V-A
Data locality achieved in cloud file systems is lower than that of other production Hadoop cluster [14,15].	As data locality plays an important role in the performance of I/O scheduling, the data distribution policy adopted by cloud benchmarks should be carefully taken into consideration when pre-populating a testbed.	V-B

- Based on the analysis results and our early experiences on cloud file system benchmarking, we describe several open problems and challenges, including modeling I/O workloads in cloud environments, modeling cloud file systems, and scaling workload against cluster capacity. We believe these issues are worthy of further exploration for advancing benchmark research.

Table I summarizes our observations and implications, and serves as a roadmap for the rest of the paper. Section II reviews some previous work on file system benchmarks. Section III provides a brief introduction of the cloud platform ACloud and its underlying file system, and then gives an overview of the summary of trace and the cluster configuration. A detailed analysis of the I/O traces and data characteristics is presented in Section IV and V. Several open problems in the community of benchmarking research are discussed in Section VI, and Section VII concludes this paper.

II. Related Work

The research topic of file system benchmark is not new, and a myriad of related research work that can shed light on our work exists. In this section, we review some existing research work closest to ours, and illustrate the motivations of our work.

A. Big data systems benchmarks

With the advent of big data techniques, there have been plenty of benchmarks developed to evaluate the performance of big data systems. These benchmarks are mainly designed for two kinds of systems: *transactional processing systems* and *analytical processing systems*.

For example, YCSB (Yahoo! Cloud Serving Benchmark) [16], was proposed to compare the performance of transactional processing systems including Cassandra [17], HBase [18], PNUTS [19], and a simple sharded MySQL implementation. YCSB supplies several workloads with different combinations of `insert, read, update` and `scan` on database tables. Similarly, Shi *et al.* [20] developed two benchmarks with a collection of structured queries, to compare the performance of Cassandra, HBase, Hive and HadoopDB.

Beyond the benchmarks for transactional processing systems, dozens of benchmarks have been proposed for big analytical processing systems, such as CloudSuite [21], BigBench [22], DCBench [23], Hibench [24], GraySort [25], CloudRank-D [26], and several other research work in [27]–[30], and so on. These benchmarks supply a set of analytical jobs or OLAP queries to test the performance to MapReduce-style systems. Chen *et al.* [31] combined experiences from the TPC-C benchmark with emerging insights from MapReduce workloads, and presented a vision for a big data benchmark that contains a mix of application domains. Besides, Palvo *et al.* [32] proposed a general benchmark with a collection of analytical jobs to compare the performance between Hadoop and parallel DBMS. Generally, these benchmarks can be effectively utilized to evaluate the performance of standalone transactional or analytical processing systems hosted in a cloud platform. However, they are unsuitable for benchmarking cloud file systems. Cloud file systems work at the underlying layer of Hadoop and DBMS in the software stack, so the workload of transactional operations or analytical tasks are uncontrollable to synthesize I/O request streams for benchmarking the performance of file systems.

B. File systems workload characterization and benchmarks

A deep understanding of the I/O workload characteristics is crucial for designing file system benchmarks. Traeger *et al.* [33] examined 415 file system benchmarks from over 100 papers spanning nine years and found that in many cases benchmarks do not provide adequate evaluation of file system performance. Many researchers conducted extensive analysis of file system workloads on various data-intensive systems[34]–[39]. The I/O workloads investigated by these studies are mainly generated by traditional cluster computing systems such as high-performance computing or MapReduce systems.

Compared to traditional cluster computing systems, the workload in a cloud platform is much more heterogeneous, complex and dynamic [40,41]. The characteristics of I/O requests in clouds are quite different and complex. Due to the lack of a publicly available trace, file system workload analysis in cloud platforms has not been well studied yet.

Despite some recent studies [12,13,41] that characterized the workloads on cloud storage systems, it is far from enough to implement a realistic benchmark for cloud file systems. The lack of understanding in the I/O workload and data characteristics of cloud file systems has forced researchers and designers to rely on outdated or marginally related information such as enterprise workload studies to model cloud file system behavior.

Compared to traditional file systems, workload characterization on cloud file systems poses special challenges for benchmarking efforts. These challenges arise from the characteristics of cloud file systems, including (1) system complexity, which makes it difficult to develop request processing models. (2) workload heterogeneity and dynamicity, which hamper efforts to identify representative workload behavior, (3) high data volume and large cluster scale, which make it challenging to replay the workload and reproduce system behavior, and (4) rapid system evolution, which requires benchmarks to accommodate changes in the underlying systems. Without real-life empirical knowledge, system researchers and engineers could easily make incorrect assumptions about their own workloads, thus yielding inaccurate performance evaluation results.

III. Overview of System and Trace

In this section, we will present an overview of ACloud, describe the procedures of read and write operations, and give a summary of the workload trace.

A. Overview of ACloud

ACloud hosts two public cloud services: Data Storage Service (DSS) and Data Processing Service (DPS). DSS is similar to Amazon's Simple Storage Service (S3). It exposes a set of Web service API which applications could invoke to read and write data objects. At the end of Oct. 2013, DSS was used by thousands of Web applications to store/access various files, such as images, videos, logs, pages, and so on. DPS is similar to Amazon EMR (Elastic MapReduce) [42] and Microsoft Azure MapReduce [43]. DPS is designed for various companies and organizations to build data warehouses and execute offline data analytical jobs. DPS supports executing a variety of MapReduce jobs, including log analysis, Web indexing, data warehousing, machine learning, and so on. By the end of Oct. 2013, over fifty thousand analytical jobs are conducted on DPS every day.

The underlying cloud file system of ACloud is called *PFS*. PFS adopts a GFS-style architecture to interconnect a large amount of servers. To enhance the availability and performance, the PSF contains multiple metadata servers (called *MasterServers*), which collaboratively direct a large set of data servers (called *ChunkServers*). The MasterServers are responsible for managing the file system namespace and regulating access to files by clients. The consistency of meta-data among the masters is maintained through the Paxos protocol [44]. The ChunkServers are a set of slave nodes responsible for storing data chunks.

B. Applications Classification and Representatives

Various cloud applications are supported by DSS and DPS, including mail systems, Web search engines, log collection systems and so on. Different applications produce diverse I/O patterns and workload characteristics. According to their I/O patterns, the applications in ACloud are categorized into three groups:

- **Read-heavy applications**. Read-heavy applications cover the applications generating read-dominant request streams. Representatives include a site search engine, which provides a keyword-based search service for retrieving information from all pages hosted by a large website. This search engine generates a large amount of read operations for scanning the index files. Read-heavy applications tend to issue read-intensive operations that access large blocks of the data at a time.
- **Write-heavy applications**. Write-heavy applications involve the applications generating write-dominant request streams. Representatives include (1) a database backup system, which periodically synchronizes the data with the online database; (2) a log collection system, which continuously collects and saves system log. The I/O requests generated by the log collection system is highly sequential, consisting of predominately writes.
- **Balanced-read/write applications**. Balanced-read/write applications refer to the applications generating balanced read/write request streams. For example, a mail server system, which utilizes a key-value storage engine to manage the mail data. The read and write requests from the mail sever system are basically balanced.

C. Write and Read in PFS

To support write-heavy applications, PFS borrows ideas from LevelDB [45] and utilizes Log-Structured Merge (LSM) tree [46] to reduce the I/O latency. The LSM tree offers fast random updates, inserts and deletes without sacrificing lookup performance [47]. Figure 1 depicts the procedures of handling write and read requests in PFS.

TABLE II: Summary of the workload trace.

Workload trace	
Log Period	Oct.12-25, 2013
Number of requests	642 million(DSS), 210 million(DPS)
Number of files in PFS	245 million
Number of chunks in PFS	202 million
Number of applications	1321(DSS), 96(DPS)
Storage used	15PB
Local file systems	Ext3
Default chunk size	64MB

For a write request, one of the PFS MasterServers first writes the change to the *write-ahead log* (aka. *"redo log"*), then writes it to an in-RAM buffer called `MemTable`, which holds newly updated items, and finally returns success to the client. The `MemTable` is organized by the LSM tree and replicated among multiple servers for enhancing the reliability of the redo logs. The `MemTable` will be dumped to disks when the memory buffer is full. With the help of the LSM, the random updates have been transformed into sequential updates, which can achieve a higher performance.

For a read request, one of the PFS MasterServers initially looks up the `MemTable`, as the `MemTable` always contains the last update. If found, the result will be returned immediately, otherwise the PFS MasterServers will merge the on-disk tree and the `MemTable`, and return the results to the requester.

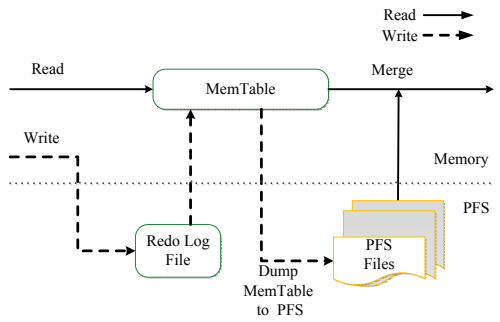


Fig. 1: The procedures of write and read operations executed on PFS.

D. Trace Summary

The trace discussed in this paper was collected during the period from Oct. 12 to 25, 2013. During this two-week period, billions of I/O requests are generated by various applications. To limit the system performance degradation caused by trace collection, we sampled the I/O requests with a fixed rate, yielding over 850 million I/O requests totally. Table II gives an overview of the trace.

IV. Workload Analysis

In this section, we analyze the features of the I/O workloads, including the arrival rate of requests, the size of requests, and so on. In one aspect, the analysis emphasizes the difference of features between DSS and DPS. In the other aspect, it focuses on the specific features of cloud file systems compared with traditional file systems. Design implications are italicized at the end of the corresponding paragraph.

A. Request Arrival Pattern

The arrival rate of requests is a key attribute for describing the workload scale. To synthesize a realistic workload for benchmarking, both qualitative and quantitative descriptions of the request arrival pattern are required. Figures 2(a) and 2(b) depict the fluctuation of arrival rate of requests. Figure 2(a) plots the arrival rate over two weeks, while Figure 2(b) plots the arrival rate over three days. The *X*-axis indicates the days of the period, and the *Y*-axis represents the average number of Arrived Requests Per Second (ARPS for short). As expected, the ARPS generated by the DSS and DPS services appear with daily periodicity. It is worth noting that the workload peaks of DSS and DPS emerge at different time intervals. For DSS, the workload peak occurs from 8:00 pm to 10:00 pm. For DPS, two workload peaks are observed. The first one occurs early morning every day, and the second one lasts from about 1:00 pm to 10:00 pm. **Implication 1.** *This observation implies that the ARPS produced by cloud benchmarks should vary over time and show a regular daily periodicity, rather than a static or a random rate. The workload peak period differs notably between different cloud services.*

Besides the ARPS, the model of request inter-arrival times (IAT for short) is also significant for benchmarks, as the IATs play an important role in the performance of a task scheduling policy [48]. Conventionally, the I/O requests are assumed to be submitted independently and the arrival of requests follows a *Poisson process* [49]. Poisson processes generate independent and identically distributed inter-arrival time that follows an (negative) exponential distribution.

However, we find this assumption is not always true. Thousands of requests arrive in one second, but their arrival time differ with each other on the precision of microseconds. At first, we collect the IATs between two subsequent requests (Figure 3) and find the distribution of the IATs considerably deviates from exponential distribution. Then, we plot the probability density function (PDF) of the ARPS, as shown in Figure 4. The black solid curve represents the probability of the fitted Poisson distribution, and the blue dotted curve represents fitted Lognormal distribution. After curve fitting, we find that the best fitting distribution is Lognormal distribution, rather than Poisson distribution.

To further verify this observation, we employ Q-Q plots to evaluate the similarity of two distributions. Q-Q plots are a simple graphical means to compare two distributions. The idea is to find the percentiles of the two distributions, and plot one set as a function of the other set. If the distributions match, the percentiles should come out at the same distances, leading to a straight line. Figure 5 shows the Q-Q plots for Poisson and Lognormal distributions. Based on these four subfigures, we can observe a good imitation of a straight line in Figures 5(b) and 5(d).

All plots in Figures 4 and 5 suggest that the distribution of arrival rates is very nearly Lognormal, rather than Poisson. We speculate this surprising observation could be explained by the I/O requests are invoked after various scheduling operations. For example, request are routed for load balancing on DSS, or requests are generated after a series of job/task scheduling operations on DPS. The latency of scheduling, including waiting times and processing times, changes the request arrival

rates. **Implication 2.** *For designing a workload generator for cloud benchmarking, it is inappropriate to simply assume that the arrival rate of requests follows a Poisson distribution.*

B. Session Behaviors

To study workload modeling [48], Schroeder *et al.* defined two system models, closed system model and open system model. In a closed system model, new request arrivals are only started when the previous request is completed, possibly followed by user think time. On contrary with an open system model, new jobs arrive independently with each other. Based on these definitions, various server systems are either assumed to be closed or open system models when benchmarking.

As described in subsection IV-A, the ARPS does not follow a Poisson distribution, and we have inferred that the arrivals of I/O requests are not completely independent. That is to say, for a portion of I/O requests, they are inter-dependent, and forms a *session*. Within a session, new request arrivals rely on previous request completions. For benchmarking, the number of requests in a session, which is often referred to as *session size* in literature, should be taken into consideration.

To analyze the session size, a preliminary step is to identify the session timeout interval. Due to the diversity of cloud applications, the exact session timeout interval is hard to determine. We conduct an empirical evaluation of approximate session timeout among I/O request streams in one day, and the results are depicted on Figure 6(a). The X-axis of Figure 6(a) represents the values of different timeout ranging from 5 seconds to 6 hours, while the Y-axis is the number of sessions counted in the context of different timeouts. Figure 6(a) demonstrates that when the timeout reaches thirty minutes, the number of sessions become stable. Therefore, the interval of thirty minutes could be regarded as session timeout interval approximately.

Figure 6(b) presents the average session size at each hour-interval during one day. For the requests from DSS, the session size generally ranges from 10 to 20 during the interval from 0:00 am to 8:00 am, and changes to values between 6 and 8. For the requests from DPS, the session size exceeds 10. According to the research conclusions in [48], when the session size is less than 5, the system behaves similarly to an open system; when the session size is more than 10, the system behaves like a closed system. Therefore, during the period of daytime, the file system PFS is partly-open under the DSS request streams. **Implication 3.** *This observation implies that a substantial portion of I/O requests provided by benchmarks should arrive as a sequence of a session. The assumption that cloud file systems behave like an open or closed system will not be satisfied.*

C. Periodicity Pattern

The periodicity analysis is a common methodology for predicting the request arrivals. Understanding the periodicity pattern for a certain workload is necessary for designing a benchmark. However, the periodicity analysis of I/O requests has not attracted enough attention. Some benchmarks provide requests with a daily period, some even possibly do not show any regular period. In this subsection, we perform a periodicity analysis of the I/O requests traces to reveal the underlying periodicity of the sequence of arrival events.

Discrete fourier transform is often used to detect the periodicity. Figure 7 provides the amplitudes of each discrete frequency. As shown in Figure 7(a), we observe five frequencies of high power-spectrum amplitudes: two-week, one-day, 12-hour, 8-hour and 6-hour. The signals of two-week and one-day are to be expected. Besides these two signals, the other three (12-hour, 8-hour and 6-hour) also have high amplitudes. These signals imply three major periods that exist among the request inter-arrival process. While in Figure 7(a), only two signals of two-week and one-day are evident. **Implication 4.** *The I/O requests in cloud file systems show multiple periodicities, rather than a single periodicity. For different cloud services, their periodicities would probably differ notably.*

D. Request Sizes

When generating a stream of requests, the workload generator needs to determine not only when to submit a request, but also which size the request should be. The request size plays a major role in the processing latency, and affects the throughput of cloud file systems.

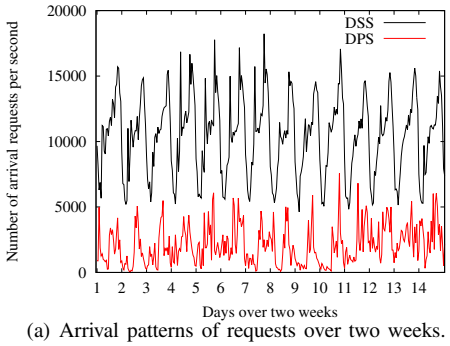
In this subsection, we attempt to derive a model for determining the size of requests in benchmarks. Figures 8(a) and 8(b) depict the cumulative distribution function of request size. Figures 8(c) and 8(d) show the probability function of request sizes. Ninety-eight percent of read requests are smaller than 400KB. Ninety-three percent of write requests are smaller than 400KB.

Researchers often use Zipf distribution to represent the file size on the Internet [50]. Motivated by this method, we plot the probabilities of write request size over the ranks of request size on a log-log scale in Figure 9(a). After curve fitting, we find that the size of write requests follows a Zipf distribution. The green line is the fitting curve using the least squares method (Zipf parameter $s = 0.7375$), while the red line is the fitting curve using the maximum likelihood method (Zipf parameter $s = 0.7748$).

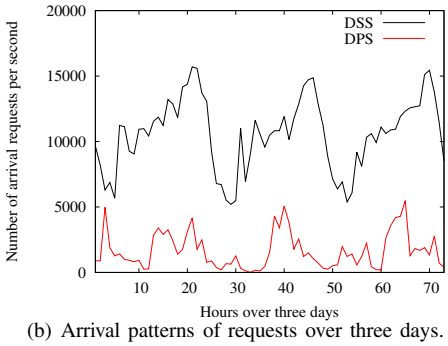
For read requests, we enumerate a series of well-known distributions, including Pareto, Weibull, Exponential, Lognormal, and examine their ability to fit the distribution of read request sizes. Interestingly, we observe that read request sizes fit a Lognormal distribution. The red curve in Figure 9(b) is the CDF of the best fitted Lognormal distribution, while the parameters of mean and standard deviation are 8.08 and 2.58, respectively. **Implication 5.** *For designing a workload generator in cloud benchmarks, the size of write requests provided by cloud benchmarks should follow a Zipf distribution, while the size of read requests should follow a Lognormal distribution.*

V. Data characteristics

A deep understanding of the data characteristics in cloud file systems is crucial to effectively pre-populate data in a benchmark. In this section, we analyze several data characteristics including file size and directory depth, data locality, and daily traffic on the file system of ACloud.

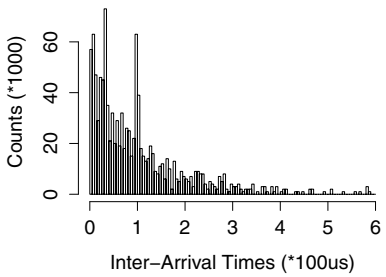


(a) Arrival patterns of requests over two weeks.

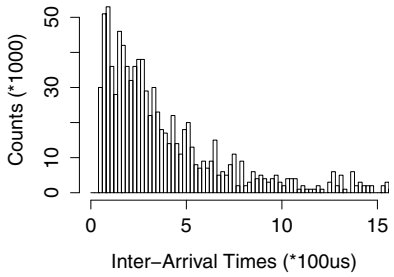


(b) Arrival patterns of requests over three days.

Fig. 2: Patterns of request arrival rate.

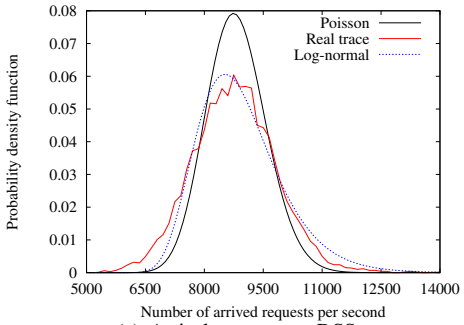


(a) The IATs of the requests on DSS.

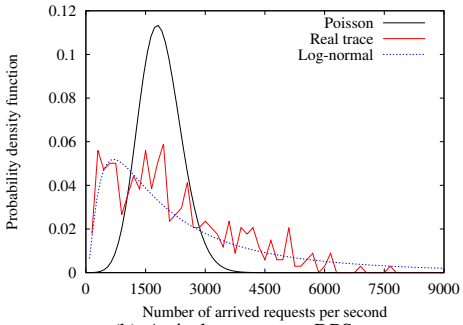


(b) The IATs of the requests on DPS.

Fig. 3: The distribution of the IATs.



(a) Arrival process on DSS



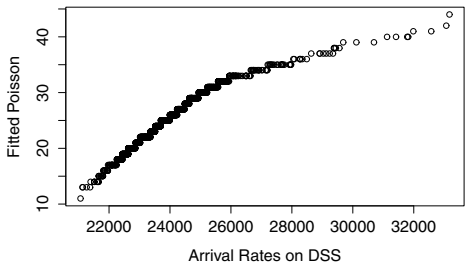
(b) Arrival process on DPS.

Fig. 4: The distribution of request arrival rates.

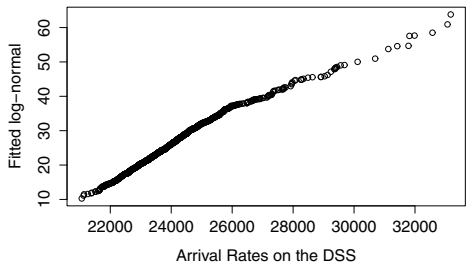
A. File Size and Depth

Large and small files present different challenges to the file system. For large files, finding contiguous space can be difficult, while for small files, minimizing initial seek time is emphasized. A large volume of small files also increase the overhead of metadata servers. In PFS, there exists about 245 million files, which occupies a total of 15PB of storage space. The size of each file ranges from 0B to 2TB. In this experiment, we divide file range into eight size intervals and count the number of files for each size interval. Figures 10(a) and 10(b) present the distribution of files grouped by size. We observed that about 50% of files are smaller than 256KB, and over 90% of files are smaller than 64MB. That is to say, more than 90% of files are smaller than a single chunk size (default 64MB). **Implication 6**. *When pre-populating data in cloud benchmarks, those files that are smaller than 64MB should be the majority in the testbed.*

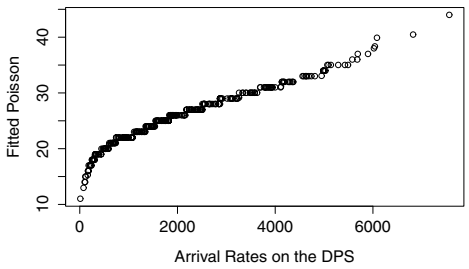
In addition, we also observed the distribution of file directory depth, which is one data characteristic that affects the I/O latency. Figure 10(c) shows the distribution of file directory depth. The maximum directory depth for an individual file is set to 20 in PFS. We count the number of files for every depth, and find that file directory depth follows a normal distribution. The directory depths range from 3 to 20, and the median value of file directory depth is 9. **Implication 7.** *As directory depth affects in the performance of meta-data access, the directory depth distribution of pre-populated data in cloud benchmarks should match the real cases.*



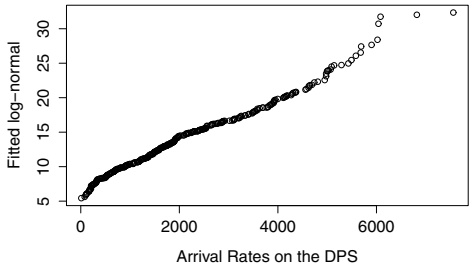
(a) Comparison with Poisson for DSS.



(b) Comparison with Lognormal for DSS.

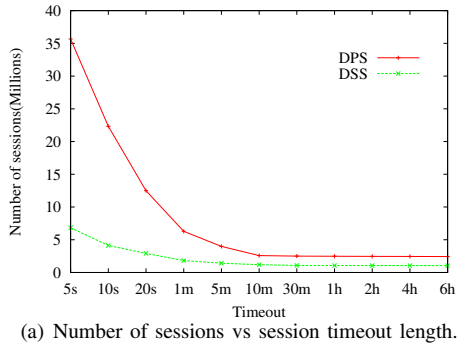


(c) Comparison with Poisson for DPS.

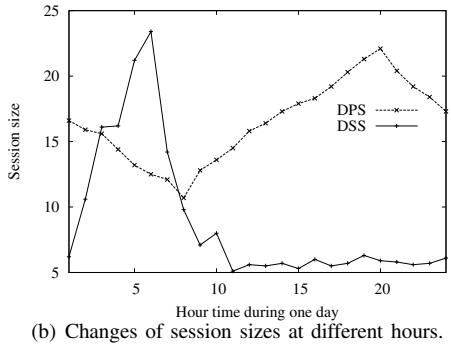


(d) Comparison with Lognormal for DPS.

Fig. 5: Q-Q plots for evaluation of distribution similarity.



(a) Number of sessions vs session timeout length.



(b) Changes of session sizes at different hours.

Fig. 6: Comparative analysis of session sizes between DPS and DSS.

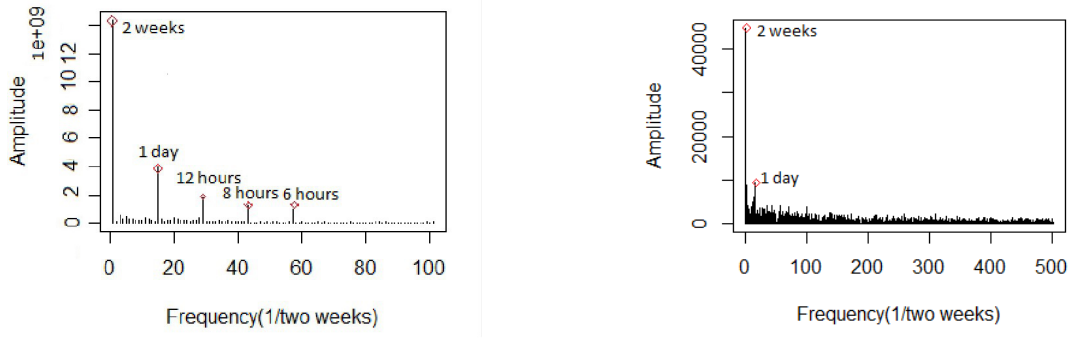
B. Data Locality

Data locality is one of the critical features to reflect the performance of file systems. A common scheme for processing data-intensive jobs is to move jobs on the node which hosts the related data (no need to fetch data from the other nodes). This is always referred to as *node-locality*. If the node-locality is not satisfied, the data needs to be fetched from either a node at an identical rack (*rack-locality*) or a remote node at other racks (*rack-remote*). Data locality is calculated as the ratio of I/O data bytes being accessed at local nodes.

In distributed systems, the higher the data locality is, the smaller network overhead and latency for data access are. Prior studies have reported that tasks on Hadoop clusters tend to achieve a high data locality [14,15,51], which is over 80%. However, it is observed that the data locality in ACloud is lower than the one observed in large-scale production Hadoop systems. Figure 11 shows that the data locality for read writes is about 46%, while for write requests, the value is 83%. Data locality achieved in cloud file systems is lower than the one achieved in other Hadoop systems, such as Facebook Hadoop [14], Taobao Hadoop [15]. **Implication 8**. *As data locality plays an important role in the performance of I/O scheduling, the data distribution policy adopted by cloud benchmarks should be carefully taken into consideration when pre-populating a testbed.*

C. Application Variability

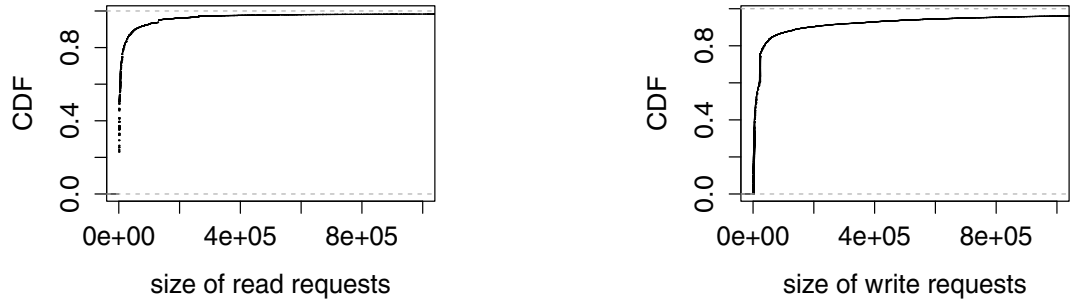
We analyzed the characteristics of file size and file popularity for each application. Ninety-six DPS applications and 1,321 DSS applications are hosted by ACloud during the two-week period. Each application has its own directory and files, and none of the files are shared by multiple applications. Figure 12 depicts the variability of applications in terms of file popularity, storage space consumption and average file size. We find that the distributions of both total volume size and file popularity are highly-skewed. Therefore, the skewness in terms of file popularity pose challenges for global resource schedulers to meet SLAs.



(a) Spectrum of the requests from DSS.

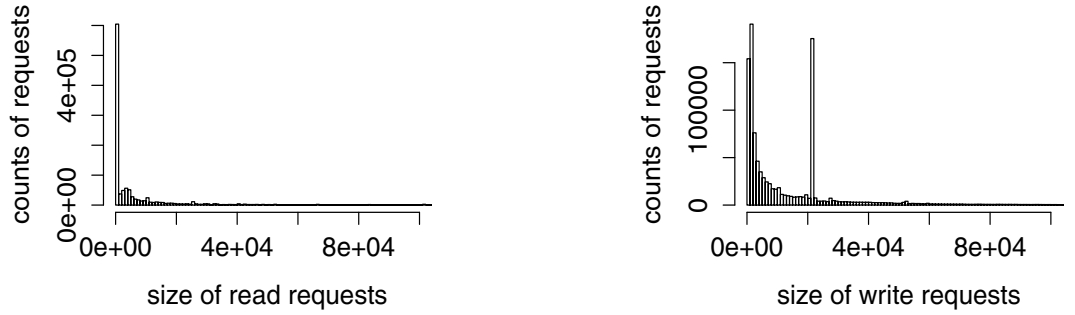
(b) Spectrum of the requests from DPS.

Fig. 7: Comparative analysis of periodicity pattern between DSS and DPS.



(a) CDF of the size of read requests.

(b) CDF of the size of write requests.



(c) Histogram of the size of read requests.

(d) Histogram of the size of write requests.

Fig. 8: Comparative analysis of read and write request size distributions in DSS.

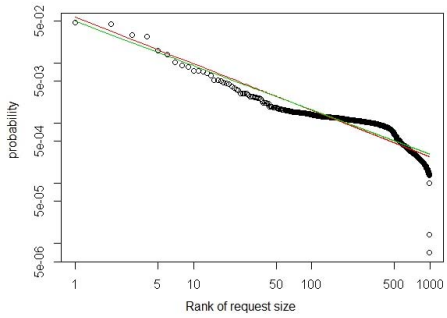
VI. Discussion

Based on the above observations and our early experiences on evaluating cloud file systems, we describe some open issues worthy of future exploration for making a realistic benchmark.

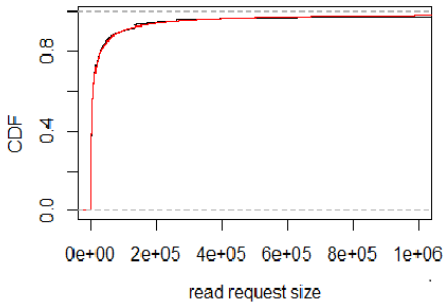
A. Modeling I/O workloads in cloud environments

For evaluating the performance of a cloud file system, the request stream needs to reflect real cases, in terms of the request size, inter-arrival time, and request arrival rate fluctuation over time, and so on. However, the features of I/O workload are complex and high-dimensional. It is non-trivial to identify which features affect the performance and which features do not. Only the features that do not affect the performance can be safely ignored.

The rapid revolution of I/O workloads also poses challenges for making a realistic benchmark. To accommodate the revolution and variation, the workload generator policy should be parameterized using empirical knowledge. In addition, the interference and constraints among the I/O requests, which

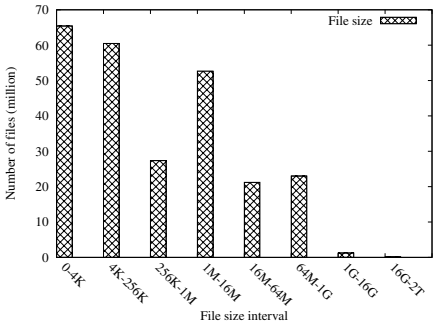


(a) The log-log scale of probability vs. rank for the write requests.

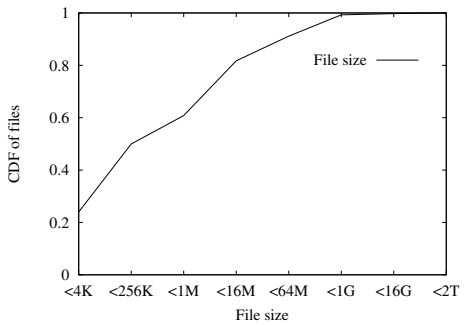


(b) Curve fitting for the read request size distribution. The red curve is fitted Lognormal distribution, and the black curve is the actual data.

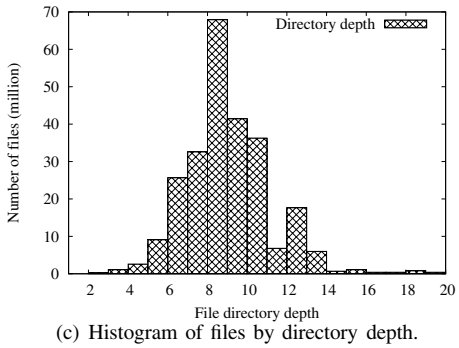
Fig. 9: The best fitting distributions for request size distributions.



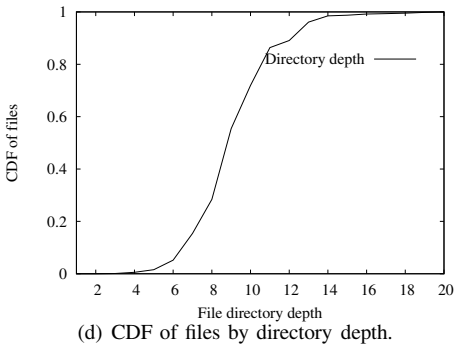
(a) Histogram of files grouped by size.



(b) CDF of files by size.

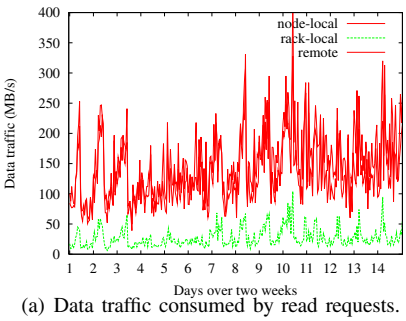


(c) Histogram of files by directory depth.

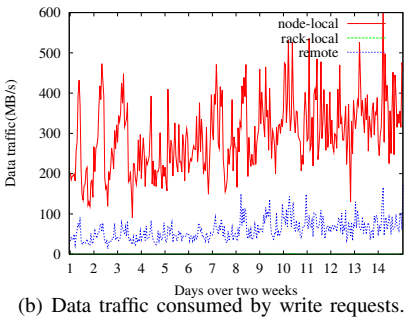


(d) CDF of files by directory depth.

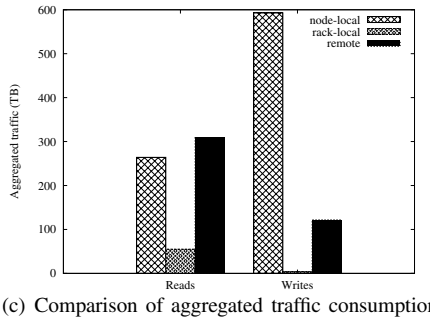
Fig. 10: Distributions of files by the file size and directory depth.



(a) Data traffic consumed by read requests.

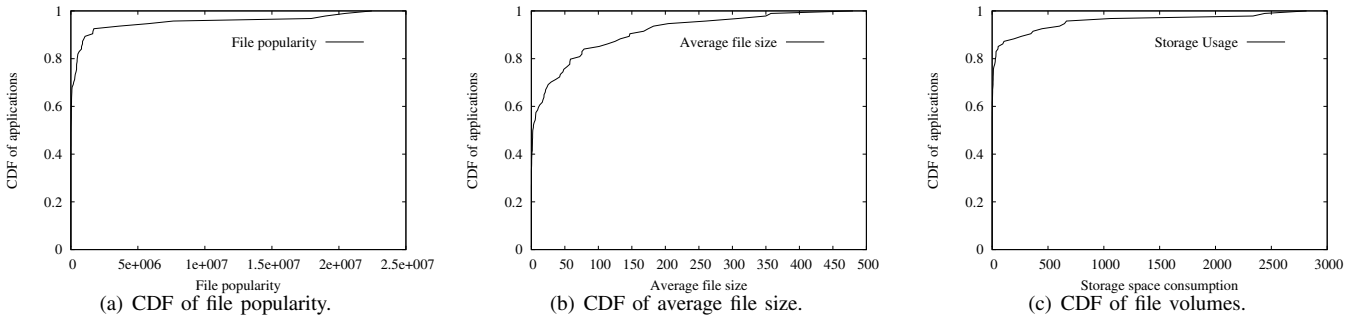


(b) Data traffic consumed by write requests.



(c) Comparison of aggregated traffic consumption

Fig. 11: Data traffic consumed by read and write requests.



(a) CDF of file popularity. (b) CDF of average file size. (c) CDF of file volumes.

Fig. 12: File popularity, storage space usage and file size of various applications.

are often omitted by existing benchmarks, should be deeply investigated and synthesized for designing benchmarks.

B. Modeling cloud file systems

In some cases, benchmarks are not only used to evaluate the system performance, but also used to identify the system bottlenecks or pinpoint the source of request failures. Therefore, understanding the model of cloud file systems is particularly important. Cloud file systems involve a large-scale cluster, and interact with many hardware/software components, including local operating systems, I/O devices, caches, disk, data center network, and so on. All these interactions make the system modeling much more complex. As cloud file systems have attracted a lot of attention from both industrial and academic areas, new improvements or features are continually emerging. Benchmarks are required to evolve in parallel with the development of cloud file systems. We believe that the current state of cloud file systems, as well as their benchmarks, has ample space for improvement.

C. Scaling down workload against cluster capacity

For benchmarking cloud file systems, it is costly to build a production-scale cluster to replay the historical workload and reproduce the system behavior. A common solution is to build a small testing cluster, pre-populate appropriate amounts of data and workloads. However, it is still vague about how to scale down the workload against a given cluster capacity. It will be particularly meaningful if benchmark results can be normalized for cluster capacity on which they are performed. If so, researchers can accurately compare results from benchmarks that were conducted on different clusters.

VII. Conclusions

Building a realistic benchmark is a critical and challenging task. It is especially difficult for cloud file systems. Many system researchers and engineers face challenges on making a benchmark that reflects real-life workload cases, due to the system complexity and vagueness of I/O workload characteristics. This paper did not propose a benchmark for cloud file systems, but presented some early experiences to explore the characteristics of data and I/O workload in a production environment. We observed that request arrival rate follows a Lognormal distribution rather than a Poisson distribution, requests present multiple periodicities, cloud file systems behave a partly-open system model rather than a purely open or closed model, the size of write requests follows a Zipf distribution, while the size of read requests follows a Lognormal distribution. Based on the comparative analysis results, we derived some interesting implications and open questions, which can help system researchers and engineers make a realistic benchmark test upon their own systems.

In the future, we will continue to work on implementing a realistic and flexible benchmark for cloud file systems. We will investigate suitable models of I/O requests and analyze their effectiveness under different scenarios. We plan to make the benchmark practical and available to the cloud computing community. In addition, the open questions described in Section VI are also our focus in the future.

Acknowledgment

This research was supported by NSF of China (No. 61300033). Weisong Shi is in part supported by the Introduction of Innovative R&D team program of Guangdong Province (No. 201001D0104726115) and Hangzhou Dianzi University. Wan Jian is supported by National Sci-tech Support Plan Projects of China (No. 2012BAH24B04). We would like to thank software engineers from the A company for their support on log collection and early comments on the paper. We are especially grateful to the anonymous reviewers for their helpful comments.

References

[1] Amazon Elastic Compute Cloud, "http://aws.amazon.com/ec2/."

[2] Windows Azure, "http://www.windowsazure.com/."

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *Operating Systems Review*, vol. 37, no. 5, 2003, pp. 29–43.

[4] D. Borthakur, "HDFS architecture guide," *HADOOP APACHE PROJECT http://hadoop. apache. org/common/docs/current/hdfs design. pdf*, 2008.

[5] D. Borthakur, J. Gray, J. S. Sarma, K. Muthukkaruppan, N. Spiegelberg, H. Kuang, K. Ranganathan, D. Molkov, A. Menon, S. Rash *et al.*, "Apache hadoop goes realtime at facebook," in *SIGMOD*. ACM, 2011, pp. 1071–1080.

[6] F. Wang, J. Qiu, J. Yang, B. Dong, X. Li, and Y. Li, "Hadoop high availability through metadata replication," in *Proceedings of the first international workshop on Cloud data management*. ACM, 2009, pp. 37–44.

[7] E. Stefanov, M. van Dijk, A. Juels, and A. Oprea, "Iris: A scalable cloud file system with efficient integrity checks," in *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 2012, pp. 229–238.

[8] M. L. Mazurek, E. Thereska, D. Gunawardena, R. Harper, and J. Scott, "ZZFS: A hybrid device and cloud file system for spontaneous users," in *FAST*. USENIX Association, 2012, pp. 16–16.

[9] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler, "Apache hadoop YARN: Yet another resource negotiator," in *SoCC*. ACM, 2013.

[10] J. Katcher, "Postmark: A new file system benchmark," Technical Report TR3022, Network Appliance, 1997. www. netapp. com/tech_library/3022. html, Tech. Rep., 1997.

[11] S. P. Council, "SPC-2 Benchmark, december 2005."

[12] C. Abad, N. Roberts, Y. Lu, and R. Campbell, "A storage-centric analysis of mapreduce workloads: File popularity, temporal locality and arrival patterns," in *IISWC*, 2012, pp. 100–109.

[13] S. Liu, X. Huang, H. Fu, and G. Yang, "Understanding data characteristics and access patterns in a cloud storage system," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, May 2013, pp. 327–334.

[14] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," in *EuroSys*, 2010, pp. 265–278.

[15] Z. Ren, J. Wan, W. Shi, X. Xu, and M. Zhou, "Workload analysis, implications, and optimization on a production hadoop cluster: A case study on taobao," *Services Computing, IEEE Transactions on*, vol. 7, no. 2, pp. 307–321, April 2014.

[16] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *SoCC*, 2010, pp. 143–154.

[17] A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 35–40, Apr. 2010. [Online]. Available: http://doi.acm.org/10.1145/1773912.1773922

[18] L. George, *HBase - The Definitive Guide*. O'Reilly, 2011.

[19] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni, "PNUTS: Yahoo!'s hosted data serving platform," *PVLDB*, vol. 1, no. 2, pp. 1277–1288, 2008.

[20] Y. Shi, X. Meng, J. Zhao, X. Hu, B. Liu, and H. Wang, "Benchmarking cloud-based data management systems," in *Proceedings of the second international workshop on Cloud data management*. ACM, 2010, pp. 47–54.

[21] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," in *ACM SIGARCH Computer Architecture News*, vol. 40, no. 1. ACM, 2012, pp. 37–48.

[22] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen, "Bigbench: Towards an industry standard benchmark for big data analytics," in *SIGMOD*. ACM, 2013, pp. 1197–1208.

[23] Z. Jia, L. Wang, J. Zhan, L. Zhang, and C. Luo, "Characterizing data analysis workloads in data centers," in *IISWC*. IEEE, 2013, pp. 66–76.

[24] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The hibench benchmark suite: Characterization of the mapreduce-based data analysis," in *Data Engineering Workshops (ICDEW)*. IEEE, 2010, pp. 41–51.

[25] J. Gray, "Graysort benchmark," *Sort Benchmark Home Page–http://sortbenchmark. org*.

[26] C. Luo, J. Zhan, Z. Jia, L. Wang, G. Lu, L. Zhang, C.-Z. Xu, and N. Sun, "Cloudrank-d: benchmarking and ranking cloud computing systems for data processing applications," *Frontiers of Computer Science*, vol. 6, no. 4, pp. 347–362, 2012.

[27] Z. Jia, R. Zhou, C. Zhu, L. Wang, W. Gao, Y. Shi, J. Zhan, and L. Zhang, "The implications of diverse applications and scalable data sets in benchmarking big data systems," in *Specifying Big Data Benchmarks*. Springer, 2014, pp. 44–59.

[28] C. Baru, M. Bhandarkar, R. Nambiar, M. Poess, and T. Rabl, "Benchmarking big data systems and the bigdata top100 list," *Big Data*, vol. 1, no. 1, pp. 60–64, 2013.

[29] E. Dede, Z. Fadika, M. Govindaraju, and L. Ramakrishnan, "Benchmarking mapreduce implementations under different application scenarios," *Future Generation Computer Systems*, vol. 36, no. 0, pp. 389 – 399, 2014.

[30] Z. Ming, C. Luo, W. Gao, R. Han, Q. Yang, L. Wang, and J. Zhan, "Bdgs: A scalable big data generator suite in big data benchmarking," *arXiv preprint arXiv:1401.5465*, 2014.

[31] Y. Chen, F. Raab, and R. Katz, "From tpc-c to big data benchmarks: A functional workload model," in *Specifying Big Data Benchmarks*, 2014, vol. 8163, pp. 28–43.

[32] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, "A comparison of approaches to large-scale data analysis," in *SIGMOD*. ACM, 2009, pp. 165–178.

[33] A. Traeger, E. Zadok, N. Joukov, and C. P. Wright, "A nine year study of file system and storage benchmarking," *ACM Transactions on Storage (TOS)*, vol. 4, no. 2, p. 5, 2008.

[34] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, "Workload analysis of a large-scale key-value store," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1. ACM, 2012, pp. 53–64.

[35] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *WORKS*. IEEE, 2008, pp. 1–10.

[36] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, and K. Riley, "24/7 characterization of petascale i/o workloads," in *CLUSTER*. IEEE, 2009, pp. 1–10.

[37] Y. Chen, K. Srinivasan, G. R. Goodson, and R. H. Katz, "Design implications for enterprise storage systems via multi-dimensional trace analysis," in *SOSP*, 2011, pp. 43–56.

[38] T. Shibata, S. Choi, and K. Taura, "File-access patterns of data-intensive workflow applications and their implications to distributed filesystems," in *HPDC*. ACM, 2010, pp. 746–755.

[39] F. Wang, Q. Xin, B. Hong, S. A. Brandt, E. L. Miller, D. D. E. Long, and T. T. Mclarty, "File system workload analysis for large scale scientific computing applications," in *IEEE Conference on Mass Storage Systems and Technologie(MSST)*, 2004, pp. 139–152.

[40] Y. Chen, A. S. Ganapathi, R. Griffith, and R. H. Katz, "Analysis and lessons from a publicly available google cluster trace," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-95, Jun 2010. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-95.html

[41] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *SoCC*, 2012, p. 7.

[42] P. Riteau, K. Keahey, C. Morin *et al.*, "Bringing elastic mapreduce to scientific clouds," in *3rd Annual Workshop on Cloud Computing and Its Applications: Poster Session*, 2011.

[43] T. Gunarathne, T.-L. Wu, J. Qiu, and G. Fox, "Mapreduce in the clouds for science," in *CloudCom*. IEEE, 2010, pp. 565–572.

[44] J. Rao, E. J. Shekita, and S. Tata, "Using paxos to build a scalable, consistent, and highly available datastore," *Proceedings of the VLDB Endowment*, vol. 4, no. 4, pp. 243–254, 2011.

[45] LevelDB, 2014. [Online]. Available: http://code.google.com/p/leveldb

[46] P. ONeil, E. Cheng, D. Gawlick, and E. ONeil, "The log-structured merge-tree (lsm-tree)," *Acta Informatica*, vol. 33, no. 4, pp. 351–385, 1996.

[47] M. A. Bender, M. Farach-Colton, J. T. Fineman, Y. R. Fogel, B. C. Kuszmaul, and J. Nelson, "Cache-oblivious streaming b-trees," in *Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*. ACM, 2007, pp. 81–92.

[48] B. Schroeder, A. Wierman, and M. Harchol-Balter, "Open versus closed: A cautionary tale," in *NSDI*, 2006, pp. 18–18.

[49] S. M. Ross, *Stochastic processes*. John Wiley & Sons New York, 1996, vol. 2.

[50] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *INFOCOM*, vol. 1. IEEE, 1999, pp. 126–134.

[51] Y. Chen, S. Alspaugh, and R. H. Katz, "Design insights for mapreduce from diverse production workloads," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2012-17, Jan 2012.