Decoupling Datacenter Storage Studies from Access to Large-Scale Applications

Christina Delimitrou*, Sriram Sankar†, Kushagra Vaid† and Christos Kozyrakis*
*Stanford University, †Microsoft
{cdel, kozyraki}@stanford.edu, {srsankar, kvaid}@microsoft.com

Abstract—Suboptimal storage design has significant cost and power impact in large-scale datacenters (DCs). Performance, power and cost-optimized systems require deep understanding of target workloads, and mechanisms to effectively model different storage design choices. Traditional benchmarking is invalid in cloud data-stores, representative storage profiles are hard to obtain, while replaying applications in different storage configurations is impractical both in cost and time. Despite these issues, current workload generators are not able to reproduce key aspects of real application patterns (e.g., spatial/temporal locality, I/O intensity).

In this paper, we propose a modeling and generation framework for large-scale storage applications. As part of this framework we use a state diagram-based storage model, extend it to a hierarchical representation, and implement a tool that consistently recreates DC application I/O loads. We present the principal features of the framework that allow accurate modeling and generation of storage workloads, and the validation process performed against ten original DC application traces. Finally, we explore two practical applications of this methodology: SSD caching and defragmentation benefits on enterprise storage. Since knowledge of the workload's spatial and temporal locality is necessary to model these use cases, our framework was instrumental in quantifying their performance benefits. The proposed methodology provides detailed understanding of the storage activity of large-scale applications, and enables a wide spectrum of storage studies, without the requirement to access application code and full application deployment.

Index Terms—Modeling of computer architecture, Super (very large) computers, Mass storage, Modeling techniques.

1 Introduction

As cloud data-stores and social networking emerge, user data is increasingly being stored in large-capacity and high-performance storage systems. These systems account for a significant portion of the total cost of ownership (TCO) of a datacenter (DC) [7], [10]. Specifically, for online services, data retrieval is often the bottleneck to application performance [9], [10], making efficient storage provisioning a first-order design constraint.

The main challenges for large-scale online services are three-fold: first, they cannot be approximated by single machine benchmarking, due to user behavior patterns, second, privacy concerns make source code, and datasets unavailable to storage system designers, and third, replaying applications in all possible system configurations is highly impractical. It is hence imperative to invest in frameworks that enable extensive workload analysis, modeling and generation. This increases the appeal of a representative model that captures key aspects of the workload's storage profile, and a tool that reproduces this storage behavior via a synthetic load. This framework enables large-scale storage studies, decoupled from the requirement to access application code.

Despite the merit in this effort, previous work on I/O workload generation [2], [6], [11], [12] lacks the ability to capture the spatial and temporal locality of I/O accesses. In this work, we provide a framework for research on large-scale storage systems that addresses these issues. This infrastructure includes probabilistic, state diagram-based models that capture information of *configurable granularity* on the workload's access patterns. The models are developed from production traces of real DC applications based on previous work [9]. We extend these models to a granular, hierarchical representation to identify the optimal level of detail for each application. Furthermore, we design a tool that recognizes these models and recreates synthetic access patterns that closely match those of the original applications. We perform extensive validation of our methodology to ensure resemblance in both I/O characteristics and performance metrics.

We use our methodology (model and tool) to evaluate two important DC storage design challenges. First, we explore the applicability of Solid State Devices (SSD) caching in DC workloads. Using the modeling framework, we show that for most of the examined DC applications, SSD caching offers significant storage system speedup without application change (31% on average for a 32GB SSD cache). In the second use case, we evaluate defragmentation for DC storage. We observe that, over a period of time, user data gets accumulated and files become highly fragmented. Leveraging tracing information [5], we rearrange blocks on disk to improve the workloads' sequential characteristics. Using our framework, we show that defragmentation offers a significant boost in performance (18% on average), in some cases greater than incorporating SSDs.

Succinctly, the main contributions of this work are:

- A concise statistical model that accurately captures the I/O access patterns of large-scale applications, including their locality, and inter-arrival times. It is also hierarchical, which allows configurable level of detail to accommodate the features of each application.
- A tool that recognizes this model, and recreates synthetic access patterns with same I/O characteristics and performance metrics as in the original application. No previous tool (e.g., IOMeter) simulates spatial and temporal I/O locality of DC workloads.
- This methodology enables storage system studies that were previously **impossible without full application deployment, and without access to real application**

. Manuscript submitted: 18-Oct-2011. Manuscript accepted: 22-Nov-2011. Final manuscript received: 24-Nov-2011.

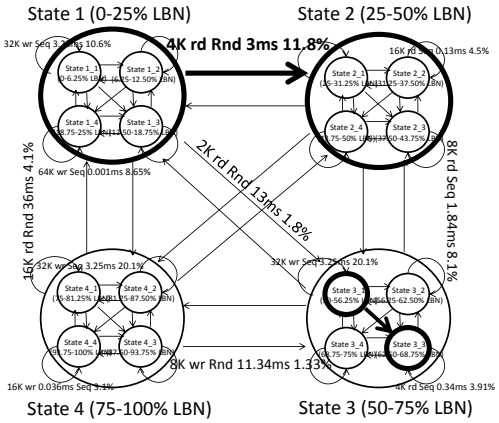

Fig. 1. Hierarchical (Two level) State Diagram Model

code. We demonstrate the applicability of our tool in evaluating SSD caching and defragmentation. These locality-based studies have been unexplored due to lack of a tool that allowed their evaluation.

In the remainder of the paper we present: the modeling and generation framework, the model's validation and a comparison with a popular workload generator (IOMeter) (Section 2), the tool's applicability to DC storage challenges (Section 3), as well as topics for future work (Section 4).

2 Modeling and Generation Process

In this Section we present the modeling and generation process implemented in our framework. More details on this, as well as extensive validation results can be found in [3].

2.1 State Diagram Model

Our approach requires a model that captures the I/O features and locality of storage activity, i.e., clusters accesses based on their spatial locality. For this, we use the Markov Chain representation proposed by Sankar et al. [9]. The models are trained based on real storage traces from production servers of a large-scale DC deployment, and capture I/O accesses in one or multiple servers, and from one or multiple users.

- *Basic Model:* According to the model, states correspond to ranges of logical blocks on disk (LBNs) and transitions represent the probabilities of switching between LBN ranges. Each transition is characterized by a set of features that reflect the workload's I/O behavior and consist of the block size, randomness, type of I/O (read, write) and inter-arrival time between subsequent requests. In its simplest form, the model consists of a number of states, e.g., four states, each of which corresponds to 25% of the total LBNs. The model works as follows (highlighted part of Figure 1): If an I/O corresponds to State 1, there is an 11.8% probability that the next I/O will be a 4K read, random access with an inter-arrival time of 3ms that corresponds to State 2.

- *Hierarchical Model:* To convey information of finer granularity, we have extended the previous model to a hierarchical representation. Figure 1 demonstrates one such model with two levels. For this example, each state in the one level diagram is subdivided in four states and becomes a new state diagram. The two-level diagram has 16 states.

Perhaps counter-intuitively, the number of transitions in the new diagram is not 256, but 76. As shown in Figure 1, level-two (fine-grained) transitions only exist within large states, but not across them. This means that a transition happens either between two major states (State 1 to State 2), or between two minor states (State 3_1 to State 3_3). The transition count for a number of levels is given by

$$4^{l-1}16 + \sum_{i=1}^{l-1} 4^{i-1}12 \qquad (1)$$

while for the flat model that explores all transitions, it is 16^l, where l is the number of levels. Reducing the number of transitions without sacrificing information happens in two steps: (i) we choose the number of states per level that minimizes the inter-state transitions, (ii) we choose the optimal number of levels per application. This way spatial locality is mostly confined within states.

TABLE 1
Scalability of the model in terms of state and transition count.

Levels	State Count	Transition Count	
		Hierarchical Model	Flat Model
1	4	16	16
2	16	76	256
3	64	316	4096
4	256	1276	65536
5	1024	5116	1048576
10	1048576	5242876	109951162776

Table 1 shows how the number of states and transitions scales for up to 10 levels. For the flat representation the number of transitions increases exponentially with the number of states, while the hierarchical model has a linear relation with state count. This choice does not cancel the value of a flat model, but rather proposes that a hierarchical model is just as beneficial without making the number of transitions intractable. Comparing the throughput of models constructed with the hierarchical and the flat representations shows less than 5% difference in throughput. The proposed model structure guarantees scalability even for applications that require many levels.

2.2 Generation Tool Design - DiskSpd

The model, previously discussed, is the first step in recreating accurate DC I/O loads. The second step, involves a tool that recognizes the model and generates storage workloads with high fidelity, using some configuration knobs.

For this purpose we use DiskSpd, a workload generator [4], which performs read and/or write I/Os in burst mode on either disks or files, given the I/Os' block size, randomness, and initial block offset. The former consist of a subset of the most relevant features of DiskSpd for the current study.
The main features we introduce for accurate generation are:

1) The ability to issue I/Os with specified inter-arrival times, both static and following time distributions.
2) The ability to preserve the spatial and temporal locality of I/O accesses, as well as the features and weights of each transition in the state diagram.
3) The ability to modify the intensity of the generated I/Os by scaling the inter-arrival time of independent I/O requests. This enables high performance storage systems evaluations (e.g., SSDs).
4) The ability to reflect the storage activity fluctuations of the original application by changing the intensity of I/O requests (i.e., inter-arrival times) over short periods of time. Essentially each model is composed by multiple models of different intensities that capture the transient features of the I/O requests.

TABLE 2
I/O features - Performance metrics validation for Messenger.

	Metrics	Original	Synthetic	Deviation
Messenger	Rd:Wr	2.8:1	2.8:1	0%
	% Rnd I/Os	90.68%	89.43%	-1.38%
	Block Size Distr.	*8K(87%)* *64K(7.4%)* *1K(1.6%)*	*8K(88%)* *64K(7.8%)* *1K(1.7%)*	0.1%-1%
	Thr Weight Distr.	*T1 (19%)* *T2 (11.6%)* *T3 (1.6%)*	*T1 (19%)* *T2 (11.68%)* *T3 (1.6%)*	0%-0.05%
	Avg Int-Arr. Time	4.63ms	4.78ms	1.02%
	Throughput	255.14	263.27	3.1%
	Avg Latency	8.09ms	8.48ms	4.8%

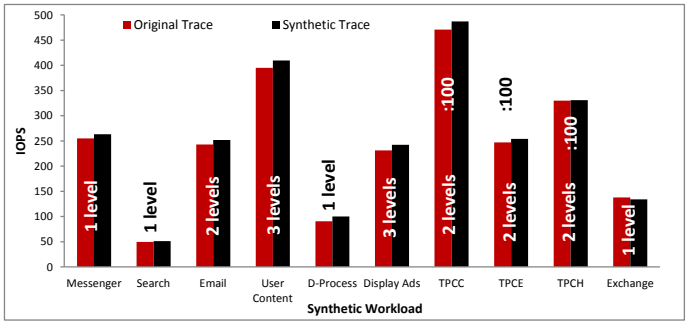

Fig. 2. Throughput comparison between original and synthetic workloads for the 10 applications.

2.3 Validation

Validating the accuracy of the framework is necessary to ensure that original and synthetic workloads are similar in their storage activity, and to identify the optimal level of detail for each application. We perform the following steps:

1) Collect traces from production servers
2) Create models with a configurable number of levels
3) Run the synthetic workload and collect new trace
4) Compare I/O characteristics and performance metrics between original and synthetic workload.

We use ten DC applications, which include SQL-based, online services and computationally intensive workloads. Experiments are run on an SQL-provisioned server, with 8 cores, 10 disk partitions and a total of 2.3TB of HDD storage.

For each of the ten applications we evaluate the similarities in terms of I/O features (block size, rd/wr, rnd/seq, inter-arrival time and thread weight) and performance metrics (throughput (IOPS) and latency) between original and synthetic applications. We verify that thread weights are satisfied with **less than 0.05%** deviation from their original values. Table 2 shows the comparison for these metrics for Messenger. The results are similar for the remaining applications. In all cases the deviation in I/O features is less than 5%, and for performance metrics at most 6.7%, and on average 3.38%. Figure 2 shows the throughput comparison between original and synthetic load for all applications. The difference in IOPS is always **less than 5%**, verifying the accuracy of the modeling and generation process. Furthermore, in order to ensure the consistency of our results, we verify that throughput variance between different runs of the same synthetic workload is **less than 1%**.

Figure 2 also plots the optimal number of levels per application, which is the one for which the synthetic trace resembles the original workload best. We define it as the first number of levels for which the performance metrics stabilize (less than 2% difference in IOPS). This way, we convey the best possible accuracy with the least necessary model complexity. In most cases, one to three levels are sufficient for the I/O characteristics to stabilize. Finally, we verify the resemblance of storage activity fluctuation between original and synthetic workloads. We capture activity fluctuations by creating multiple models, over 30-minute time intervals, and switching between these models when generating the synthetic workload. We observe that no significant fluctuations in storage activity occur at a granularity less than 30-minutes, for the examined applications.

2.4 Comparison with IOMeter

IOMeter is the most well-known open-source workload generator [6]. It offers many capabilities as far as access features are concerned, but has limited information on I/O spatial locality, making it unsuitable for several DC storage studies. Furthermore, IOMeter implements outstanding I/Os, but cannot represent inter-arrival times, which seriously inhibits its intensity scaling capabilities. Finally, it does not allow specific file accesses, which would make it impractical to evaluate the benefits of defragmentation (see Section 3.2).

Here we compare the performance (throughput and latency) of IOMeter and DiskSpd. For this purpose no change to capture I/O locality is conducted in IOMeter, and the parameters for the tests are defined using the tool's default knobs. Note that no notion of spatial locality is introduced in these simple tests. From the results we observe that both tools behave similarly with a maximum throughput deviation of 3.4%. Their main difference becomes evident when introducing the notion of spatial locality. To demonstrate that DiskSpd takes locality into account, while IOMeter does not, we use an optimization technique that is presented in more detail in Section 3.1. SSD caching takes advantage of frequently-accessed blocks, and improves performance by avoiding accessing the disk often. If a tool takes locality into consideration, we expect a performance improvement when the synthetic workload is run using SSD caches. We compare the performance of the ten synthetic traces run with DiskSpd and the corresponding I/O tests that best resemble them using IOMeter. No notion of spatial locality is incorporated in the latter. Figure 3 shows how performance changes as we progressively add SSDs to the system for Messenger and User Content. The important point in these figures is not the precise speedup but the significantly different behavior of the tools. It becomes evident that IOMeter does not reflect the spatial and temporal locality of the original access pattern. There either is no speedup, as is the case for Messenger, due to incorrect block caching, or there is inconsistent speedup (User Content) to what would have been expected as caching becomes more intense (more SSDs - better speedup).

3 Use Cases

One of the main benefits from using the proposed framework is enabling storage studies, which would otherwise require access to application code or full application deployment. Here we evaluate two possible use cases for the tool: SSD caching and defragmentation. Both are locality-dependent, and have been unexplored using workload generation tools.

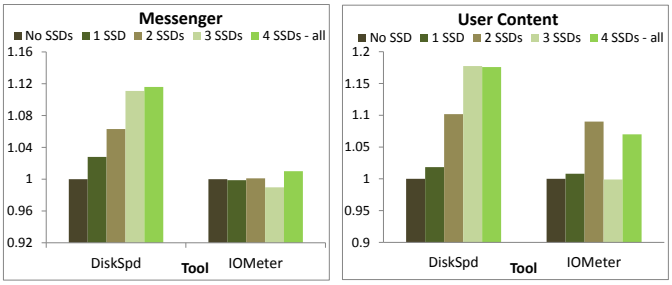

Fig. 3. IOMeter vs DiskSpd speedup comparison for (a) Messenger, and (b) User Content.

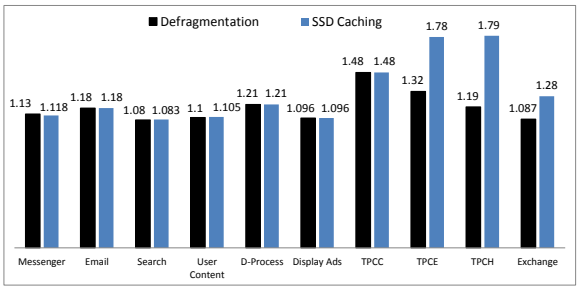

Fig. 4. Storage Speedup for Defragmentation/SSD Caching

3.1 SSD caching

Studying the spatial locality of the ten applications reveals that for most of them, I/Os are aggregated in a small LBN range. This motivates incorporating SSDs to improve performance. Estimating performance, however, is not easy, since writes in SSDs are highly unpredictable, and often as slow as disk. Here we use the previous models (open-loop approach) to predict speedup. Experiments are performed on an SQL-provisioned server with 4 SSD caches (8GB each) [1].

Figure 4 shows the storage speedup when all 4 SSD caches are on for the examined applications. We observe that especially for the I/O intensive TPCH, TPCE and D-Process, the performance benefit is significant (31% on average across all workloads for 4 SSDs, and 79% maximum for TPCH). Studying the clustering of accesses in the corresponding models reveals that these three applications have the highest I/O aggregation. More levels offer better speedup approximation. An important note is that in Figure 4, we refer to storage speedup and not speedup for the entire application. These workloads are not necessarily limited by storage but their performance (and efficiency) improvement are strong incentives towards the use of SSD caching nonetheless.

3.2 Defragmentation

Most DC applications experience high levels of fragmentation, as user-requests get accumulated over time, with Random I/Os often exceeding 80%. This motivates the use of defragmentation to improve performance. From tracing [5] we extract the name of the accessed file, estimate fragmentation levels, and perform a block rearrangement to improve the sequential characteristics. Estimating performance after defragmentation using new models based on defragmented traces, can act as an offline method to identify its benefits, as well as the optimal moment to perform defragmentation. These are usually latency-critical applications that cannot afford the overhead of a continuous online evaluation.

In most cases, sequential I/Os increase by 20%, which corresponds to a storage speedup of 8-45% (Figure 4). This implies that clustering I/Os is more beneficial than taking advantage of parallel spindles, due to faster sequential access completion. D-Process and Email, which have high write-to-read ratios, benefit most from defragmentation. Since these are random-write dominated applications, improving their sequential characteristics allows better utilization of the full disk rotation. Defragmentation also benefits the TPC benchmarks that access continuous entries in database tables.

Comparing SSDs and defragmentation shows that for some workloads (e.g., Messenger, Email) defragmentation offers better speedups without increasing the system's cost. The decision on which method to use is up to the storage system designer, given the application and system of interest.

The model enables accurate performance estimations without needing to deploy the optimizations in the entire DC.

4 Future Work and Conclusions

We have proposed a framework for modeling, characterization and regeneration of DC storage workloads. We have extended a probabilistic model to capture granular information on a workload's I/O pattern and implemented a tool that recreates DC workloads with high fidelity. This tool can be used for a wide spectrum of studies in large-scale systems, without requiring access to DC application code or full application deployment.

In contrast with previous work, we take into account I/O spatial and temporal locality, a critical feature for DC applications. We have performed extensive validation of the generated I/O traces against ten real large-scale applications, and have evaluated two possible uses for the framework, and quantified the performance improvement. We believe that, compared to previously available workload generators, this framework can be used to make confident design decisions in DC systems. Our future work focuses on the development of a model that aggregates all different system parts and a tool that recreates a synthetic load for complete DC applications, with possible applications to virtualization, application consolidation and DC performance and power modeling.

References

[1] Adaptec MaxIQ. 32GB SSD Cache Performance Kit. http://www.adaptec.com/en-US/products/CloudComputing/MaxIQ/SSD-Cache-Performance/
[2] I. Ahmad. "Easy and Efficient Disk I/O Workload Characterization in VMware ESX Server". In Proc. of IEEE IISWC, Boston, MA, 2007.
[3] C. Delimitrou, S. Sankar, K. Vaid and C. Kozyrakis. "Accurate Modeling and Generation of Storage I/O for Datacenter Workloads". In Proc. of EXERT, CA, March 2011.
[4] DiskSpd: File and Network I/O using Win32 and .NET API's on Windows XP http://research.microsoft.com/en-us/um/siliconvalley/projects/sequentialio/
[5] ETW: Event Tracing for Windows: http://msdn.microsoft.com/en-us/library/bb968803%28VS.85%29.aspx
[6] IOMeter, performance analysis tool. http://www.iometer.org/.
[7] C. Kozyrakis, A. Kansal, S. Sankar, K. Vaid, "Server Engineering Insights for Large-Scale Online Services". In IEEE Micro, vol.30, no.4, July 2010.
[8] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron, "Migrating enterprise storage to SSDs: analysis of tradeoffs". In Proc. of EuroSys, Nuremberg, 2009.
[9] S. Sankar, K. Vaid, "Storage characterization for unstructured data in online services applications". In Proc. IEEE IISWC, TX, 2009.
[10] S. Sankar, K. Vaid. "Addressing the stranded power problem in datacenters using storage workload characterization". In Proc. of the first WOSP/SIPEW, San Jose, CA, 2010.
[11] SQLIO Disk Subsystem Benchmark Tool. http://www.microsoft.com/downloads/en/details.aspx?familyid=9a8b005b-84e4-4f24-8d65-cb53442d9e19&displaylang=en
[12] H. Vandenbergh. Vdbench: User Guide 5.00 Oct. 2008.