Storage I/O Generation and Replay for Datacenter Applications

Christina Delimitrou*, Sriram Sankar†, Kushagra Vaid† and Christos Kozyrakis*
*Electrical Engineering Department, Stanford University, Stanford, CA.
{cdel, kozyraki}@stanford.edu
†Microsoft, Redmond, WA.
{srsankar, kushagra.vaid}@microsoft.com

I. Introduction

With the advent of social networking and cloud data-stores, user data is increasingly being stored in large capacity and high performance storage systems, which account for a significant portion of the total cost of ownership of a datacenter (DC) [3]. One of the main challenges when trying to evaluate storage system options is the difficulty in replaying the entire application in all possible system configurations. Furthermore, code and datasets of DC applications are rarely available to storage system designers. This makes the development of a representative model that captures key aspects of the workload's storage profile, even more appealing. Once such a model is available, the next step is to create a tool that convincingly reproduces the application's storage behavior via a synthetic I/O access pattern.

Previous efforts on storage I/O workload generation rely on traces which greatly depend on the underlying system. Publicly available tools lack the ability to detect and utilize the spatial and temporal locality of I/O access patterns, causing workloads to significantly deviate from the application's true characteristics.

In this work, we provide a toolset for research on large-scale storage systems. The main contributions of this framework are:

- A concise statistical model that accurately captures the I/O access pattern of large-scale applications including their spatial locality, inter-arrival times and type of accesses. 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 loads with same I/O characteristics as in the original application. No previous tool (eg: IOMeter [5]) can simulate spatial and temporal locality of DC workloads.
- This methodology enables storage system studies that were previously **impossible without full application deployment and access to the real applications**. We demonstrate the applicability of our tool in evaluating SSD caching and defragmentation. These spatial locality-based DC challenges have been relatively unexplored due to lack of a tool that allowed their evaluation.

II. Modeling, Generation and Use Cases

In this Section we describe the modeling and generation methodology as well as two possible use cases for the toolset.

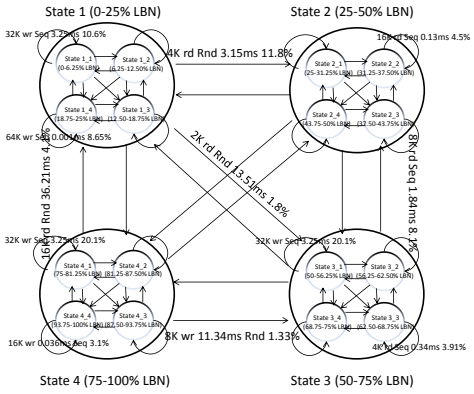

Fig. 1: Two levels State Diagram

A. Model and Generation Tool

For our model we use the Markov Chain representation proposed by Sankar et al. [1]. According to this, states correspond to ranges of logical blocks on disk (LBN) 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: block size, randomness, type of I/O (rd, wr) and inter-arrival time between subsequent requests. The probability for each transition is the percentage of I/Os that correspond to it. In order to convey more detailed information on the I/O access pattern we have extended this model to a hierarchical representation (Figure 1), where each state in the one level diagram is subdivided in four states and becomes a new state diagram.

The model, previously discussed, consists of 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. For this purpose we use DiskSpd [4], a workload generator which we enhance with a series of features. Succintly these features include:

1) The ability to issue I/O requests with specified inter-arrival times, both static and following time distributions. Unlike all previous work, our model is based on inter-arrival times instead of outstanding I/Os making it more representative of an application's behavior [2].
2) The ability to generate requests from multiple threads with individual access characteristics (block size, rd/wr, seq/rnd), while preserving the thread weight (i.e. portion of accesses) for each transition.

Metrics	Original Load	Synthetic Load	Deviation
Rd:Wr Ratio	1.8:1	1.8:1	0%
Random %	83.67%	82.51%	-1.38%
Block Sizes	8K(87%) 64K(7.4%) 1K(1.6%)	8K(88%) 64K(7.8%) 1K(1.7%)	0.33%
Th. Weights	T1(19%) T2(11.6%)	T1(19%)T2(11.68%)	0% - 0.05%
Avg Int.Time	4.63ms	4.78ms	3.1%
IOPS	255.14	263.27	3.1%
Avg Latency	8.09ms	8.48ms	4.8%

TABLE I: I/O - Performance Metrics Validation (Messenger)

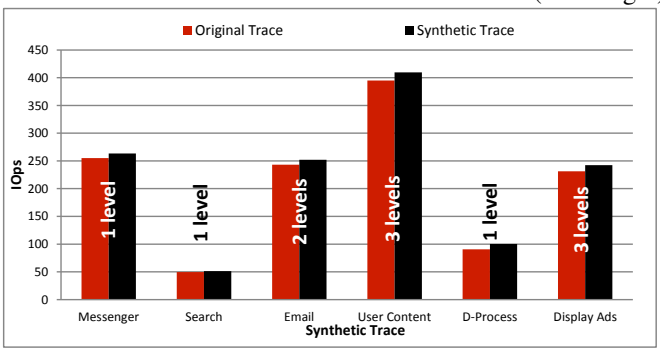

Fig. 2: Comparison between Original and Synthetic Trace

3) The ability to modify the intensity of the generated I/Os through an intensity knob which scales the inter-arrival time of I/O requests. This is especially useful in high performance storage systems (e.g.: Solid State Drives).

B. Tool Validation

For all our experiments we use traces from production servers of six popular large scale DC applications. Messenger, Display Ads and User Content are the SQL portions of an Online Messenger, an Ads Display and a Live Storage application respectively. Email and Search are latency critical online services and D-Process is a distributed computing application that resembles Map-Reduce [7]. These applications cover the majority of large-scale workloads in modern DCs.

The first step in order to create the synthetic workloads, is collecting 24-hour long traces from production servers, hosting these applications. From these traces we create models of configurable detail which are then given as input to the tool.

Validating the accuracy of the model and the tool is necessary in order to ensure that original and synthetic workloads are similar in their storage activity. Furthermore, since we adopt an open-arrival approach (we do not guarantee a specific number of outstanding I/Os as in a closed system) I/O fidelity is not trivial. The process we are performing is as follows:

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

For this part of our experiments we use an SQL-provisioned server with 8 cores, 10 disk partitions and a total of 2.3TB of HDD storage. For each application we evaluate the similarities in the features of I/O requests (block size, rd/wr, rnd/seq, inter-arrival time and thread weight) and the performance metrics (throughput and latency). Table 1 shows this comparison between original and synthetic workload for Messenger. The results are similar for the other applications. For all metrics the deviation between original and synthetic load is less than 4.8%. 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 calculate the variance between different runs of the same synthetic workload and guarantee a difference in throughput **less than 1%** in all cases. For each application we choose an optimal number of levels which is the first for which the performance metrics stabilize (less than 2% difference in IOPS). This way, we convey the best accuracy with the least necessary model complexity.

In order to demonstrate the merit of this methodology we perform a comparison with the most well-known workload generator (IOMeter). We verify that for simple tests with no spatial locality notion the two tools behave similarly (less than 3.4% difference in throughput), while in studies where spatial locality is critical, the results are fundamentally different.

C. Use Cases

One of the main benefits from using this toolset is the opportunities it offers in evaluating storage studies that would otherwise require access to the application code or full application deployment. For this work, we evaluated two possible use cases for the tool, SSD caching and the benefits from defragmentation in a large-scale environment.

Firstly, we explore the applicability of SSD caching in DC workloads. Using our tool, we show that for some applications, SSD caching offers significant storage system speedup without application change (18% on average for a 32GB SSD cache). In the second use case, we motivate the need for defragmentation in the DC. User data gets accumulated over a period of time and files get highly fragmented. Using tracing information, we rearrange blocks on disk to improve the application's sequential characteristics. Using the tool shows that defragmentation offers a significant speedup (14% on average), in some cases greater than incorporating SSDs.

Part of our future work includes evaluating the energy efficiency of these use cases, as well as extending a similar modeling methodology to encompass other parts of the system, towards a complete DC workload model.

REFERENCES

[1] S. Sankar, K. Vaid, "Storage characterization for unstructured data in online services applications". In IISWC, 2009.

[2] D.Narayanan, E.Thereska, A.Donnelly, S.Elnikety, A.Rowstron, "Migrating enterprise storage to SSDs: analysis of tradeoffs". In EuroSys 2009.

[3] S. Sankar, K. Vaid. "Addressing the stranded power problem in datacenters using storage workload characterization". Proceedings of the first WOSP/SIPEW, 2010.

[4] DiskSpd: File/Network I/O using Win32/.NET API's on WindowsXP http://research.microsoft.com/en-us/um/siliconvalley/projects/sequentialio/

[5] IOMeter, performance analysis tool. http://www.iometer.org/.

[6] ETW: Event Tracing for Windows: http://msdn.microsoft.com/en-us/library/bb968803%28VS.85%29.aspx

[7] J. Dean and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". OSDI'04. CA, December, 2004.