Benchmarking Disk-based Archival Storage Tiers using Appropriate Archival Workloads

DongJin Lee   Michael O'Sullivan   Cameron Walker
Department of Engineering Science
The University of Auckland
Auckland, New Zealand
{dongjin.lee, michael.osullivan, cameron.walker}@auckland.ac.nz

ABSTRACT

This paper presents benchmark experiments for designing an optimal archival storage system. The benchmark utilizes archival workloads developed from an analysis of historical file size distributions. The workloads provide more appropriate measurements of system performance *as an archive* than traditional approaches. We use these benchmarks to measure disk-based archival systems.

We then consider designing an archival system based on our benchmark measurements and produce a low cost design for a commodity disk-based archival storage system. Combining results of predictions along with our optimization-driven design, we discover an ideal building block for an archival storage system.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems—*Design studies, Measurement techniques*

General Terms

Design, Measurement, Performance

Keywords

Archival, Benchmark, Disk, File Size, Storage, Workload

1. INTRODUCTION

Information is becoming more valuable and access to that data is critical in many organizations. Modern hierarchical storage management (HSM) maintains high capacity by using different tiers of storage to provide fast access to important, frequently accessed data at high cost and slower access to less important, infrequently accessed data at low cost. A widely used archive storage system is made up of tape-based disks and provides high capacity, but it lacks arbitrary retrieval performance. Disk-based storage systems have recently received increasing interest due to their decreasing cost and high performance, but benchmarking with an appropriate archival workload and building optimal systems (e.g., number of disks, nodes, etc.) has not been studied.

In this paper, we generate file workloads for an archival storage tier which is determined by the size of files in the archive. We use empirically measured file size distributions from large HPC archival sites [1] with which we develop a model for the typical distribution of archival file size. Our generated workload is compared with different kinds of distributions to confirm the robustness of our model, and is then benchmarked against two different sets of servers running on a distributed object-based file system (Ceph [5]). We measured the performance of these servers in several different configurations, e.g., single server with many disks, multiple servers with a few disks each.

2. ARCHIVAL STORAGE WORKLOAD

For online (e.g., *scratch* volume, non-archival) tiers, the requirements prioritize heavy I/O performance, so stress testing by generating a large number of low-level I/O blocks may be appropriate via an I/O work generator tool, e.g., `IOzone`. However, for archival storage, stress test benchmarking is not appropriate. The results from the block-based benchmarks are unrealistic as they do not generate files or file sizes that correspond to a typical archival workload.

The archival process periodically selects a list of files (using selection policies such as by last-accessed time) and aggregates them into an ordered batch. These batches are then written sequentially to the archive volume (data migration), we call this operation *sequential-write*. The process also spends a significant amount of time reading files from the volume (data retrieval). Usually this operation reads files or fragments of files from multiple locations within the archive in an arbitrary pattern, we call this operation *random-read*. The I/O performance varies with different OS, file systems and disks themselves, but the critical factors affecting the time required for the access-patterns (sequential-write or random-read) are the number of files and the size of each file. Hence, in order to ensure that our storage design and its performance metrics are based on the typical archival workload, we obtained archival datasets and statistically modeled their file size distributions.

Figure 1 shows the fit using Gamma and Generalized Gamma distribution functions which also account for possible variations by incorporating confidence interval envelopes, so as add robustness of each fits with a possible lower-bound (more large files) and an upper-bound (more small files); see [3] which details our methodologies with performance comparisons amongst the models. We then generate workload from the fitted models (fileset distributions with varying capacity utilization) and benchmark a disk-based storage system configured with high- and low- performance machines. Here we show results using an Intel Atom `D525`; other setups are discussed in the full paper [2].

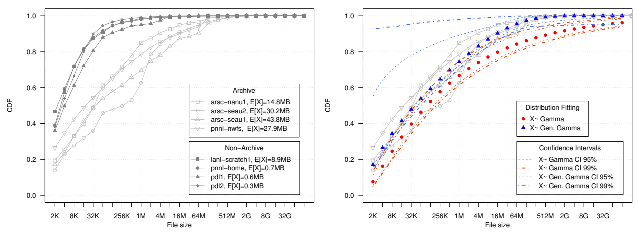

Figure 1: File size distributions with CI envelopes

3. DESIGN EXPERIMENTS AND RESULTS

Single node: We performed the experiments on a single machine with the following setups.

Changing the number of disks on the node – The sequential-write performance decreases with increasing capacity utilization. This is due to a larger proportion of the disk platter being used and the decrease is more pronounced with more disks. The decrease is also exacerbated for multiple disks from the disk management overhead.

Changing the RAM on the node – We experimented with different levels of RAM and found that more RAM helps for random-reads (there was no noticeable effect for sequential-write), but only when the capacity utilization is small. Some of the node's RAM is used as cache for the OS and some for the Object Storage Disk, i.e., disk, so this assists performance when the volume of reads is low. However, once the total volume of reads is at any reasonable level the cache has little effect. Additionally, in a real-world storage archive, we would not expect files to be re-read soon after they are written or read, so caching would not help.

Changing the controller – We experimented with the controller by connecting the disks via a RAID controller instead of the built-in motherboard SATA controller and SATA extenders. The effect of the RAID controller was similar to the effect of extra RAM, i.e., it improved performance for low I/O volumes. This was due to the cache on the RAID controller acting in a similar manner to the RAM cache for the OSD. However, when used in write-through mode (i.e., bypassing the controller's cache) we observed an overall improvement in sequential-write performance. We expect this is because the RAID controller's hardware deals with writing the data instead of the node's CPU, hence the improvement.

Multiple nodes: We performed the experiments on different configurations by changing both the number of nodes and the number of disks per node. We had 6 nodes and 12 disks, so we tested (number of nodes, disks per node) configurations, e.g., (2, 1), (2, 2), . . . , (6, 1), (6, 2). Figure 2 shows the sequential-write performance of *per node* and *multiple disks* for these configurations.

We observe the performance per disk drops (when the number of nodes is kept constant) due to the overhead of managing more disks. This is also shown by the form of the total sequential-write performance, i.e., tailing off to a constant value. Furthermore, the performance drops as the number of disks per node is kept constant and the number of nodes increases. This is expected as the total number of disks is increasing, thus increasing management overhead. Note that both plots illustrate different increase functions at different capacity utilizations; less capacity utilization clearly has higher per-disk increase values. We expect increasing the number of disks much above 6 nodes would lead to a performance decrease as the overhead of disk management on the node outweighs the benefit of more disks.

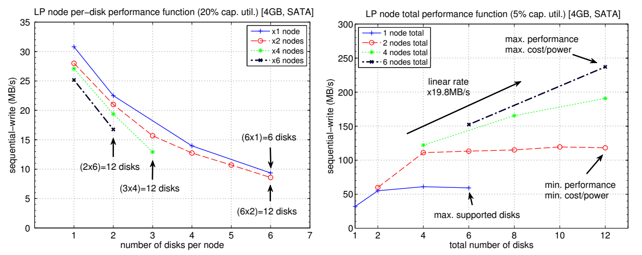

Figure 2: Performance function of multiple nodes

Summarizing our observations, we find the following behavior. First, the highest total performance is obtained with multiple nodes (6 nodes, 2 disks each), but this configuration also incurs the highest cost and power consumption. This configuration has the lowest per-disk performance. Second, by contrast, the highest per-disk performance is obtained with the least nodes (2 nodes, 6 disks each), but the total performance is low due to the limited number of nodes. However, the smaller number of nodes leads to low cost and less power consumption. Lastly, there are configurations that balance performance and cost/power consumption. For example, nodes with (4 nodes, 3 disks each) configuration get reasonable performance, cost and power consumption.

The measurements were then integrated with vendor specifications and combined into an optimization-based automatic design tool that produced the best archival storage tier design. We have discovered an ideal building block for an archival storage system, suggesting a combination of the various high- and low-performance servers, and median-range servers. Nevertheless, the best configuration to use depends on the priorities of the storage architect; see [4] for details of the optimization-based automatic design tool.

4. CONCLUSION

In this paper, we have developed a file size distribution workload benchmark that models the typical workload experienced by an archival storage tier. This benchmark is then thoroughly tested on multiple nodes of different configurations. Lastly, our measurements are then utilized for an optimization-driven design so as to discover an ideal building block for an archival storage system; the resultant design suggested a combination of the measured servers might be appropriate. Combining both our measurements and resultant predictions enables us to determine properties of a component that is beneficial for the archival storage tier design, even if such a component does not currently exist (see [4]).

5. REFERENCES

[1] S. Dayal. Characterizing hec storage systems at rest. In *Technical Report, CMU-PDL-08-109*. Carnegie Mellon University Parallel Data Lab, July 2008.

[2] D. Lee, M. O'Sullivan, and C. Walker. Benchmarking disk-based archival storage tiers using appropriate archival workloads. *SIGMETRICS Performance Evaluation Review*, 40(2), 2012.

[3] D. Lee, M. O'Sullivan, C. Walker, and M. Mackenzie. Robust benchmarking for archival storage tiers. In *PDSW*, 2011.

[4] M. O'Sullivan, C. Walker, and D. Lee. Designing data storage tier using integer programing. In *SAC*, 2012.

[5] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn. Ceph: A scalable, high-performance distributed file system. In *OSDI*, pages 307–320, 2006.