OSDsim - a Simulation and Design Platform of an Object-based Storage Device

WeiYa Xi Wei-Khing For DongHong Wang Renuga Kanagavelu Wai-Kit Goh
Data Storage Institute, Singapore
xi_weiya@dsi.a-star.edu.sg

Abstract

Modeling and simulation is an effective way to design and evaluate the performance of a network storage system. OSDsim is designed and developed for simulation of an Object-based Storage Device (OSD) system. An OSD File System (OSDFS) has been designed and developed for OSDsim. It can provide high throughput for big object requests and also maintain high disk utilization for small object requests. The OSDFS can be configured to suit different workload patterns. The disk module designed and developed for OSDsim makes use of the dynamic disk profile extracted from the actual disk drive through interrogative and empirical methods. Therefore disk drive simulation should be accurate. OSDsim has been validated and the write error is within 5%.

1. Introduction

OSD defines a new object-based interface for a storage device. This new approach for data storage is currently the subject of much intensive research and development, both in industry as well as in academia. Some significant initiatives include the work of the Storage Network Industry Association (SNIA)'s OSD Technical Work Group (TWG) [5] to draft industry standards and that of the ANSI T10/OSD Committee to facilitate interoperable solutions [6]. The Network Attached Secure Disk [2] project at the Parallel Data Lab at Carnegie Mellon University enables direct data transfers between clients and online storages without invoking the file server in common data access operations.

In an OSD system, file system functionalities are divided into two logical components: a file manager and a storage manager. A file manager takes care of tasks including data hierarchy, naming, file to object mapping, access control *ect.*. While the storage manager is in charge of actual data retrieving from disk drive. The storage manager component is located on an Object-based Storage Target (OST) together with disk drive or other storage media. A new and efficient file system which can provide high throughput and maintain high disk utilization needs to be designed for an OSD system.

In this paper we present the OSDsim, especially designed and developed for simulation of an OSD system. The file system designed can provide high throughput for large object requests and still maintain high disk utilization for small object requests. In the following sections, the OSDsim, including all modules for the simulator, is first introduced. Detail descriptions are provided for the sub-modules of the OSDFS storage component and the disk drive and its parameters extraction. After that is experimental work and validation of the OSDsim. Details are provided on the performance of OSDsim in the validation test using actual experimental data to compare with the OSDsim's simulation results. Finally the results of further simulation and analysis are presented to compare the performance of different region settings for OSDFS.

2. OSDsim

OSDsim comprises four main modules. They are the Object-based Storage Client (OSC) module, Object-based Storage Target (OST) module, Meta-Data Server (MDS) module and Network module. Figure 1 shows the OSDsim module structure with sub-modules.

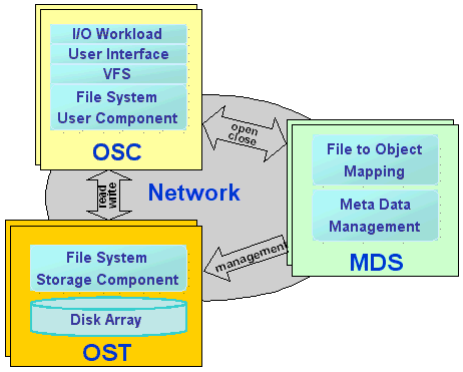

Figure 1. The OSDsim structure and modules

2.1. OSC Module

The OSC module consists of four main sub-modules. These are the I/O workload module, user interface, Virtual File System (VFS), and OSDFS user component.

1. *The I/O workload sub-module:* This sub-module generates synthetic traces for the system simulation. It also provides the interface for taking in external traces.

2. *User interface:* The user interface interprets user commands such as MOUNT, MKDIR, RMDIR, WRITE, DELETE, MV, CP and READ.

3. *VFS:* This sub-module implements the directory lookup, file look up and memory management for the system.

4. *OSDFS user component:* Some main functions implemented in this sub-module includes converting the user interface commands into OSD SCSI commands, and keeping the file structure hierarchy.

2.2. MDS Module

There are two main sub-modules in the MDS module. One is for file to object mapping and the other is for meta data management.

1. *File to object mapping:* Currently one file to one object mapping has been implemented. A unique object id is also assigned to each object.

2. *Meta data management:* This sub-module is responsible for managing and retrieving object meta data information. B tree is used to store the object meta data information. Each node of the B tree contains the information of User_object_id, Group_object_id, and inode information.

2.3. Network Module

OSDsim is actually the enhanced and extended version of SANSim [13] which simulate in detail Fibre Channel (FC) to frame level. The network module for OSDsim adopts the FC module in the SANSim

2.4. OST Module

The OST module includes two main sub-modules: a OSDFS storage component and the disk drive.

2.5. OSDFS Storage Component Sub-module

The OSDFS storage component sub-module located on the top of the disk drive is responsible for data allocation and retrieval. Four commands: FORMAT, DELETE, READ, and WRITE have been implemented.

Figure 2 shows the design structure of this OSDFS storage component. The whole disk is divided into equally sized regions with each region consisting of four parts: a region head which stores regions information, a free onode map which records free onode space, an onode table which keeps the information on each onode meta data and the area for data blocks which is the location for actual object data.

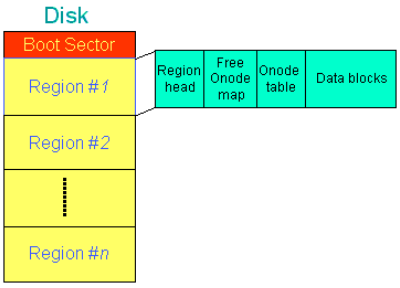

Figure 2. Design structure of the OSDFS storage component

There are three main objectives in the designing of the OSDFS storage component. The firstly is that it should be efficient to search for continuous free space. Secondly it should provide high throughput for large object data and lastly it should be able to maintain good disk utilization for small object data.

An efficient region management scheme based on extent was adopted. With this the searching time for continuous free space is shorter than bitmap searching used by general file systems. This is one of the reasons why our file system can outperform the general file system like ext2[7] and ext3[4]. The performance comparisons for file systems can be found in the paper which is also submitted for MSST2006 titled *"Adaptive Extents-Based File System for Object-Based Storage Devices"*. Variable-sized blocks are used for data allocation and object data are grouped according to their sizes. Large object data can be allocated to big regions and small object data allocated to small regions. Therefore, OSDFS can provide high throughput for large object data and still maintain high disk utilization when there are a lot of small object data.

OSDFS also modifies the onode write scheme from a general purpose file system as to improve write performance. In a general file system an onode (meta data of an object data on disk) writing normally takes place right after each object data writing. This scheme is not efficient as

the onode table and data blocks are in different part of the regions. OSDFS adopts the scheme of writing N ($N > 1$) onodes together after writing N objects data. The write performance is improved through a reduction of by seeking distances.

Read performance can be improved by computing the data location for non-fragmented objects instead of requiring the retrieval of data location from the disk . Figure 3a shows how the system computes the location of the data blocks for incoming object requests for the non-fragmented object data. For the object requests with data fragmentation, Figure 3b shows that the onode data location information has first to be read from the onode table on disk and then directed to data blocks.

2.5.1. Disk Drive Sub-module OSDsim needs a disk drive simulation module which can simulate disk drive activities accurately. The simulated disk drive must be a SCSI disk drive available in the market so that OSDsim can be validated using an actual physical system. Presently, there is no such disk drive simulation module available. Some storage system simulators come with a very detailed disk drive module, for example DiskSim [3]. However many of the disk drive parameters are not readily available to the public. Furthermore, many of the simulators use only very simple disk drive modeling which then affect the accuracy of the simulation results. They compute the disk performance based only on average values such as average seeking times and rotational delays. This is not good enough for our requirements for the OSDsim.

We obtained disk drive parameters using two methods: interrogative method and empirical method [12] [11]. We modify the SCSIBench [8] to suit OSDsim. We are able to obtain parameters such as cache size, zone information, cylinder and track skew, LBN to PBN mapping defect regions and actual seeking curve. Such information are gathered and passed to our disk drive module for the disk simulation. Figure 4 shows the seeking profile obtained experimentally for the disk drive Seagate SCSI ST318437LW. Figure 4 is the seeking profile with a full seek. Figure 5 is the expanded view for the first 5000 cylinders. From Figure 4 we can see that there are four points which are obviously out of the normal seek profile. For such obviously erroneous points, we use the previous seeking times instead.

The disk drive sub-module consists of four main components: bus interface, cache/scheduler, command processing and the component consisting of mechanics, data transfer and data mapping as shown in Figure 6. As the disk drive module makes use of actual disk parameters, the simulation of the disk should give better accuracy.

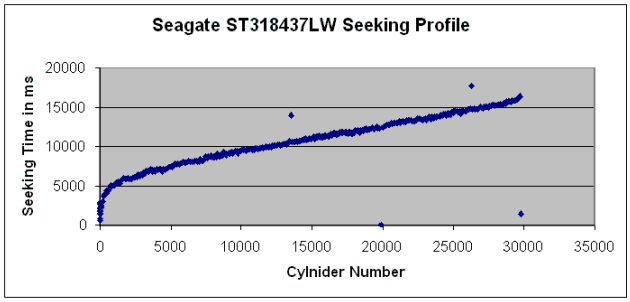

Figure 4. Seeking profile with a full seek

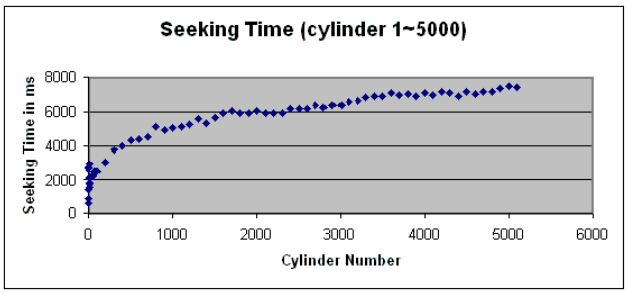

Figure 5. Seeking profile for the first 5000 cylinders

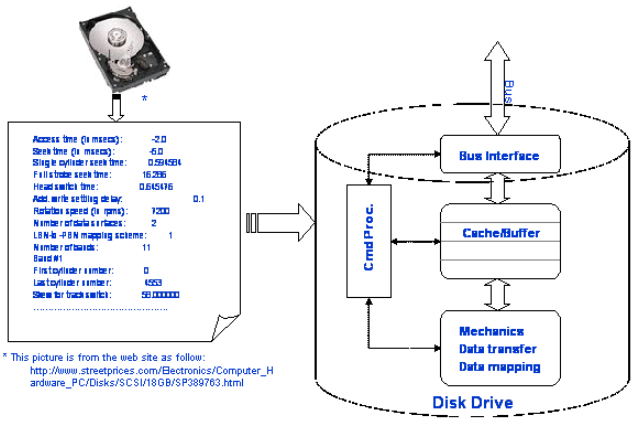

Figure 6. Disk drive module

3. Simulation Validation

3.1. Experiment Set Up

We implemented OSDFS storage component at the user level on a PC with a 1 GHZ PentiumIII CPU and 512MB RAM running Redhat Linux, Kernel 2.4.20. The operating system was installed on a 40GB ATA Maxtor D740X-6L disk drive and the target performance testing was on the Seagate SCSI disk drive model ST318437LW.

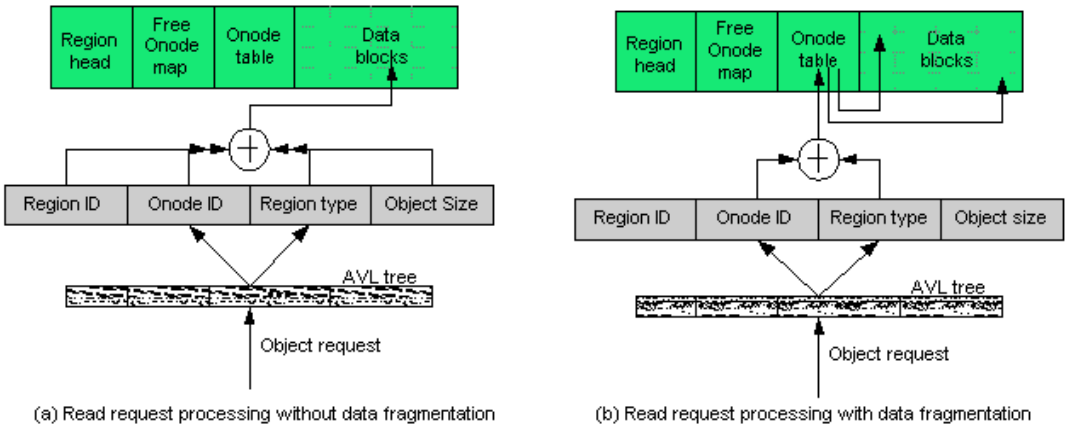

Figure 3. Diagram to illustrate how a read request is processed

3.2. Comparison of Experimental Data and Simulation Data

Both read and write performance for different size ranges from 1KB to 512KB of the object requests were tested on the system. The system was made to write 5000 object requests first and then to read these back. The performance in terms of throughput were recorded from the system and computed. There were then compared with the simulation results from OSDsim by applying the same requests. Figure 7 & 8 shows the performance comparisons for both experimental and simulation results.

Figure 7 is the performance for write requests comparison for both experimental and simulation. The biggest error is about 4.7% for the write performance when request size is 64KB.

Figure 8 is the performance for read requests for both experimental and simulation. The largest error, of about 12.7%, at the point of request size 512KB.

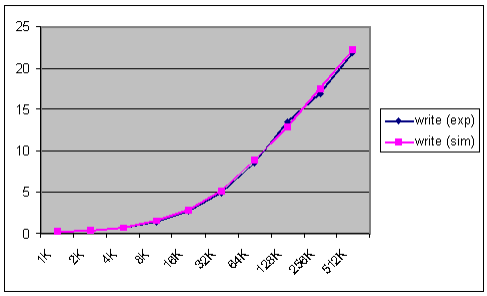

Figure 7. Write performance

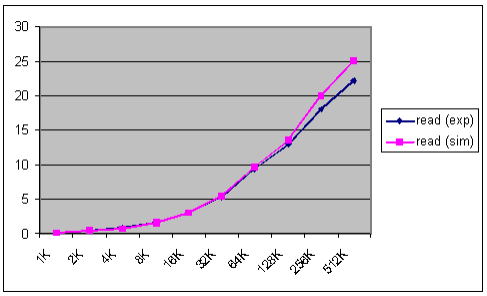

Figure 8. Read performance

4. Simulation and Analysis

4.1. Simulation Environment

Simulation and analysis were also performed with the OSDFS storage component configured with different types of region settings. The disk drive used for this is a Seagate SCSI disk drive, model ST31843LW. The simulation is based on closed-loop [9] which means that a subsequent request is issued only when the previous request complete its execution.

The assumptions made for the simulations are as follow:

1. The OSDFS file to object mapping is one to one.
2. System cache is large enough for the boot sections, the region head as well as the free onode map.
3. The write performance is only measured for the data writing on the disk in *Step 3* in the following simulation steps.
4. Read performance is for the reading of all the data written in *Step 3* in the following simulation steps.

The following steps were employed in the simulation procedure:

Step 1: Format the disk.

Step 2: Create an aged file system by performing data writing and deletion. Data with object size close to size of data unit are continuously written to the disk until the disk is full. Deletion requests are then performed with every other object delete. As an result, the free space left on the disk are not continuous and the largest will be equal to region data unit size

Step 3: Perform write requests on the disk again. The workload size distribution follows the pattern summarized from the actual trace obtained from HP-UX machines. These consist of twenty machines located in laboratories for undergraduate classes [10]. This workload is referred as Instructional workload (INS) and the workload size distribution is shown in Figure 9. The write performance is computed based on the write requests performed in this step.

Step 4: Perform read requests on the disk. The read requests read out all the data written in *Step 3*.

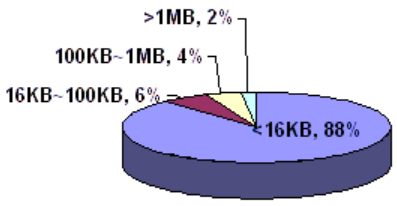

Figure 9. Workload size distribution

4.2. Simulation Results and Analysis

The performance of OSDFS was tested for the five different types of region settings shown in Table 1. Figure 10 shows the read and write performances in terms of throughput for various type of region settings.

Table 1. Region Settings

No. of regions	Size of data unit (KB)
1	4
2	4, 128
3	4, 128, 512
4	4, 128, 512, 2048
5	4, 128, 512, 2048, 8196

We can see from Figure 10 that for all the different types of region settings, the read performances are better than the write performances. The larger the number of regions used for the OSDFS, the better the performances.

Figure 11 is the overall disk space utilizations for various region settings. We can see that the larger the unit size of a region, the smaller the disk space utilization.

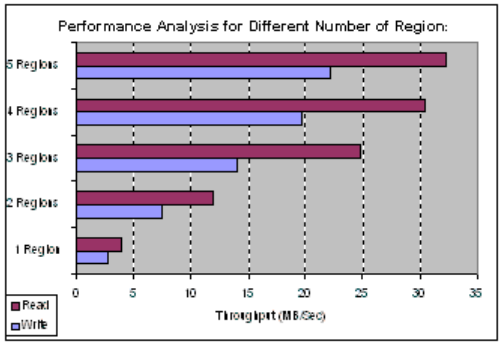

Figure 10. Performance analysis for different number of regions

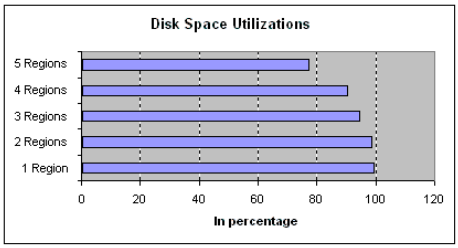

Figure 11. Disk space utilizations for different regions settings

Figure 12 shows the average number of fragmentation per object. The number is computed base on data written in *Step 3* in the simulation procedure. We can see from this figure that with five regions, this value is the smallest while with only one regions this value is the largest. This is the main reason why the performance for a setting of five region is better than that with only one region under the workload of INS.

5. Summary and Related Work

So far we have not seen much work done on OSD simulation. The only work we are aware of was done in IBM [1] in which an IBM Object Storage Device for Linux was developed. The simulator does not design/develop an ob-

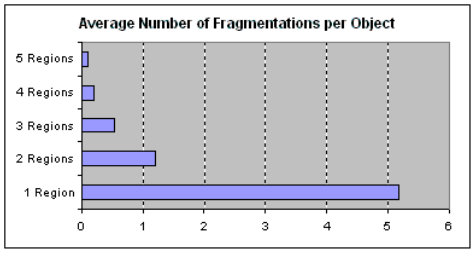

Figure 12. . Average number of fragmentations per object

ject file system. It only uses the local file system to store object data.

In our work, we have designed and developed an OSD simulator for an OSD system. The OSDFS storage component developed for the simulator is capable of providing high throughput for large object requests while, at the same time, maintaining high disk utilization for small object requests. The disk drive module developed for the simulator simulates not only critical mechanical dynamics but also the cache/buffer, spare regions and other activities. Actual parameters which define important characteristics are extracted from actual disk drives and used in the simulations to ensure that simulation errors remain very small.

Validation experiments with the actual system implemented with OSDFS at the user lever and installed with an actual physical disk drive have been conducted to compare the simulation results with measured performance data obtained from physical system. These show OSDsim to perform well with simulation results for write performance to be within 5% and read performance within 13% of actual measured data.

References

[1] http://www.haifa.il.ibm.com/projects/storage/objectstore/osdlinux.html.

[2] http://www.pdl.cmu.edu.

[3] http://www.pdl.cmu.edu/disksim/.

[4] http://www.redhat.com/support/wpapers/redhat/ext3/.

[5] http://www.snia.org/tech_activities/workgroups/osd/.

[6] www.t10.org.

[7] R. Card, T. Ts'o, and S. Tweedie. Design and implementation of the second extended filesystem. *In Proceedings of the First Dutch International Symposium on Linux*, 1994.

[8] Z. Dimitrijevi, R. Rangaswami, D. Watson, and A. Acharya. Diskbench: User-level disk feature extraction tool. 2002.

[9] G. R. Ganger. System-oriented evaluation of i/o subsystem performance. *PhD dissertation*, 1995.

[10] D. Roselli, J. Lorch, and T. Anderson. A comparison of file system workloads. *In Proceedings of the 2000 USENIX Technical Conference*, pages pp. 41–54, 2000.

[11] N. Talagala, R. Arpaci-Dusseau, and D. Patterson. Microbenchmark-based extraction of local and global disk characteristics. *Technical Report CSD-99-1063, University of California, Berkeley*, 1999.

[12] B. L. Worthington, G. Ganger, Y. N. Patt, and J. Wilkes. On-line extraction of scsi disk drive parameters. *In proceedings of the ACM Sigmetrics*, pages pp. 146–156, 1995.

[13] Y.-L. Zhu, C.-Y. Wang, W.-Y. Xi, and F. Zhu. Sansim - a simulation and design platform of storage area network. *In Proceedings of 21st IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies*, 2004.