PULSTORE*: Automated Storage Management with QoS Guarantee in Large-scale Virtualized Storage Systems

Lin Qiao † Balakrishna R. Iyer § Divyakant Agrawal † Amr El Abbadi † Sandeep Uttamchandani *

§ IBM Silicon Valley Lab
balaiyer@us.ibm.com

† University of California, Santa Barbara
{lqiao, agrawal, amr}@cs.ucsb.edu

* IBM Almaden Research Center
sandeepu@us.ibm.com

1. Large-scale Virtualized Storage System

Traditionally storage has been purchased and attached to a single computer system. Such storage is accessible only through the computer system to which it is locally attached. In the last 10 years, especially in corporate data centers, storage is being increasingly purchased independent of the processors, and independently managed and administered. Because of the standardization of disk IO protocols, storage can be easily shared amongst various heterogeneous processors running various applications. The shared storage is accessed over a network interconnecting the processors to the shared disk subsystem, known as the *storage area network* - a network on which processors send IO calls to virtual disks. It is the task of the storage controller to manage the mapping of virtual disks to physical disks, a task known as *storage virtualization*, similar to memory virtualization of processors. The storage virtualization layer has been exploited to provide diverse storage functions.

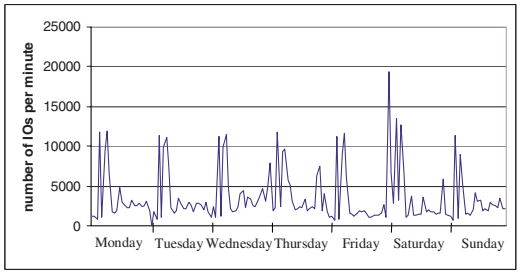

Figure 1: IO Workload from Cello92 Trace

The storage virtualization layer may be used not only statically but also dynamically to change the mapping of virtual disks to physical disks. This can be exploited to dynamically change the physical disk configuration. Many IO workloads exhibit cyclic behaviors, and alternate between bursts of high activity and periods of low activity. A significant number of practical instances show that the IO workload lends itself to be predictable. For example, Figure 1 shows the IO workload for a week extracted from HP Cello92 IO traces [8] collected from an HP-UX file server. It is highly repetitive. It is also observed that there is an order of magnitude difference between the peak and average IO rates. Configuring the IO sub-system for the peak leads to over-provisioning and waste. In a virtualized storage management system, it is possible to provision on demand, especially when the storage is shared amongst multiple applications, each of which would have peak requirements at different times.

2. The Challenges

If a reasonable prediction of IO workload can be made, the storage virtualization layer could optimize the mapping of physical disks to virtual disks to satisfy applications' IO response time requirements. This problem is usually referred to as *on-demand utility provisioning* [7]. This is analogous to the problem of processor allocation to shared concurrent workloads. The difference is that disks contain data. Hence if we change the number of disks supporting a specific collection of data we also need to redistribute the data. This task involves data movement, which generates even more IO. Under some circumstances, it may not always be advisable to change the number of disks. If the number of disks are to be changed, then the change needs to be done in advance. When must it be done? These are questions that will be answered in this paper.

In this paper, we tackle the problems of moving data in a storage hierarchy under both capacity/performance constraints and on-demand resource provisioning constraints. The challenges are two folds. First, for a single data movement action, as it interacts with the applications, we need to limit its impact on application performance. In particular, we need to control the invocation time of a data movement such that there is no performance constraint violation during and after the data movement. Second, as workload varies over time, it is likely that the previous storage

* A pulsating star dims and brightens as its surface expands and contracts.

Proceedings of the Second International Conference on Autonomic Computing (ICAC'05)
0-7695-2276-9/05 $ 20.00 **IEEE**

provisioning action may have been either too small or too large, resulting in either capacity/performance constraint violation (under-provisioning) or wasteful storage configuration (over-provisioning). Hence it is important to dynamically generate a sequence of carefully tuned data movement actions in order to adapt to the changing workload.

3. The System Model

Our study is based on a hierarchical storage structure, storage QoS goals, an online data migration model, and a storage utility cost model.

Many storage systems are constructed using a hierarchical model. Moving upward through the hierarchy, each layer provides faster access speed but is more expensive. Such a storage hierarchy scheme provides the tradeoffs between access speed and cost through data movements.

The storage level quality-of-service (QoSS) specifies the IO performance for a particular storage object that must be guaranteed under any workload. QoSS goals are usually associated with logical disks [5]. In the performance prediction models [1, 6, 4], the IO latency can be represented as a function of the workload and the logical disk configuration. Since these two parameters are a function of time, latency can simply be predicted as a function of time, denoted as $L(t)$. The QoSS goal is expressed as a bounded latency: i.e., $L(t) \leq L_{QoS}$ at any time t.

Data migration is initiated when the storage system experiences performance degradation or anticipates disk failure. Furthermore, data migration needs to be *online*, i.e. no application should be interrupted during data migration. We study two aspects which directly impact the performance of a striping storage system. We can either increase the physical disk speed or the striping width to decrease the IO latency, or vice versa to increase the IO latency.

The storage cost over time for a particular logical disk is called the *resource utility cost*, or $\mathcal{U}$. It is the summation of the cost of the provisioned storage system, C_t, at each time t over the whole time period: $\mathcal{U} = \sum_t C_t$. Our goal is to minimize $\mathcal{U}$ while guaranteeing that no QoSS violation occurs.

4. The System Design

We present a design of a system, named PULSTORE, whose goal is to produce a migration sequence so that the performance outcome satisfies the QoSS specification while minimizing the resource utility cost. In order to address the this problem, PULSTORE first provides solutions for a single migration, and then generates a sequence of migration actions which are either offline optimal or online approximate solution for a given time period. The overview of PULSTORE is summarized as follows.

1. *Safe time zone for a single migration.* An analytical model is developed to determine the safe time zone to start a migration, so that during the span of migration, the application IO performance outcome still satisfies QoSS constraints, even with the extra migration IOs. The solution is based on the monotonic property of the workload fraction remaining on source and destination disks, when the migration invocation time varies.
2. *Optimal migration sequence.* The optimal scheduling scheme provides a sequence of migration invocations $\{(\mathcal{D}_1, \mathcal{I}_1), ..., (\mathcal{D}_n, \mathcal{I}_n)\}$, when we assume that all the future workload is completely known. The output of the optimal solution, including the minimal utility cost and the migration sequence.
3. *Approximate migration sequence.* In order to reduce the computational cost, we propose an approximate algorithm to handle the workload with a fixed size prediction window, and generate a migration sequence. The algorithm delays up-migration as late as possible and executes down-migration as early as possible to minimize the utility cost. Moreover, it checks the prediction window to resolve conflicts between migrations. The migration sequence is adjusted accordingly when the prediction window moves forward.

Acknowledgement The work was supported in part by NSF awards CNF-04-23336, IIS-02-20152 and EIA-00-80134. Lin Qiao was also supported in part by IBM Corporative Fellowship.

References

[1] S. Chen and D. Towsley. A performance evaluation of RAID architectures. *IEEE Transactions on Computers*, 45(10):1116–1130, 1996.

[2] E. I. Cohen, G. M. King, and J. T. Brady. Storage Hierarchies . *IBM System Journal*, 28(1):62–76, 1989.

[3] IBM Corp. *AIX Logical Volume Manager from A to Z: Introduction and Concepts*. IBM Redbooks, 2000.

[4] E. K. Lee and R. H. Katz. An Analytic Performance Model of Disk Arrays. In *Proceedings of the 1993 ACM SIGMETRICS*, pages 98–109, 1993.

[5] C. Lu, G. A. Alvarez, and J. Wilkes. Aqueduct: Online data migration with performance guarantees. In *USENIX Conference on File and Storage Technologies*, pages 219–230, 2002.

[6] O. I. Pentakalos, D. A. Menasce, M. Halem, and Y. Yesha. Analytical performance modeling of hierarchical mass storage systems. *IEEE Transactions on Computers*, 46(10):1103–1118, 1997.

[7] J. W. Ross and G. Westerman. Preparing for utility computing: The role of IT architecture and relationship management. *IBM System Journal*, 43(1):5–19, 2004.

[8] C. Ruemmler and J. Wilkes. A trace-driven analysis of disk working set sizes, HP Laboratories Technical Report HPL-OSR-93-23, April 1993.

[9] P. Scheuermann, G. Weikum, and P. Zabback. Data Partitioning and Load Balancing in Parallel Disk Systems. *VLDB Journal: Very Large Data Bases*, 7(1):48–66, 1998.