Italian for Beginners: The Next Steps for SLO-Based Management*

Lakshmi N. Bairavasundaram, Gokul Soundararajan, Vipul Mathur
Kaladhar Voruganti, Steven Kleiman
NetApp, Inc.

Abstract

Literature is rife with compelling ideas on simplifying storage management using service-level objectives (SLOs). However, very few of these ideas have actually been productized, despite the fact that many of the original ideas came from industry and were developed more than a decade ago. While many good research ideas do not become products, in this case, we believe that there are important reasons why adoption has been slow. This paper investigates the reasons for slow adoption and discusses ideas that can truly simplify storage management using SLOs.

1 Introduction

Consolidating and virtualizing hardware resources has been the mantra for constructing cloud services, both public and private [3, 4]. The benefits of consolidation in large data centers include reduced costs of hardware, power, and cooling. In such a multiplexed environment, applications are in competition for data-center resources, such as, CPU, memory, and I/O bandwidth. At the same time, the cloud service is still required to meet per-application goals for performance, data protection, etc. Recent research [19] has extended and explored automated management using service-level objectives (SLOs) [17, 30, 36] to meet application goals in the cloud while keeping management costs low.

SLOs are a specification of an application's requirements, primarily in technology-independent terms. The term SLO may refer to application needs at different levels of the software stack; we focus on storage. To satisfy business needs, an application may specify performance (e.g., average I/O latency), capacity, reliability, and security needs for its data. SLOs have also been referred to as "service-level requirements" or "service-level agreements" or "quality of service" (ignoring some differences). With SLOs, administrators can focus more on *what* they need from storage than on *how* it is achieved.

SLO-based management is attractive in principle and a series of research efforts, many of them from industry [17, 36], have developed various techniques that cover the entire spectrum of activities: monitoring workloads and systems, analyzing the interaction of workloads with configurations and with each other, planning remedial actions when SLOs are not met, and executing the actions in an efficient fashion. However, very few of these ideas have been productized, especially in the context of storage systems. Unless we first address productization, we may not be solving the real problems, whether for the cloud or elsewhere.

In this paper, we analyze why SLO-based management, despite having a compelling vision, suffers from poor adoption in products and what we can do to better enable such adoption. Our analysis shows that (a) SLOs are not as simple to specify as we would like, (b) system models, which proactively assess system behavior, have considerable error relative to manual approaches, and (c) the cost of remedying a modeling error is too high.

In order to address these issues, we offer the following research directions:

- We should focus on process, not product; processes that limit the scope of SLO-based management to a pre-defined, well-known set of SLOs ("qualified SLOs") will greatly improve (i) the ability of users to select SLOs, (ii) the accuracy of models, and (iii) the ability of vendors to support systems.
- We should leverage the similarity of workload instances across the data center or even the entire customer base to gather data on workloads and continually update system models and support workflows.
- We should develop lightweight *dynamic reconfiguration* techniques to mitigate the cost of modeling errors.

*The title is a nod to research by Wilkes et al. on SLO-based management for storage systems, summarized as "Traveling to Rome" [36]. Our paper is about the important next steps.

SLO-Based Management	Research	Products
SLO SPECIFICATION: *Simple-to-use, technology-agnostic representation; attributes form one atomic unit.*		
Capacity	GBs of user data (deduplication [38] not handled), quotas	Typically, GBs of actual storage and quotas
Performance	latency, IOPs, bandwidth, shares, I/O priorities	Primarily, shares and I/O priorities; latency and IOPs in some cases
Reliability and availability	9s of availability, recovery point and time objectives, cost of data loss or unavailability [16]	Only technology-dependent: RAID level, backup and mirroring relationships
Security and compliance	Encryption, retention, secure deletion, audit	Encryption, retention, secure deletion, audit
Multidimensional specification	Combined performance and reliability [30], language for all attributes [36]	Attributes are independently specified
MONITORING AND REPORTING: *End-to-end (app, hypervisor, network, storage) monitoring, statistics reported in SLO terms.*		
	In SLO terms, end-to-end (for performance) [31]	Often, technology-dependent SLO terms, end-to-end (for performance) [1, 7, 15]
IMPACT ANALYSIS: *Analysis for: provisioning a workload or removing a workload on a system; dynamically changing the configuration.*		
Analysis for workload provisioning	Black-box [36, 20], white-box [36] models that analyze impact of adding workloads	Not available
Analysis for dynamic reconfiguration	Not available	Not available
REMEDIAL ACTIONS: *Wide spectrum of remedial actions from low-cost to high-cost; planner for automatically examining the cost of different actions and selecting the appropriate one; cooperation between layers of software stack.*		
Range of actions	Limited range: I/O prioritization [13, 34], cache partitioning [29, 33], coarse-grained data migration [12], hardware change [2]	Limited range: I/O prioritization [6, 32], cache partitioning [27], coarse-grained data migration, hardware change
Action selection	Selection techniques available [2, 36]	Not available
Cooperative planning	Not available	Not available

Table 1: **Survey:** We reviewed research papers [12, 13, 16, 20, 29, 30, 31, 33, 34, 36] and products [1, 7, 9, 14, 15, 24, 27, 32] to determine the extent to which research has been productized as well as how well they match with an ideal system.

2 Is Adoption Really Slow?

We perform a survey of research papers and products available in the market to evaluate the extent of SLO-based management proposed as research ideas and its adoption in products.

For a long time, the primary method for implementing service levels has been creating a silo for each application (*i.e.*, independent allocation of resources, often entire storage systems) and using different "tiers" of storage depending on the application's needs. While the degree of SLO-based management available in products has increased (in order to handle multi-tenant sharing), many of them have not moved past the tiering approach, and the differences are evident in our comparison.

As shown in Table 1, we compare research systems and products against the ideal case along multiple axes: (i) SLO specification, (ii) monitoring and reporting, (iii) dynamic analysis of a system's ability to support a new workload (beyond simple thresholds), and (iv) techniques to handle SLO violations and ability to compare multiple remedial actions.

We find that while research systems are not quite ideal, products are significantly lacking in SLO-based management features. The crucial differences are seen in the technology-dependent nature of SLO specification for many attributes, the lack of multidimensional SLO specification, the lack of system models for impact analysis, and the lack of an automated planner for selecting remedial actions for SLO violations. Neither research nor products have adequate solutions for end-to-end management (monitoring, reporting, and action coordination) across multiple layers of the software stack.

3 Reasons for Non-Adoption

John Wilkes [36] believes that the greatest barrier to adoption is the developers' ability to convince administrators that the system can be trusted; he suggests that these systems be simple-to-use, more predictable, and open about the decision-making process. While we agree, we believe that there are additional important issues to be addressed in order to improve adoption, and we detail them in this section.

3.1 SLO Specification

While specifying workload requirements are significantly simpler than specifying the storage-system configuration for a workload, many administrators may not know precise workload characteristics and requirements either. Complicating the problem is that the administrator needs to take into account multiple dimensions – that is, understand the impact of performance, protection, security, cost, etc. in a holistic manner – to provide the final specification to the storage system. Thus, the real goal of simplifying storage management is not addressed simply

by introducing SLO-based management. This observation is borne out in I/O prioritization techniques being productized [6, 32]. It is significantly easier for the administrator to say that one workload is more important than the other (or specify a share of the total capacity), than to specify each workload's requirements. However, I/O prioritization may not help in actually satisfying the applications' requirements.

Some prior work has examined ways to simplify SLO specification [36, 30]. The easiest specification is none at all, as exemplified by techniques to iteratively derive these requirements [36] instead of having administrators specify them. However, such an approach works primarily only for performance attributes and can also make support much harder for vendors; imagine a support scenario where the requirements are determined dynamically and SLO violations still occur. With respect to reliability requirements, Keeton et al. [16] suggest a simpler abstraction for users based on the dollar cost of data loss or unavailability (as compared to metrics such as recovery time objectives). However, other service-level attributes do not lend themselves easily to such a model.

3.2 Modeling Errors

Without SLO-based management, users are expected to manually model the impact (performance, etc.) of workload additions or system-configuration changes based on their expertise in examining system utilization levels. With SLO-based management, this burden shifts to vendor's knowledge of internal workings of the system (*i.e.*, white-box models [36]) or to statistical machine-learning techniques (*i.e.*, black-box performance models [20]).

In particular, the modeling errors occur because, for white-box models, the model builders usually trail the product feature release cycle as it takes longer for them to capture the complex interplay between the various features. Similarly, it is not possible to train black-box models for all the permutations and the lack of this training data gives large errors in the extrapolation region. Irrespective of the approach, errors occur.

Modeling errors can also be expected to increase with increasing system complexity. We believe that given lower complexity – one application per system, modeling primarily RAID latency/bandwidth, only a few infrastructure layers – many users in the past may have been better at modeling than computer counterparts (although Alvarez et al. show otherwise in one specific experiment [2]). However, current storage systems are highly complex and modeling the interactions between various sub-systems is non-trivial.

3.3 Reconfiguration Cost

More important than modeling errors per se, is the impact of modeling errors. The penalty for an incorrect model is often an SLO violation that requires heavy-weight corrective actions, such as migrating the dataset or purchasing a new system. Thus, administrators leave systems underutilized to minimize SLO violations; *i.e.*, keeping storage administration simpler by spending more.

In order for administrators to embrace system-managed complexity, the impact of a mistake has to be lowered. This improvement is essential even when models are highly accurate; when workloads are inherently very dynamic, one can provision storage for either the peak load or the average load, but not both, thus causing poor utilization or periodic SLO violations respectively. Thus, a lack of dynamic low-cost techniques to handle modeling errors and also counter the dynamic nature of workloads, results in poor adoption of automated SLO-based management.

4 Why SLOs Now?

This section looks at the recent technical trends that make automated management essential, showing why we should be concerned about poor adoption of SLOs.

4.1 Configuration Complexity

Prior work [36] uses configuration complexity as motivation, focusing on the number and type of disk drives needed to support a given workload. The complexity today is significantly higher due to a plethora of new features like deduplication [38], use of flash memory [22], use of multiple levels of caches [22, 23], striping of data across nodes [28], etc. It is hard to predict properties like performance when different features are combined. To make things worse, system properties change over time forcing administrators to keep up with all of the changes in the products of multiple vendors. This resulting increase in complexity may favor computer-based models; their errors will increase as complexity increases but at a lower rate than manual modeling, thus encouraging adoption.

4.2 Multi-Tenant Sharing

In a multi-tenant environment, applications with potentially different requirements may share the same storage system. The availability of flash, as well as the pervasive use of caches, permit the same system to support different "tiers" of storage depending on requirements [11]; *e.g.*, data with high performance requirements can make greater use of flash than data with lower requirements. The availability of a large number of workloads to provision allows service providers to "thin-provision" resources, assuming that not all workloads will require peak performance at the same time.

While consolidation has great benefits, it makes storage-management difficult. Specifically, with flash

and hard disks in one array, administrators need to control the amount of flash provided to each application. As flash technology changes or application needs change, the flash allocation should be changed as well. Thus, these tasks should be automated using SLOs.

4.3 Scale and Dynamism

In cloud storage environments, two important problems are much worse as compared to that in prior work [36]: scale and dynamism. The increase in the amount of data stored as well as the number of storage devices, objects, users, policies, and sites that need to be managed in an integrated manner make storage management particularly challenging. Further, new applications are provisioned at a fast rate and their requirements may change significantly over time. Therefore, administrators should ideally not be involved in processing every provisioning request or in managing the load on individual systems. Again, these tasks should be automated using SLOs.

5 Research Directions

The lessons of the past give us insights on how to ease the adoption of SLOs.

5.1 Process, Not Product

Much of the focus on SLO-based management has been on developing a better storage-management product. Less time has been devoted to research on improving processes, including requirement specification by users, building and updating of system models by the vendor, and post-sales support for products and SLOs.

We propose that the vendors identify SLOs for popular workloads ahead of time by (a) leveraging the expertise of select customers and partners, (b) developing tools to translate between requirements specified at the application and storage levels, and (c) developing tools to resolve differences in specification derived from various sources. The starting point for this approach would be to obtain information such as application name and its configuration, *e.g.*, Microsoft® Exchange with N mailboxes, as used in best-practices documents [25]. We term SLOs generated thus as "qualified SLOs." Thus, most users would select a qualified SLO based on the application name or type or even the stack being deployed. The SLO that users select is also an atomic unit (*e.g.*, performance is not independent of security); systems should prevent the specification of inconsistent goals [17].

We believe that the process defined above would make SLO specification (selection) easier for users, reduce the burden on the storage vendor in creating models of the storage systems as well as in supporting both the storage systems and their models; there are fewer, but important, workloads to focus on when creating models; and when a support request is made, the support engineers always have prior workload-based training as well as current context (expected SLO, delivered service, expected workload, actual workload) to troubleshoot with.

5.2 Dynamic Low-Impact Reconfiguration

For users to trust automated management, the impact of errors needs to be lowered. These systems should be nimble in handling dynamism of workloads when the original provisioning is aggressive (for non-peak load).

Systems should include low-time and low-cost mechanisms to handle SLO violations, including automatically and non-disruptively reconfiguring resources or migrating small amounts of data based on current application needs and resource usage. The HP AutoRAID system [37] is an early example. Recent efforts have explored dynamic data layouts that take advantage of flash storage [11]. Mechanisms that separate namespace and location [10, 28] enable object-granular migration.

Further research could focus on taking advantage of *hardware fluidity*. Specifically, caching could be managed throughout the storage stack dynamically with techniques for deciding whether to create new caches, grow/shrink existing caches, etc.

5.3 Community Wisdom

While the large scale of operation is typically problematic from a management viewpoint, we believe that it can also be leveraged to reduce the difficulty of management.

First, many instances of the same application may be provisioned in a single data center (especially in a cloud service). A phenomenal amount of data is now available for training system models and for comparing actual workloads against those the models are trained with. Second, since storage systems have various degrees of reporting back to vendors [5], modeling data from the large customer base can be leveraged to improve in-house models further. Third, scalable data analytics is available through new computing paradigms [8]. Combinations of the above techniques have been developed in the context of handling misconfigurations [35] and bugs [18]. We propose that the techniques be extended to improve system models, construct support workflows, and guide customers towards best practices.

5.4 End-to-End Management

This research direction is prompted less by the failings of the past than by the need to avoid emerging pitfalls. Specifically, the number of infrastructure layers has increased significantly: hypervisors, caches, etc. As a result, an application's performance and reliability depend on multiple pieces of infrastructure, managed by different people or tools, making it hard to derive guarantees.

SLO-based management should move towards a more end-to-end approach. First, we need protocols that can be used to specify SLOs throughout the stack and to aggregate information from all monitoring stations. Orchestration tools [7] aid in such monitoring to a limited extent. Second, we need mechanisms to coordinate control. For example, data migration can be performed at different layers – storage-level [26] and hypervisor-level [12] – to handle SLO violations. When these tools work independently, they may expend more resources than needed to address SLO needs; worse, they would lead to many more SLO violations, more useless work, and potentially, unavailability. Thus, without cooperation, even bug-free components can cause the system as a whole to be unstable – an example of emergent "mis"behavior [21].

6 Conclusion

> "Simplicity, simplicity, simplicity! I say, let your affairs be as two or three, and not a hundred or a thousand;" *Henry David Thoreau*

As we research the new directions proposed herein and develop ways to make storage management simpler for users, we should strive to keep it simple enough for ourselves (*i.e.*, the developers). Methods (*e.g.*, building system models) that require constant or hard, manual refinement may break down because development or support for SLOs cannot keep up with products.

Acknowledgements

We would like to thank the anonymous reviewers and our shepherd, Pradeep Padala, for their insightful comments that helped improve the paper. We also thank members of the NetApp Advanced Technology Group (ATG) for their valuable comments and suggestions.

References

[1] Akorri, Inc. *Akorri BalancePoint*. http://akorri.com/products-features.htm.

[2] Alvarez et al. Minerva: An Automated Resource Provisioning Tool for Large-Scale Storage Systems. *ACM TOCS*, 19(4), 2001.

[3] Amazon, Inc. *Amazon Web Services*. http://aws.amazon.com.

[4] Armbrust et al. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, University of California, Berkeley, 2009. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[5] Bairavasundaram et al. An Analysis of Latent Sector Errors in Disk Drives. In *SIGMETRICS'07*.

[6] Bhargava. FlexShare Design and Implementation Guide. Technical report, NetApp, 2006. http://www.netapp.com/us/library/technical-reports/tr-3459.html.

[7] BMC, Inc. *BMC Atrium Orchestrator*. http://documents.bmc.com/products/documents/13/00/101300/101300.pdf.

[8] Dean and Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI'04*.

[9] EMC, Inc. *EMC Navisphere Quality of Service Manager*. http://www.emc.com/collateral/software/data-sheet/c1152-emc-navisphere-quality-service-manager.pdf.

[10] Garrison and Reddy. Umbrella File System: Storage Management Across Heterogeneous Devices. 5(1), 2009.

[11] Guerra et al. Cost Effective Storage using Extent Based Dynamic Tiering. In *FAST'11*.

[12] Gulati et al. BASIL: Automated I/O Load Balancing Across Storage Devices. In *FAST'10*.

[13] Gulati et al. PARDA: Proportional Allocation of Resources for Storage Access. In *FAST'09*.

[14] HP, Inc. *HP Storage Essentials SRM Software Suite*. http://h18000.www1.hp.com/products/quickspecs/12191_na/12191_na.pdf.

[15] IBM, Inc. *IBM Tivoli Software*. http://www-01.ibm.com/software/tivoli/.

[16] Keeton et al. Designing for Disasters. In *FAST'04*.

[17] Kephart and Chess. The Vision of Autonomic Computing. *IEEE Computer*, 36(1), 2003.

[18] Liblit. *Cooperative Bug Isolation*. PhD thesis, University of California, Berkeley, 2004.

[19] Lim et al. Automated Control for Elastic Storage. In *ICAC'10*, Washington, DC, Jun 2010.

[20] Mesnier et al. Modeling the Relative Fitness of Storage. In *SIGMETRICS'07*.

[21] Mogul. Emergent (Mis)behavior vs. Complex Software Systems. In *EuroSys'06*.

[22] NetApp, Inc. *NetApp Flash Cache*. http://media.netapp.com/documents/ds-2811-flash-cache-pam-II.pdf.

[23] NetApp, Inc. *NetApp FlexCache*. http://media.netapp.com/documents/tr-3669.pdf.

[24] NetApp, Inc. *NetApp Provisioning Manager*. http://media.netapp.com/documents/ds_2742_provisioningmgr.pdf.

[25] NetApp, Inc. Microsoft Exchange 2007 on VMware Infrastructure 3 and NetApp iSCSI Storage. Technical report, NetApp, 2009. http://media.netapp.com/documents/tr-3683.pdf.

[26] NetApp, Inc. *NetApp DataMotion for Volumes*, 2010. http://media.netapp.com/documents/ds-3122.pdf.

[27] Pillar Data, Inc. *Managing Quality of Service (QoS) in a Shared Storage Environment*. https://pillardata.box.net/shared/static/qyvnx4vvid.pdf.

[28] Shepler et al. Network File System (NFS) Version 4 Minor Version 1 Protocol, 2010. RFC 5661, http://tools.ietf.org/html/rfc5661.

[29] Soundararajan et al. Dynamic Resource Allocation for Database Servers running on Virtual Storage. In *FAST'09*.

[30] Strunk et al. Using Utility Functions to Provision Storage Systems. In *FAST'08*.

[31] Thereska et al. Stardust: Tracking Activity in a Distributed Storage System. In *SIGMETRICS'06*.

[32] VMWare, Inc. *vSphere Resource Management Guide*. http://www.vmware.com/pdf/vsphere4/r41/vsp_41_resource_mgmt.pdf.

[33] Wachs et al. Argon: Performance Insulation for Shared Storage Servers. In *FAST'07*.

[34] Wang and Merchant. Proportional share scheduling for distributed storage systems. In *FAST'07*.

[35] Wang et al. Automatic Misconfiguration Troubleshooting with PeerPressure. In *OSDI'04*.

[36] Wilkes. Traveling to Rome: A Retrospective on the Journey. In *R2D2'08*.

[37] Wilkes et al. The HP AutoRAID Hierarchical Storage System. In *SOSP'95*.

[38] Zhu et al. Avoiding the Disk Bottleneck in the Data Domain Deduplication System. In *FAST'08*.