Towards a Middleware for Configuring Large-scale Storage Infrastructures

David M. Eyers
Computer Laboratory
University of Cambridge
Cambridge, United Kingdom
david.eyers@cl.cam.ac.uk

Ramani Routray, Rui Zhang
IBM Research – Almaden
San Jose, California, USA
routrayr@us.ibm.com
ruizhang@us.ibm.com

Douglas Willcocks, Peter Pietzuch
Department of Computing
Imperial College London
London, United Kingdom
{dtw107,prp}@doc.ic.ac.uk

ABSTRACT

The rapid proliferation of cloud and service-oriented computing infrastructure is creating an ever increasing thirst for storage within data centers. Ideally management applications in cloud deployments should operate in terms of high-level goals, and not present specific implementation details to administrators. Cloud providers often employ Storage Area Networks (SANs) to gain storage scalability. SAN configurations have a vast parameter space, which makes them one of the most difficult components to configure and manage in a cloud storage offering.

As a step towards a general cloud storage configuration platform, this paper introduces a *SAN configuration middleware* that aids management applications in their task of updating and troubleshooting heterogeneous SAN deployments. The middleware acts as a proxy between management applications and a central repository of SAN configurations. The central repository is designed to validate SAN configurations against a knowledge base of best practice rules across cloud deployments. Management applications contribute local SAN configurations to the repository, and also subscribe to proactive notifications for configurations now no longer considered safe.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Reliability, availability, and serviceability

Keywords

SAN, Middleware, Best Practice, Machine Learning

1. INTRODUCTION

Cloud computing has begun to flourish in recent years [3]. Its growth has led to a significant increase in the demand for storage, and thus scalable, manageable storage infrastructures. The promise to customers of low Total Cost of Ownership (TCO), combined with the ability of cloud computing to cope with those customers' dynamic infrastructure requirements, have fueled the emergence and growth of cloud service providers such as Amazon S3 and EC2[1]. The highly heterogeneous, multi-vendor interconnection of devices created by the cloud service providers allows multiple qualitative gradations of service. During the whole lifecycle starting from data placement to retirement of workload, it remains a monumental task for the cloud providers to perform the planning, configuration and migration on demand [2, 12]. Such operations are always error-prone given the complexity of interrelated physical and logical provisioning.

Cloud providers may choose from a number of different approaches to effect scalable storage. In this paper we focus on providers who employ Storage Area Network (SAN) technologies, since the authors have particular expertise in their use. Although Cloud providers are generally highly secretive about their infrastructure, the authors have interacted with such providers in a business capacity. Recent high demand for SANs has led to a proliferation of vendors and devices. The sheer heterogeneity and size of SANs means that they frequently represent the greatest challenge in the configuration of cloud storage. Often, expert teams will configure the initial deployments for a customer. Management applications are responsible for re-configuration, monitoring and tuning the performance of SAN systems and troubleshooting system faults.

Cloud providers' storage needs often evolve from what they were at the time of their original specification. Such reconfigurations will occur in response to customers' needs. A critical step in a cloud provider's workflow is to convert customers' application workload requirements into a set of parameters that can be translated to the cloud provider's infrastructure in terms of resources (such as CPU, memory, storage, I/O, and throughput). This includes customers' initial resource usage and the facilities for on-demand, dynamic growth or elasticity of applications. Many cloud computing providers will make these changes to their SAN configurations using their own IT staff. This may introduce instabilities and inefficiencies into the SAN. Tracking down these problems is often expensive in terms of external consultant man-hours [10].

Accurate and up-to-date configuration best practices have proven to be a valuable tool for addressing the aforementioned challenges [1]. Validating configurations against best practice rules before deployment is a form of "what-if" analy-

[1]`www.amazon.com/ec2/` and `www.amazon.com/s3/`

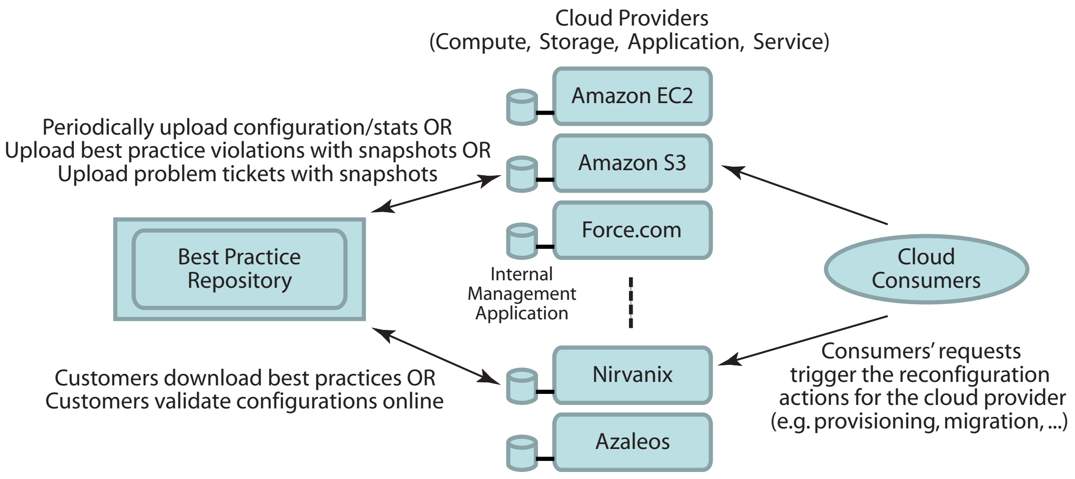
Cloud Providers
(Compute, Storage, Application, Service)
Amazon EC2
Amazon S3
Force.com
Nirvanix
Azaleos
Internal Management Application
Periodically upload configuration/stats OR
Upload best practice violations with snapshots OR
Upload problem tickets with snapshots
Best Practice Repository
Customers download best practices OR
Customers validate configurations online
Cloud Consumers
Consumers' requests trigger the reconfiguration actions for the cloud provider (e.g. provisioning, migration, ...)

Figure 1: Customers taking advantage of cloud storage infrastructure trigger storage reconfiguration

sis that may unearth potentially problematic settings. When problems do occur, the best practice rules can also be used to assist in faster root cause analysis. For example, IBM has a SAN Central team whose job it is to examine all known SAN configuration issues and develop appropriate best practices. These best practices have helped the SAN Central team to reduce the time required to resolve configuration errors from two weeks to two days: around 80% of the configuration problems that the team sees are caused by the violation of best practices in the IBM repository.

SAN configuration experts tend to have a great deal of real world experience, and apply intuition above and beyond what is provided to them in terms of specified configuration guidelines. However, the generation of the best practices using manual methods is costly, requiring many man-years of data gathering and analysis. In preference to the use of expensive, time-consuming, and error-prone domain expert analysis, it would be desirable for machine learning techniques to derive best practices. However, the generation of effective best practice rules relies heavily on collecting a large number of problematic configuration samples that together present enough "valuable lessons" to learn from. For example, SPIKE [10] works with synthetically generated data, where the problem reports not only contain the entities reporting problems but also the rest of the SAN configuration that may be working correctly. We found that to correctly identify and generate four best practices, SPIKE requires 500 problem reports. It is unable to generate any best practice rules using less than the first 150 problem reports.

Given the plethora of SAN devices on the market, and the myriad configuration possibilities, it is not practical to expect enough samples to be collected through the current state-of-the-art method, in which problematic configurations are manually reported from a few SAN deployments (often from the same provider). Rather, storage providers should be able to effortlessly contribute their share of bad configuration experience for the common good. All storage providers will benefit from a comprehensive best practice knowledge base from which they can then test their current and future SAN configuration practices. A key motivation behind this paper is the identification of the infrastructure gap currently preventing us from reaching this vision.

The primary contribution of this paper is the introduction of a *SAN configuration middleware* for validating SAN configurations. This middleware can check whether proposed changes to the configuration violate a collection of best practice rules that are provided by human domain experts or automated machine learning techniques. The results are exposed to a set of the management applications, for example, for planing and problem diagnosis, that support the SAN infrastructure. These applications are presented with a higher-level abstraction that unifies the heterogeneity of the underlying SAN infrastructure and provides them with a cleaner, less complex view of effects of configuration changes.

A high level overview of our proposed infrastructure is shown in Figure 1. The cloud providers are depicted in the middle of this figure, with their customers shown on the right. The SAN systems of each cloud provider interact with the best practice repository on the left. The best practice repository stores cloud provider configuration snapshots, uses machine learning algorithms to update the best practices collection, removes aged practices from the database, and indexes configuration data to facilitate efficient searching. The middleware contributes back newly acquired knowledge about both valid and invalid SAN configurations to the central knowledge base after first anonymizing any company-sensitive information that might be within the SAN configuration report. Finally, the middleware can subscribe to updates in the central knowledge base that might indicate that a configuration that was considered safe in the past is now considered to be at risk. This enables management applications to react to such problems in near real-time, benefiting from the collective insights about SAN configurations across the global cloud.

The rest of this paper is organized as follows. Section 2 provides a brief overview of SAN technologies. A case study that motivates the need for a SAN configuration middleware is presented in Section 3. We introduce our middleware in Section 4. Related research work is examined in Section 5. Finally, Section 6 provides concluding remarks.

2. BACKGROUND

This section provides a brief introduction to SAN technologies. The goal of a Storage Area Network (SAN) is

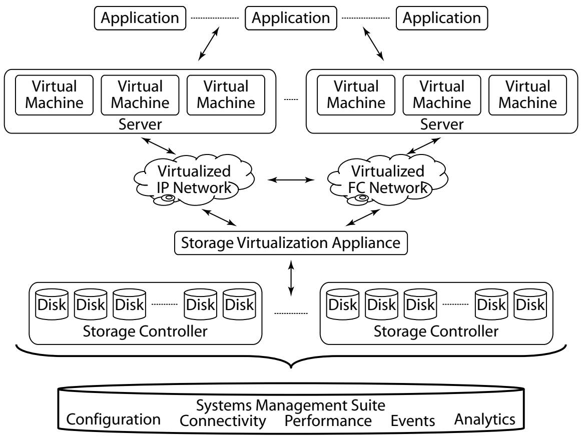

Figure 2: An overview of a typical Storage Area Network (SAN) deployment

to provide block-based storage as a service over a (usually) dedicated network to attached hosts. The components of a typical SAN deployment are shown in Figure 2.

A host usually has one or more host bus adapters (HBA) with ports that are connected to switch ports via Fibre Channel cables. These Fibre Channel switches are often linked together to form a network topology (AKA Fibre Channel fabric) that has properties such as redundant data paths from hosts to storage subsystems, along with security primitives such as zoning.

The storage subsystems and tape libraries are typically connected to switch ports and provide block storage access in the form of volumes to the hosts. The admissibility of particular data paths is vetted using access control provided by storage controller known as masking and mapping.

With the advent of virtualization and emergence of new technologies, the storage infrastructure in data centers has become increasingly heterogeneous over time. Uniform end-to-end management is key to the provision of high-value analytics such as problem determination, storage provisioning planning, optimized resource usage, application migration and configuration analysis. Storage and/or system management software products, such as IBM TotalStorage Productivity Center (TPC[2]), IBM Director[3], Micrososft System Center[4], HP System Insight Manager[5], and EMC Control Center[6], can provide tools to address such management tasks. The current products for unified management are point solutions: they aim to allow administrators to perform similar tasks, such as creating a volume, on a variety of different manufacturers' devices. They do not facilitate sharing of salient information across the customer base to avoid mistakes being made.

3. CASE STUDY

With popular success stories of applications hosted in the cloud[7], we motivate the need for a SAN configuration middleware for cloud storage providers by exploring a hypothetical scenario: a cloud provider (both compute and storage cloud) facilitating online retail stores as a service. A seasonal sale drives an increase of 50% more online shoppers than anticipated. This causes a spike in compute as well as storage requirements. The customer (an e-business website) uses the service of the cloud for two applications, each having specific quality-of-service (QoS) requirements:

(1) The online shopping application comprises of application servers, HTTP servers, edge servers, and database servers. Database servers use storage from the cloud to store transactional data generated from the online shopping. Data is stored on high-end storage (for reliability and availability). All of the changes to this storage are replicated on the other high-end storage controller. This is for disaster recovery purposes. Nightly offline backup is done onto the tape-library.

(2) The online store also has its email application and data in the cloud. Storage for the email application is housed on mid-range storage, as this gives a better cost/risk balance. The email application is hosted on mid-range servers, and is backed-up using incremental-daily and full-weekly strategies on low-tier storage.

This data center has three types of human administrators: application administrators, network administrators and storage administrators. We assume that the cloud management is done using one or more software suites such as IBM's TotalStorage Productivity Center (TPC). TPC allows the admins to be alerted to changing conditions in the SAN environment, such as: "notify me when file system capacity utilization reaches 80%". Alerts can be connected to transfer volume, or when a particular port utilization rises over a threshold. Resource provisioning is semi-automated: while some resources are allocated automatically, other resource requests originate from request-tickets and are managed through workflows, so as to capture an audit-trail. The

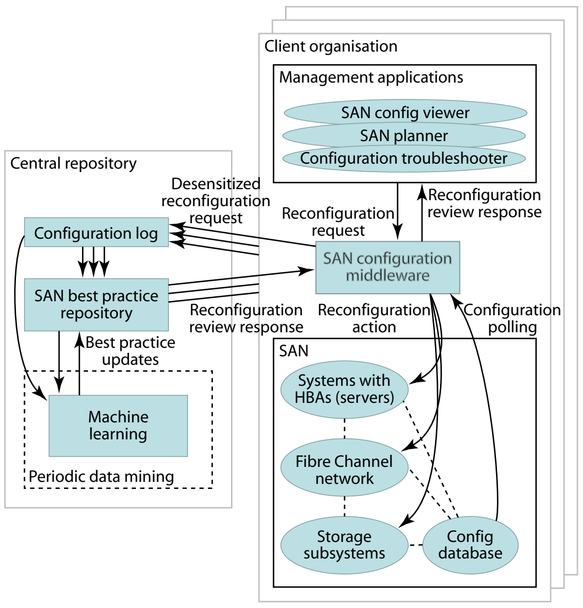

Figure 3: An overview of the SAN configuration middleware.

[2]www.ibm.com/software/tivoli/products/totalstorage-data/
[3]www.ibm.com/systems/management/director/
[4]www.microsoft.com/systemcenter/en/us/
[5]h18002.www1.hp.com/products/servers/management/hpsim/
[6]www.emc.com/products/family/controlcenter-family.htm
[7]www.informationweek.com/cloud-computing/blog/archives/2008/09/case_studies_in.html

following work items might be raised in the course of the applications' operation:

(1) The application file system alerts the application administrator that storage space utilization has risen above 80%, and thus more storage is required. In some cases, allocation of more storage is automated, but in others it gets approved and provisioned through a workflow.
(2) Every night, a script has to be run to enable a backup traffic flow for the tape library. Since the path through the FC network to the tape library should be disabled at other times, there is a periodic need to fine-tune the time-window provided for access to the tape library.
(3) The application servers have to be updated with a code patch that implements a new feature.
(4) Some new servers are to be introduced that provide applications to perform particular business analytics.
(5) Seasonal sale drives cause the introduction of more clone application servers, database servers and allocation of more storage.

Either through manual or automated workflows, all three administrators meet and decide how to incorporate requests such as those listed above. Each request-ticket has a turnaround time based on the service level agreement. Implementation of the changes (introduction of new servers, extension of file systems, extension of databases, connection of servers to the FC fabric, creation of zones, addition of zones to active zonesets, creation of new volumes, assignment of new volumes to servers, migration of applications) is done through hybrid semi-automatic planning. It is then approved by the administrators through explicit procedures for agreement and then deployed.

For example, to accommodate the second request stated above, network connectivity and configuration changes are planned for deployment. However, validating the proposed plan against the best practices raises a violation of the best practice "No zone should contain both tapes and disks".

With our SAN configuration middleware, we add the capability to validate the proposed changes to the SAN configuration against a library of best practices before and after the deployment, as discussed in the following section. Also, by periodically validating the existing cloud configuration against the set of best practices, the cloud provider can maintain the health of its data center, and obviate problems such as performance degradation, data loss, and so forth.

4. SAN CONFIGURATION MIDDLEWARE

Our proposal is a modification of a typical SAN infrastructure as shown in Figure 3. We introduce new elements into the SAN deployment to support configuration validation. Each client organization that manages a SAN infrastructure also runs a *SAN configuration middleware* that is situated between the SAN infrastructure and a set of management applications. The middleware also interacts with a central *SAN best practice repository* that is maintained and operated by a third-party organization. The purpose of the repository is to validate configurations and exploit best practice knowledge across a large set of heterogeneous SAN deployments by client organizations.

Initially, we expect that a central repository will provide the most straightforward means to establish the trust of the client organizations, and to ensure that all of the data required by the machine learning algorithms is available. In future, it may be beneficial to employ a distributed approach to the best practice repository. This would also mitigate a single point of failure, although if the repository was unreachable, clients could switch to reactive validation in order to continue their operations.

4.1 Middleware operation

As illustrated in Figure 3, the SAN configuration middleware can access up-to-date configuration information from the local configuration database that is part of SAN infrastructure. The middleware supports two broad modes of configuration validation: *reactive* and *proactive*. There are also two types of client participation: *sharing* and *non-sharing*.

(1) For reactive validation, the middleware periodically (e.g. each week) collects the SAN configuration data from the local database, and sends it to the central repository for validation. If any best practice rules are discovered to be violated, alerts will be sent back to management applications at the originating site. For example, the middleware could facilitate a SAN configuration viewer application highlighting problematic areas of the SAN infrastructure, as informed by reactive querying. Reactive validation is comparatively non-intrusive, however, it has the disadvantage that unsafe configurations can be run for a time before alerts are raised.

Clients that are *sharing* participants will permit sanitized and anonymized snapshots to be stored at the central repository for examination. The snapshots are tagged with an organizational ID to indicate data provenance, and to facilitate time-series analysis. An important aspect of sharing participants is that a collection of (presumed) good configurations is formed. Non-sharing clients can still have their configurations validated: if validation fails, the client will be prompted as to whether they will accept the problematic configuration being uploaded and examined in more detail.

(2) Proactive validation requires that the configuration middleware sit between existing SAN management applications and the devices being configured. The middleware must be able to intercept requests for configuration modification. All changes to the SAN configuration will need to be successfully validated before they can proceed, including initial configuration, and any mitigating actions performed after a critical event (e.g. an outage is reported). The proactive validation approach still permits clients to choose whether or not to be sharing participants.

The dotted rectangle in Figure 3 indicates that the machine learning (ML) component of the infrastructure runs separate from, and in parallel to the proactive and reactive queries from clients. The results from ML analysis may lead to inquiries being made back to willing clients for further information about particular SAN configurations. The host of the central repository may run experiments in the lab. Finally, there may be a declaration of a new best practice rule. Each new rule will be explicitly published.

Best-practice policies are encoded in a declarative policy language and stored in the repository [7]. These best practices are categorized into *parametric* and *non-parametric*; parametric ones accept input parameters must be provided by administrators as thresholds. The middleware exposes primitives such as *scope* and *profile* to help customers customize their configuration analysis environment. Scope defines the setup that needs to be validated—it can range from the entire environment to a single Fibre Channel fabric or a set of Fibre Channel zone sets. These scopes can be selected on the basis of the policies to be verified. Cloud providers can manually decide or have the system automatically pick a set of applicable best practices to run on a particular scope, which is called a "profile".

The API for reactive verification begins with a call to validateConfiguration(Scope, Profile). A set of Violation references is returned, if the configuration does not validate successfully. Each Violation indicates responsible entities, along with parent entities. For example, a Fiber Channel Port that violates a best practice would also carry information about its encompassing HBA, server, and switch interconnections.

The middleware at each client site receives a re-validation alert from the repository when a model update or retraining takes place and either (a) any best practice rule recently used to validate the local configuration has been changed (not removed), or (b) new rules have been added that might invalidate the current local configuration.

4.2 Abstracting configuration parameters

Since multiple deployments of the SAN configuration middleware are in different cloud data centers, each one will have to handle a specific underlying SAN infrastructure with a custom set of configuration parameters. To ship information to and from the SAN best practice repository from each of the middleware deployments, we need to have an abstract format for SAN configuration snapshots: heterogeneous SAN configuration parameters are translated to a common representation. The transformation to the abstract representation involves removal of any confidential information within clients' configurations.

The repository uses an object-oriented, hierarchical format that extends the CIM/SMI-S profiles[8] defined by standards bodies. Each snapshot is tagged with a timestamp and any open or resolved problem tickets that were raised in the data center. Problem tickets also follow a standardized format: the user is requested to classify tickets into categories such as performance degradation, data access problem, security violation, data loss, and so on, along with a set of the entities that the user believes are involved with the problem.

Another benefit of a high-level configuration description is that it can enable customers to express their management goals in a higher-level manner. To achieve this, customers may specify their motivation for a reconfiguration, rather than worrying about the specific reconfigurations that need to occur. For example, the goal to "increase storage capacity" on a given volume can be expressed without managing the specifics of the particular SAN infrastructure.

4.3 Best practice repository

It is expected that there will be a large number of rules in the best practice repository. This section discusses some approaches for structuring the repository to facilitate efficient querying.

Let R be the set of best practice rule IDs. Each $r \in R$ is an identifier for a rule written in a particular policy language. Useful rules can be encoded by as simple a language as one that indicates that a set of entities and attributes should not be in the same SAN configuration: e.g. tapes and disks should not share a zone, Windows and Linux OSes should not be mixed in a zone, or HBA Firmware Version x should not be used with Solaris. Irrespective of the policy languages that are used, we are able to narrow down a candidate set $C \subseteq R$ for any query, by relationships between $r \in C$ and three sets P, E and A that we describe below. Further metadata for $r \in R$ includes the reporting organization, organizations that are subscribed to changes in r, etc.

Problem types P: The set P of problem types contains values such as performance degradation, data access problem and data loss. All problem tickets, and bad SAN configurations, are classified into a subset of P.
Entities E: The set E lists the possible components of a SAN deployment. Example elements of E include server, data-path, FC switch, IP switch, application, disk array, and HBA. Most elements of E will be parametrized by a number of attributes, e.g. the manufacturer of the component, the firmware version, and so forth.
Attributes A: Finally, the set A contains all possible values of all of the attributes of each $e \in E$, e.g. IBM might be the vendor of a particular disk array.

Let the three functions $f_P : R \to \mathcal{P}(P)$, $f_E : R \to \mathcal{P}(E)$, and $f_A : R \to \mathcal{P}(A)$, map each rule onto the relevant parts of P, E and A. Although the values in the sets P, E and A form a structured ontology, significant filtering will be possible simply on the basis of set-oriented matching. For example, if a customer does not have a tape library, we want to avoid evaluating any parts of the best practice repository that relates to tape libraries. More formally, for a candidate rule set $C \subseteq R$, we have $\forall c \in C$, tape $\notin f_E(c)$.

The selectivity of each element of P, E, and A for subsets of R can be measured easily, and used to form indices over R. Since the matching criteria can be considered to be independent, we can look to partial match retrieval techniques from database research. For small numbers of filtering attributes, bitmap indices may be appropriate. More generally, techniques using Bloom filters can be used to form signatures of the attributes in rule sequences. These signatures allow rapid determination that a block does not contain rules of interest with a given probability of false positives.

The knowledge base of best practice rules must stay up-to-date. Updates will occur for the following reasons:
(1) A significant number of new configuration snapshots have been submitted. The best practice rules should be updated to reflect the new data. We anticipate that these updates could be quite frequent if the repository is used by many SAN clouds and thus new configuration snapshots or problem reports begin to arrive at high speed. If any applied ML techniques support incremental model updates, then only a subset of the model needs to be considered. However, techniques such as inductive logic programming do not support incremental learning: the complete model has to be re-trained in such cases.
(2) After some time, some of the snapshots used for training may be declared as being obsolete, and retraining of the rules will need to occur without using these snapshots. This will avoid a stale set of best practice rules exerting an influence that is not applicable to current SAN deployments. For example, after a given device interaction problem has been fixed across all SAN deployments by updates in hardware or software, the best practice rules relating to this specific problem will become obsolete and may be discarded.

5. RELATED WORK

As computer systems become increasingly distributed and

[8]See the DMTF Common Information Model: `dmtf.org/standards/cim`, and the SNIA Storage Management Initiative Specification: `snia.org/forums/smi/tech_programs/smis_home`

heterogeneous, there has been a plethora of work on wide-area middleware abstractions that help manage this growing complexity.

In Grid computing, applications have benefited from Grid middleware for a long time. *Globus* [5] and the *OGSA* specifications [6] provide a set of services to help engineer, deploy and execute wide-area Grid applications. Similar to our SAN configuration middleware, Grid middleware is faced with a heterogeneous set of resources that must be managed. However, the focus of Grid middleware is on software abstraction instead of configuration management of physical system resources. Monitoring middleware such as *Ganglia* [9] provide mechanisms for collecting statistics from a set of cluster machines. In contrast to our proposal, multiple middleware deployments do not share information to improve global system behavior.

Work in the broad area of systems management seeks to address the challenges that arise in terms of validating or providing guidance to management actions. Performance models have been developed for various cloud components that project the QoS impact of run-time resource allocation actions or deployment-time capacity planning [13, 14, 4]. Conversely, many of these models can also be used for performance problem localization and remediation. It is now possible to perform "What-if" analyses with regard to service/application reliability and up-time or root cause analysis in the event of major outages using various availability models [11, 8]. Such analyses are more quantitative in nature than the type of configuration validation discussed in this paper, but nevertheless bear interesting similarities in relation to complexity and cost of modeling.

The latest generation of these models are also "learned" from data collected by management middleware [14, 4, 8]. This is in contrast to the case of configuration best practices, which deal with more abstract concepts such as component types common across deployments. In learning these complex relationships, the need for a large amount of data has been acknowledged. Although performance/availability data can be quickly accumulated over time, the performance and availability characteristics of different environments are often distinct and thus data may not be shared across deployments.

6. CONCLUSIONS

Cloud applications are supported by storage infrastructure in data centers. The heterogeneity and complexity of storage systems makes it difficult to configure them correctly with current management applications. We propose a solution by introducing a layer of abstraction in the form of a SAN configuration middleware. The middleware collects configuration data, abstracts it into a standard representation and detects configuration problems. This is done by exploiting a repository of best practice configuration rules that is shared among multiple middleware deployments. As a result, the middleware provides a uniform, higher-level abstraction to SAN management applications, making it easier and more cost effective to reconfigure and troubleshoot storage infrastructures.

We believe that this work is an important first step towards the vision of a general purpose management middleware for cloud storage. A uniform middleware abstraction across multiple data centers will enable the coordinated enforcement of policies for cloud applications. Such policies may implement automated infrastructure management for cost reduction, regulate power consumption across data centers to achieve "green" operation and enforce security safeguards for privacy protection. Supporting these policies will require sophisticated and widespread tuning of the numerous infrastructure-level "knobs" in a cloud offering. The cooperation of multiple middleware deployments across data centers that help generate and then benefit from a shared configuration knowledge base, will be key to achieving these goals in the future.

7. REFERENCES

[1] D. Agrawal, J. Giles, K.-W. Lee, et al. Policy-based validation of SAN configuration. In *POLICY '04: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks*, page 77, Washington, DC, USA, 2004. IEEE Computer Society.

[2] E. Anderson, S. Spence, R. Swaminathan, et al. Quickly finding near-optimal storage designs. *ACM Transactions on Computer Systems*, 23(4):337–374, 2005.

[3] M. Armbrust, A. Fox, R. Griffith, et al. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[4] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using Magpie for request extraction and workload modelling. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI'04),*, December 2004.

[5] I. Foster. Globus toolkit version 4: Software for service-oriented systems. In *IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779*, pages 2–13, 2005.

[6] I. Foster, C. Kesselman, J. Nick, et al. The physiology of the grid: An Open Grid Services Architecture for distributed systems integration, 2002.

[7] S. Gopisetty, S. Agarwala, E. Butler, et al. Evolution of storage management: transforming raw data into information. *IBM J. Res. Dev.*, 52(4):341–352, 2008.

[8] R. Herbrich, T. Graepel, and B. Murphy. Structure from failure. In *Proceedings of the Annual Conference of National Information Processing Symposium*, 2005.

[9] M. L. Massie, B. N. Chun, and D. E. Culler. The Ganglia distributed monitoring system: Design, implementation and experience. *Parallel Computing*, 30:2004, 2003.

[10] P. Sarkar, R. Routray, E. Butler, et al. SPIKE: best practice generation for storage area networks. In *SYSML'07: Proceedings of the 2nd USENIX workshop on tackling computer systems problems with machine learning techniques*, pages 1–6, Berkeley, CA, USA, 2007. USENIX Association.

[11] Y.-S. Su and C.-Y. Huang. Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. *J. Syst. Softw.*, 80(4):606–615, 2007.

[12] J. Ward, M. O'Sullivan, T. Shahoumian, and J. Wilkes. Appia: Automatic storage area network fabric design. In *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, page 15, Berkeley, CA, USA, 2002. USENIX Association.

[13] Q. Zhang, L. Cherkasova, and E. Smirni. A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In *ICAC '07: Proceedings of the Fourth International Conference on Autonomic Computing*, page 27, Washington, DC, USA, 2007. IEEE Computer Society.

[14] S. Zhang, I. Cohen, J. Symons, and A. Fox. Ensembles of models for automated diagnosis of system performance problems. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 644–653, Washington, DC, USA, 2005. IEEE Computer Society.