CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2011)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.1716

Configuring large-scale storage using a middleware with machine learning

David M. Eyers[1,*,†], Ramani Routray[2], Rui Zhang[2], Douglas Willcocks[3] and Peter Pietzuch[3]

[1]*Computer Laboratory, University of Cambridge, Cambridge, U.K.*
[2]*IBM Research—Almaden, 650 Harry Road, San Jose, CA, U.S.A.*
[3]*Department of Computing, Imperial College London, London, U.K.*

SUMMARY

The proliferation of cloud services and other forms of service-oriented computing continues to accelerate. Alongside this development is an ever-increasing need for storage within the data centres that host these services. Management applications used by cloud providers to configure their infrastructure should ideally operate in terms of high-level policy goals, and not burden administrators with the details presented by particular instances of storage systems. One common technology used by cloud providers is the Storage Area Network (SAN). Support for seamless scalability is engineered into SAN devices. However, SAN infrastructure has a very large parameter space: their optimal deployment is a difficult challenge, and subsequent management in cloud storage continues to be difficult.

In this article, we discuss our work in *SAN configuration middleware*, which aims to provide users of large-scale storage infrastructure such as cloud providers with tools to assist them in their management and evolution of heterogeneous SAN environments. We propose a middleware rather than a stand-alone tool so that the middleware can be a proxy for interacting with, and informing, a central repository of SAN configurations. Storage system users can have their SAN configurations validated against a knowledge base of *best practices* that are contained within the central repository. Desensitized information is exported from local management applications to the repository, and the local middleware can subscribe to updates that proactively notify storage users should particular configurations be updated to be considered as sub-optimal, or unsafe. Copyright © 2011 John Wiley & Sons, Ltd.

Received 31 May 2010; Revised 16 November 2010; Accepted 23 December 2010

KEY WORDS: SAN; configuration policy; middleware; best practices; machine learning

INTRODUCTION

The sophistication and deployment of cloud computing has significantly gathered pace in the recent years [1]. In most cases, instances of cloud services require persistent data in the cloud, and thus there has been a coupled demand for scalable, manageable and reliable storage systems. Even outside the cloud (or in so-called 'private clouds,' that operate within one organization), there is an increasing demand for scalable storage systems, just to meet the needs of storing ever-larger data sets for analysis and data mining. Much of the growth of cloud computing has been on the basis of cloud providers being able to lower the Total Cost of Ownership (TCO) for their

*Correspondence to: David M. Eyers, Department of Computer Science, University of Otago, PO Box 56, Dunedin 9054, New Zealand.
†E-mail: dme@cs.otago.ac.nz

cloud clients, as well as the elasticity provided by cloud infrastructure: economies of scale mean that the cloud providers can cope with their clients' dynamic requirements. The emergence and significant expansion of cloud service providers such as Amazon S3 [2] and EC2 [3] have been the result.

Within any data centre, it is usual that a highly heterogeneous collection of devices from different vendors forms over time. This is often the result of rolling upgrades, and allows the equipment to be used to provide a number of different graduations of service. However, this heterogeneity makes the overall life cycle from data placement to retirement of workload a highly onerous task. There will be requirements to plan, configure and to perform the migration of data on demand [4, 5]. Owing to the complexity of the interrelated physical and logical systems, these provisioning tasks are often error-prone.

This article focuses on assisting those organizations that employ Storage Area Network (SAN) technologies: the authors have field expertize and experience regarding SAN deployment and reconfiguration. Although in common use, SANs are not the only means of achieving reconfigurable, large-scale storage systems. Although cloud providers and other organizations that use data centres, are generally highly secretive about the configuration of their infrastructure, the authors have interacted with a number of these organizations in a business capacity.

There has been a proliferation of SAN devices, and indeed vendors, based on the growing demand for them. One of the greatest challenges in terms of configuration of large-scale storage will often be dealing with the extraordinary degree of heterogeneity and the consequent complexity involved in SAN configuration. Teams of experts will often be employed to perform the initial deployment of SAN infrastructure for customers. From that point onwards, management applications will be installed that provide for the monitoring of the SAN, and troubleshooting faults in the system. In addition, the software will facilitate the reconfiguration of the SAN system, and its performance tuning [6].

It is highly unusual for the configuration that is deployed initially to remain satisfactory for very long into a SAN's life cycle. Evolution will frequently be required due to capacity or bandwidth issues, driven by client demand. The key step in the evolution workflow is to determine an appropriate plan for converting a client's application workload requirements into parameters that are expressed in terms of the underlying infrastructure (for example, CPU, network throughput, memory allocations, storage capacity and I/O). This configuration of parameters in a cloud provider's case needs to include both the initial resource allocation, and the considerations for elasticity in that cloud client's demand. Owing to the expense of consultant engineers from SAN manufacturers, most large-scale storage users will choose to use their own staff to make changes to their SAN configurations. However, this can lead to inefficiency of SAN performance, and at worst, possible instabilities in the configuration. It has been shown that tracking down these types of configuration issues using external consultants can be extremely expensive [7].

Any modified configuration should be tested to ensure that it meets the basic requirements for performance and security as needed by the clients of the storage system. For example, cloud providers must be able to establish, for each configuration, the estimated resource usage, availability and performance metrics based on the management tools and capabilities that are available. Unfortunately, the presumption is usually that if these metrics are all met, the configuration is performing as you designed it to, and thus the application user objectives are being met. However, this is not always the case due to subtle interactions between subsystems that may not be clearly visible. Furthermore, the cost to organizations of down-time is very high—consumers of online services are highly agile and can often move their business to competitors' services easily.

When addressing SAN configuration challenges, it has been shown that accurate and up-to-date configuration best practice descriptions are a highly valuable tool [8]. The best practice rules provide a reference point against which to perform 'what-if' analyses *before* the new configuration is actually deployed, in an effort to discover any potentially problematic parameters. In addition, when problems occur in the field, the best practice rules can help administrators to more quickly complete root cause analysis. The SAN Central team at IBM has the job of examining all known SAN configuration issues so that they can develop and document best practices. From experiences in the field, about 80% of the configuration problems that the SAN Central team have encountered also violate one of the best practices contained within the IBM repository. Use of the best practices

DOI: 10.1002/cpe

has led to a drastic reduction in the amount of time required to resolve configuration errors, from the order of weeks to the order of days.

Experts in SAN configuration usually have a significant amount of real-world experience, and are able to apply intuition to situations above and beyond what the specified configuration guidelines provide to them. Unfortunately, it is extremely costly—in the order of man-years of data collection and analysis—to generate best practices using manually driven methods. Rather than having to rely on expensive, potentially error-prone, and time-consuming analyses performed by expert technicians, a better situation would be for *machine learning* (ML) techniques to be able derive best practices, or at least to focus the efforts of the previously manual efforts. One problem with the use of ML is that training can only be done effectively—for the derivation of salient rules—if a large number of problematic configurations can be collected. In our past work on SPIKE [7], synthetically data were generated so as to match the distribution of observed field data, and to create problem reports that contain both the problematic entities and a complete description of the rest of the SAN configuration, which may not have anything wrong with it. In order to generate four best practices from that set, the SPIKE system requires 500 problem reports. Further, SPIKE was unable to produce any best practice rules before it had digested 150 problem reports.

Given this significant need for training data, combined with the huge variety of different SAN devices on offer, and the vast parameter spaces of configurations, it is not going to be practical to expect state-of-the-art manual reporting methods of reporting problem cases to provide a sufficient number of samples to generate good best practices. Further, the attention of manual methods necessarily focuses on a few SAN deployments at a time, and often those SANs will have been deployed for one large-scale storage customer. Instead, it would be better if storage providers were able to contribute information about their bad configurations effortlessly, for the sake of the overall greater good. The expansion of a comprehensive *best practice knowledge base* would benefit all of the storage providers who cooperate: they can test their current and future SAN configurations comprehensively. This article explores our proposal for filling the gap in the infrastructure that currently prevents storage experts from reaching the vision of automated best practice generation.

This article's main focus is the introduction of a *SAN configuration middleware* for validating SAN configurations. The middleware facilitates checking whether proposed changes to a configuration violate best practice rules—regardless of whether the rules have been generated from the work of human domain experts or from the application of ML techniques. The middleware exposes its results to the set of management applications that are used for planning and problem diagnosis within the SAN infrastructure. By adding another layer of abstraction, we present these management applications with a view that avoids the need to manage the heterogeneity of the SAN infrastructure itself, and thus can reduce the apparent complexity of configuration changes.

Figure 1 provides a high-level overview of the infrastructure that we propose for the case of cloud storage providers. These providers are shown at the centre of the figure; their customers are displayed on the right of the figure. All the SAN systems of the cloud providers connect with the repository of best practices shown on the left of the figure. The repository of best practices stores configuration snapshots derived from the data of each cloud provider, it uses ML algorithms to update the best practices collection, removes old best practices from the knowledge base when they are no longer useful, and provides indices over the configuration data to allow for fast searches to be performed. Newly acquired knowledge about both good and bad SAN configurations is sent to the central knowledge base by the middleware, after the anonymizing of any information within the SAN configuration report that might be sensitive to the companies that are participating in this system. At each site, the middleware can subscribe to salient updates from the central knowledge base. This might mean that a local configuration that was considered to be safe, is subsequently determined to actually be risky. Thus, external experience can be brought to bear on the management applications, allowing them to react near real-time to problems and insights (or reflect them to their operators) that are determined from throughout the global cloud connected to the shared repository.

This article is organized as follows. In the following section, we provide a brief overview of SANs, and the subsystems that they include. We also introduce the ML techniques that we

DOI: 10.1002/cpe

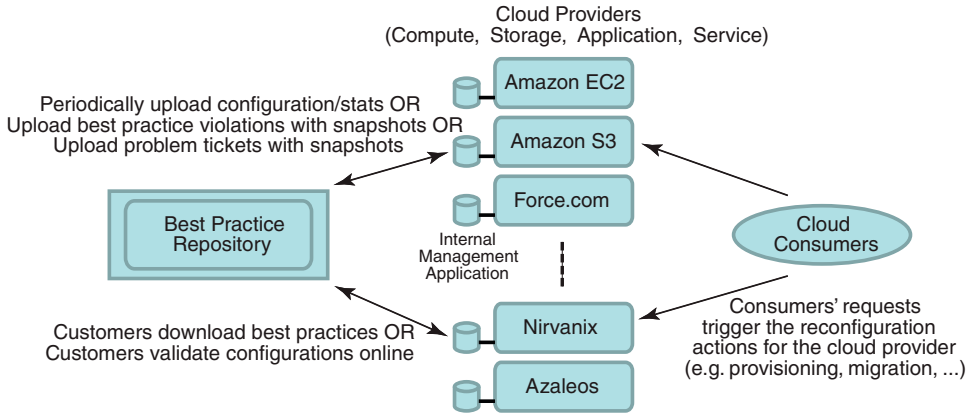

Figure 1. Customers taking advantage of cloud storage infrastructure trigger storage reconfiguration.

apply in our work. We follow this with the presentation of a case study that motivates the need for a SAN configuration middleware. Our particular middleware system is then presented in the following section: we discuss its operation, the abstraction of SAN configuration parameters, key types of best practice rules, and some techniques for optimizing searches being performed over the repository. The end of the article includes coverage of related research work, before providing some concluding remarks.

BACKGROUND

In this section, we provide a quick introduction to the SAN technology, and the ML approaches that we apply in our work.

Storage area networks

SANs operate at the block level as opposed to the file system level (NAS does the latter). SANs provide this block-level service over a dedicated network to attached hosts. The key point is that the SAN devices provide for a more manageable interface to the underlying storage devices (e.g. hard disks) than would be possible were the storage to be directly connected to the host servers that utilize it. Figure 2 shows the components within a typical SAN deployment.

Hosts that use SAN storage connect to storage network switches using one or more host bus adapters (HBA). Often these switches, and the HBAs, will use Fibre Channel cables. The network formed by the interconnected Fibre Channel switches is referred to as the Fibre Channel fabric. These storage networks are likely to incorporate properties such as redundant network paths between storage subsystems and the hosts. Also, security is integrated, usually with the concept of 'zoning'.

The storage subsystems, tape libraries and other storage network devices are connected to switch ports so that they can provide block storage access to the hosts. The storage is encapsulated in the notion of a 'volume'. Admission control is applied to data paths through the storage network fabric: masking and mapping are two common access control functions that are provided by the storage controller.

In contrast to the SAN approach, Network Attached Storage (NAS) provides a file-based interface to a storage service, as opposed to a block-based interface. It is expected that the boundaries between NAS and SAN systems will blur in the future. This is due to NAS heads increasingly exposing data paths to their back-end storage. The interlinks between storage components are being further opened up by technologies such as iSCSI, that facilitate reuse of existing IP technology within large-scale storage systems.

DOI: 10.1002/cpe

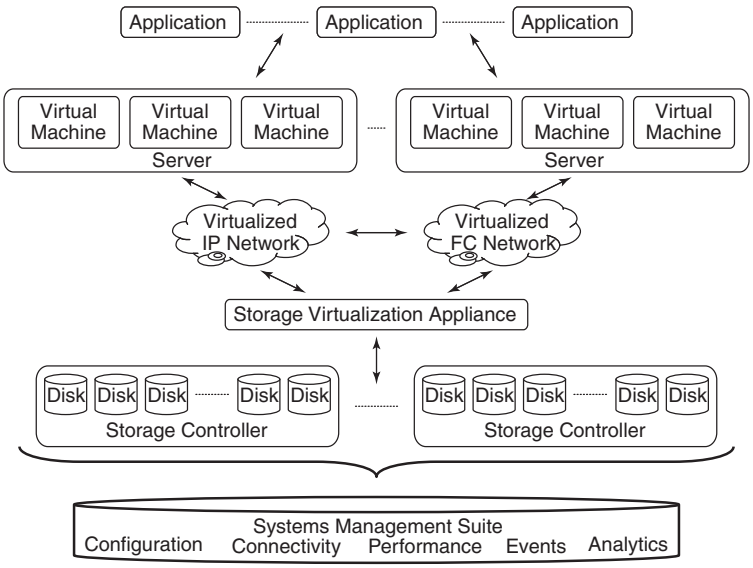

Figure 2. An overview of a typical Storage Area Network (SAN) deployment.

Another approach is Direct-Attached Storage (DAS). In this case, the disks are contained within dedicated infrastructure that is separate from the hosts that utilize the storage. At the high-end, DAS devices strongly resemble SAN devices, but for using a single point-to-point link instead of a network topology.

Finally, it may be that software in the application layer is in the best position to manage an extensible storage system. Such systems often use local disks. It is likely that the application-level storage abstraction will be applied throughout clusters of computers: this sort of storage system is appropriate for service providers that run highly customized operating systems (e.g. Google), or have homogeneous application loads that can be partitioned effectively (e.g. e-mail storage, or certain types of database systems).

Recent technological introductions, such as virtualization and new types of devices and interconnects, have caused the storage infrastructure in data centres to become more and more heterogeneous. In the face of this, uniform end-to-end management is key to the provision of mission-critical analytics. These analytics include storage provisioning, performance tuning, migration of volumes and applications, configuration analysis and fault determination. Many storage and system management software products are on the market that aim to assist with the aforementioned management tasks. These include IBM TotalStorage Productivity Center (TPC) [9], IBM Director [10], Microsoft System Center [11], HP System Insight Manager [12] and EMC Control Center [13]. Nonetheless, the currently available products for unified management are generally special-purpose solutions: they will facilitate administrators performing particular tasks such as the creation of storage volumes on a variety of devices produced by different manufacturers. However, they do not aim to share information across the customer base to advise against potentially problematic configurations, and to validate the decisions made by local administrators.

Machine learning

ML refers to a collection of widely used techniques that provide computer systems with the ability to automatically learn to recognize complex patterns. ML techniques utilize some form of provided *domain knowledge* and a set of observed examples referred to as *training data*.

Broadly, learning is classified into two categories: *supervised learning* and *unsupervised learning*. In supervised learning, the possible set of output classifications from the system is known in advance. Training data input to the system is labeled with the output classification that is desired from the system for each data. In unsupervised learning, it is desired that the

DOI: 10.1002/cpe

system produce some sort of classification from input data, but those data are not labeled—often unsupervised learning systems perform some type of clustering of the input data. There are various ML techniques applicable to different domains and data types.

The class of system management problems that we have explored present multi-dimensional interrelationships as input and require deterministic, discrete, interrogable outputs from the ML system. This type of supervised learning is well suited to ILP (Inductive Logic Programming) techniques [14]. ILP evaluates positive and negative examples based on background knowledge to generate hypotheses. Other ML techniques applied, such as decision trees, were discovered to deal poorly with multi-dimensional data [7], and some impose significant preprocessing costs.

We have evaluated four specific implementations of ILP: Aleph [15], HR [16], Progol [17] and ProGolem [18]. Both Aleph and Progol are top-down, relational ILP systems based on inverse entailment. In contrast, HR is a Java-based automated reasoning tool for theorem generation.

Many ILP systems are written to support a fairly specific input domain. For example, we ran into limitations in the form of a lack of support for arithmetic operations, comparison and cardinality in Progol. We then discovered that the preprocessing and background knowledge encoding were going to be too expensive for use within the HR tool (although parts of this system are undergoing redevelopment at the moment, and should improve the situation).

We found the ProGolem to be useful for our needs: it generates its hypotheses in the form of first-order logic expressions with quantifiers.

A CASE STUDY OF LARGE-SCALE STORAGE USE

There are numerous success stories emerging of applications being hosted within the cloud [19]. We thus motivate the need for our SAN configuration middleware against the backdrop of a hypothetical scenario involving a cloud service provider. In this case, the cloud provider aims to facilitate the running of online retail stores, packaged as a service. This involves both compute needs (e.g. analytics and search), and storage.

To justify the need for SAN reconfiguration, we assume that a highly successful seasonal sales drive is run by one of the cloud provider's clients, and that this causes a 50% increase in the number of online shoppers that had been previously projected by the cloud provider. The client of the cloud provider, in this case, is an e-business web site, and makes use of cloud facilities to host two specific applications. These applications have contrasting requirements in terms of quality-of-service (QoS), however.

1. The main use is for the web-facing online shopping application itself. This is a typical, multi-tier web application that is built using application servers, web servers, edge servers (for caches, etc) and database servers.
   Focusing on storage, the database servers use cloud storage to persist the transactions occurring due to the online shopping. These data require both reliability and availability, and thus are stored on high-end equipment. The changes made to the storage here are replicated onto another high-end storage controller, as required for the e-business' disaster recovery process. For long-term storage onto offline media, a backup to a tape library is performed on a nightly basis.
2. A less critical use for the cloud infrastructure by the same e-business, is their e-mail system. Although the data in the e-mail system are important, in terms of a cost to risk balance, only mid-range storage infrastructure is needed to house it. Again based on cost, the e-mail application has incremental backups done on a nightly basis, and full backups done weekly. The target for these backups is onto low-range storage infrastructure.

Any data centre will employ a number of administrators of various types. In this study, we consider application administrators, network administrators and storage system administrators. The cloud that hosts the e-business uses some sort of software suite to perform management tasks such as IBM's TPC. This sort of product provides salient notifications to the administrators regarding the

DOI: 10.1002/cpe

current operating conditions of the SAN infrastructure. A common alert might be something along the lines of: 'notify me when file system capacity utilization reaches 80%'. In SAN monitoring, alerts might relate to particular volume statistics, or they may relate to the conditions on a particular network port—causing an alert when utilization of a port's bandwidth rises over a determined threshold, for example.

Provisioning of resources in such an environment is only semi-automated. While some resources can be allocated automatically, others will be the result of resource requests that arrive to the administrators through request tickets, and are managed through workflows defined by the cloud provider. This is necessary to record an adequate audit-trail regarding the service level agreements between the cloud provider and its client. Examples are given below as to the type of resource requests that might be made in the course of operation of the above applications.

A storage capacity problem: The main application file system raises an alert that the utilization of storage space has risen above 80%. In this case, more storage is required. Within certain quanta, this sort of allocation can be automated (and if so is also carefully monitored), but in other cases it may require approval through a specific workflow that leads to provisioning: for example in the case that there is an accounting impact of the change.

A security-related request: The tape library that stores backups of the company e-mail system should not be participating on the storage network except when backups are in progress. This restriction can be effected using zoning to preclude paths through the Fibre Channel network to the tape library. However, there may be a need to fine-tune the window of time that is provided for the tape library to be accessed: for example in the case that the storage used by the e-mail system grows significantly.

Software updates: Either through security or application feature requests, applications are continuously updated. A request may arrive that involves the application servers having a patch applied that implements a particular new feature.

New systems: The organization might purchase a new software system that performs a type of business analytics. This application's own requirements will need to be met in terms of storage system configuration, and careful consideration will need to be given to any access that the new application requires to existing storage subsystems.

Server capacity requests: The seasonal sales drive discussed above, which causes an impact on the front-end servers, will cause the deployment of replicated servers in the application and database domains, and will probably also have an associated storage allocation increase.

When issues such as those described above arrive are raised with the administrators, either manually or through automated workflow systems, the administrators will together determine how to best incorporate the changes required. It is common for request tickets to have a connection to service level agreements, and thus to define a particular expected turn around time for the issue to be resolved. The changes that are required will be done using semi-automated planning, and may include: the installation of new servers, modification of file systems, reconfiguration of the patching of servers to the FC fabric of the storage network, creation of security zones, capacity modifications to databases, modification of the zones that are contained in the active zonesets, storage volume creation, storage volume assignment and migration of applications and data. All the changes need to be approved by the administrators through some sort of explicit validation process before they are deployed.

As a specific example, the second request described in the list above that relates to zone security will require network connectivity changes, and also potentially changes to the storage subsystem devices. However, simply shifting the time periods in which the tape library is in a particular zone may risk violating one of the well established, real-world, best practices, that 'no zone should contain both tapes and disks'.

If the cloud provider in this case was using our SAN configuration middleware, we would be able to validate the proposed changes to the tape-library zone assignment against a comprehensive library of best practices, both before and after the proposed change. We discuss this process in the following section. Beyond catching serious potential misconfigurations, by using periodic validation against this library of best practices, the cloud provider can be more confident that the

DOI: 10.1002/cpe

health of their data centre is at its best, and that they have avoided potential problems that might lead to data loss, or more general forms of sub-optimal performance.

A MIDDLEWARE FOR CONFIGURING SANS

We propose that SAN infrastructure is modified to include a middleware component, as shown in Figure 3. The middleware and other elements introduced into the SAN infrastructure support the validation of its configuration. As seen on the right-hand side of this figure, each large-scale storage system 'client organization' that manages SANs also runs an instance of a *SAN configuration middleware* that is interposed between the existing 'management applications', and the 'SAN' itself. In addition to the reconfiguration interactions within the client organization, the middleware performs external interactions with a central service that provides a *SAN best practice repository*. This is shown as 'desensitized reconfiguration request' message flows from right to left and vice versa for the relevant responses. The best practices are managed by an organization independent to the large-scale storage provider. The repository is used as a basis for the validation of SAN configurations, incorporating the aggregated knowledge of best practices that have been derived from a large set of all of the different large-scale storage users' SAN deployments. The repository is informed by a collection of data in a 'configuration log'. The aggregated data is developed into best practice rules by performing periodic data mining of the configuration log using the 'ML' component.

We have elided one detail in the above description. From the perspective of the individual users of the SAN configuration middleware, validation against best practices appears to occur at a central service. This is intended to be a logical view of the best practice repository rather than a physical view. On the one hand, having the illusion of a central repository will most easily facilitate the establishment of trust from the storage system users of the central repository. However, it would

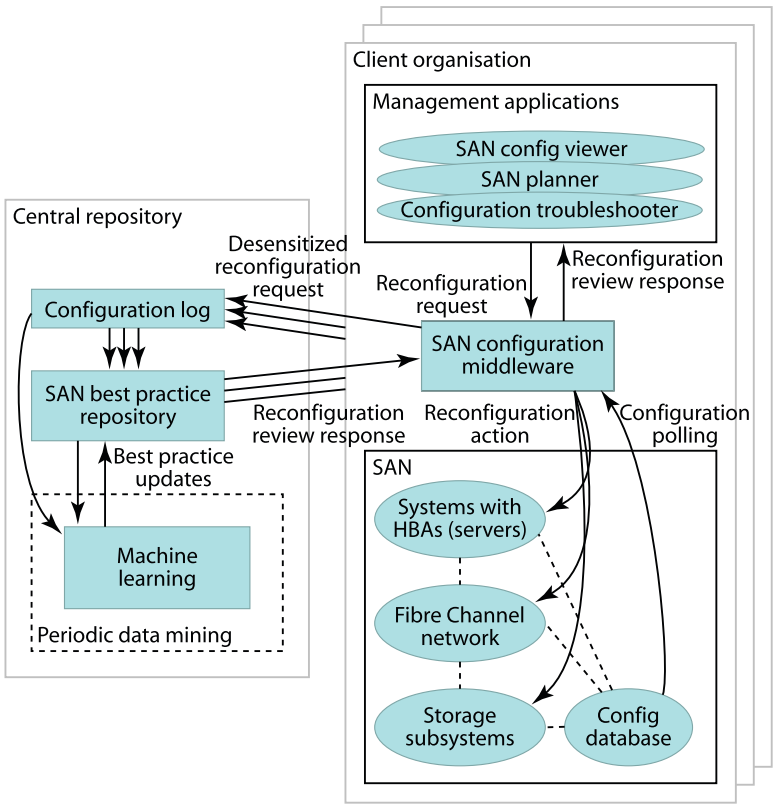

Figure 3. An overview of the SAN configuration middleware.

DOI: 10.1002/cpe

also cause scalability and reliability risks, due to being a single point of failure. We expect that a highly available, distributed deployment of the repository would be the actual means used to effect physical deployment. Having said that, if the middleware repository was not available, the SAN configuration middleware can switch to reactive validation (described below) in order to allow local operation to continue. Eventually, it may be useful to further partition the best practice repository, operating as an entirely distributed system. However, it remains an open research question as to whether the ML that needs to be done over the configuration data can operate in this environment. On the other hand, it might allow for less desensitizing of the data to need to occur: distributed participants may be able to assist distributed training without needing to disclose their configuration data.

How the middleware operates

Figure 3 indicates that the SAN configuration middleware is able to access the current configuration information from the local configuration database that is included as a subsystem of the SAN infrastructure. There are two broad modes of operation supported by the middleware in terms of configuration validation, *reactive* and *proactive*. In addition, there are also two types of participation by the large-scale storage users who run the SAN configuration middleware instances: *sharing* and *non-sharing*.

Reactive validation. In the reactive mode of validation, the middleware collects SAN configuration data periodically from the local configuration database, and sends these batches of information to the central repository for validation. The central validator determines whether any of the best practice rules have been violated. If so, these rules are collected and sent back as an alert to the management applications operating at that client's site. These alerts could be used, for example, to provide extra information to an existing local SAN configuration viewing tool, allowing it to highlight areas of potential problems in the SAN infrastructure informed from the aggregated knowledge base, as opposed to only operating on the basis of alerts raised by devices within the local SAN. The reactive validation approach has the advantage of being non-intrusive to organizations that utilize it. However, this is also a disadvantage: configurations of the SAN that are potentially unsafe can be allowed to run for some time before the alerts can be generated by the central repository.

By default, the aforementioned process documents the *non-sharing* case. If the client is additionally a *sharing* participant, then sanitized and anonymized snapshots of that SAN user's data will be stored at the central repository for subsequent detailed examination. The snapshots are tagged with an ID to indicate the organizational provenance of the data, and with timing information so that subsequent analyses can determine interactions between SAN middleware alerts and the configuration changes that were made. Participants that share their data assist learning by providing both good and bad configurations. Note that it is still the case that non-sharing clients can have their configurations validated. If there is a failure in the validation, then the non-sharing client will be asked whether they give permission for further specifics of the configuration that is causing the problem to be uploaded and examined further—potentially just by ML algorithms.

Proactive validation. The proactive validation mode of operation requires that the SAN configuration middleware be placed directly between existing SAN management applications and the actual SAN devices that are being configured—in other words, all requests to make configuration changes must be able to be intercepted by the middleware. Thus, it is possible for the middleware to ensure that all changes to the SAN configuration must be successfully validated against the best practice repository before they will be allowed. This validation would apply equally to the initial configuration parameters, and any changes that are made in response to some sort of management event (e.g. responding to resource allocation request tickets, or the reporting of an outage). Note that the use of proactive validation does not require that clients be sharing participants in the infrastructure: it is still possible for validation of all actions to occur even if the storage system user does not give permission for these data to be retained by the central repository.

DOI: 10.1002/cpe

In Figure 3, the dotted rectangle indicates that the ML component of the infrastructure runs independently from, and also in parallel to, the proactive and reactive queries from clients. This is because results from ML analyses may require that further enquiries are made back to sharing clients for information about their SAN configurations. Also, ML processes may require running for long periods of time, as a consequence of the very large search spaces of possible configurations. This will often make the timescales unsuitable for real-time decision making. Indeed, the organization that runs the central repository may want to perform carefully targeted experiments in their own labs to validate the new problem reports. If all of these stages progress, then there will be the explicit declaration of a new best practice rule. This publication will propagate to the relevant instances of the SAN configuration middleware instances.

We do not explore the ML methodology in detail in this paper—more specific examples are described in [20].

A declarative policy language [21] is used to encode the best practice rules that are stored in the central repository. For convenience, these best practice rules are divided into two main classes: *parametric* and *non-parametric* rules. Parametric rules accept particular input parameters that are used as thresholds in the rules. To help the organizations running SAN configuration middleware instances manage their analyses, notions of *scope* and *profile* are exposed. The scope defines the extent of the SAN configuration that requires validation. In a particular query, validation might centre around a single Fibre Channel fabric, it might relate to a set of Fibre Channel zones or it might involve the entire SAN configuration at the organization's data centre. A *profile* in the SAN configuration space is a collection of sets of best practices to validate for some particular scope of configuration. Profiles are provided to help simplify management, and potentially to accelerate the process of validation for large and complex SAN deployments.

When performing reactive verification, a call to an API target validateConfiguration is made, that takes as arguments Scope and Profile. The result of this call is a returned set of Violation references, which is empty if the configuration does, in fact, validate successfully. The Violation references are accompanied with indicators of the potentially responsible (or at least involved) entities, as well as what has been considered as their parent entities. As a specific example, if a Fibre Channel Port violates some best practice, the result Violation reference will indicate its encompassing HBA, the server it is connected to and also the FC switch interconnections that are involved.

Whenever the central repository performs an update of its model, or salient retraining takes place, the SAN configuration middleware at each site receives a re-validation alert if: (a) a change has been made to a best practice rule that was recently used to validate a local configuration or (b) extra rules have been added that might potentially show the current local configuration to be invalid.

Abstractions utilized over configuration parameters

Each deployment of the SAN configuration middleware will see a different underlying SAN infrastructure (i.e. the myriad different data centres of different cloud providers), each with its own custom set of configuration parameters. Transporting these data to and from the best practice repository will require the use of an abstract data format for SAN configuration snapshots. We translate the heterogeneous SAN configuration parameters into a common representation. One important side effect of the mapping functions defined to do this translation is the removal of parameter values that might disclose confidential information about the environment of particular clients.

The best practice repository stores data in an object-oriented, hierarchical format that extends the CIM/SMI-S profiles. For further details, see the DMTF Common Information Model specification [22], and the SNIA Storage Management Initiative Specification [23]. These profiles are defined by standard bodies that include most of the significant industry participants. All snapshots are tagged with timestamps and collect information about the open or unresolved problem tickets that have been raised in that particular data centre. The problem tickets also require translation into a standard format. In this case, particular classifications are required to be applied by the user raising

DOI: 10.1002/cpe

the ticket, for example whether the issue affects performance degradation, loss of data, violation of security or general problems involving access to data. The user is requested to report on the entities that they believe are involved with the problem.

Using a high-level configuration description has benefits beyond it meeting the needs of the SAN configuration middleware. Large-scale storage users can express their management goals using the same description language. In the typical sense of declarative representations, organizations such as cloud providers can use the high-level descriptions to indicate their motivation for intending to perform a particular reconfiguration, rather than having to consider the specific steps that are required to bring about the desired change. A common requirement is to make storage capacity increases—this can be expressed as such, without having to tightly couple the changes being made to the specifics of particular SAN subsystems.

The best practice repository

This section explores the way in which we manage rules within the best practice repository. We define R to be the set of IDs for all possible best practice rules. Each $r \in R$ will be an identifier that is associated with a rule written in some particular policy language. In principle, policy languages can be of unbounded complexity, and Turing complete. In practice, however, it is possible to express useful rules using much simpler policy languages. One such rule representation simply indicates the set of entities and attributes that should not feature in the same SAN configuration (mutual exclusion). This can express a real-world rule such as 'tapes and disks should not share a zone', or hypothetical examples 'Windows and Linux operating systems should not be mixed in a zone' or 'HBA firmware version X should not be used with the Solaris operating system'.

In order to describe aspects of how we manage the best practice repository, we need to fix the meaning of a number of sets that we use. We describe the sets P, E and A:

Problem types P: We define the set P to include all the general terms collected to describe problems in SAN systems. Values will include performance degradation, data access problem and data loss. The set P will include all the terms that users must refer to when lodging problem tickets, and that are used to classify bad SAN configurations.

Entities E: We define the set E to contain all the values that can describe the different potential components that are used to build a SAN. Values will include server, data-path, FC switch, IP switch, application, disk array and HBA. Although these top-level terms will be useful for description, many of them will carry extra attributes that describe aspects such as the manufacturer of a given component, the firmware version of some particular entity and so on.

Attributes A: Related to the set E above, the set A contains all the possible values for attributes that can be applied for some $e \in E$. Thus, an example value might include IBM as an indicator of the manufacturer of some particular piece of SAN equipment.

Best practice categorization

In this section, we outline some of the best practice rule representations that were found to be useful by manual, expert analysis of the results of field studies (see [20] for further details). In particular, we describe Cartesian restrictions, connectivity conditions, exclusion requirements, many-to-one and one-to-one mapping restrictions.

Cartesian: In Cartesian restrictions, given a set of values $v_1, \ldots, v_m$ for attributes $a_1, \ldots, a_m$ in A, we seek to avoid configurations in which an element e_i belonging to E satisfies

$$\bigwedge_{j=1}^{m} e_i[a_j] = v_j,$$

where square brackets are used to denote the accessing of an attribute of an entity. An example of this type of restriction would be to want to avoid all HBAs made by Vendor A of type B that are not running firmware versions f20 or f21.

DOI: 10.1002/cpe

Connectivity: For this type of constraint, if we are given an association m between two distinct entities e_a and e_b, we require that the number of instances of the association m_j exceeds some threshold value k. This type of best practice relates to requirements such as having more than one independent network path from a given host to the storage subsystems that it uses.

Exclusion: Exclusion constraints take two sets of values $v_{11}, \ldots, v_{1m}$ and $v_{21}, \ldots, v_{2m}$ for attributes $a_1, \ldots, a_m$, and seek to avoid configurations of the elements e_i and e_j (both of which belong to E) being configured such that:

$$\left(\bigwedge_{k=1}^{m} e_i[a_k] = v_{1k}\right) \wedge \left(\bigwedge_{k=1}^{m} e_j[a_k] \neq v_{2k}\right).$$

In other words, an example of such a constraint would be that tape libraries should not be contained in a zone if that zone also contains disk subsystems.

Many-to-one: In this type of rule, configurations are avoided if they do not have the same values for a set of attributes $a_1, \ldots, a_m$ across all the entities e_i within the configuration. A common use of this type of rule would be to require that a particular host computer only uses HBAs that are produced by one manufacturer, and have identical model and firmware versions across all instances.

One-to-one: Finally, in this type of rule we avoid configurations, in which the values of a set of attributes $a_1, \ldots, a_m$ is not unique and distinct for all the entities e_i that are contained within that configuration. A use of this type of rule is the requirement that ports in a storage network must all have unique port world-wide names (WWNs).

Optimizing repository searches

For the best practice repository to be useful, it is likely to contain a very large number of rules. A number of techniques can be applied to structure the repository so that this size does not preclude being able to query its data efficiently. This section explores some of the possible approaches.

We employ the definitions of the sets R, P, A and E as given above. Whatever policy language is used for r, we can significantly narrow down the candidate set $C \subseteq R$ for any given query, using relationships between $r \in C$ and sets P, E and A that we describe below. Other useful metadata for $r \in R$ includes which organization reported the rule, the organizations that have current subscriptions for notification when r changes and so on.

Given the sets P, E and A, we define three functions: $f_P : R \to \mathscr{P}(P)$, $f_E : R \to \mathscr{P}(E)$ and $f_A : R \to \mathscr{P}(A)$. These functions map each rule $r \in R$ into the relevant parts of P, E and A. It is worth highlighting that these three sets are in fact operating over data that contain a much richer structure than is encoded here (e.g. internal interrelationships). Even so, significant filtering of information will be possible and effective just using set-oriented matching, in many cases.

Consider our discussion of the best practice rule that tapes and disks should not be mixed in a zone. If a particular large-scale storage system owner does not have any tape libraries within their infrastructure, we want to avoid evaluating parts of the best practice repository that can only be grounded on the basis of a tape library being present. This can be expressed more formally as for a candidate rule set $C \subseteq R$, we have $\forall c \in C$, $\mathsf{tape} \notin f_E(c)$.

Once elements of the sets P, E and A have been collected, it is straightforward to measure the selectivity of each element with respect to particular subsets of R. This knowledge can be used to form efficient indices over R. The method that we have discussed above allows each matching criteria to be considered independently, and partial patch retrieval approaches can be applied from database research. For example, when the number of filtering attributes is small, bitmap indices may be an appropriate mechanism. In more general situations, Bloom filters have been shown to be effective in the formation of signatures over the attributes in rule sequences. Once the signatures have been collected, they can be used to rapidly determine that a block does not contain rules of interest, and thus can be skipped, tuned for a given probability of false positives occurring (lower false positive rates incur larger space requirements).

DOI: 10.1002/cpe

It is crucial that the knowledge base of best practices stays current. Updates to the repository may occur for a number of reasons, including:

- Whenever a large number of new configuration snapshots are submitted by sharing participants, there is the possible need to update existing best practice rules to reflect the salient trends in the new data. If a large number of organizations deploy the SAN configuration middleware, we expect that these updates to the repository could be rather frequent: as more and more organizations deploy different configurations, larger numbers of bugs will be reported back to vendors and manufacturers.
  If configuration snapshots and problem reports arrive at high speed, it may be possible to apply ML techniques that only perform incremental model updates. Thus, only a subset of the existing model needs to be considered. However, for techniques such as ILP, incremental learning may not be straightforward. The Progol and ProGolem systems that we used do not support incremental learning, although other ILP systems do. Even so, beyond the incremental addition of positive and negative snapshots, we may have the need to purge snapshots and their dependent rules, which may fall outside ILP systems' capacity for incremental learning. In the future, we plan to investigate whether incremental learning can be used effectively in our SAN middleware.
  Regardless of whether the complete model needs to be retrained or not in each case, the results of model update will usually only cause some subset of the rules to change. Clients of the configuration middleware will only need to react to those entries in their cached rule sets that are updated.
- The snapshots that have been used for training may eventually be considered obsolete, or might be revoked for other reasons. When this occurs, rules that used to depend on these snapshots will now need to be reevaluated without using those snapshots. Otherwise, stale knowledge might result in exerting an undue influence that is no longer applicable to current SANs. One possible illustration of this effect would be if there was a particular interaction problem between a pair of device types, but that hardware and/or software updates had mitigated the problem. The best practice rules that were previously required to protect against this SAN component bug would no longer be required (or at least should not be included for checking by default).

RELATED WORK

In the recent years, the research community has discovered log files and configuration data as a resource to troubleshoot configuration and performance problems [24, 25]. The goal is to try to solve problems that occur when validating or attempting to guide the actions of administrators. For example, the Artemis system [26] provides a framework to collect, store, analyze and visualize data from system log files. It is extensible with a plug-in mechanism to include ML algorithms for the data analysis. In contrast to our work, the focus is on data collection and visualization instead of concrete ML approaches for knowledge generation.

Systems management research has applied ML to data that are collected by management middleware. For example, ML techniques have been used to generate robust signatures for performance problems in complex distributed systems [27]. These types of automatically extracted signatures can help with root cause analysis [24]. However, configuration best practices contrast with these approaches in that best practices deal with more abstract concepts—common component types across deployments, for example. As opposed to learning patterns from a homogeneous data source, determining best practices requires the ML to derive complex relationships, and needs large amounts of training data. The performance and availability data collected over time by the current system management software tend to lead to proposed solutions that are only applicable at sites that have equivalent infrastructure.

In addition, a number of performance models have been proposed that project the QoS impact of run-time resource allocation actions in cloud contexts, as well as determining appropriate capacity

DOI: 10.1002/cpe

planning at the time of deployment [28–30]. These models can also be used as a basis for the localization and repair of performance problems. Along the lines of our focus on reconfiguration, a number of systems facilitate 'what-if' analyses to determine whether service and/or application reliability, or indeed general up-time, will be impacted by a change [31, 32]. Many of the tasks in these systems are more easily made quantitative than the tasks that we are trying to address.

There has been a large amount of work on wide-area middleware abstractions that attempt to help manage the growing complexity of heterogeneous, distributed computer systems. In Grid computing, the nature of the infrastructure makes it natural to deploy middleware, and thus management of Grid resources with middleware is commonplace. The *Globus* [33] and *OGSA* specifications [34] both provide standard sets of services that facilitate the engineering, deployment and execution of wide-area applications on Grids. As for our SAN configuration middleware, Grids comprise heterogeneous resources that must be managed by the Grid middleware. In contrast, however, Grid infrastructure tends to focus on software abstraction instead of management of the configuration of the underlying physical system resources.

Middleware systems that collect performance and diagnostic statistics from clusters of machines are also mature technologies, e.g. *Ganglia* [35]. However, unlike SAN configuration middleware, while these systems collect data from multiple deployments of their middleware, they do not reflect these measurements back to the middleware instances in order to try to improve behaviour throughout the whole, distributed system.

CONCLUSIONS

Cloud computing, and other large-scale computing users, need the support of extensible storage systems in their data centres. However, the complex and heterogeneous infrastructure makes it difficult to correctly configure overall systems with the current management tools. Our proposed solution introduces a layer of abstraction: a SAN configuration middleware. This middleware collects configuration data from each deployment site, translates it into a standard, homogeneous representation, delivers it to a centralized knowledge base, and thus detects problems or potential problems in configurations. This knowledge base—a repository of best practices—is shared between all the middleware deployments for their collective benefit. The middleware provides a uniform, high-level abstraction that can be linked to SAN management applications, making reconfiguration and troubleshooting of storage infrastructures easier and much more cost effective.

The introduction of configuration middleware into the SAN space is an important step toward the vision of general purpose, policy-based management for cloud storage infrastructure. Providing a uniform middleware abstraction across multiple data centres will enable previously impossible extents of coordination in the enforcement of configuration polices. Optimal policies can lead to cost reductions, regulation of power consumption, increases in the achievement of 'green' operation and safer maintenance of privacy protection. Meeting these desirable, but high-level policies will require tuning a large number of parameters in each large-scale storage provider. Cooperation across data centres to both generate and to consult a shared configuration knowledge base will be key to the future achievement of these important management goals.

REFERENCES

1. Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I *et al*. Above the clouds: A Berkeley view of cloud computing. *Technical Report UCB/EECS-2009-28*, EECS Department, University of California, Berkeley, February 2009. Available at: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html [May 2010].
2. Amazon Simple Storage Service (S3). Available at: http://aws.amazon.com/s3/ [May 2010].
3. Amazon Elastic Compute Cloud (EC2). Available at: http://aws.amazon.com/ec2/ [May 2010].
4. Anderson E, Spence S, Swaminathan R, Kallahalla M, Wang Q. Quickly finding near-optimal storage designs. *ACM Transactions on Computer Systems* 2005; **23**(4):337–374. DOI: http://doi.acm.org/10.1145/1113574.1113575.
5. Ward J, O'Sullivan M, Shahoumian T, Wilkes J. Appia: Automatic storage area network fabric design. *FAST'02: Proceedings of the First USENIX Conference on File and Storage Technologies*. USENIX Association: Berkeley, CA, U.S.A., 2002; 15.

DOI: 10.1002/cpe

6. Tate J, Lucchese F, Moore R. *Introduction to Storage Area Networks*. Vervante, 2006. ISBN: 0738495565. Available at: http://www.amazon.ca/Introduction-Storage-Area-Networks-Tate/dp/0738495565.
7. Sarkar P, Routray R, Butler E, Tan Ch, Voruganti K, Yang K. SPIKE: Best practice generation for storage area networks. *SYSML'07: Proceedings of the Second USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*. USENIX Association: Berkeley, CA, U.S.A., 2007; 1–6.
8. Agrawal D, Giles J, Lee KW, Voruganti K, Filali-Adib K. Policy-based validation of SAN configuration. *POLICY '04: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks*. IEEE Computer Society: Washington, DC, U.S.A., 2004; 77.
9. IBM Total Storage Productivity Center (IBM TPC). Available at: http://www-306.ibm.com/software/tivoli/products/totalstorage-data/ [May 2010].
10. IBM Systems Director. Available at: http://www-03.ibm.com/systems/management/director/ [May 2010].
11. Microsoft System Center. Available at: http://www.microsoft.com/systemcenter/en/us/default.aspx [May 2010].
12. Hewlett Packard Systems Insight Manager (HPSIM). Available at: http://h18002.www1.hp.com/products/servers/management/hpsim/index.html [May 2010].
13. EMC Control Center. Available at: http://www.emc.com/products/family/controlcenter-family.htm [May 2010].
14. Muggleton SH. Inductive logic programming. *New Generation Computing* 1991; **8**(4):295–318.
15. Srinivasan A. Aleph. Available at: http://web.comlab.ox.ac.uk/activities/machinelearning/Aleph/ [May 2010].
16. Colton S. The HR program for theorem generation. *Proceedings of CADE'02*, Copenhagen, Denmark, 2002.
17. Muggleton SH. Progol. Available at: http://www.doc.ic.ac.uk/∼shm/progol.html [May 2010].
18. Muggleton S, Santos JCA, Tamaddoni-Nezhad A. ProGolem: A system based on relative minimal generalisation. *ILP* (*Lecture Notes in Computer Science*, vol. 5989), Raedt LD (ed.). Springer: Berlin, 2009; 131–148.
19. Case studies in cloud computing. Available at: http://www.informationweek.com/cloud-computing/blog/archives/2008/09/case_studies_in.html [May 2010].
20. Routray R, Zhang R, Eyers DM, Willcocks D, Pietzuch P, Sarkar P. Policy generation framework for large-scale storage infrastructures. *POLICY '10*: *Proceedings of the 11th IEEE International Symposium on Policies for Distributed Systems and Networks*. IEEE Computer Society: Fairfax, VA, U.S.A., 2010.
21. Gopisetty S, Agarwala S, Butler E, Jadav D, Jaquet S, Korupolu M, Routray R, Sarkar P, Singh A, Sivan-Zimet M, Tan C-H, Uttamchandani S, Merbach D, Padbidri S, Dieberger A, Haber EM, Kandogan E, Kieliszewski CA, Agrawal D, Devarakonda M, Lee K-W, Magoutis K, Verma DC, Vogl NG. Evolution of storage management: Transforming raw data into information. *IBM Journal of Research and Development* 2008; **52**(4):341–352.
22. DTMF. Common Information Model (CIM). Available at: http://www.dmtf.org/standards/cim [May 2010].
23. Storage Management Initiative Specification (SMI-S). Available at: http://www.snia.org/forums/smi/tech_programs/smis_home/ [May 2010].
24. Cohen I, Zhang S, Goldszmidt M, Symons J, Kelly T, Fox A. Capturing, indexing, clustering, and retrieving system history. *SIGOPS Operating Systems Review* 2005; **39**(5):105–118. DOI: http://doi.acm.org/10.1145/1095809.1095821.
25. Kiciman E, Maltz D, Platt J, Goldszmidt M. Mining web logs to debug distant connectivity problems. *ACM SIGCOMM Workshop on Mining Network Data* (*MineNet*), Pisa, Italy, 2006.
26. Cretu G, Budiu M, Goldszmidt M. Hunting for problems with Artemis. *USENIX Workshop on the Analysis of System Logs* (*WASL*), San Diego, CA, 2008.
27. Bodik P, Goldszmidt M, Fox A. HiLighter: Automatically building robust signatures of performance behavior for small- and large-scale systems. *Usenix Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, Cambridge, MA, 2007.
28. Zhang Q, Cherkasova L, Smirni E. A regression-based analytic model for dynamic resource provisioning of multi-tier applications. *ICAC '07*: *Proceedings of the Fourth International Conference on Autonomic Computing*. IEEE Computer Society: Washington, DC, U.S.A., 2007; 27. DOI: http://dx.doi.org/10.1109/ICAC.2007.1.
29. Zhang S, Cohen I, Symons J, Fox A. Ensembles of models for automated diagnosis of system performance problems. *Proceedings of the International Conference on Dependable Systems and Networks*. IEEE Computer Society: Washington, DC, U.S.A., 2005; 644–653. DOI: http://dx.doi.org/10.1109/DSN.2005.44.
30. Barham P, Donnelly A, Isaacs R, Mortier R. Using Magpie for request extraction and workload modelling. *Proceedings of the Sixth Symposium on Operating Systems Design and Implementation* (*OSDI'04*), San Francisco, CA, 2004.
31. Su YS, Huang CY. Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. *Journal of Systems and Software* 2007; **80**(4):606–615. DOI: http://dx.doi.org/10.1016/j.jss.2006.06.017.
32. Herbrich R, Graepel T, Murphy B. Structure from failure. *Proceedings of the 2nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, Cambridge, MA. USENIX Association: Berkeley, CA, U.S.A., 2007; 10:1–10:6.
33. Foster I. Globus toolkit version 4: Software for service-oriented systems. *IFIP International Conference on Network and Parallel Computing* (*Lecture Notes in Computer Science*, vol. 3779). Springer: Berlin, 2005; 2–13.
34. Foster I, Kesselman C, Nick J, Tuecke S. The physiology of the grid: An open grid services architecture for distributed systems integration, 2002. Available at: http://www.globus.og/research/papers/ogsa.pdf [May 2010].
35. Massie ML, Chun BN, Culler DE. The Ganglia distributed monitoring system: Design, implementation and experience. *Parallel Computing* 2004; **30**:817–840.

DOI: 10.1002/cpe