SPIKE: Best Practice Generation for Storage Area Networks

Prasenjit Sarkar[1], Ramani Routray[1], Eric Butler[1], Chung-hao Tan[1], Kaladhar Voruganti[1], Kiyoung Yang[2]
[1] IBM Almaden Research Center [2] University of Southern California

Abstract

This paper presents SPIKE, an automated algorithm to generate best practices by analyzing Storage Area Network (SAN) configuration errors. Best practices are a useful tool in problem diagnosis as most configuration problems are caused by the violation of best practices in the storage network domain. However, the manual generation of best practices is tedious, error-prone and costly in terms of time and manpower. SPIKE uses a combination of information-retrieval principles, entity ranking and decision-tree classification to statistically infer the best practices for the prevention of SAN configuration problems. Preliminary results from an initial implementation of SPIKE indicate speed and accuracy improvements over manually generating best practices.

1 Introduction

Adherence to best practices is essential for successful configuration and deployment of complex systems. In such scenarios, experts rely on experience as well as repositories of best practice guidelines to proactively prevent any configuration problems while deploying a system in a data center. The reasons for this are not hard to find. First, the entities in a data center are associated via complex physical and logical relationships, which may number an order of magnitude more than the entities themselves. Second, the diagnosis of a problem requires the collection, filtering, and correlation of a huge amount of data from various sources over different periods of time. Third, most data center deployments are not integrated enough to present a single, usable text or graphical interface to navigate through the problem diagnosis process. Instead, a system administrator must correlate data across multiple point tools using techniques that are tedious, time-consuming and prone to errors. Fourth, the technology in a data center is continually evolving primarily due to the need for product differentiation. This hampers the ability of system administrators to diagnose configuration problems because their expertise lags behind the state of the art. Finally, the standardization process that dictates inter-vendor interoperability may be immature and continuously evolving leading to hard-to-diagnose configuration problems.

This paper addresses the problem of automatically generating best practices for a particular system domain. Automation is essential because the manual generation of best practices is costly. In most system domains, it requires a large team to analyze large sets of configuration data using rudimentary tools to generate the best practices for that system domain. The generation of the best practices using manual methods requires many man-years of data gathering and analysis. In the case of storage subsystems, IBM has a Storage Area Network (SAN) Central team, whose job is to examine all known storage area network configuration issues and come up with best practices. These best practices have helped the SAN Central team reduce the time required in resolving configuration errors from 2 weeks to 2 days; as 80% of the configuration problems are caused by the violation of best practices. However, the generation of the best practices required 20 man-years of data gathering and analysis. More importantly, the best practices are dynamic and change as new technology is introduced.

Due to the many challenges in generating best practices for a domain, there needs to be a systematic way to gather all the data needed for problem diagnosis. The success of any automated mechanism is determined by the quality and quantity of the data. Since data sets have a large number of entities, attributes and associations that can number in the hundreds of thousands, there is a need for effective dimensionality reduction so that the data sets can be analyzed efficiently. Finally, system administrators want the purest subset of entities, attributes and associations that contribute to a configuration error. This would require the use of a highly accurate data classification tool that can overcome incomplete dimensions as well as noise in the data sets.

The key contribution of this paper is an automated algorithm to generate best practices for SAN configuration problem, termed SPIKE (Storage Practices usIng Knowledge Engineering). SPIKE uses a combination of information-retrieval principles, entity ranking algorithms and decision-tree classification to statistically infer the best practices relevant to a set of SAN configuration problems.

The rest of the paper is organized as follows: Section 2 describes the domain in greater detail and categorizes the typical SAN configuration problems obtained based on field data. Section 3 describes the algorithm to construct the best practice for a SAN configuration problem. Section 4 describes preliminary results from an initial implementation. We present related work in Section 5 and conclude in Section 6.

2 SAN Configuration Problem

This section describes the Storage Area Network model with a brief description of the core entities and associations. This is followed by a description of our categorization of SAN problems. Then we define the elemental structure for a best practice based on this categorization and use this definition to lay the foundation of the key steps in SPIKE.

The goal of a storage area network is to provide block storage as a service over a typically dedicated network to attached hosts. The three principal entities are hosts, switches and subsystems. A host typically has one or more Host Bus Adapters with ports that are connected to switch ports via fibre cables. Switches, which provide connectivity between the hosts and storage devices, can be connected to each other in a complex topology, depending on the scale of the Storage Area Network. Storage subsystems and tape libraries connected to switch ports provide block storage access in the form of volumes to the hosts via access control lists.

While the above represents the core domain model, more advanced storage area networks such as those with virtualization at different levels can exist. For example, in-band virtualization device present in the network. Our storage area network model does account for most of other dimensions of complexities.

Each entity in the storage area network has attributes of these types:

- *Direct attributes:* These are inherent properties of the entity in the storage area network. In the case of a host, the direct attributes include: IP addresses, hostname, operating system, memory size among others.
- *Associations:* These are properties that are based on associations present in the storage area network. The key difference between an association and a direct attribute is an association links multiple types of storage network entities.
- *Derived Attributes:* These are properties that are logically derived from the direct attributes and associations present in the storage area network. These derivations are principally made to compose the best practice for a SAN configuration problem. However, if one were to automatically generate best practices for a SAN configuration problem, the identity of the derived attributes is not known in advance. For example, number of members in a zone is a derived attribute.

Now that we have a better understanding of the storage area network domain, next an overview of the typical problems that affect a storage area network is given. This is achieved by collected the different types of configuration problems associated with a SAN and use a classification mechanism to categorize the problems. At first glance, it may seem that the conditions in the SAN configuration problems do not present a structure for categorization. However, upon closer inspection, one can gain more insight by analyzing the Boolean expression representing the best practices corresponding to the SAN configuration problems. Based on this categorization technique, we grouped the best practices for SAN configuration problems into five categories [13].

To explain the five categories in detail, we assume a set of entities E with elements e_1 .. e_n, each with attributes a_1 .. a_t. We also assume associations A with instances A_1 .. A_k where each association A_1 groups a subset of entities in E. The categories that represent the best practices are:

- **Cartesian:** Given set of values v_1 .. v_m for attributes a_1 .. a_m, avoid configurations where an element e_i belonging to E satisfies $e_i.a_1 = v_1$ & .. & $e_i.a_m = v_m$. For example, avoid all HBAs of Emulex type 9002 that do not have firmware levels of 3.81a, or 3.81b.
- **Connectivity:** Given an association A_i, avoid configurations where the number of instances of the association A_i between two entities e_a and e_b does not exceed a certain threshold k. e.g., avoid all configurations where a host does not have at least 2 network paths to a storage subsystem.
- **Exclusion:** Given a set of values v_{11} .. v_{1m} and v_{21} .. v_{2m} for attributes a_1 .. a_m, avoid configurations of elements e_i and e_j belonging to E that satisfy $e_i.a_1 = v_{11}$ & .. & $e_i.a_m = v_{1m}$ & $e_j.a_1 \neq v_{21}$ & .. & $e_j.a_m \neq v_{2m}$. e.g., tape drives should not exist in a zone if it contains disk drives.
- **Many-to-one:** Avoid configurations where the value of a set of attributes a_1 .. a_m is not the same for all entities e_i in an instance of an association A_k. e.g., all host computers in the same storage network zone should have the same operating system type.
- **One-to-one:** Avoid configurations where the value of a set of attributes a_1 .. a_m is not different and unique for all entities e_i in an instance of an association A_k. e.g., all ports in a storage network fabric must have a different port world-wide name (WWN).

At first glance, some of the operators in the categories may seem unduly restrictive, but they represent the known universe of SAN configuration best practices.

As can be seen from the categorization of the best practices for SAN configuration problems, we need to identify three important elements in the construction of the best practice for a given SAN configuration problem:

- *Associations*: An association that is responsible for the SAN configuration problem.

- *Attributes*: The entity attributes and their corresponding values that are responsible for the SAN configuration problem.
- *Derived Attributes:* The derived attributes and their corresponding values that are responsible for the SAN configuration problem.

All the three elements may not be present in a best practice. As seen from the categorization, the presence of an association in the best practice is optional. However, there must be at least one attribute-value pair or one derived attribute-value pair in the best practice. Other than this, there are no restrictions on the composition of the best practice. The next section describes the steps needed to generate these elements of a best practice for this domain.

One key thing to note is that statistical best practice generation is very specific with respect to versions and does not tend to implicate all versions of an entity. This protects against situations where configuration problems fixed in later versions may invalidate a best practice.

3 The SPIKE system

SPIKE uses of two important stages for generating best practices for SAN configuration problems. The first stage is the collection of configuration data and problem reports from the various SAN deployments. The second stage is the actual generation of best practices by analyzing the collected data. The second stage is composed of three important sub-stages that are mandated by the nature of the best practices: (i) association identification, (ii) attribute identification (feature selection), and (iii) derived attribute identification. Each of these three sub-stages is detailed in the following sections with examples. It is important to note that SPIKE is generic in nature and not tailored towards any of the types of configuration problems listed in Section 2.2. Thus, the discovered associations, attributes and derived attributes are used in the generation of the best practice for a given SAN configuration problem.

3.1 Data Collection

The first stage in SPIKE is collect data from multiple SAN deployment sites. When a SAN administrator reports an irresolvable problem to the help desk, the help desk personnel advises the administrator to provide a current snapshot of the database of the storage resource management (SRM) tool managing the SAN. The SAN administrator also provides a problem report that is associated with the snapshot of the database. When there is a sufficient number of problem reports and attached database snapshots, the best practice generation tool is applied to the collected data in the warehouse and likely best practices to prevent the reported problems are produced.

3.2 Generation Algorithm

As mentioned earlier, the second stage is split into three sub-stages, with each sub-stage focusing on a different element of the best practice. For the purpose of the description, we assume that there is a SAN configuration problem P that is reported that affects a section of storage network entities that are a proper subset of E.

3.2.1 Association Identification

The basis of association identification is to use information-retrieval metrics to identify the most promising association indicative of a SAN configuration problem.

The identification of the association proceeds in the following steps:

1. SPIKE examines every SRM snapshot database corresponding to a problem P.
2. For each such snapshot database, SPIKE calculates the set of entities that are reporting problems. Let us call this set E1.
3. From the entity-relationship model in the snapshot database, SPIKE generates the list of associations A.
4. For every association a in the list of associations A, SPIKE generates a list of the instances of the association, aInst.
5. For every instance $aInst_i$ in the list of associations aInst, SPIKE generates a set of entities corresponding to that association instance called $E(aInst_i)$.
6. Following this generation, SPIKE merges two entity sets $E(aInst_i)$ and $E(aInst_i)$ if the two association instances are independent. The merger step gives more statistical significance to association instances with a smaller number of members. We define two association instances to be independent if the instances do not have behavioral implications on each other in terms of the storage area network operations.
7. Following this merger process, SPIKE comes up with a new list of set of entities called $E_{merge}(aInst)$.
8. For each instance of the set $E_{merge}(aInst)$, SPIKE applies the metrics of precision and recall with respect to the set E1 (calculated in the second step) and rank the instances based on the calculated metrics.
9. SPIKE chooses the top instance of the set $E_{merge}(aInst)$ and for that instance, report the relevant associations as the most promising for the particular SAN configuration problem P.

In Section 2.2, it was stated that it is possible to generate a best practice without an association. Therefore, SPIKE adds the null association to the list of associations

considered for the algorithm. The null association trivially represents all the entities in E and a selection of the null association as the top-ranked association implies the absence of an association in the best practice.

The next critical sub-stage is to identify the attributes and their corresponding values that may be related to the SAN configuration problem.

3.2.2 Attribute Identification

SPIKE uses a classification approach in order to determine the purest subset of attributes that are indicative of a SAN configuration problem P. To aid in the classification process, SPIKE uses decision trees primarily because the results are easily interpretable. The relatively deterministic nature of the domain of interest biased the design of SPIKE against the use of naïve Bayesian classifiers.

However, there is a significant challenge in using a decision tree in this case. The key challenge is there are multiple types of entities with disjoint attribute sets, while decision trees typically take a single entity type as input.

To alleviate this problem, SPIKE can combine all the entity types into an aggregated entity type using database joins. However, a naïve approach of joining all entities is going to result in an extremely large number of instances for the aggregated entity.

The next challenge is determining if SPIKE can do better than a full aggregation of all entity types. To achieve this, SPIKE uses a ranking metric for all entity types based on their relevance to the reported SAN configuration problem. Then this ranking metric used to decide the number and order of database joins to perform on the entity types.

In this step, SPIKE computes the ranking of every entity type based on the concept of information entropy. In canonical terms, the information entropy of an event x is the sum, over all possible outcomes i of x, of the product of the probability of outcome i times the log of the inverse of the probability of i (which is also called i's surprisal - the entropy of x is the expected value of its outcome's surprisal). Formally, this is represented as:

$$H(x) = \sum_{i=1}^{n} p(i) \log_2 \left(\frac{1}{p(i)} \right)$$

Next, the concept of information entropy is used in ranking the various entity types. We define a data path between two entities as a list of association instances that can be used to perform a database join of the entities in question. Trivially, there exists a data path between an entity and itself through the null association.

For each entity, SPIKE computes the number of data paths from entities in as well as outside the set of entities E. As defined earlier, the set E represents the collection of entities reporting the SAN configuration problem. We define a true data path as one to an entity in the set E and a false data path as one to an entity outside the set E.

SPIKE uses the data path statistics to define the entropy for an entity type T. Assume that there are P true and N false data paths in the entity type T. Let us assume that instances of the entity type T divide the above-mentioned data paths into k partitions, each containing P_i true and N_i false data paths, where i varies from 1 to k.

$$Entropy(P,N) = -\left(\frac{P}{P+N} \times \log_2 \frac{P}{P+N} + \frac{N}{P+N} \times \log_2 \frac{N}{P+N} \right)$$

Based on the above, SPIKE computes the information gain accruing from the entity type T as follows:

$$Gain(T) = H(T) - \sum_{i=1}^{k} \frac{P_i + N_i}{P+N} \times Entropy(P_i, N_i)$$

Based on this concept of information gain, SPIKE follows these steps to identify the attributes and values of interest:

1. Entity types are ranked in order of their computed information gain metric: Gain(T) for an entity type T.
2. The top-ranked entity type is chosen in terms of information gain as the current entity type and applies the decision tree algorithm on the current entity type.
3. The current entity type is joined with the next-ranked entity type that is connected to the current entity type to form the new current entity type and the decision tree algorithm is applied to the new current entity type.
4. If there is no increase in classification accuracy in step 3, the classification process is halted and SPIKE moves on to step 5. Otherwise, step 3 is repeated to continue the classification process.
5. The classification process results in is a list of attributes and values belonging to various entity types. These attributes and values are used in the construction of the best practice relevant to the SAN configuration problem P.

When, the set of attributes and values is not enough to construct the best practice for a SAN configuration problem P, derived attributes and their corresponding values are generated.

3.2.3 Derived Attribute Identification

The derived attribute identification step is based on the principle of generating derived attributes by analyzing the associations in the storage network domain. Given a SAN

configuration, there are a number of associations as described in Section 2.1. We assume that each association can be represented as a derived entity in the entity-relationship model corresponding to the storage network domain. This implies that there is a table corresponding to each association in the input configuration data. Therefore, every association type can also be input into the decision-tree classifier for derived attribute identification.

The derived attribute generation algorithm is invoked when SPIKE considers an association type for joining into the input table for classification (Step 2 or Step 3 in Section 3.2.2). When such an association type is selected, SPIKE adds summary information on the entities involved in the association that are already part of the input table. This summary information is considered to be the derived attributes for the association type.

For each entity type identified in the previous step that is linked with an association type, SPIKE computes the summary information on each of the attributes in the entity type. Typically, the these six aggregate functions have been employed for computing the summary; average, min, max, count, count distinct and sum. In typical SAN configurations, however, most of the attributes are categorical attributes. Hence, SPIKE utilizes count and count distinct aggregate functions.

SPIKE first obtains the distinct values for each attribute, then the following two attributes for each distinct value:

- *count attribute:* the count of data instances with each distinct value is stored. The name of this attribute is N_<distinct value>
- *boolean existence attribute*: the existence of the distinct value within the association is stored. When the data is sparse, this attribute helps the classification technique to correctly obtain the best practices. The name of this attribute is HAS_<distinct value>

4 Preliminary Results

In this section, we perform an exhaustive evaluation of an initial implementation of SPIKE with data synthetically generated from a fault injection routine. While synthetic data cannot substitute for real-world data, the evaluation provides an initial validation of SPIKE. To this end, we injected SPIKE with problem reports under various conditions and observed the behavior of SPIKE in terms of correct best practices generated as well as false positives. The problem reports not only contain the entities reporting problems but also the rest of the SAN configuration that may be working correctly. Both the problematic and non-problematic data is used for decision tree classification. Another thing to note in interpreting the data is it takes 2000 problem reports for an expert team to come to similar conclusions.

In the evaluation of SPIKE, we found that the association identification was able to correctly identify the relevant association within 10 problem reports and was relatively less sensitive to environmental factors than the attribute identification. Consequently, the major emphasis of our evaluation is focused on the generation of the correct attribute and value pairs for the sake of brevity.

In one experiment, we measure the sensitivity of SPIKE to the number of faults of the same type presented in the input SAN configuration data. We vary the number of problem reports as input to SPIKE from 50 to 500 in steps of 50. The fault injection routine introduces four different problems of the Cartesian type into the problem reports and we observe the number of problem reports required by SPIKE to generate the problem reports. We choose the Cartesian type as the default problem type because the number of attributes and values in this type typically exceeded those in the other problem types. As can be seen in Figure 1, SPIKE cannot generate any best practice using up to 150 problem reports. Following this, the number of best practices generated slowly increases until SPIKE is able to generate all four best practices using 500 problem reports. The number of false positives decline gradually with the increase in the number of problem reports.

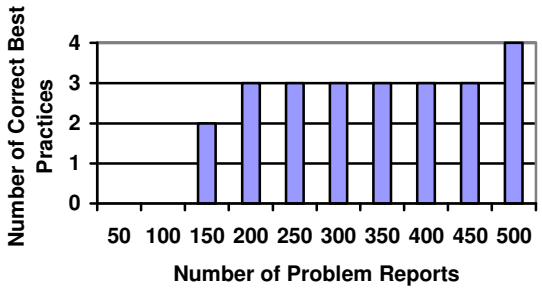

Figure 1. Sensitivity to the number of problems of the same problem type

In a related experiment, we also vary the number of different problems of the same problem type from 1 to 4. We observed in this experiment, that the number of problem reports required to find the first two best practices varies from 150 to 200 and to find the third varies from 250 to 300. As before, SPIKE finds the fourth best practice when there are 500 problem reports.

In another experiment, we measure the sensitivity of SPIKE to partial and complete mis-reporting of SAN configuration problems by system administrators. For the partial mis-reporting case, we vary the number of problem reports as input to the best practice generator from 50 to

500 in steps of 50. The fault injection routine introduces a single problem of the Cartesian type into the problem reports selectively erases a subset of the entities reporting a problem based on the degree of misreporting. In this experiment, we vary the percentage of misreporting from 0% to 50% in steps of 10%. SPIKE is able to correctly predict the best practice within 150 problem reports even with an error rate of 20%. The number of required problem reports climbs to 200 with an error rate of 30%, and the algorithm is unable to generate the best practice with 500 problem reports at an error rate of 40% or more. For the complete misreporting case, SPIKE is able to correctly generate the best practice within 150 problem reports with an error rate of up to 10% and 200 problem reports with an error rate of 20%. If the error rate is increased beyond this, SPIKE is unable to generate the correct best practice even with an input of 500 problem reports.

5 Related Work

Recent research has focused on the black-box approach in order to handle the complexity and the non-deterministic behaviors of the managed system. One typical setup is the symptom-based approach. By observing system states from the results of probes, a symptom-based approach tries to predict the cause of failure from past experience. Unsurprisingly, machine learning techniques are adopted in most recent papers in the black-box approach. Among those various learning algorithms, Decision Tree [4], Naïve Bayes Classifier [12, 9, 7], Bayesian Network [5, 14] and Clustering [3, 6, 10, 9] are widely used due to their advantages of interpretability and modifiability [6, 17].

From the perspective of machine learning, failure diagnosis can be viewed as an anomaly detection problem [9, 10, 17], particularly if the majority of the training samples define good cases. The concept of feature selection and model selection are also explored in several recent papers [12, 18]. A history-based approach is another useful technique which reduces the solution space significantly [16, 17]. Diagnosis of configuration problem has been studied in several areas such as Windows Registry [16, 10, 8], router configuration [7] and general Internet application [11]. Other recent research activities focus on performance problems [1, 6], software failure [3, 4] or general fault localization techniques [14, 12, 2].

6 Conclusions

Best practices are a useful tool in problem diagnosis as most configuration problems are caused by the violation of best practices in the storage network domain. The paper presents SPIKE, an automated best practice generation tool that uses a combination of information-retrieval principles, entity ranking and decision-tree classification to statistically infer the best practices for SAN configuration problems. We have presented preliminary results from an initial implementation of SPIKE and hope to validate SPIKE with real world problem reports.

References

[1] M. K. Aguilera, P. Reynolds, and A. Muthitacharoen. Performance debugging for distributed system of black boxes.
In Proc. 19th ACM SOSP, October 2003.

[2] A. Beygelzimer, M. Brodie, S. Ma, and I. Rish. Test-based diagnosis: Tree and matrix representations. In Proc. 9th
IFIP/IEEE IM, May 2005.

[3] M. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer. Pinpoint: Problem determination in large, dynamic internet services. In DSN, June 2002.

[4] M. Chen, A. X. Zheng, J. Lloyd, M. Jordan, and E. Brewer. Failure diagnosis using decision tree. In Proc. 1st IEEE
ICAC, May 2004.

[5] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. S. Chase. Correlating instrument data to system states: A
building block for automated diagnosis and control. In Proc. 6th USENIX OSDI, December 2004.

[6] I. Cohen, Z. Steve, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox. Capturing, indexing, clustering, and retrieving system. In *Proc. 20th ACM SOSP*, October 2005.

[7] K. El-Arini and K. Killourhy. Bayesian detection of router configuration anomalies. In *Proc. ACM SIGCOMM Workshop on Mining Network Data*, August 2005.

[8] A. Ganapathi, Y.-M.Wang, N. Lao, and J.-R.Wen. Why pcs are fragile and what we can do about it: A study of windows registry problems. In *DSN*, June 2004.

[9] G. Hamerly and C. Elkan. Bayesian approaches to failure prediction for disk drives. In *Proc. 18th ICML*, June 2001.

[10] E. Kiciman and Y.-M. Wang. Discovering correctness constraints for self-management of system configuration. In
Proc. 1st IEEE ICAC, May 2004.

[11] K. Nagaraja, F. Oliveria, R. Bianchini, R. P. Martin, and T. D. Nguyen. Understanding and deailing with operator
mistakes in internet services. In *Proc. 6th USENIX OSDI*, December 2004.

[12] I. Rish, M. Brodie, and N. Odintsova. Real-time problem determination in distributed system using active probing. In *Proc. 9th IEEE/IFIP NOMS*, April 2004.

[13] Dakshi Agrawal, James Giles, Kang-Won Lee, Kaladhar Voruganti, Khalid Filali-Adib: Policy-Based Validation of SAN Configuration. POLICY 2004: 77-86

[14] M. Steinder and A. S. Sethi. End-to-end service failure diagnosis using belief networks. In *Proc. 8th IEEE/IFIP NOMS*, April 2002.

[15] P.-N. Tan, S. Michael, and K. Vipin. *Introduction to Data Mining*. Addison Wesley, 2006.

[16] Y.-M. Wang, C. Verbowski, J. Dunagan, Y. Chen, H. J. Wang, C. Yuan, and Z. Zhang. Strider: A black-box, statebased approach to change and configuration management and support. In *Proc. 17th USENIX LISA*, Oct 2003.

[17] A. Whitaker, R. S. Cox, and S. D. Gribble. Configuration debugging as search: Finding the needle in the haystack. In *Proc. 6th USENIX OSDI*, December 2004.

[18] S. Zhang, I. Cohen, M. Goldszmidt, J. Symons, and A. Fox. Ensembles of models for automated diagnosis of system performance problems. In *DSN*, June 2005.