Policy-Based Validation of SAN Configuration

Dakshi Agrawal, James Giles, Kang-won Lee, Kaladhar Voruganti, Khalid Filali-Adib

with help and encouragement from
Mandis Beigi, Seraphin Calo, Sandeep Gopisetty, Dilip Kandlur, David Olshefski, Dinesh Verma, and other colleagues from IBM IGS, Research and Tivoli.
Outline

- Problem Statement and Proposed Solution
  - Background on SAN, Policies
  - Configuration Validation Using Policies

- SAN Configuration Validator
  - Architecture
  - Data Model
  - Aspects of policy management

- Conclusion and Future Work
Overview: Storage Area Network (SAN)

- Workstation
- DB server
- Streaming server
- File server
- Fabric switch
- Disks
- Optical disks
- Disks arrays
- Tape subsystem
- Arbitrated loop
Problem Statement

- **SAN management is a difficult task!**

- **Field representatives have designed “recipes” or best practices for SAN deployment**
  - goal is to minimize configuration problems

- **Provide an automated solution to audit a SAN configuration**
  - encourage more economical SAN designs
  - encourage consolidation of best practice
Examples of SAN Interoperability Problems

- Switches are not interoperable with each other
  - Switches from vendor X and Y cannot be cascaded

- Different OS hosts cannot be put in the same zone
  - Windows and Linux hosts cannot be in the same zone

- Software compatibility
  - Multi-pathing software from storage vendor X on host A will not let you see the storage from vendor Y
Solution: Policy-Based SAN Configuration Validation

- Policy infrastructure can be used to manage interoperability constraints:
  - updating, distributing, evaluating

- Allows different actions to be taken
  - actions depend on type of interoperability problems

- Policy infrastructure can be used by other components of SAN management software.
  - virtualization, backup, storage planning
Policy Model

- **Scope**
  - A context or domain within which the policy applies. Examples:
    - “SAN Configuration Validation:IntraHBA”
    - “Storage Access Control:FileAccess”
    - “Storage Allocation:DiskQuota”

- **Condition**
  - Specifies when a policy is to be applied. Condition is specified as a Boolean expression. The “if clauses” within a policy.

- **Decision**
  - A policy guidance. The “then clause” within a policy.

- **Priority**
  - A statement of priority, expressed as an integer.
Example Policy for SAN Configuration Validator

- **Scope**
  - “SAN Configuration Validation”
  - “IntraHBA”

- **Condition**
  - If (hba.Manufacturer.Name = “QLogic”) AND (hba.firmware.level < 3.41)

- **Decision**
  - Two name-value pairs:
    - synopsis= “QLogic HBAs should have firmware level greater than or equal to 3.41”. HBA with id: ” + hba.id + “ violates this policy.”
    - severity = ERROR

- **Priority**
  - currently not used
How it works (walk through the validation process)

Policy Specification Tool -> Remote Centralized Policy Database

Local Policy Database -> Policy Evaluator

Policy Event Generator -> SMI-S Database interface

SAN Database -> SAN

Action Handler
Main policy scope identifies a broad area of policy application.
- policy evaluator can be configured to follow different modes of operation for different main scopes, e.g. choosing an evaluation engine, solicited/unsolicited.

Subscope establishes a narrower context under which policies are written, distributed, and evaluated.
Subscope

- **Subscope establishes a context for policies**
  - policies under subscope “InterHostHBA” should be related to properties of a host and a resident HBA.

- **Subscope limits the data that can occur in a policy**
  - Event generator needs to send only relevant data to the evaluator
  - a policy author may find that none of the defined subscopes are suitable for a particular policy
Defining a policy scope taxonomy

1. Collect sample policies to be implemented as plain English statements.

2. Model subscopes and corresponding data available for policies. SMI-S is a good starting point for SANs.

3. Translate plain English statements to policy 4-tuple using the data model.

4. Iterate between Step 2 and Step 3, until a satisfactory list of subscopes has been achieved.
SAN Configuration Policy Analysis

- Collected a comprehensive set of SAN configuration rules of thumb used by field experts.

- Analyzed the collected rules and extracted 64 representative policies.

- Defined policy language constructs to express these policies (in addition to traditional operators such as Numeric, Boolean, String operators).
Subscopes for SAN Configuration Validation

- One subscope for each component
  - HBA, Host, Storage Device, Switch, Link, Zone etc.

- One subscope for each pair of associated components
  - HostHBA, HBASwitch etc.

- Subscopes for special collections of components
  - AllSwitches, AllHosts, etc.
Policy Aspects

▪ **Policy language**
  – XML based: ease of parsing, compatible with Web Services
  – Seamlessly extensible: addition of new functions and data types

▪ **Collection policies**
  – requires adding new functions to the policy language

▪ **Two evaluation engines**
  – general purpose engine
  – hyperspace-based fast engine

▪ **Policy evaluator is capable of gathering data required for policy evaluation**
Collection Operators

- We determined five collection operators are necessary to fully express SAN policies.
  
  1. Given a collection C with elements O1, … On, where each element has m attributes p1, …, pm.

  2. Cartesian operator: Given sets of values A1, …, Am for attributes p1, …, pm, respectively, return all elements Oi that satisfy (Oi.p1 ∈ A1) AND … (Oi.pm ∈ Am).

  3. Exclusion: Given sets of values A1, …, Am and B1, …, Bm, return all elements in C that satisfy (Oi.p1 ∈ A1) AND … (Oi.pm ∈ Am) and (Oj.p1 ∈ B1) AND … (Oj.pm ∈ Bm)

  4. Many-to-One: Test if the value of an attribute pi be the same for all elements in C.

  5. One-to-One: Test if the value of an attribute pi be different for all elements in C.

  6. Graph: Given a directed graph G = (E, C) and two elements Oi and Oj, return all directed paths between them.
How it works (walk through the validation process)

1. Policy Specification Tool
2. Remote Centralized Policy Database
3. Local Policy Database
4. Policy Evaluator
5. Action Handler
6. Policy Event Generator
7. SMI-S Database Interface
8. SAN Database
9. SAN
Related Work

- EMC Storage Architect and CA BrightStor SAN Designer have SAN configuration checking tools
- PCIM, SNIA SMI-S standards
- This approach is compatible with different policy evaluation engines
- Ongoing work on policy based management for IP network Security and QoS services
Conclusion and Future Work

- Incorporating configuration validator features into other storage management functions:
  - storage capacity planning tools
  - storage provisioning tools

- Working on how to scale this system across thousands of devices with thousands of policies

- Extending this work to check for inconsistencies across integrated application/database/file system/SAN deployment
Examples of SAN Best Practice Rules

- There should be at least 2 paths from the host to the storage controller.
- Hosts should be connected to department class (16 port) switches.
- Storage Controllers should be connected to director class (64 port) switches.
- The same HBA should not be used to access both tape and disk storage.