CIMDIFF: Advanced difference tracking tool for CIM compliant devices

Ramani Routray
IBM Almaden Research Center
routrayr@us.ibm.com

Shripad Nadgowda
IBM India Systems and Technology Lab.
shripad.nadgowda@in.ibm.com

Abstract

Total Cost of Ownership (TCO) for any enterprise scale data center is significantly dependent upon the effectiveness of the system management solutions and procedures deployed. Complexity of managing a data center increases as various enterprise applications demand diverse sets of requirements, leading to a very heterogeneous environment often fueled by diverse emerging technologies. Emergence of the industry standard Common Information Model (CIM) has introduced uniformity and interoperability into this complex managed environment.

In this paper, we describe a tool CIMDIFF that provides syntactic and semantic difference tracking for CIM compliant devices in both spatial and temporal flavors. Since this problem is NP-hard, in this paper we present an efficient technique that combines domain specific object oriented knowledge with hierarchical structure of CIM-XML to derive meaningful differences. We demonstrate the value of this tool for a) tracking difference in device characteristics b) verification of proper operation as well as automated validation of management software given the limited resources of testing infrastructure. An experimental evaluation of this tool in a complex data center is provided.

1 Introduction

Any modern day enterprise-scale data center is heterogeneous in nature. Varieties of Service Level Agreements (SLAs) mandated by wide range of deployed applications along with acquisition of emerging technologies drive the heterogeneity . System Management solution(s) deployed to manage the data center provide the basic tuning knobs to adjust in order to meet the SLA goals in terms of reliability, availability, performance etc.. With the advent of virtualization, ensuring the performance guarantees, security, and fault isolation have become even more challenging. E.g. to isolate an application bottleneck; configuration, performance metrics have to be collected across server(s), network element(s) and storage controller(s). These metrics have to be investigated after correlating across physical and virtual layers. Whether, its the internal data center of an enterprise or the service offerings such as cloud computing [1, 2], *System Management* is the key. To enable seamless management in the complex and heterogeneous environments, open industry standards are necessary. Standards like Common Information Model (CIM) [3] from Distributed Management Task Force (DMTF) [4], Storage Management Initiative Specification (SMI-S) [5] from Storage Networking Industry Association (SNIA) [6], Simple Network Management Protocol (SNMP) [8], WBEM [9], SMASH [7], WMI [10] etc.. have provided excellent base to ensure interoperability between a wide array of multi-vendor data center elements, including fiber channel and IP networking components, storage components, servers, operating systems, software infrastructure and applications. System management and storage management solutions like IBM TPC [20], IBM Director [21], EMC Control Center [13], HP Systems Insight Manager [19], Microsoft Systems Center [22] have demonstrated the use of these standards to bring unified interoperable open management to complex heterogeneous managed environment.

CIM provides a common extensible base definition of management information for systems, networks, applications and services. CIM's common definitions enable vendors to exchange semantically rich management information between systems over the network. Vendor devices expose the management information through a software module called *CIM Agent*. CIM agent is either embedded in the device hardware or externally installed and configured to point to the managed device to report the management information. One CIM agent can report management information of multiple devices based on the configuration settings. CIM agents may be automatically discovered using Service Location Protocol

(SLP) or are explicitly specified by the system administrator. CIM agent software is a set of software modules called CIM Providers that are plugged into the Common Information Model Object Manager (CIMOM). CIM Providers are like servlets, that are plugged into the CIMOMs which are like the application servers. Open source CIMOMs [16, 17, 18] available are commonly used in the industry. In this paper, we refer the complete managed device information reported by CIM Agent as *CIM Repository*. This information is either cached in an internal format in the device CIM agent or retrieved on-demand through device instrumentation. Standard CIM Client [15] is used to query information from the CIM agent or to invoke configuration change operations. Adapters are also available to map SNMP to CIM Object, which is returned to the management applications. Information exchanged over network conforms to CIM-XML format. Since XML is hierarchically structured, difference tracking in such format is known to be NP-hard due to the nesting.

In this paper, we propose a tool CIMDIFF that can efficiently track syntactic and semantic difference across CIM repositories in both spatial and temporal favors. Using this tool, user/administrator can get answers to questions of following nature: a)Spatial Difference: What are the difference in characteristics between the two storage subsystems in my data center such as capacity, available space, number of disk drives, number of fiber channel ports etc..? What are the difference in configuration characteristics across two virtual machines (VMs) ? b)Temporal Difference: What are the configuration changes to the storage subsystem from past one week such as volumes created/deleted/modified, volumes assigned/unassigned etc..? What are zone configuration changes to the fiber channel fabric in last one week ? This tool uses an interesting hashing technique combined with KnowledgeBase of standard recipes that helps track differences at i) CIM construct level ii) device characteristics level to track the device configuration changes. Semantic difference provides deep insight to administrators about the data center. Syntactic difference has proved to be very useful in real production testing environment of storage/system management suite. CIMDIFF can also be used to aid conformance testing [23] by tracking effect of CIM Agent version changes when vendors update versions of their CIM Providers.

2 Background

CIM Repository reported by a CIM agent contains the complete management information of device(s). For example, a storage subsystem CIM Repository would contain information regarding the storage subsystem, storage pools, storage volumes, storage pool to storage volume association, fiber channel ports, masking/mapping information etc.. CIM Agent also reports the storage volume performance and fiber channel port performance statistics on demand. Sample CIM-XML information stream reported for a IBM DS6000 Storage subsystem is represented in Figure 1.

2.1 Model and Problem Definition

The CIM Repository (C_r) is a collection of CIM Instances (C_i) that are instances of CIM Class(es) (C_c). The structure of a CIM Instance is shown in Figure 1. Each CIM Class (C_c) defines the structure of either an entity class or an association class. An example of an entity class is `IBMTSDS-ExtentPool` that represents a Storage Pool in a IBM Storage Subsystem. Following the inheritance model of CIM, `IBMTSDS-ExtentPool` class extends from standard CIM defined class `CIM-StoragePool`. Similarly, entity class `IBMTSDS-Volume` represents a Storage Volume that extends from CIM defined `CIM-StorageVolume`. Example of an association class is `IBMTSDS-AllocatedFromExtentPool` that represents the association between Storage Pool(s) and Storage Volume(s) extending from standard CIM class `CIM-AllocatedFromStoragePool`. Single CIM Repository can contain CIM Instances of one device or multiple devices depending upon the number of devices attached to the CIM agent.

We denote a tree T by its nodes N. In the context of this paper, CIM Repository (C_r) is a tree. Children of a node are represented as $n \in N$. A CIM Instance is composed of one CIM Objectpath and zero or more CIM Properties. Node N can be a CIM Instance (C_i) or a CIMObjectpath(C_{op}) or a CIM Property(C_p). Edit operation e applied to original tree T_1 transforms the tree into T_2 is described as $T_1 \ \overrightarrow{e} \ T_2$. CIMDIFF computes *minimum-cost optimal edit script*, a sequence of basic edit operations that depicts the transformation $T_1 \ \overrightarrow{e} \ T_2$. Definitions are formally represented as:

$$C_r = \{\ N^+\ \}$$
$$N = \{\ C_i | C_{op} | C_p\ \}$$
$$C_i = \{\ C_{op}, C_p^*\ \}$$
$$C_{op} = \{\ C_p^+\ \}$$

Edit Operations: Three edit operations (e) that are evaluated for computing difference between two CIM repositories are:

- **Insert:** Insert operation creates a new node N in the tree T. Nodes N that are CIM Instance (C_i) qualify as an insert in a CIM Repository (C_r).

Figure 1: Structure of CIMInstance

- **Delete:** Delete operation is the removal of a node N in the tree T. Node N that are CIM Instance (C_i) qualify as a delete in a CIM Repository (C_r).

- **Update:** Update operation is the update of a node N in the tree T. Node N that are CIM Instance (C_i) qualify as an update in a CIM Repository (C_r), only if the CIMObjectpath(C_{op}) is unchanged across original and modified versions. Insertion, deletion or modification of one or more CIM Property(C_p) that are non-key in a CIM Instance (C_i) contribute to an update.

2.2 Types of Tracking

CIMDIFF accepts two CIM repositories as input and tracks the differences. A CIM repository is accessible through standard CIM Client using credentials (URL, User, Password, Interop Namespace). Difference tracking provided by CIMDIFF are across two dimensions. First dimension provides two categories of difference tracking: i)Syntactic ii) Semantic. Second dimension provides two categories: i)Spatial ii)Temporal.

Syntactic: CIMDIFF tracks the syntactic difference between two CIM Repositories by exploiting the syntax of CIM structure. This mode is helpful for system management tool developers and testers to perform automated testing.

Sematic: CIMDIFF tracks the semantic difference between two CIM Repositories by using the combination of CIM structure and domain knowledge. This mode is helpful for system administrators.

Spatial: CIMDIFF tracks the spatial difference between two CIM Repositories that belong to two devices that are of same or different model. For example, spatial difference can be tracked between two IBM DS8000 storage subsystems. Spatial difference can also be tracked between a IBM DS8000 storage subsystem and a EMC Symmetrix storage subsystem.

Temporal: CIMDIFF tracks the temporal difference between two CIM Repositories of the same device that are captured at different point-in-time. For

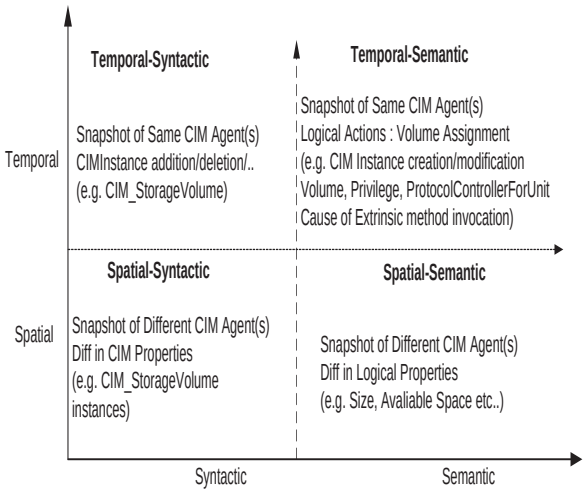

Figure 2: CIMDIFF Dimensions

example, temporal difference for a IBM DS8000 storage subsystem can be tracked between its current state and its state before one week.

With the combination of above two dimensions, CIMDIFF can track difference between two CIM repositories in four different fashion as described in Figure 2.

In the context of this paper, we use a tool called iSAN [26]. We have developed iSAN and open-sourced it through Eclipse Aperi [12] that allows to capture point-in-time snapshot of the complete CIM Agent i.e. the CIM Repository and host it. CIM Repositories created at different point in time capture device configuration changes due to regular data center tasks such as capacity provisioning, performance tuning, volume migration etc..

3 System Overview

This section provides an overview of CIMDIFF. It discusses architecture, inputs, outputs and the key components of the tool in the following subsections. CIMDIFF can reside on a separate server or can be integrated into the CIM agent to provide an integrated time-travel feature of configuration changes similar to the idea of ITIL's [24] (Information Technology Infrastructure Library) CMDB (Configuration Management Database) at the device level. CIMDIFF relies heavily on open standards and vendor extensions of open standards to derive the differences.

3.1 Architecture

Using Eclipse Aperi SAN Simulator (iSAN snapshotting framework) [26, 12], CIM repository can be persisted in a flat file or in a relational database. Device CIM agent is accessible via standard CIM Client [15] using *URL, User, Password, Interop Namespace*. *URL* is composed of *Protocol*, *IP address* and *Port*. CIMDIFF is a browser based web application that requires two CIM agent credentials as input to compute the difference. CLI interface of the tool is also exposed to the user. We use the terminology *Source CIM Agent* and *Target CIM Agent*. If both source and target CIM agent are pointing to different devices, CIMDIFF computes the spatial difference. CIMDIFF computes temporal difference, if both source and target CIM agents are pointing to the exact same device. Each device has a live CIM agent that reports the current information. Earlier point-in-time snapshots are captured and hosted by iSAN.

For example: an external `CIM Agent` deployed on a server reports two `IBM DS8000 Storage Subsystems`. Snapshot of this `CIM Agent` will be represented as one `CIM Repository` . `CIM Repository` will be associated with several `CIM Classes` such as `IBMTSDS-ComputerSystem` and `IBMTSDS-Volume` etc.. For each of these `CIM Classes`, there will be one or more `CIM Instance`. Each `CIM Instance` will have one `CIM Objectpath`. On a parallel hierarchy, there will be two `CIM Devices / CIM-ManagedElement` associated with the `CIM Repository`. Each `CIM Device` will have associated `CIM Classes`. Each `CIM Class` will have one or more `CIM Instances` associated with respect to the `CIM Device`. These two hierarchies essentially are created to answer CIM queries:

```
enumerate(CIM-StorageVolume)
associate(CIM-ComputerSystem ->
CIM-StorageVolume)
```

Basically, `CIM Repository` is a XML document with multiple hierarchy (Nodes having multiple parent) denoted by solid and dotted arrows in Figure 3.

There has been a lot of work on difference tracking algorithms for text data [27, 28], for relational data [29], for tree or XML data [30, 31]. Our tool is built on the base of text and XML difference tracking but different because of the domain specific meaningful change tracking functionality rather than the text or XML difference tracking. In addition, multi-parent nature of the nodes in the XML structure and the semantic domain knowledge of `CIM Recipes` that derive the difference makes our tool unique. Figure 4 describes the functional blocks of this tool. CIMDIFF has four main functional blocks: i)Hash Maker ii) KnowledgeBase iii) Hierarchy Resolver iv) Difference Tracker.

Hash Maker component calculates the hash using the popular SHA-256 or MD5 algorithm for the `CIM Repository`. Hash values are created and stored

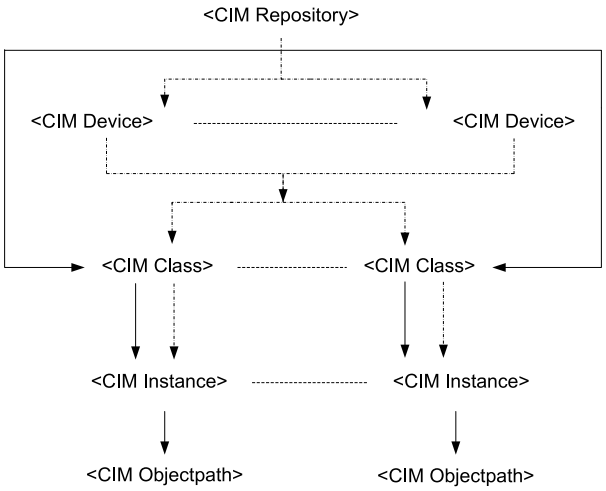

Figure 3: CIM Repository Structure

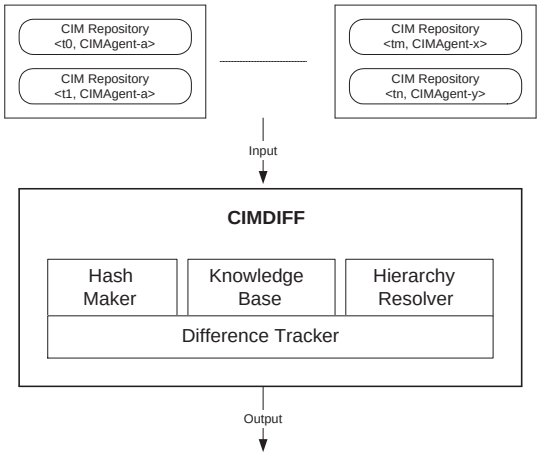

Figure 4: CIMDIFF Components

respective to nodes. Each node can contain multiple hash values since it can be parent of multiple nodes (CIM Agent Hash, CIM Device Hash). Hash values are built using bottom-up approach. Hash value of a parent node N is calculated from the hash values of the children nodes n. Leaf nodes that contain the hash values are the `CIM Objectpath`. Hash value calculation can also be integrated with the snapshot making process. Otherwise, it can be created later for a `CIM Repository` and correlated against it. Hash Maker component makes the difference tracking process efficient because, the hash values stored at nodes help CIMDIFF determine the tree/subtree isomorphism much efficiently. Tree/subtree isomorphism determination helps in the temporal-syntactic difference tracking. This component helps CIMDIFF traverse the least number of nodes necessary to perform difference tracking.

KnowledgeBase component stores the standard recipe invocation sequences and the changes associated with it. For example, if a StorageVolume was created and assigned (masked/mapped) to a server by execution of `CIM Recipe(s)` meaning a set of extrinsic methods were executed. Extrinsic method invocation on a `CIM Agent` results in `CIM Instance` creation/deletion/modification. Goal of this component is to store the popular standard recipe formats and their effect on `CIM Instances`. This component helps track the temporal-semantic changes across `CIM Repositories` of same `CIM agent(s)`. Similarly, any vendor-extended properties are also canned into KnowledgeBase for spatial-semantic difference tracking. KnowledgeBase is populated with standard SMI-S recipes to start with. It is updated with the latest variance in recipe formats as well as any new recipes that are introduced. KnowledgeBase can be centrally created and then distributed across installations to reflect the updates. Users can also edit custom recipe formats into knowledge fragments and update their KnowledgeBase. An example SMI-S recipe for storage volume creation is described in Figure 5

HierarchyResolver component is a wrapper around a MOF [11] parser. It traverses the CIM hierarchy through the `CIM Class` hierarchy and filters out the string mismatches. This component helps track the spatial-semantic changes across `CIM Repositories` of different `CIM Agent(s)` of similar type. For example: two storage pool instances (one from IBM DS8000 and other from IBM DS4000) have different vendor-extension `CIM Class`; `IBMTSDS-VolumeSpace` and `LSISSI-StoragePool` respectively. This module helps discard the regular string differences and picks the relevant difference in properties such as `Total Space` , `Total Available Space`. This module performs this task by i) resolving hierarchy since both these classes belong to same super class `CIM-StoragePool` ii) checking the KnowledgeBase for relevant `CIM Properties`

Difference Tracker component works in conjunction with the three components described earlier. It orchestrates the invocation of the CIMDIFF components. This component implements the algorithm to compute the minimum-cost optimal edit script. Outline of CIMDIFF algorithm is explained in Figure 6.

Edit script is then grouped into clusters based on the rules defined in KnowledgeBase to derive meaningful changes from the device configuration perspective. `CIM Objectpath` is used as the main correlation mechanism in the aggregation process to avoid ambiguity. CIM Properties are interpreted based on the rules to derive semantic differences. It also presents the output to the user and provides primitives for analysis across dimensions described in Figure 2 via both browser and java swing based graphical user interface. This component in con-

```
//Create Storage Volume on an Array [10GB RAID-5 Volume]
// RequestedSize = 10 * 1024 * 1024 * 1024 // 10 GB

1. Enumerate all StoragePools associated with $StorageArray
2. For each StoragePool {
     if(StoragePool is non-primordial) {
       get StorageCapabilities associated with StoragePool
       if(StorageCapabilities == Volume Requirements) {
         if (StoragePool has free space available) {
           invoke GetSupportedSizes and(or) getSupportedSizeRange
           //Round up using integer arthimetic
           get valid capacity of newly created volume
           Select StoragePool and Select VolumeSize }
       } } }
3. Services[] = AssociatorNames(StorageArray)
4. StorageConfigurationService = Services[0]
5. InArguments[ElementType] = 2 // Storage Volume
   InArguments[Goal] = GeneratedStorageSetting
   InArguments[Size] = VolumeSize
   InArguments[InPool] = StoragePool
   InArguments[TheElement] = null
   ReturnValue = InvokeMethod(
                   StorageConfigurationService,
                   CreateOrModifyElementFromStoragePool,
                   InArguments, OutArguments)
6. if ((ReturnValue == 0 || ReturnValue == 4096 &&
   OutArguments[TheElement] != null)) {
       //Reference to the newly created volume
       CreatedVolume-> = OutArguments[TheElement] }
```

Figure 5: Recipe - Storage Volume Creation

junction with HierarchyResolver also tracks the change of CIM model definition [11].

Figure 7 explains the input specification of CIM agent information to the tool. Spatial difference can be tracked between two similar types of `CIM Repositories` but not necessarily of same model. Two input CIM agent can report two storage subsystems or two servers. Spatial difference could be either syntactic or semantic. Spatial difference tracking is helpful for checking the difference in characteristics and configuration across similar devices. Spatial-syntactic difference tracking would derive information like i) the difference in number of `CIMInstances` per class between two `IBM DS4000` storage subsystems or between two different servers or between a `IBM DS4000` and a `IBM DS6000` storage subsystem ii) difference in `CIM Properties` iii) difference in `CIM Property` value. Same information in spatial-semantic form would be presented as i) difference in number of storage volumes, storage pools, file systems ii) difference in total available space, difference in total consumable space, difference in server main memory (RAM).

Temporal difference can be tracked between `CIM Repositories` of same `CIM Agents` snapshotted at different point in time. This kind of information is helpful for tracking the temporal change in the characteristics and configuration of the same device. As shown in Figure 7, point-in-time snapshots of CIM agent can be captured and hosted using the open source tool iSAN. Temporal-syntactic difference would present information such as i) the difference in number of `CIMInstances` per class ii) dif-

```
calculateMinCostEditScripts(C^r1,C^r2)
{
  /* Hashing */
  Map C^r1 to T^1, Map C^r2 to T^2
  Hash T^1, Hash T^2
  /* Traverse CIM Agent Hierarchy */
  /* Traverse Managed Device Hierarchy */
  Traverse CIM Agent hierarchy &
  Traverse Managed Device hierarchy
  {
    /* Check tree isomorphism based on Hash */
    /* Compute Insert, Delete, Update */
    for all Node N in T^1 and T^2
      for all CIMClass(es)
        for all CIMInstance(s)
          Compare CIMObjectpath
            Compare CIMProperties
  }
}
```

Figure 6: Outline of minimum-cost optimal edit script computation

ference in `CIM Properties` because of configuration change of the device such as volume creation, masking/mapping for storage subsystem or file system creation on server. Large amount of change in number of `CIM Instances` and/or `CIM Properties` due to device configuration change and its presentation via temporal-syntactic change tracking might be useful for systems management solution test automation but can be overwhelming for a system administrator. Grouping these changes by logical actions such as storage provisioning, file system provisioning is termed as temporal-semantic change.

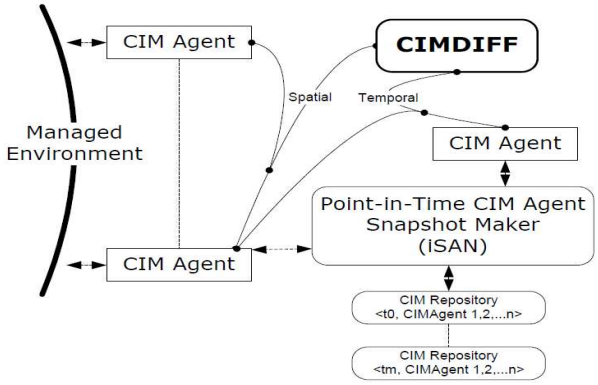

Figure 7: Input for CIMDIFF

4 Experimental Evaluation

Our experimental test bed is part of a production SAN environment. It contains servers with Linux and Windows operating systems. Production SAN also contains interconnecting fiber channel switches from multiple vendors, IBM and non-IBM storage subsystems. Each server has its own `CIM Agent` hosted on the server itself. Most of the fiber channel switches have external `CIM Agents` reporting the fabric information except a very few switches that has embedded `CIM Agent`. IBM and non-IBM storage subsystems have external `CIM Agents` hosted on separate servers. Management information is retrieved from `CIM Agents` using standard SBLIM CIM Client [15]. In this managed environment, server `CIM Agent` have a 1:1 mapping of `CIM Agent` to managed element. For fiber channel switches and storage subsystems, this environment has a maximum of 1:4 mapping of `CIM Agent` to managed device. CIMDIFF uses iSAN [26] to create the snapshots of the `CIM Agents`. `CIM Repositories` created from the snapshot process were stored in IBM DB2 UDB relational database. CIMDIFF also uses embedded database Derby [25] for handling data centers with very few devices. We integrated the HashMaker module of CIMDIFF with iSAN [26] to create the hash at relevant nodes during the snapshot process itself. Calculating hash did not impose a major overhead in terms of snapshot creation time. Calculating hash based on the standard CIM hierarchy denoted by solid arrows in Figure 3 introduced very minimal overhead during the snapshot process. Creation of device based hash denoted by dotted arrows took most of the time. An evaluation of hash creation overhead during a snapshot creation process for a `CIM Repository` of 2887 `CIM Instances` is described in Figure 8.

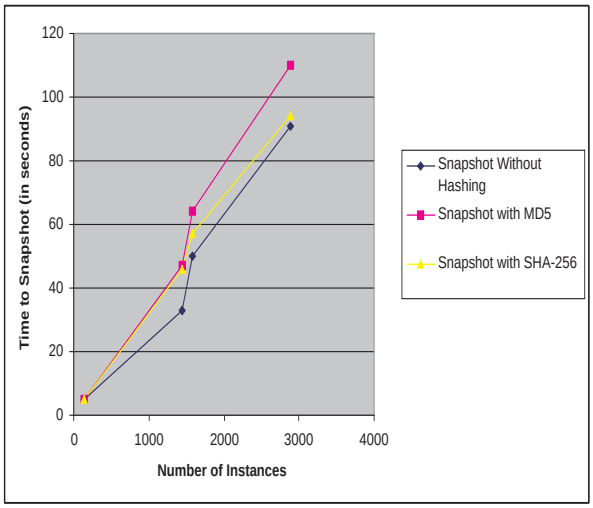

Figure 8: Hash Creation Overhead

CIMDIFF is implemented completely in java. It is hosted as a servlet based web application in a IBM Websphere Application Server container. Browser based user interface and user options are shown in Figure 9. CIMDIFF provides user interface and primitives to manage the CIM agent credentials. It also provides interface to manage and host snapshots of CIM agents. Knowl-

Table 1: Setup (Spatial Difference)

CIMDIFF	
CIM Repository	Description
CIMRepository-An	CIMAgent-A Storage Subsystem [IBM DS6000]
CIMRepository-Bm	CIMAgent-B Storage Subsystem [IBM DS4000]

edgeBase can be easily updated by uploading knowledge fragments. Knowledge fragments can be written by following a very simple XML based rule engine.

Based on the input CIM agent credentials, CIMDIFF automatically detects whether the mode is spatial or temporal. As described in Figure 9, syntactic and semantic output are presented in tabbed format. Due to the large and detailed nature of the output returned by CIMDIFF, we present selective portion of it in a tabular fashion to depict the nature of difference tracking.

Spatial Difference Tracking: Source CIM agent and target CIM agent credentials are supplied by the user (as shown in Figure 9). Since, both the CIM agents point to two different IBM DS8000 storage subsystems, spatial difference option is highlighted. Spatial difference can be evaluated between two similar types of devices, e.g. between a IBM DS4000 and a IBM DS8000 storage subsystem. Table 1 shows the setup for spatial difference tracking and Table 2 shows the types of syntactic differences that are tracked. Automation test suites used by discovery engines of system management tools [20] use the syntactic results to verify the proper operation of the tools.

Semantic information is derived based on the standards and(or) the KnowledgeBase. Standard CIM class CIM-StoragePool has CIM Properties `TotalManagedSpace` and `RemainingManagedSpace`. By accumulating the values across all the storage pools, TotalSpace and FreeSpace are calculated. Similarly, by using standard properties from `CIM-StorageSetting`, RAID level is calculated. These properties are common across vendor implementations(IBM storage subsystem represents its storage pool through the class IBMTSDS-ExtentPool or LSISSI-StoragePool based on the model. But, both these classes extend from CIM-StoragePool). CIMDIFF refers to rules across vendor extended properties to derive meaningful semantic differences. Rules can be easily added and updated by specifying new rules in simple XML format. Sample semantic difference values are shown in Table 3

Temporal Difference Tracking: If both source and tar-

Table 2: Spatial Syntactic Difference

CIMDIFF	
CIMRepository-An	CIMRepository-Bm
74 CIM Classes	78 CIM Classes
1874 CIM Instances	2436 CIM Instances
IBMTSDS-Volume PowerOnHours CIM Property	LSISSI-StorageVolume NO Corresponding Property
IBMTSDS-Volume NO Corresponding Property	LSISSI-StorageVolume RaidLevel CIM Property
IBMTSDS-Volume 212 CIM Instances	LSISSI-StorageVolume 120 CIM Instances
IBMTSDS-ExtentPool 7 CIM Instances (Extends from: CIM-StoragePool)	LSISSI-StoragePool 5 CIM Instances (Extends from: CIM-StoragePool)
IBMTSDS-DiskDrive 36 CIM Instances	LSISSI-DiskDrive 12 CIM Instances

Table 3: Spatial Semantic Difference

CIMDIFF	
CIMRepository-An	CIMRepository-Bm
Remaining Managed Space 840GB	Remaining Managed Space 300GB
Remaining RAID-5 Space 640GB	Remaining RAID-5 Space 300GB
Remaining RAID-1 Space 140GB	Remaining RAID-1 Space 0GB

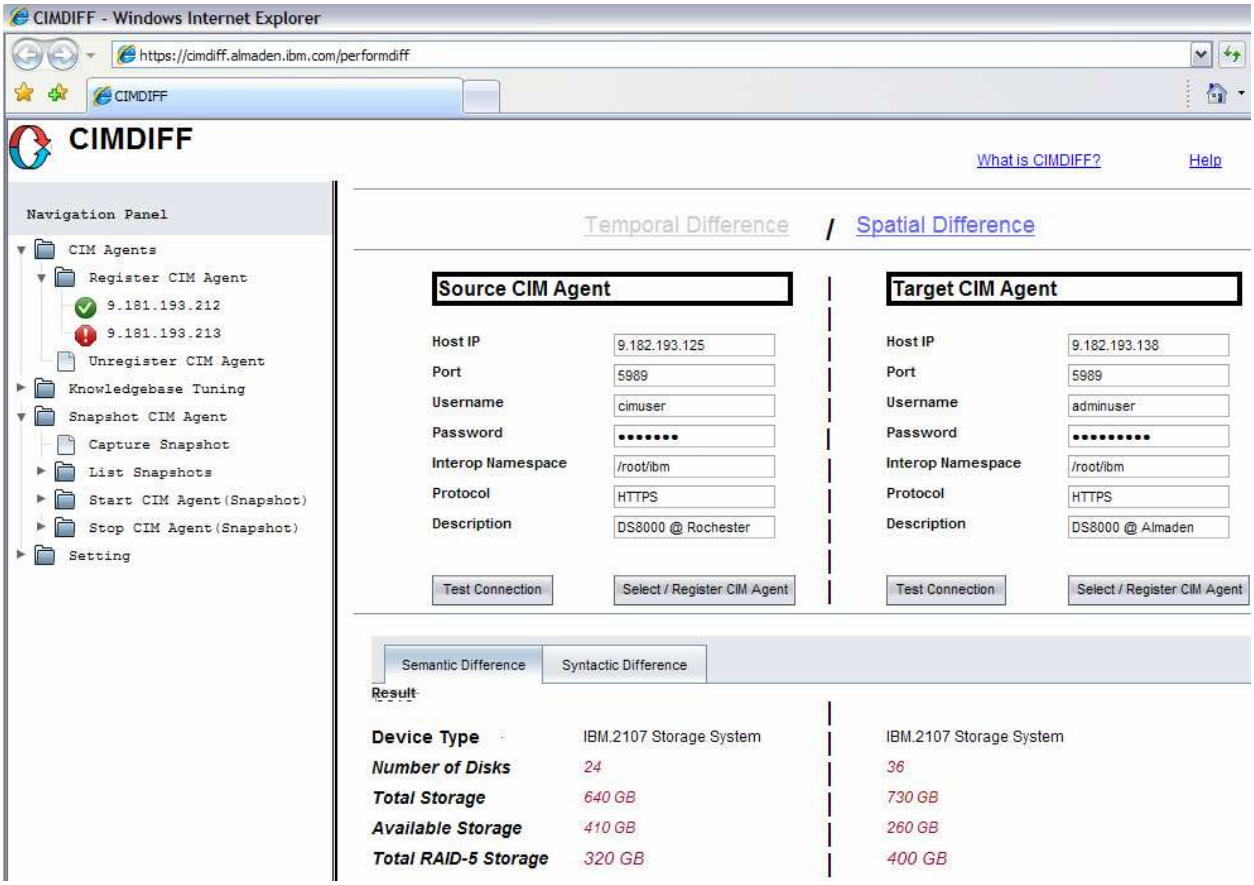

Figure 9: CIMDIFF User Interface

Table 4: Setup (Temporal Difference)

CIMDIFF	
CIM Repository	Description
`CIMRepository-A0`	`CIMAgent-A` Storage Subsystem [IBM DS6000] Snapshot at time `t0`
`CIMRepository-A1`	`CIMAgent-A` Storage Subsystem [IBM DS6000] Snapshot at time `t1`

get CIM agent point to the same device, CIMDIFF automatically detects and derives the temporal difference. Table 4 depicts the setup for a temporal difference scenario.

Due to the configuration actions performed on the storage subsystems such as volume creation/deletion, volume assignment/unassignment or zoning actions on fiber channel switches, temporal differences are reflected in the `CIM Repositories`. Table 5 shows some of

Table 5: Temporal Syntactic Difference

CIMDIFF	
`CIMRepository-A0`	`CIMRepository-A1`
1874 `CIM Instances`	1942 `CIM Instances`
`IBMTSDS-Volume` 212 `CIM Instances`	`IBMTSDS-Volume` 224 `CIM Instances` [SAME] 216 `CIM Instances` [DELETED] 5 `CIM Instances` [NEW] 17 `CIM Instances` [MODIFIED] 1 `CIM Instances`

Table 6: Temporal Semantic Difference

CIMDIFF	
CIMRepository-A0	CIMRepository-A1
1874 CIM Instances	1942 CIM Instances
	VOLUME CREATION VOLUME ASSIGNMENT
IBMTSDS-Volume 212 CIM Instances	IBMTSDS-Volume 224 CIM Instances
IBMTSDS-Privilege 18 CIM Instances	IBMTSDS-Privilege 20 CIM Instances
IBMTSDS- ProtocolControl lerForUnit 48 CIM Instances	IBMTSDS- ProtocolControl lerForUnit 52 CIM Instances
....	....

the syntactic difference due to the configuration actions. These difference are useful for automated testing validation of management software.

Modification of CIM Instance is derived because CIMObjectpath did not change but one or more of the other non-key CIMProperties changed across time.

In this scenario, syntactic difference might be overwhelming and does not provide with much meaningful information for administrator. But, by tracking the change in CIMInstances and then grouping the changed instances based on the recipe KnowledgeBase, we can show a potential list of extrinsic methods that were executed. As shown in Table 6, change between time t0 and time t1 was because of volume creation, volume assignment(masking, mapping). Correlating this information through the CIMObjectpath provides more value in terms of configuration change activities on the device. Sample KnowledgeBase is described in Table **??**

We used this tool to capture snapshot of our data center (small scale SAN environment) to i) track the temporal configuration changes for devices ii) compare and contrast similar type device configuration through spatial changes. This provided the system administrators with deep insight about change in the data center environment. Systems Management solutions [21] or Storage Resource Management solutions [20, 13] also provide this kind of feature at a much higher and better granularity because these solutions correlate data from across CIM Agents. CIMDIFF does not correlate data from across CIM Agents (e.g. addition/ removal of a port-to-port connectivity tracking by correlating the server port and fiber channel port). CIM Agent Snapshots of multiple CIM Agents can be aggregated together for evaluation. This distinction of aggregation versus correlation is a known limitation of CIMDIFF. We have also tested CIMDIFF and did not encounter any scalability issues for few hundreds of thousands of CIM Instances in the CIM Repositories.

5 Discussion and Related Work

General problem of detecting changes from snapshots of textual, relational or hierarchical structured (XML) data has been studied in great details. GNU diff utility is a popular tool in this context. Our calculation of edit scripts with respect to CIM Classes based on CIMInstance and CIMObjectpath is similar to the notion of LCS (Longest Common Subsequence) used in the difference tracking of CVS. Meaningful change detection algorithms [30, 31] have also a similar notion of difference tracking. Comparison of CIM standards and the vendor reported CIM Instances have a close analogy with XML schema / DTD and well-formed XML documents. In [33, 34, 37], change detection problems have been addressed for ordered trees. Authors [35] have also proved the meaningful change detection complexity to follow quasi linear time for NP-hard scenarios. Similarly, algorithm [30] also provides a similar approach by transforming the change detection problem to a problem of computing a minimum-cost edge cover of a bipartite graph. CIMDIFF is effectively domain specific logical difference tracking that is guided by industrial standards and customizable knowledge base as compared to the traditional text difference tracking, XML document difference tracking or change detection. Similar to our HashMaker approach, there has been similar work in the XML document change detection area that deals with constructing node signatures using exclusive-or (XOR) [32].

In virtualized environments [36], configuration change and movement of virtual resources are more often than a physical environment. CIMDIFF can very easily [39] show the difference in change in environment due to migration, provisioning [36] etc..

CIMDIFF provides an important primitive to the data center administrator as well as a very vital test tool for System Management solution developers and testers. SNIA CTP [23] can also potentially use this feature to make the testing process more efficient. CIM Agents before and after incorporating the changes to CIM Provider during testing iteration conformance cycle can use this tool. CIMDIFF provides valuable difference tracking powered by the knowledgebase. Currently, knowledgebase is manually populated by encoding the information from standard CIM recipes. Execution patterns of standard CIM recipe and their effects can be automatically derived by using machine learning. We

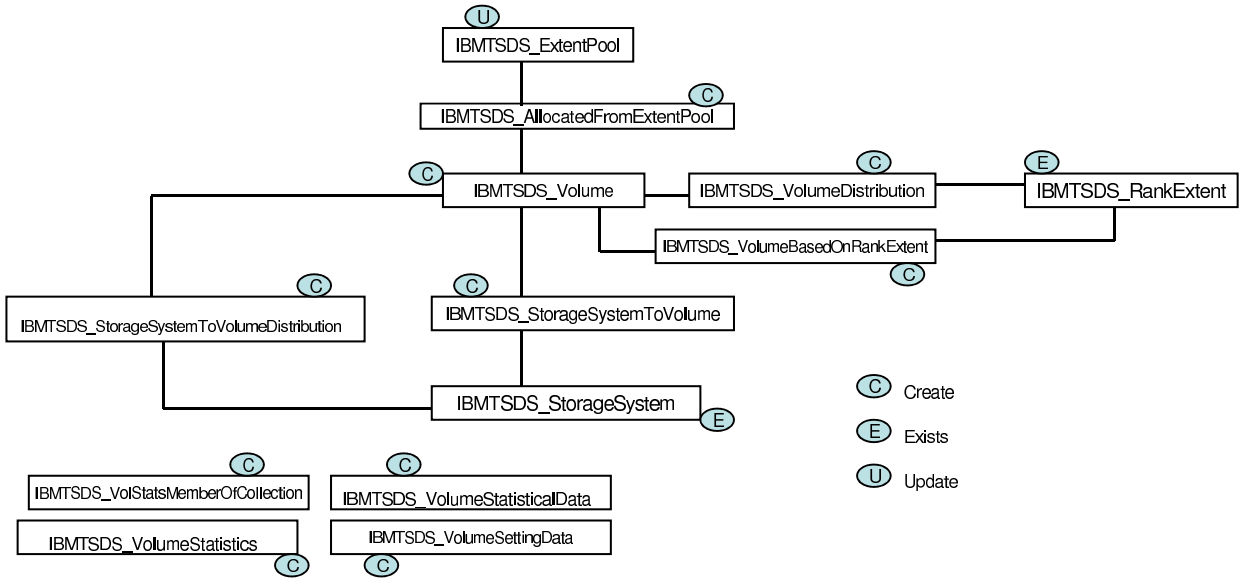

Figure 10: KnowledgeBase - Volume Creation

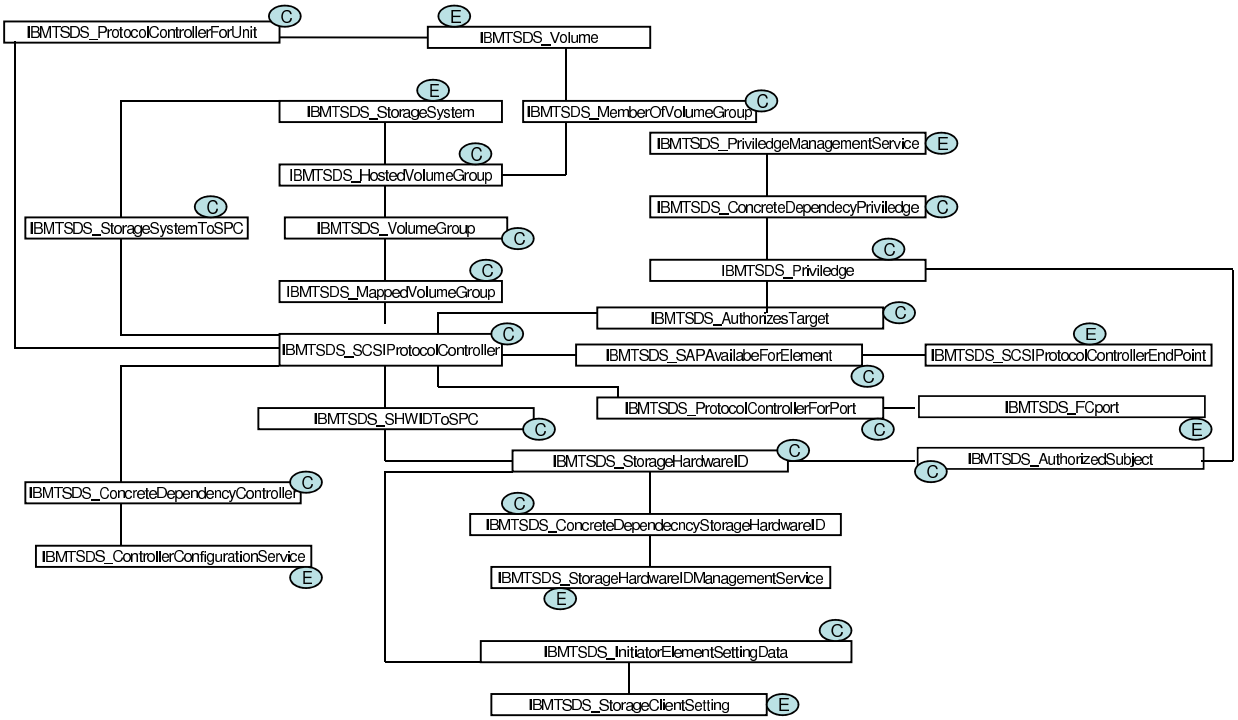

Figure 11: KnowledgeBase - Volume Assignment

are also extending a semi-autonomic interface for this tool to let the administrator create a knowledge fragment by grouping a discrete set of syntactic difference and associating with a particular semantic configuration change from the output panel of this tool itself.

Without creating snapshot of `CIM Agent` at intervals, change information also can be derived by: i) if `CIM Agent` supports lifecycle indication monitoring ii) by continuous subscription to all indications. But, such mechanism could be costly and would not provide semantic difference.

6 Conclusion

In this paper we presented an architecture, working of CIMDIFF that provides an important tool to system administrators with tracking data center configuration and characteristic changes. CIMDIFF also demonstrates the difference tracking based on the domain knowledge of CIM [3] standard. It builds on the standard text difference and XML difference tracking technologies. CIMDIFF uses an efficient hashing and knowledgebase interpretation on top of the multi-hierarchical XML data. Demonstrated approach provides an interesting solution to the nested structure that poses NP-hard problem.

CIMDIFF also limits itself to `CIM Repositories` aggregation rather than correlation functionality provided by systems management solution. CIMDIFF definitely provides an important technology for management solution testing as well as conformance [23] testing. Since most of the modern day data center entities have CIM agents, CIMDIFF offers a neat and quick tool for administrators in the data center.

References

[1] Amazon Elastic Compute Cloud EC2. `http://aws.amazon.com/ec2/`

[2] Amazon Simple Storage Service S3. `http://aws.amazon.com/s3/`

[3] Common Information Model (CIM). `http://www.dmtf.org/standards/cim`

[4] Distributed Management Task Force (DMTF). `http://www.dmtf.org`

[5] Storage Management Initiative Specification (SMI-S) `http://www.snia.org/forums/smi/tech_programs/smis_home/`

[6] Storage Networking Industry Association (SNIA). `http://www.snia.org`

[7] Systems Management Architecture for Server Hardware (SMASH) `http://www.dmtf.org/initiatives/smash_initiative/`

[8] IETF Simple Network Management Protocol (SNMP) `http://www.ietf.org/rfc/rfc1157.txt`

[9] Web-Based Enterprise Management (WBEM). `http://www.dmtf.org/standards/wbem/`

[10] Windows Management Instrumentation (WMI) `http://msdn.microsoft.com/en-us/library/aa384642(VS.85).aspx`

[11] Managed Object Format (MOF). `http://www.dmtf.org/education/mof/`

[12] Eclipse Aperi Project. `http://www.eclipse.org/aperi`

[13] EMC Control Center. `http://www.emc.com/products/family/controlcenter-family.htm`

[14] SNIA Interoperability Lab.. `http://www.snia.org/forums/smi/tech_programs/lab_program/`

[15] SBLIM CIM Client. `http://sblim.wiki.sourceforge.net/CimClient`

[16] Pegasus CIMOM. `http://www.openpegasus.org/`

[17] Sun WBEM. `http://wbemservices.sourceforge.net/`

[18] SNIA CIMOM. `http://www.opengroup.org/snia-cimom/`

[19] Hewlett Packard Systems Insight Manager (HP SIM). `http://h18002.www1.hp.com/products/servers/management/hpsim/index.html`.

[20] IBM TotalStorage Productivity Center (IBM TPC) `http://www-306.ibm.com/software/tivoli/products/totalstorage-data/`

[21] IBM Systems Director `http://www-03.ibm.com/systems/management/director/`

[22] Microsoft System Center `http://www.microsoft.com/systemcenter/en/us/default.aspx`

[23] SNIA Conformance Testing Program `http://www.snia.org/forums/smi/tech_programs/ctp/`

[24] ITIL `http://www.itil-officialsite.com/home/home.asp`

[25] Apache Derby `http://db.apache.org/derby/`

[26] R. Routray, S. Gopisetty, P. Galgali, A. Modi and S. Nadgowda. iSAN: Storage Area Network Management Modeling Simulation In *Proceedings of IEEE International Conference on Networking, Architecture, and Storage (NAS)*, 2007

[27] R. Wagner, M. Fischer. The String-to-String Correction Problem In *Journal of the ACM, Volume 21, Issue 1*, 1974

[28] E. Myers. An O(ND) Difference Algorithm and Its Variations In *Algorithmica*, 1986

[29] W. Labio, H. Garcia-Molina. Efficient Snapshot Differential Algorithms for Data Warehousing In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB)*, 1996

[30] S. Chawathe, H. Garcia-Molina. Meaningful Change Detection in Structured Data In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1997

[31] Y. Wang, D. DeWitt, J. Kai. X-Diff: an effective change detection algorithm for XML documents In *Proceedings of 19th International Conference on Data Engineering*, 2003

[32] L. Khan, L. Wang, Y. Rao. Change Detection of XML Documents Using Signatures In *Proceedings of Workshop on Real World RDF and Semantic Web Applications*, 2002

[33] K. Zhang, D. Shasha. Simple fast algorithms for the editing distance between trees and related problems In *SIAM Journal on Computing* , 1989

[34] D. Shasha, K. Zhang. Fast parallel algorithms for the unit cost editing distance between trees In *Proceedings of the first annual ACM symposium on Parallel algorithms and architectures*, 1989

[35] D. Shasha, K. Zhang. Detecting Changes in XML Documents In *Proceedings of International Conference on Data Engineering*, 2001

[36] M. Dehus, D. Grunwald. STORM: Simple Tool for Resource Management In *Proceedings of 22nd Large Installation System Administration Conference*, 2008

[37] B. Nguyen, S. Abiteboul, G. Cobena, and M. Preda Monitoring XML data on the Web In *Proceedings of ACM SIGMOD*, 2001

[38] K. Begnum, M. Disney, E. Frisch, and I. Mevg Decision Support for Virtual Machine Re-Provisioning in Production Environments In *Proceedings of 21st Large Installation System Administration Conference*, 2007

[39] VMware CIM APIs. `http://www.vmware.com/support/developer/cim-sdk/`