Linux Operating System Configuration Management Framework: A Scalable and Efficient Approach Using Open Source Utilities

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Srinivas R. Kalidindi

November 2007

This thesis titled

Linux Operating System Configuration Management Framework: A Scalable and Efficient Approach Using Open Source Utilities

by

SRINIVAS R. KALIDINDI

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

_____

Chang Liu

Assistant Professor of Electrical Engineering and Computer Science

_____

Dennis Irwin

Dean, Russ College of Engineering and Technology

Abstract

KALIDINDI, SRINIVAS R., M.S., November 2007, Computer Science

Linux Operating System Configuration Management Framework: A Scalable and Efficient Approach Using Open Source Utilities (120 pp.)

Director of Thesis: Chang Liu

With the steady growth in Information Technology sector, managing complex infrastructure has proven to be a challenge for organizations, both big and small, in terms of maintaining a consistent environment across the board. In a product development industry, this consistency in the environment plays a key role in maintaining the overall productivity of the organization.

Current Information Technology administrators have access to a wide array of open source and commercial tools for infrastructure management in terms of hardware as well as software configuration changes. As hardware support is provided by the appliance vendor, the complexity in managing the underlying operating system configuration changes poses a challenge. Although there is a wide array of open source tools available to achieve this objective, they may be far more complex than required, and may not be entirely portable within an existing environment. In many cases, the commercial solutions are not affordable depending on the size of the organization.

This thesis investigates the currently available configuration management tools, and presents Linux operating system configuration management framework (LOSCMF) developed using open source tools and utilities. Linux operating system configuration management framework (LOSCMF) solution has been designed to perform complex operating system configuration changes by addressing the following three drawbacks found in the current configuration management solutions; usage of additional agents (system daemons), usage of complex configuration language, and usage of special scripting and programming languages. Each of the computer systems installed with the LOSCMF are grouped based on their current configuration level. A system administrator defines the necessary configuration changes that need to be applied either for a particular group or a logical combination of groups. These configuration changes are applied until a

computer system confirms to a defined configuration state. The LOSCMF has both command line as well as web based reporting interfaces that can be used by the administrators to generate reports as per the requirement. The LOSCMF has been written using object oriented Perl, PHP and Bourne shell, and the underlying architecture is independent of the actual command or the scripting language, which would be performing the configuration change. The system administrators can choose to use any scripting or programming language of their choice to perform the configuration change. The LOSCMF solution can be used as a prototype within medium to large sized organizations for maintaining Linux operating system configuration changes. This thesis presents some of the benefits achieved using the LOSCMF approach such as a decrease in the administrative time, decrease in the support call volume, as well as a simple and effective mechanism for administrators to extract useful information on the fly.

Approved: _____

Chang Liu

Assistant Professor of Electrical Engineering and Computer Science

Acknowledgements

I thank my advisor, Dr. Chang Liu, for sparing his valuable time to supervise this Master's thesis. He has been an excellent advisor for my period of study at Ohio University.

I would like to thank my other committee members, Dr. Shawn Ostermann and Dr. Jeffrey Dill for being a part of my thesis committee. I would like to thank Dr. Xiaoping Shen for being the external college representative in my thesis committee.

I would like to thank my family for their encouragement and support and for providing me the opportunity for studying at Ohio University. Finally I would like to thank all my friends at Ohio University.

I dedicate this thesis to my parents.

Table of Contents

Page

List of Tables

List of Figures

List of Abbreviations

IT – Information Technology

LOSCMF – Linux Operating System Configuration Management Framework

YAST – Yet Another Setup Tool

XML – Extensible Markup Language

RHN – Red Hat Network

SNMP – Simple Network Management Protocol

MIB – Management Information Base

GPL – General Public License

TCP – Transmission Control Protocol

SSL – Secure Sockets Layer

API – Application Programming Interface

IP – Internet Protocol

SED – Stream Editor

PHP – Hypertext Preprocessor

TCL – Tool command Language

HTML – Hypertext Transfer Protocol

CVS – Concurrent Versions System

ODBC – Open database Connectivity

CGI – Common Gateway Interface

SQL – Structured Query Language

NAS – Network Attached Storage

NIC – Network Interface Card

NFS – Network File System

LSF – Load Sharing Facility

LDAP – Lightweight Directory Access Protocol

NIS – Network Information Service

WAN – Wide Area Network

RCS – Revision Control System

ASCII – American Standard Code for Information Interchange

DST – Daylight Savings Time

SOAP – Simple Object Access Protocol

COAS – Caldera Open Administration System

IDC – International Data Corporation

LCFG – Local Configuration System

X-CONF – Xml-based Configuration Management System

BCFG – Bundle Configuration System

1 Introduction - Motivation and Goals

In this chapter we discuss the motivation and the expected goals to be achieved with this thesis. Section 1.1 describes the motivation behind developing this framework, and section 1.2 describes the goals expected to be achieved through this thesis. Finally section 1.3 describes the overall organization of this thesis.

1.1 Motivation

In the current information technology age, it is required that a business environment with mission-critical systems along with the applications that run on them be operationally efficient and reliable. In order to achieve this mammoth task, the concerned system administrators and Information Technology (IT) engineers are constantly looking for effective solutions for hardware maintenance and the underlying operating system configuration management. Linux operating system platform is currently being adopted by many organizations as it provides significant advantages in terms of reliability, ease of use and security. According to International Data Corporation (IDC) [22] publication in 2003, “Linux operating system server growth is 50% Year-Over-Year”. In most cases hardware maintenance is performed by the respective vendors as part of the support contract process, but the customization of the operating system still needs to be an in house task, as the requirements vary as per the needs of an organization.

A Linux system administrator could use simple tools written in an administrative language such as Shell [23] or Perl [24] to perform the routine tasks such as adding users, scheduling backups and monitoring system utilization, and a manual configuration approach for complex operating system level changes. Although this manual approach works for small organizations, it is not scalable for mid-size to large organizations. In an organization that has personnel with various skill set levels (novice as well as experienced) within the IT department, this manual approach leads to an inconsistent environment without the use of any policies and standards. In a large organization, it is expected that the system administrators automate the daily administration tasks and devote much of the time towards solving customer issues and improve the overall efficiency of the organization.

Performing complex operating system level configuration changes requires an in depth knowledge and expertise in the related areas. This functionality can be achieved through an operating system configuration management frame work (e.g. Cfengine [25]). Configuration management is defined as follows in the IEEE Glossary of Software Engineering Terminology (Standard 729-1983) [60]: *"Configuration Management is the process of identifying and defining the items in the system, controlling the change of these items throughout their life-cycle, recording and reporting the status of items and change requests, and verifying the completeness and correctness of items"*.

Although there are a wide variety of open source as well as commercial operating system configuration management solutions, not all of them are generic in nature and can be used off-the-shelf. All of the tools require some degree of customization based on the organizational requirements. An ideal configuration management solution design should satisfy the following criterion

- Modular design
- Easy integration
- Scalability
- Customization allowed
- Decrease overhead's
- Easy learning curve

Current available open source configuration management solutions do not satisfy all or most of the above criterion. In addition they have one or more of the following drawbacks associated with them.

- Usage of additional agents (system daemons)
- Usage of complex configuration language
- Usage of special programming and scripting languages

Some of the open source Linux operating system configuration management solutions are Cfengine [25], COAS [20], Webmin [56], PIKT [11], Linuxconf [21], and YAST [26]. A detailed description and the associated drawbacks of each of these solutions is described in section 3.1. Through this thesis, we are looking into developing

a Linux operating system configuration management framework (LOSCMF) using open source tools and utilities that addresses all of the three drawbacks mentioned above. The LOSCMF design satisfies the above mentioned design criterion. Through the LOSCMF, we provide the functionality for a system administrator to implement changes uniformly within an environment as well as perform status check on the fly.

1.2 Thesis Goals

The initial goal of this thesis is to study the working of a generic configuration management process and analyze the current available configuration management solutions for their merits and drawbacks. The main goal of this thesis is to substantiate the claim that the use of the model described in section 1.2.2 will address the above mentioned ‘three’ drawbacks found in the current available solutions. We aim to show that this is a cost effective solution, which provides the ability to mask the complexity of the configuration changes from a system administrator.

1.2.1 Generic Configuration Management Process

The initial goal is to study the working of a generic configuration management process. The figure below represents the sequence of flow of events in a generic configuration management process. The purpose of a Configuration Management process is to ensure that a computer system is always in a defined configuration state by rectifying any of the present anomalies. In a configuration management solution, the known good pre-defined configurations are stored and act as the baseline parameters, which need to be followed by the computer system that it is being applied on. These parameters can either be stored in the form of profiles as mentioned in [2], or in the form of configuration files themselves. These configuration parameters are usually stored on a configuration management server. Each computer system contacts the configuration management server, usually through a daemon, and checks for the necessary profiles or parameters defined for its type. Upon matching, either the local computer system or the server compute the necessary changes those need to be performed. Once the changes are determined, the relevant fixes are either pushed through to the local system from the server or vice versa. This process repeats itself until the local computer system

completely conforms to the defined standards or is close to defined standards depending upon the scope of the changes that need to be made. Figure 1-1 below depicts the flow of events usually followed within a generic configuration management process.

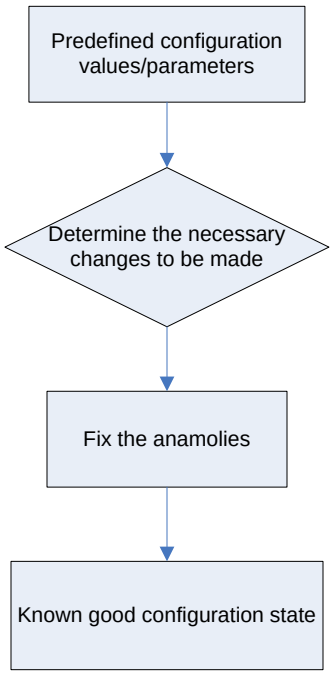

Figure 1-1: Flow Chart Representation of a Generic Configuration Management Solution

1.2.2 High Level Overview of the Linux Operating System Configuration Management Framework

This section provides a high level overview of Linux operating system configuration management framework (LOSCMF) solution being developed through this thesis.

Linux operating system configuration management framework (LOSCMF) design is similar to a generic configuration management solution described in section 1.2.1. Each of the LOSCMF installed computer system within the environment periodically parses a master configuration file set, which is stored on an accessible centralized storage location. Any updates to the master configuration file set are performed through Concurrent Versions System (CVS) [47] to maintain consistency across the environment.

This approach compensates for any deviations that might result in storing the master file set locally on each of the installed computer systems. All the installed computer systems are dynamically classified in to groups (e.g. all computer systems installed with Red Hat 7.2 Linux Operating System can belong to one group). Each of the configuration change is defined for either a particular group or a logical combination of groups (see section 5.3.1). Depending on some or all of the groups that a particular machine belongs to, the defined configuration changes are applied. This process continues through each of the LOSCMF execution cycle until the particular machine is in the defined configuration state. It (LOSCMF) even allows for changes to be applied, which require the machine to be rebooted for being effective.

Each of the changes being applied is logged both locally as well as on the centralized storage location, since collecting log information from each of the machines at a later point of time is not a feasible approach (involves additional network bandwidth utilization, system resource consumption as well as 100% uptime of the machine). The log information files are further processed by the reporting mechanism and are stored in the defined MySQL [31] database tables. The reporting mechanism can be used by the system administrators to generate necessary reports. The LOSCMF solution can be considered to include the following major components; local computer system, centralized storage location, MySQL [31] database tables, and the reporting utility.

Figure 1-2 below presents a pictorial representation of the interaction of various components within the LOSCMF.

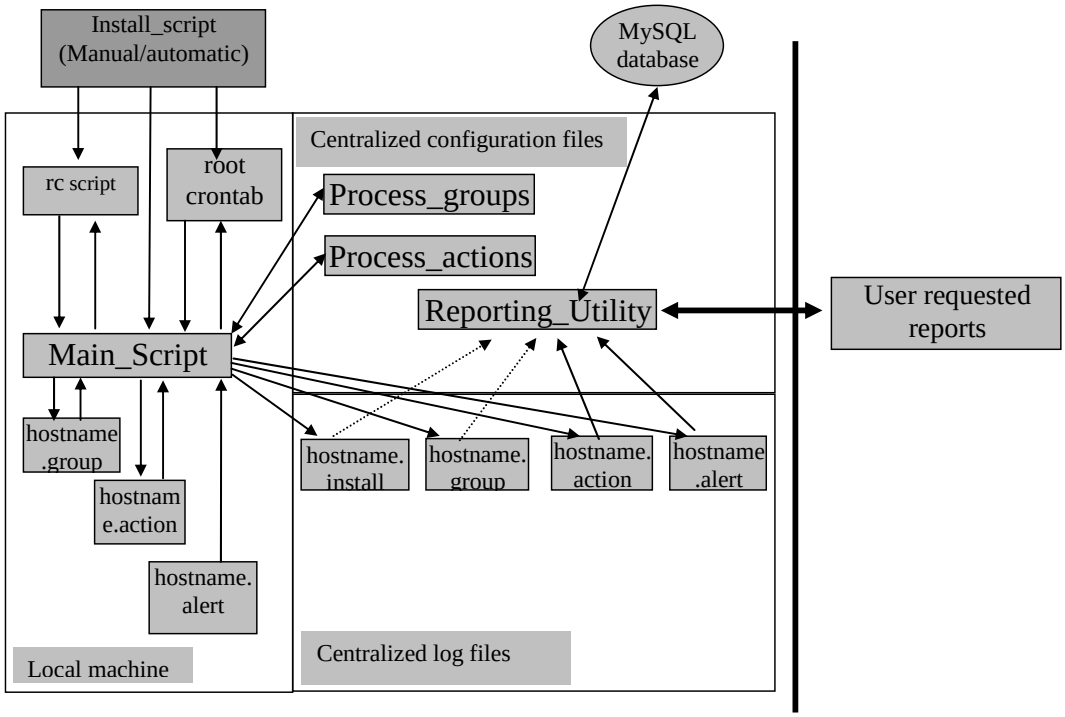

Figure 1-2: Component Interaction within Linux Operating System Configuration Management Framework

1.2.3 Validation of the Above Mentioned Goals

We have analyzed the available open source as well as commercial solutions for their merits and drawbacks in chapter 3. Each of the respective sections in chapter 3 provides a detailed summary and the drawbacks associated with each of these solutions. We have applied the LOSCMF solution that has been developed through this thesis within a medium sized organization, which has about 6000 Linux computing nodes. For the purpose of not disclosing any proprietary information, we will refer to the above mentioned medium size organization as the enterprise in this thesis. We will present a detailed comparison of our approach with the existing approaches.

Finally we will provide useful statistics in terms of the ease of use of the LOSCMF solution, the reduction in Linux operating system related support call volume, decrease in the over-all turn around time for pushing the operating system configuration changes across the environment.

1.3 Organization of Thesis

Chapter 1 includes the motivation behind this work, the goal set to achieve as a result of this work, and the organization of the thesis. Chapter 2 provides a description of the related academic work. Chapter 3 provides a brief description of the current available configuration management tools for Linux operating system. In this chapter a close analysis of both the open source and commercially available solutions is presented. Chapter 4 presents a brief description of the open source technologies being used to develop the LOSCMF framework. Chapter 5 presents a detailed description of our work, Linux Operating system Configuration Management Framework (LOSCMF) solution. Chapter 6 presents the deployment and evaluation of the LOSCMF solution being developed through this work within the enterprise. It presents some of the useful statistics that were collected, which include the overall user friendliness of the LOSCMF solution, the reduction in Linux operating system related support call volume, the total cost of ownership, and increase in overall operational efficiency. This chapter provides a tabulated comparison of the LOSCMF approach with the existing open source as well as commercial solutions. Finally we provide a real world example of the LOSCMF application within the enterprise. Chapter 7 presents the conclusions and future work

2 Related Academic Work

Advancements in the IT sector have given rise to a wide variety of configuration management solutions both in open source as well as commercial space. Although most of these solutions have been used to good effect by system administrators and IT engineers, they require complex understanding of the operating system internals and solid programming and scripting skills for being effectively deployed within an enterprise. Moreover the commercial solutions are associated with further expenses such as purchasing, licensing and maintenance costs.

Through this thesis, we have the idea of developing a Linux operating system configuration management framework (LOSCMF) solution that addresses the drawbacks associated with the current configuration management solutions mentioned in section 1.1. We have looked in to the work of Desai et al (see Desai et al in Table 2-1) [15], which describes Bundle configuration system (BCFG) [15]. BCFG is a configuration management tool for the heterogeneous cluster management at the Argonne National Laboratory. BCFG [15] is a symbolic configuration management tool and uses multi-tiered configuration description. BCFG has been designed as a metadata based configuration management system.

Anderson et al (see Anderson et al in Table 2-1) [2], describe Local Configuration system (LCFG) [48], an automatic configuration and installation system. It was originally developed by the department of Computer Science at Edinburgh University to handle a large set of Solaris [33] machines within their network. It was designed to satisfy several requirements that were not found in the then existing utilities.

Choi et al (see Choi et al in Table 2-1) [17] have developed Xml-based configuration management system (X-CONF), which is an Extensible markup language (XML) [27] based configuration management system for distributed systems. X-CONF architecture consists of XML [27] based manager and XML [27] based configuration agent components. The communication between these components is carried through Simple object access protocol (SOAP) [28].

Since our work involved evaluating some of the open source configuration management solutions, we looked in to the work of Chaudry (see Chaudry in Table 2-1)

[4]. This thesis work presents user survey of configuration engine (Cfengine) [25] usage practices.

Anderson et al (see Anderson et al in Table 2-1) [3] described an autonomous reconfiguration approach by integrating LCFG [48] and SmartFrog [3] framework. SmartFrog is a resource configuration system designed to manage complex services spread over multiple computing nodes.

Anderson et al [2] described LCFG [48], a large scale UNIX [29] configuration system. LCFG [48] approach uses a configuration language and a central repository of configuration settings. We are interested in such an application that would allow us to develop the Linux Operating System configuration management framework as well as satisfy the criterion mentioned in section 1.1. Desai et al [15] in their work have developed BCFG [15], a metadata based configuration management system. Its configuration language allows for extensive configuration reusability and verification. BCFG [15] includes the functionality for validating the changes that need to be performed, detecting the current anomalies and providing the functionality for constructing configuration fragments as per the requirement. BCFG [15] has been implemented on a 320 node testbed [15] cluster at the Argonne National Laboratory. Choi et al [17] have developed an XML [27] based configuration management solution. They have presented a general management model that can be applied to configuration management of distributed systems using XML [27] based schema. Anderson et al [3] discuss about the prototype implementation of a framework by integrating LCFG [48] and SmartFrog [3] systems. They have demonstrated that this framework can be used to construct a real service with autonomic fault recovery [3]. Chaudry [4] has performed a thorough and detailed analysis of configuration engine (Cfengine) [25] in his Master's thesis. The user surveys were analyzed and the results were presented back to Mark Burgess, a professor at Oslo University College, and who is the original developer of Cfengine [25]. Chaudry's [4] work is aimed at helping the Cfengine [25] developers to enhance the existing version of Cfengine [25].

In section 6.2 of this thesis we address the drawbacks of the related work and provide a comparison of Linux Operating System Configuration Management Framework (LOSCMF) solution with the related work.

Table 2-1: Tabular Description of Related Academic Work

Author	Work	Description
Anderson et al	Paul Anderson, Alastair Scobe, "Large Scale Linux Configuration with LCFG" UKUUG Winter Conference, 2002 [2]	Here the authors talk about a large scale UNIX configuration system. This approach uses a configuration language and a central repository for storing the configuration settings. The configuration parameters are stored in key-value pairs.
Desai et al	Desai, N.; Lusk, A.; Bradshaw, R.; Evard, R., "BCFG: a configuration management tool for heterogeneous environments," Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER'03) Page(s):500 – 503, 2003	Here the authors talk about a metadata based configuration management system for Linux clusters. It is a symbolic management tool, and its configuration language allows for extensive configuration reusability and verification. It has been implemented on a 320 node testbed [15] cluster
Choi et al	Hyoun-Mi Choi; Mi-Jung Choi; Hong, J.W., "Design and implementation of XML-based configuration management system for distributed systems," IEEE/IFIP Network Operations and Management Symposium, Volume 1, 19-23 April 2004 Page(s):831 - 844 Vol.1, 2004.	Here the authors talk about an XML [27] based configuration management system for distributed systems. Its architecture uses XML [27] based managers and XML [27] based configuration agents. It uses SOAP [28] as the transport mechanism between the manager and the agent components. It (X-CONF) has been applied to the configuration management system for NG- MON [18], which is a distributed and real-time Internet traffic monitoring and analysis system

Table 2.1: continued

Author	Work	Description
Chaudry	Raheel A Chaudry, "User survey of practices using Cfengine", Master's Thesis, University of Oslo, 2005	Raheel A Chaudry [4] has performed a thorough and detailed analysis of Cfengine [25] in the Master's thesis. He developed comprehensive user surveys relating to all aspects of Cfengine [25] usage. The user surveys were analyzed and the results were presented back to Mark Burgess, a professor at Oslo University College, who is the original developer of Cfengine [25]. Chaudry's [4] work would help facilitate the developers to work on enhancing the existing version of Cfengine [25] and make it more user-friendly
Anderson et al	Paul Anderson, Patrick Goldsack, Jim Patterson, "SmartFrog meets LCFG: Autonomous Reconfiguration with Central Policy Control", Proceedings of the 2003 Large Installations Systems Administration (LISA) Conference, 2003	Here the authors talk about an autonomous reconfiguration approach by integrating LCFG [48] and SmartFrog [3] framework. LCFG [48] is an automatic configuration and installation system. SmartFrog [3] is a resource configuration system designed to manage complex services spread over multiple computing nodes.

3 Current Configuration Management Systems

There are a wide variety of Linux operating system configuration management systems currently available. These include open source solutions like Cfengine [25], Linuxconf [21] as well as commercial solutions such as Red Hat Network (RHN) [61] or HP Open View OS manager [62]. For the purpose of our study we considered some of the most widely used ones in the open source as well as the commercial segment. Section 3.1 describes some of the open source solutions, whereas section 3.2 presents some of the commercial systems available. In section 3.1 we present some of the drawbacks associated with each of the solutions. Most of these drawbacks and limitations have been overcome with this proposed solution of the LOSCMF. In section 3.3 we present a comparison between the open source and commercial configuration management solutions in terms of the costs associated with them.

3.1 Open source Configuration Management Systems

The following are some of the most popular and widely used open source configuration management tools. Some of these tools can be used for other operating systems such as Solaris [33], and Windows [34] along with Linux.

- Cfengine
- COAS
- YAST
- Webmin
- Linuxconf
- PIKT

3.1.1 Cfengine

Cfengine [25] is an open source configuration management system. It is one of the most widely used in the information technology (IT) industry today [4]. Cfengine [25] is a distributed agent framework for performing policy based network and system administration [5]. It could be used on Windows [34] as well as UNIX platforms. Cfengine [25] consists of a number of components as mentioned in [4]. Figure 3.1 [4]

provides a pictorial representation of Cfengine [25] components. Cfengine [25] can be invoked either manually or through cron.

- **cfagent:** It is the most important component and is an autonomous configuration agent that runs on all the installed hosts and checks for any configuration anomalies.
- **cfservd:** It is a file server and remote activation service, which is used to start the Cfengine [25] remotely. It provides access control mechanism based on RSA authentication and IP address. The configuration file cfservd.conf controls the server daemon.
- **cfexecd:** It provides wrapper functionality for cfagent execution and can be used as a reporting utility as well.
- **cfenvd:** It is an anomaly detection mechanism and is part of cfagent.
- **cfenvgraph:** It is a tool for cfenvd, and can be used for providing a graphical representation of system performance.
- **cfkey:** It is a tool to generate public-private key pair for authentication.
- **cfrun:** It is the tool responsible for executing remote agents. It interfaces with cfservd that in turn starts an authorized agent.
- **cfshow:** A tool for showing the contents of the internal databases used by Cfengine [25] in its operation. The contents can be dumped in to text files

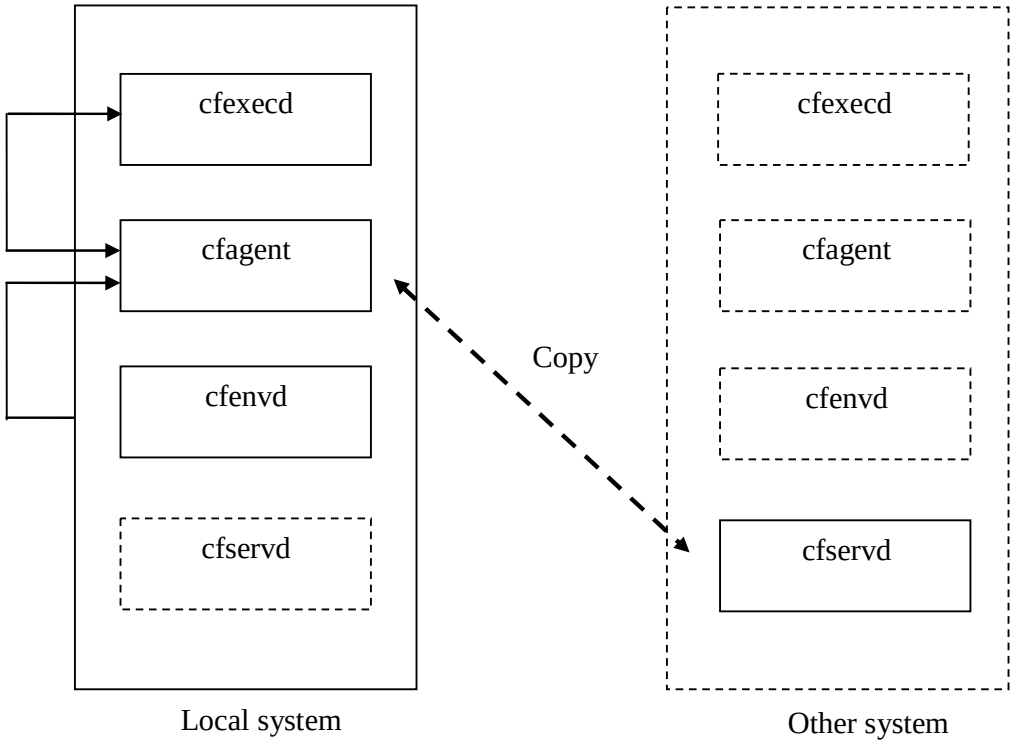

Figure 3-1: Cfengine Components [4]

Following are some of the features offered by Cfengine [25]

- Network interface configuration
- File manipulation
- Symbolic link creation
- Setting Permission bits
- Removing unwanted files
- Script execution
- Managing system processes

Some of the drawbacks associated with Cfengine [25] as described in [4] are as follows

- Complex in nature especially for small scale deployments.
- Configuration files are difficult to understand and maintain, and it is not a customized approach for Linux operating system.

- Novice system administrators can find it difficult to administer.

3.1.2 COAS

Caldera Open Administration System (COAS) [6] was originally developed for being incorporated as the main configuration tool for OpenLinux [63] distribution. It is freely available under the GNU [55] General Public License [64]. The initial development goal of COAS was to provide the ability for adapting the tool according to the underlying platform and portability across other Linux platforms. It (COAS) breaks up the system administration task in to three layers [6], native system data files, implementation of internal representation as a database, and user interaction code. The data model being used is a run-time representation of system data, which hides the underlying on-disk representation from the upper layers. The user interaction code is the top most layer and controls the information being displayed to the user. Internally the data model stores the information in a tree, with the variables being named similar to Simple network management protocol (SNMP) [35]. The underlying schema for the data model is similar to the way a Management information base (MIB) [36] describes the organization of entities within SNMP.

The user interaction code is written in Python [8]. It interacts with the underlying database engine and works on the abstract data representation. The end results are then provided to the user.

Although the COAS project started ambitiously with incorporating a huge amount of functionality, not all of it is part of the final outcome [6]. Some of the observed drawbacks associated with COAS are as follows

- Lack of information in terms of the available documentation and the overall design, and has limited public mailing lists
- Impending issues with the functionality of the tool are left to be solved by the administrators themselves.

3.1.3 YaST

Yet Another Setup Tool (YaST) [26] provides the system administrators with an integrated solution for automated installation, configuration, and administration of SUSE

[32] Linux Enterprise systems. It is freely available under General Public License (GPL) [64]. It incorporates both Graphical user interface (GUI) and ncurses [65] front ends. It can be used to configure a wide range of the system hardware, security and network parameters. It can also be used for user administration, and installation of additional software. The web updater feature present in YaST enables system maintenance. A screen shot of one of the YaST GUI for network configuration is as follows

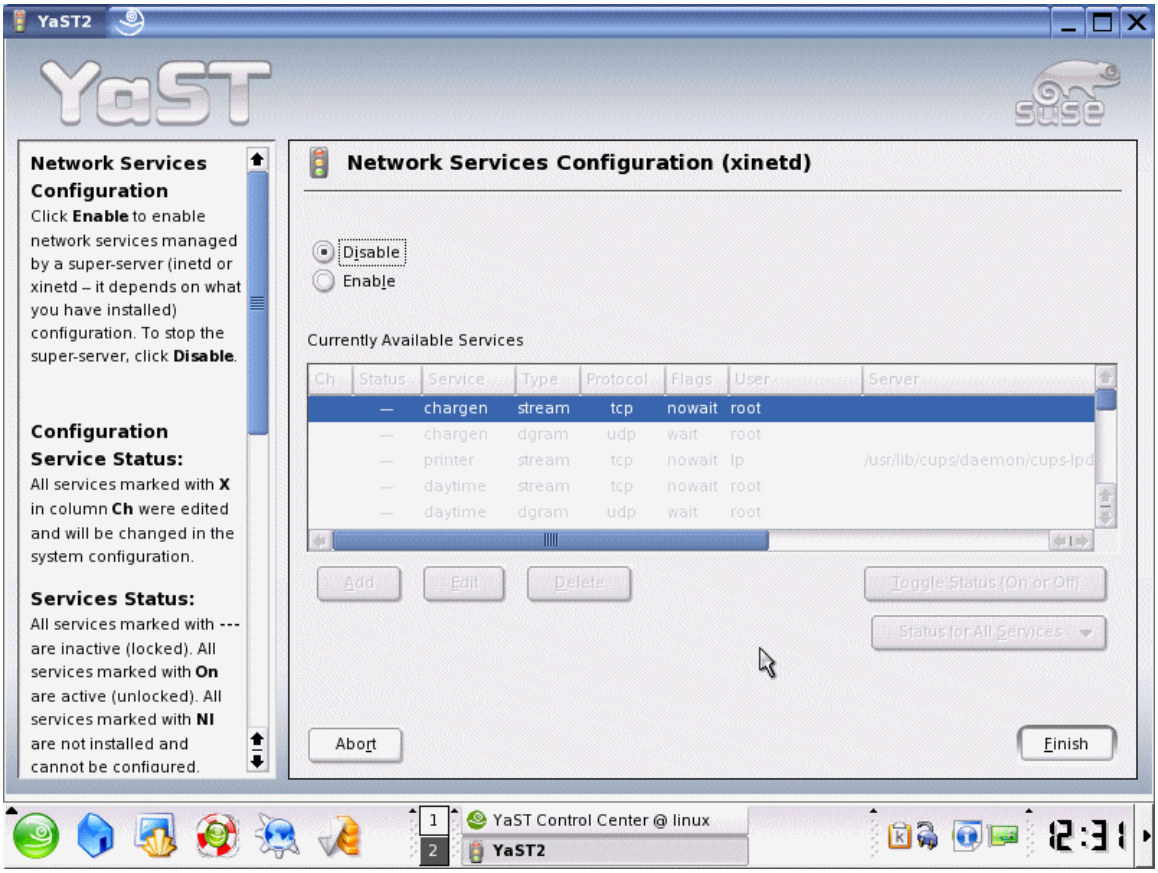

Figure 3-2: Screenshot of YAST GUI for Network Configuration

YaST is currently incorporated into the SUSE Linux distribution. It can be used to perform manual as well as automated installation of SUSE [32] Linux. Some of the drawbacks and limitations observed with YaST are as follows

Its usage is limited to SUSE Linux distribution

It is local to the system that it is installed on and cannot be used to remotely administer systems

No built-in provisions for giving users limited access, and hence it is only useful for rectifying current errors, and allowing new users to administer their systems.

It is not network aware, and hence the administrator has to log into each system that needs to be rectified.

3.1.4 Webmin

Webmin is a graphical interface useful for configuring system services on UNIX platforms. It is predictable in nature and does not modify system configuration files unnecessarily to produce incompatible results [7]. It is freely available under Berkeley standard distribution (BSD) license [80]. It is based on Perl [24] and is running its own process and Web server usually on Transmission control protocol (TCP) port 10000. It can be configured to use SSL if OpenSSL [40] is installed on the system. It is a browser based utility and can run on most of the available browsers such as Mozilla [41], Netscape [42], and also text based browsers such as Lynx [43]. Figure 3.3 below shows a sample view of Webmin [57] running on Mandrake Linux 7.0 using Netscape [42] browser.

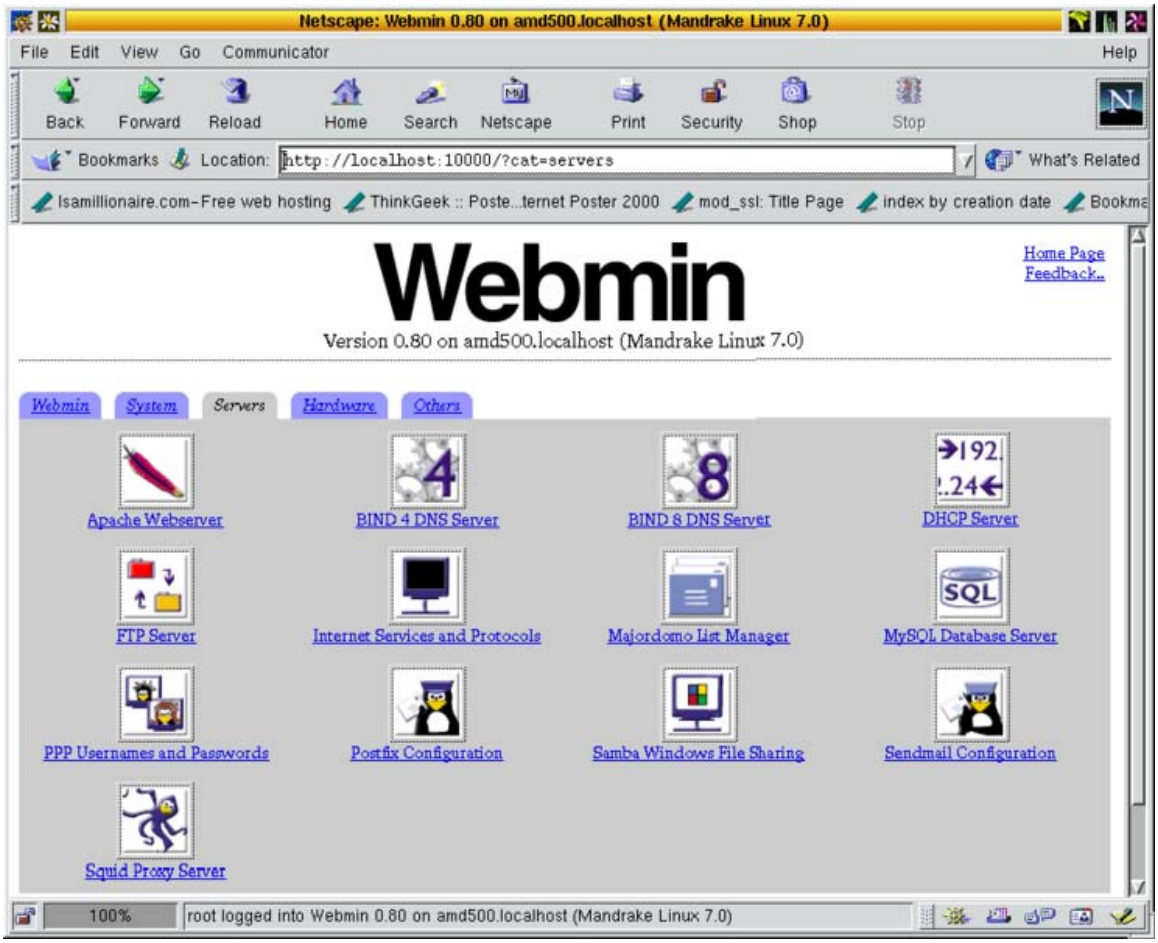

Figure 3-3: Screenshot of Webmin GUI on Mandrake Linux7.0 Using Netscape

Webmin is modular in nature with each module providing configurable options, access control features, and flexible action logging mechanisms [7]. The Webmin GUI interface provides user login mechanism with various privilege levels. Each network usually has multiple Webmin servers and they can access each other though the GUI interface. Webmin detects the underlying operating system that is installed by parsing through the user defined configuration files. Webmin provides the functionality to assign multiple users (usernames and passwords are stored within Webmin) with varying levels of access control. E.g., User1 can be provided with access to shutdown the system, whereas User2 can be provided the access to create/delete and manipulate user accounts only, and is restricted from being able to use other system functionality.

Some of the limitations and drawbacks observed with Webmin are as follows

- Lack of comprehensive documentation in most areas of usage
- Lack of security in terms of maintaining the local Webmin user accounts. The username and password information are communicated in clear text format across the network. Although this security loop hole can be slightly minimized by the ability to grant user access to only certain hosts(s) and networks, it still presents a considerable threat [7]
- Webmin makes the system more accessible to non technical people who must administer systems in such a way that they need not be granted with the actual root privileges on the server [7]
- Webmin defaults to running on port 10000 and needs access control restrictions to be set-up through firewall [7]

3.1.5 Linuxconf

Linuxconf [21] is a graphical configuration management system for the Linux operating system. It is based on GTK+ [79] written in C++. It can serve the functionality of both a configuration management system as well as an activator (to activate the changes made to the system by restarting system daemons). Linuxconf [21] works based off of configuration files. It allows system configuration even before it is fully booted through system initialization scripts. Linuxconf currently ships with Red Hat Linux and is freely available under GPL license model.

It can be used through a web based browser by first running Linuxconf on the system and adding the host(s) and network(s) that need to be allowed to connect. After the initial procedure of adding the hosts, each of them can be contacted through a web based browser. Linuxconf by default runs on port 98 and needs root user credentials for successful authentication. In a stand alone mode, it runs on port 901. Following is a screenshot of Linuxconf version 1.16r3.2 prompting for user credentials for authentication purposes.

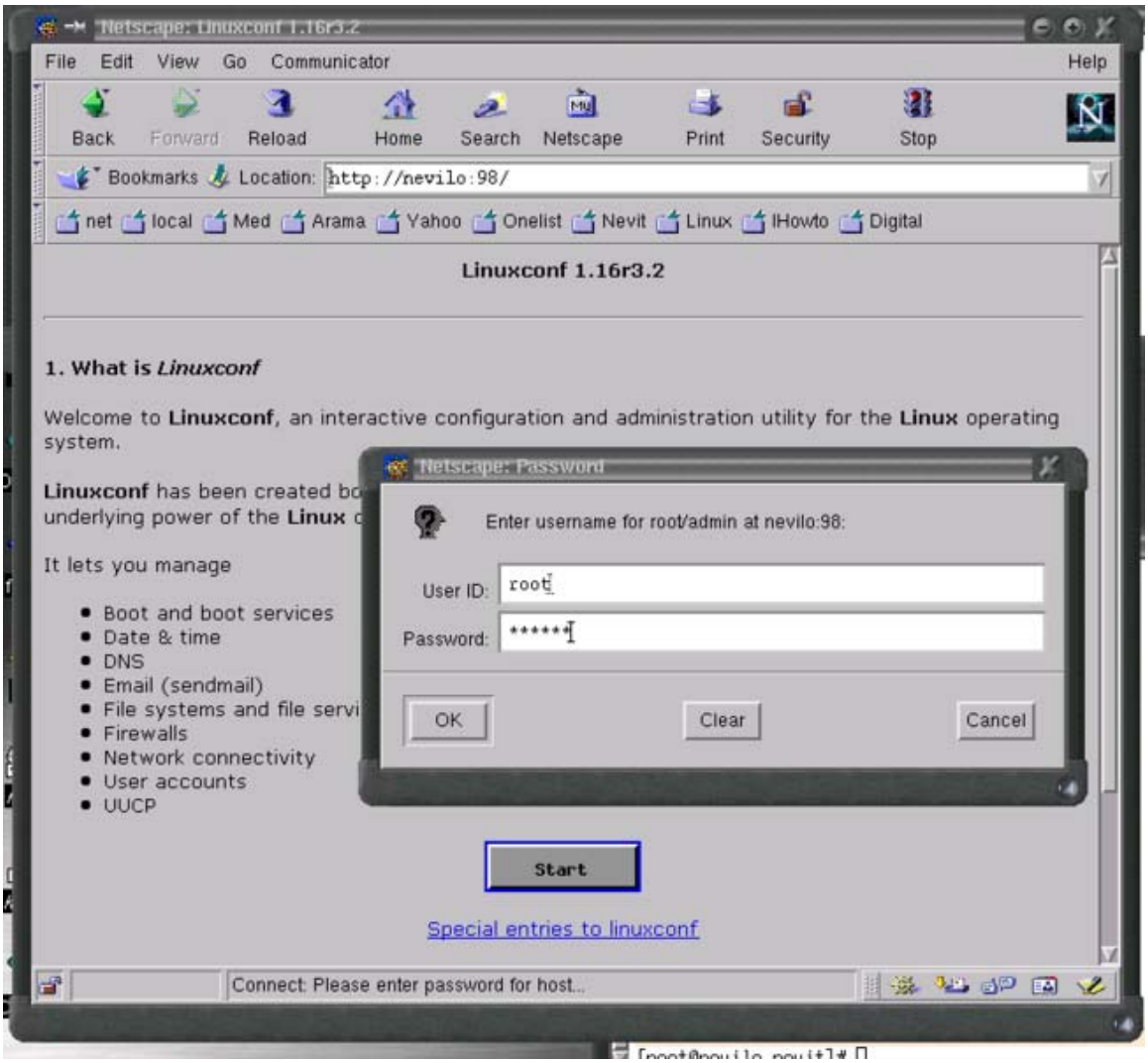

Figure 3-4: Screenshot of Linuxconf GUI on Red Hat 7.2 Using Netscape

User defined modules may be used to access various APIs within Linuxconf. Modules are currently supported in C++ only, and can be used to define new menu entries, command line options and configuration file resources.

Some of the drawbacks and limitations observed with Linuxconf are as follows

- It does not support encryption and hence requires external security measures such as IPsec [76] or other forms of IP level security to be able to connect with an installed host in the network.

- The user needs to compile a parser for each of the package that is using Linuxconf for its configuration management.
- Linuxconf does not contain man pages for additional information, and the built-in help is not very helpful

3.1.6 PIKT

PIKT [11] is multi-purpose software for monitoring and configuring computer systems [11]. It is primarily used for system monitoring, and can also be used as a configuration management utility. It uses the master slave architecture and the master system usually has eight configuration files namely, systems.cfg, defines.cfg, macros.cfg, alerts.cfg, alarms.cfg, objects.cfg, programs.cfg, and files.cfg. These files control the entire set-up. It consists of a management utility, piktc, which when invoked can pre-process the configuration files, install the target files on the client and also restart system daemons. Each client has two daemons running on it, namely, piktc_svc and piktd. The former listens for and responds to piktc requests, whereas the latter launches Pikt scripts at specified intervals.

PIKT [11] has its own embedded scripting language called Pikt and the corresponding interpreter for this scripting language is pikt. The Pikt scripting language has three basic data types, namely strings, numbers and file handles. Pikt uses AWK and GNU regular expressions.

Some of the drawbacks and limitations observed with PIKT are as follows

- Complex configuration files
- Complex installation procedure
- As it uses its own scripting language, the end user is expected to have a thorough knowledge about the scripting language for customization

3.2 Commercial Configuration Management Systems

The following are some of the most popular commercially available configuration management solutions for Linux operating system. We provide a brief summary and explain the associated drawbacks as well as the limitations. For the purpose of our study

we would be considering Red Hat Network and Hewlett Packard (HP) Open View Configuration Management OS Manager

3.2.1 Red Hat Network

Red Hat Network (RHN) [61] is a commercial systems management platform, which can help manage Linux deployment within an enterprise. It aims at lowering ownership costs within an enterprise through complete system life cycle management [61]. It provides its services through a series of modules and architectural components. These Modules include management module, provisioning module and monitoring module, whereas the architectural components include the hosted component, satellite component and proxy components.

The management module helps decrease the administrative overhead through features such as systems grouping, setting system permission levels at the group level, scheduling actions, and multi-platform management. The provisioning module helps deploy the available resources as required through features such as bare metal provisioning, configuration management and kickstart [44] configuration management. The monitoring module helps keep track of the systems and the associated applications through features such as system probes, application probes, alert notifications and reporting utilities.

The hosted model of the architectural component is the basic model for Red Hat Network [61]. Through this model, each of the individual systems connects over the public internet to Red Hat Network and exchanges software packages with the central Red Hat Network servers. Satellite server model uses an intermediate satellite server that is located within the enterprise network and is the central repository for all the updated packages provided by Red Hat Network. The server periodically connects to the external Red Hat Network and updates its local repository accordingly. The proxy server approach requires proxy servers to be added to either the hosted model or the satellite server model. In the proxy server approach the individual systems connect through a Red Hat Network proxy located within the enterprise network to either the satellite or the central Red Hat Network servers.

Some of the drawbacks associated with the Red Hat Network are as follows

- Using the hosted model, presents potential security threats to the organization
- Using the satellite server approach requires dedicated hardware, which is an additional cost
- Lacks in providing a high degree of customization
- The licensing costs grow with the growth in the number of systems

3.2.2 HP OpenView Configuration Management OS Manager

HP OpenView Configuration Management OS manager [56] is a commercial configuration management solution, which provides automated management and policy based provisioning of computer systems [56]. It can be used on a wide variety of platforms such as Windows, UNIX and Linux operating systems.

Some of the key features of HP OpenView Configuration Management OS manager as mentioned in [56] include; Automated Operating System management, Life-cycle automation, Policy based image management, Web-based administration, Verification and compliance reporting

Some of the drawbacks associated with this approach involve

- Initial software purchasing and licensing costs
- Periodic license renewal costs
- Purchasing and licensing costs are proportionate to the growth in the number of computer systems

3.3 Open source Vs Commercial Configuration Management Solutions

Open source configuration management solutions involve the cost associated with maintaining and customizing the code base as per the requirement. In most cases, this customization process can be implemented by the IT engineers within the environment and is facilitated with the help of the open source community forums and public mailing lists. Also, open source solutions do not involve any purchasing and licensing costs and provisioning of dedicated hardware like the commercial solutions. Commercial solutions on the other hand require hundreds of dollars towards purchasing and licensing costs for each managed client within the environment (e.g., Red Hat Network’s Management

Module and Provisioning Module [84]). This cost increases with increase in the total number of managed clients within the environment. Thus when compared with the cost associated with the commercial configuration management solutions, the cost associated with the open source solutions in most cases is minimal.

4 Related Technologies

This section provides a brief description of the open source technologies being used in the development of the LOSCMF. The technologies being used are as follows

Perl

MySQL

Rsync

CVS

PHP

CGI

phpMyadmin

4.1 Perl

Perl [24] is one of the most powerful programming languages currently available to system administrators. It includes various features found in other programming languages such as C, shell [23] scripting, GAWK [53], Sed [49] and Lisp [54]. It was developed by Larry Wall and first released in 1987. It is freely available and is licensed under both Artistic License [66] and GNU General Public License. Perl supports three types of variables, namely, scalars, arrays (arrays of scalars) and hashes (associative arrays of scalars).

Perl is an interpreted language and is optimized for manipulating text files as well as binary data. It uses sophisticated pattern matching techniques to efficiently scan large amounts of data. A Perl script is usually a combination of a sequence of declarations and commands. It comes installed with the default distribution of most flavors of Linux operating system. It is platform independent and highly portable when compared to other existing tools.

Perl provides the following useful features, which are essential for large projects

- Modularization
- Object-oriented techniques

- Arbitrary data structures
- Functionality to have compiler as well as runtime built-in checks
- Supports advanced set of regular expressions
- Allows system calls usage
- Provides run time loading of modules

For the purpose of developing the LOSCMF, we have used Perl version 5.6.0 with added MySQL DB modules support.

4.2 MySQL

MySQL [31] is the most popular open source SQL database management system [31]. MySQL is written in C and C++ and has been tested with a wide range of compilers. It works on various platforms and uses GNU Automake [81], Autoconf [82] and Libtool [83] for portability. The MySQL server design is multi-layered and comprises of independent modules. It is multi-threaded and uses kernel threads. It provides both transactional as well as non-transactional storage engines. SQL functions are implemented in MySQL using a highly optimized class library.

MySQL supports multiple different data types such as float, double, varchar, date, time and provides fixed length and variable length records. MySQL server offers a high level of security using privilege and encrypted password based authentication. It (MySQL) provides host based authentication. MySQL is highly scalable and can handle large databases. It can scale up to databases that contain fifty million records and can provide up to 64 indexes per table [31]. Clients on multiple platforms can connect to a MySQL server using protocols such as TCP/IP sockets. MySQL client programs can be written in many languages such as C, C++, Eiffel [67], Java [9], Perl [24], PHP [45], Python [37], Ruby [68], and Tcl [69]. MySQL has several client and utility programs such as mysqldump and mysqladmin.

For the purpose of developing the LOSCMF, we have used MySQL version 4.1.10a.

4.3 Rsync

Rsync [13] is a utility for incremental file transfers across networks. It is freely available under GNU General Public License Version 2. It uses the “rsync algorithm” and provides a fast method for syncing remote files. Some of the features offered by Rsync are as follows [13].

- It has the ability to update entire directory trees and file systems
- It has the ability to preserve symbolic links, hard links, file ownership, permission bits, devices and time stamps.
- It can be easily installed without special privileges
- It provides internal pipelining, which in-turn reduces latency for multiple files
- It can use either rsh, ssh or direct sockets as the underlying transport mechanism
- It supports anonymous rsync, which can be used for mirroring purposes

For the purpose of developing the LOSCMF, we have used Rsync version 2.6.8.

4.4 CVS

Concurrent versions system (CVS) is an open source version control system. It is widely used and freely available under GNU General Public License. It uses a client server model architecture where in the server maintains a master copy of the file set, and the client can connect and download a local copy. Once necessary changes are made to the local copy, the changes can be checked back in to the master copy on the server. CVS [47] provides the following functionality

- It allows access to multiple versions of the same file to revert any changes
- It provides the ability of concurrent editing of files
- It allows customization of the file locking mechanism being used.

For the purpose of developing the LOSCMF, we have used CVS version 1.11.21

4.5 PHP

Hypertext Preprocessor [PHP] is a general purpose server-side scripting language, which can be embedded in to Hypertext Markup Language (HTML) for web development. PHP [45] scripts are executed on the server prior to the web pages being displayed to the user. It is freely available and provides built-in support for many databases such as MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, and Generic ODBC. PHP runs on several different platforms such as Windows and Linux and is compatible with most of the web servers available today. PHP scripts usually contain scripts, HTML tags and text. PHP is a loosely typed language and hence converts the variable to the appropriate data type depending on the usage. PHP can help build web pages with the following functionality.

- Ability to query a database
- Ability to upload user files
- Ability to provide custom configuration basing on the user login

For the purpose of developing the LOSCMF, we have used PHP version 4.3.9-3.22.4

4.6 CGI

Common Gateway Interface (CGI) is a standard protocol for interfacing external application software with an information server [14]. A web server contains locations which are defined to be served by a CGI [46] program. Upon receiving a request matching to a corresponding Uniform resource locator (URL), the web server invokes the corresponding along with the user sent information as the input to the program. The resulting output from the program is collected by the web server and sent back to the user along with appropriate headers. Each invocation of the CGI request generally requires a fresh copy of the program to be executed. The location of the CGI programs on the web server is usually controlled by the web master. A CGI program can be written in any of the following languages, which allow it to be executed on the system.

- C
- C++

- Perl
- TCL
- Unix shell script
- Visual Basic
- AppleScript

4.7 phpMyadmin

phpMyadmin [70] is a extremely useful utility written in PHP [45] for the administration of MySQL over the web. It is freely available under GPL [64]. It can be used effectively for various MySQL administration tasks such as creating and dropping databases, creating editing and altering tables, dropping editing and adding fields, executing structured query language (SQL) statements. phpMyadmin requires PHP version 4.1.0 or newer and MySQL 3.23.32 or newer to be running on the system to be installed. phpMyadmin itself does not provide user management functionality, but passes the username and password information to the underlying MySQL, which then uses the defined access levels for authentication. Following is the screenshot of phpMyadmin version 2.6.0 web interface

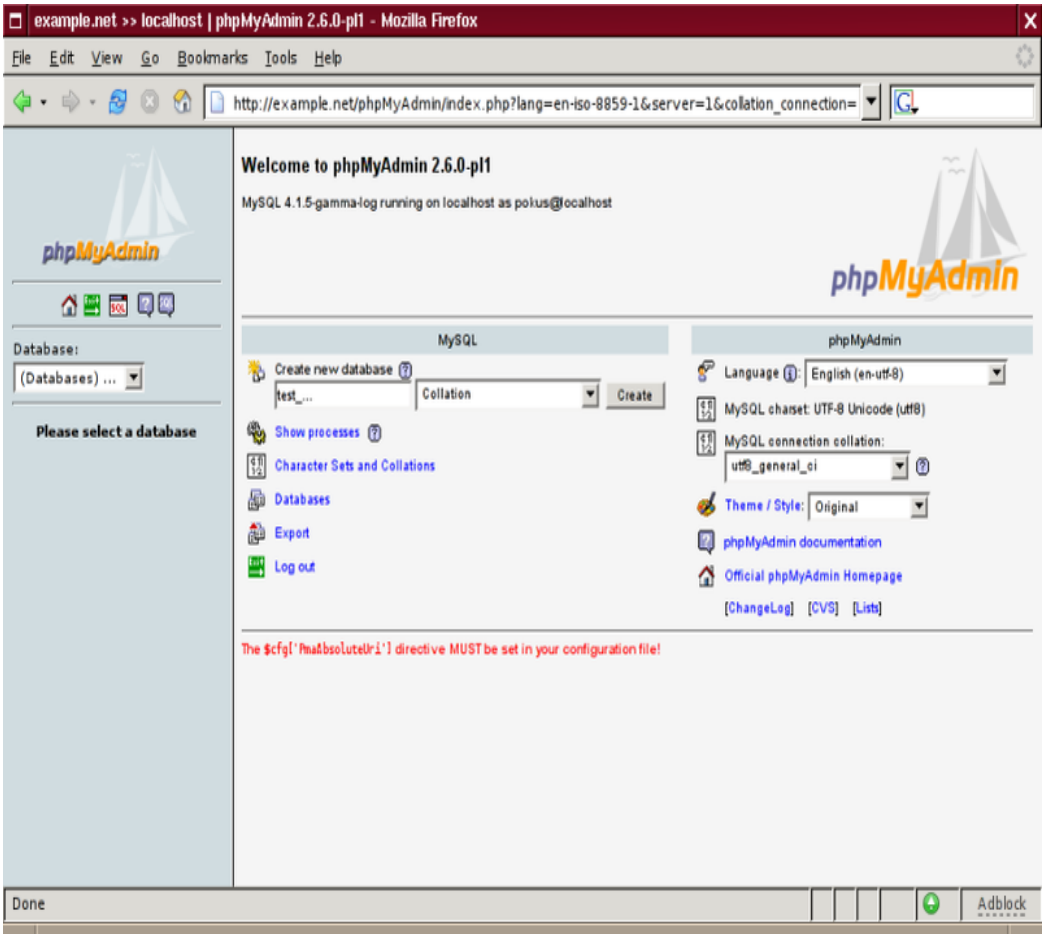

Figure 4-1: Screenshot of phpMyadmin 2.6.0 Administrative Interface

5 Linux Operating System Configuration Management Framework

This chapter provides an insight in to the technical makeup of the Linux Operating System Configuration Management Framework (LOSCMF) solution that is being developed through this thesis. Section 5.1 describes the key techniques in the design of the LOSCMF solution. Section 5.2 presents a detailed description of the components that make up this framework. Section 5.3 provides a brief description of the LOSCMF configuration files. Section 5.4 provides a detailed description of the flow of events during the LOSCMF execution cycle. Section 5.5 provides a brief description of the technical makeup of the LOSCMF solution. Appendix B provides a detailed description of the LOSCMF associated files and appendix C provides a detailed description of the LOSCMF configuration files.

5.1 Key Techniques in the LOSCMF Design

Automating the process of operating system configuration management has been around since the beginning of the need for efficiently managing IT resources within a given environment. There have been several solutions as discussed in chapter 3, which tend to fulfill one or more of the criterion mentioned in section 1.1.

Current available solutions (described in chapter 2 and chapter 3) have the following drawbacks associated with them.

- Usage of additional agents - current solutions require multiple agents (system daemons) to be constantly running for communication and change detection functionality between the server and the client components (e.g., Cfengine [25] requires about fifteen different agents to be running on both the server and the client for performing configuration changes, LCFG requires about fifty different components to be running on the client for being able to provide complete operating system configuration management capability). These system daemons being used for communication and change detection involve unnecessary overheads in terms of the system resource consumption and network bandwidth usage.

- Usage of complex configuration language - some of the current solutions use their own configuration language, which in most cases is difficult to understand and customize for individual requirements (Cfengine, BCFG, LCFG use their own configuration language, which is difficult to understand by both novice as well as experienced system administrators).
- Usage of special programming and scripting language - some solutions require the use of their own scripting and programming language for carrying out the configuration management changes (e.g., PIKT uses its own scripting language called pikt, Linuxconf requires a parser compilation for each of the packages being configured, LCFG defines the client configuration in an XML based profile)

As all the limitations and drawbacks mentioned above have not been addressed in any of the available configuration management solutions, we have developed the LOSCMF, which does not require additional agents to be running on both the client and the server components, uses a simple configuration language specified in plain text format and does not require the use of any specific scripting and programming language. In addition the LOSCMF design allows for end user customization. The section below describes the LOSCMF design that helps achieve these objectives.

- Usage of additional agents – A client upon the LOSCMF execution (either manually, through cron or system initialization script) reads the master copy of the configuration file set from the NAS server using NFS protocol. The LOSCMF execution does not require any additional agents (system daemons) to be running on both the client and the server for establishing a communication mechanism as well as determining the changes that need to be made to the current configuration. The client after reading the master set of the configuration files from the server performs the required computation locally. Once the required changes that need to be performed (basing on the client's current configuration) have been determined and applied, the client contacts the server to replicate the log information of all the changes that have been performed.

- Usage of complex configuration language – The LOSCMF uses a very simple configuration language. All the parameters are separated using "::" symbol and the configuration files are stored in plain text format. The LOSCMF group configuration file is a simple collection of 4 fields through which the desired configuration can be specified. It is flexible and easy to customize. The operators used are == (equals), =~ (matches), != (not equals), <= (less than), >= (greater than), <> (range operator). The LOSCMF action configuration file is a simple collection of 7 fields through which the desired actions can be specified (see section 5.3 for further details on the configuration files).
- Usage of special scripting and programming language – The LOSCMF does not require any special scripting or programming language. Any administrative language such as Shell, Perl, and PHP can be used for group determination and change propagation. It offers flexibility based upon the skill set level of the administrator.
- Customization - Another key feature of the LOSCMF solution is the ability to update the main process flow mechanism as per the end user requirement. The LOSCMF is dynamic in nature, in that, it initially checks for any updates in the main process flow. If the main process flow is found to be updated, another execution of the LOSCMF is triggered before the determination of groups for the host in the current context. Depending on the groups determined, the relevant actions defined in the configuration file set are executed.

5.2 Component Analysis

The objective of this thesis is to create a configuration management framework that defines the configuration of all the Linux computer systems within the environment. The LOSCMF when executed on every registered computer system in the environment, initially checks for any updates in the process flow and later parses the master configuration file set. The group configuration of each installed host is then determined dynamically and later checked against the master file set for any anomalies from the defined configuration. Upon detection of any anomalies, the corresponding fixes are applied automatically. Each of the installed hosts is classified in to groups basing on the

configuration properties that distinguish them. This framework uses a flexible system of ‘groups’, which helps to single out a specific set of hosts basing on their current configuration parameters. Following are the various components that comprise this framework.

- Client
- Server
  - NAS Server
  - MySQL Database Tables
- Groups
- Actions
- Web interface
- Reporting mechanism

5.2.1 Client

A client is defined as a Linux host present in the current environment that is installed with the LOSCMF. All the clients execute the framework either through cron [30] or system initialization script. Any configuration change that does not require a machine reboot cycle is performed in the cron [30] execution cycle. A configuration change, which requires a client reboot for being effective, will be performed through the system initialization script (e.g., changing module options for Network Interface Card). The above method of executing a configuration change that involves system reboot cycle, through the system initialization script is used to ensure that a working machine is not rendered unusable due to any propagated changes. Each client is categorized in to “groups”, and depending upon these groups the specified “actions” are performed.

5.2.2 Server

The server component is further divided in to two components, namely Network Attached Storage (NAS) [51] server and MySQL database tables

5.2.2.1 NAS Server

The master configuration files and related scripts are stored on a centralized storage location, which is usually a Network Attached Storage (NAS) location that is accessible to all the clients within the environment. Each registered host (client) will have a directory space on the NAS location to store the log information resulting from the framework execution. The log information is replicated both locally as well as on the NAS location, since collecting localized log information from each of the machines at a later point of time is not a feasible approach (involves additional network bandwidth usage, system resource consumption as well as 100% uptime of the respective machine). The master configuration files that reside on the centralized storage location are maintained under concurrent version systems (CVS) [47] control.

5.2.2.2 MySQL Database Tables

The MySQL server running on a dedicated host in one location could serve the purpose of a database master server. Each of the remote locations could communicate with the master server to periodically dump the client logs. A list of the LOSCMF database tables is provided in Table 5-1 below.

Table #1 stores information about the actions defined in the master configuration file set. Table #2 stores the log information output from each of the defined actions. Table #3 stores the group configuration information. Table #4 stores the actions performed by each of the installed hosts. Table #5 stores the archive information of the actions performed by the installed hosts. Table #6 stores the audit information regarding the actions performed by each of the installed hosts. Table #7 stores the group information for each of the installed host. Table #8 stores the archive of the group information for each of the host. Table #9 stores information about the hosts which fail to execute any group configuration scripts. Table #10 stores information about the hosts which are part of the defined implicit exception list.

Table 5-1: LOSCMF Database Tables

Table #	Table Name
1	loscmf_action_conf
2	loscmf_action_log
3	loscmf_group_conf
4	loscmf_host_action
5	loscmf_host_action_archive
6	loscmf_host_audit
7	loscmf_host_group
8	loscmf_host_group_archive
9	loscmf_error_group
10	loscmf_implicit_exceptions

5.2.3 Groups

Groups are the tokens that categorize a given host. The configuration management framework parses a group configuration file and determines all the groups that a host belongs to. It then applies “actions” that match a given group or a logical combination of groups. Essentially, it implies that the state of a given group, either on or off, determines the behavior of the program. Since the framework runs independently on each installed machine, it (machine) knows its host name, operating system version, architecture, and other related information. The groups are evaluated based on the result of either executing a script (e.g. A host either belongs to Clearcase [50] group or not) or a command.

5.2.4 Actions

Actions are the commands or scripts that will perform the actual configuration changes on the installed host. Every action is either associated with a particular group or combination of groups. An action has a return value for recording the success or failure in the log information. In addition to the return values, certain actions require logging of the action script itself as the framework offers the capability to enable this functionality (e.g. collecting the reboot history logs for each registered host requires exit value of the actual action script as well as logging of action script output).

5.2.5 Web Interface

The framework offers a simple web interface for ease of use. The user can use this interface to gather useful information. The user query interacts with the database or the backend log files, which are updated periodically by the framework execution. Sample information such as below can be easily obtained through the interface

- Status of the locations that have hosts registered with the framework's database
- Information regarding the existing groups within the framework
- Information regarding the existing actions within the framework
- Status of current actions that are being applied on registered machines at various locations
- Extracting the Action log information as mentioned in section 5.2.4 for each individual host.

5.2.6 Data Dumping and Reporting Mechanism

The reporting mechanism consists of two parts. The MySQL [31] master database tables form the backend, while Perl [24] scripts with MySQL DB modules support form the front end. The log information files are collected from various locations and the database is populated accordingly. Periodic reporting can be run using the Perl [24] scripts through the command line. Some of the sample reports could be of the form

- List of hosts that have successfully carried out Action "n"
- List of hosts that belongs to group "clearcase [50]"
- List all hosts that belong to a particular operating system version
- List hosts that have not performed a particular action.

These reports can be invoked through command line or through the web interface. The framework has a robust built in Perl [24] script to run the reports requested through the command line.

5.3 LOSCMF Configuration Language

The LOSCMF configuration language involves the definition of the group configuration file and the action configuration file. Sections 5.3.1 and 5.3.2 below

provide a brief description of the group and action configuration files. Detailed description of the configuration files can be found in appendix 0.

5.3.1 Group Configuration File

It is a simple collection of 4 fields (group name, group determination script or command, operator and expected value) through which the desired configuration can be specified

- It is flexible and easy to customize
- The operators used are == (equals), =~ (matches), != (not equals), <= (less than), >= (greater than), <> (range operator)
- Group configuration script invokes the group determination script or command (written in any administrative language such as Perl, PHP, Shell)

5.3.2 Action Configuration File

It is a simple collection of 7 fields (Action name, one time or recurring execution, reboot status, action execution script, logical combination of groups, description and optional log file) through which the desired actions can be specified

- An action can be defined for a set of groups (logical combination of groups using and '&', or '|', and not '!' operators
- Logical combination of groups allows to create exception groups for enhanced flexibility (e.g., group1&group2!group3)
- Action configuration script invokes the action execution script (written in any administrative language such as Perl, PHP, Shell) to perform the configuration change

5.4 Flow Chart Representation of the LOSCMF

This section describes the flow of events during the execution of the LOSCMF on each of the installed host. Figure 5.1 below provides a flow chart representation of the LOSCMF execution. The entire sequence of events could be divided in to three stages.

Stage 1 represents the installation phase of the LOSCMF by the administrator. The installation script creates the appropriate directory structure on the client to store the local log information and the executable scripts. It also creates a system initialization script and a cron [30] entry for root user.

Stage 2 depicts the possible ways to invoke the LOSCMF on a given system, once the LOSCMF has been installed, it can either be invoked through the system initialization script during system startup or through the cron [30] daemon.

Stage 3 represents the entire sequence of flow of the LOSCMF.

The LOSCMF execution when invoked either through the cron [30] or the system initialization script, initially detects for any changes in the actual process flow. The LOSCMF execution can be invoked manually by the system administrator upon requirement. If an updated process flow is detected, the master copy is used to replace the existing local copy. This update functionality modifies the system initialization script to be in sync with the master copy. This process repeats until the local copy of the process flow is in sync with the master copy on the NAS server. Once synchronized, the local system then looks for the other master configuration files on the NAS server.

The group configuration check script that is invoked by the process flow parses the Group configuration file, “group.conf” to determine the groups that a particular client belongs to. The determination process involves executing either a command or a script written in any administrative language such as Perl [24], Shell [23] or PHP [45] (e.g., executing the command “*uname –n”* helps determine hostname of the local host). A particular host can be part of one or more groups and all the group related information is stored locally on the client. E.g., “Host1” belongs to groups RHEL, LINUX, KERVER_2.4.21, LSF [71] if the host is a Red Hat Linux with kernel version 2.4.21-xx and is part of a Load Sharing Facility (LSF) [71] farm type.

After determining the groups that a particular client belongs to, the process flow invokes action configuration check script which parses the action configuration file, “action.conf”. The actions applicable to the current execution cycle are thus determined, and the current execution instigator (either cron or the system initialization script) is determined.

Each action execution is initially compared to previously executed actions to determine if the action is being performed for the first time or if it is being repeated. Unless if the action is recurring or the previous execution was unsuccessful, it is not performed again. An action that is already successfully logged is not repeated. E.g., consider the current action to be “Action1”, it is checked against the action log on the local host. If “Action1” is already present and is not recurring and was successful during last execution, it will not be executed again.

Logging involves exit status, start time, finish time and any additional information that needs to be logged by the action script that was executed. The local group and action logs generated by each host serve as the input files for the master database where different tables store information specific to each hosts groups, actions performed and update times. Reports can be generated on the master database by a user, which give a description of current status of hosts, groups, and actions performed on hosts.

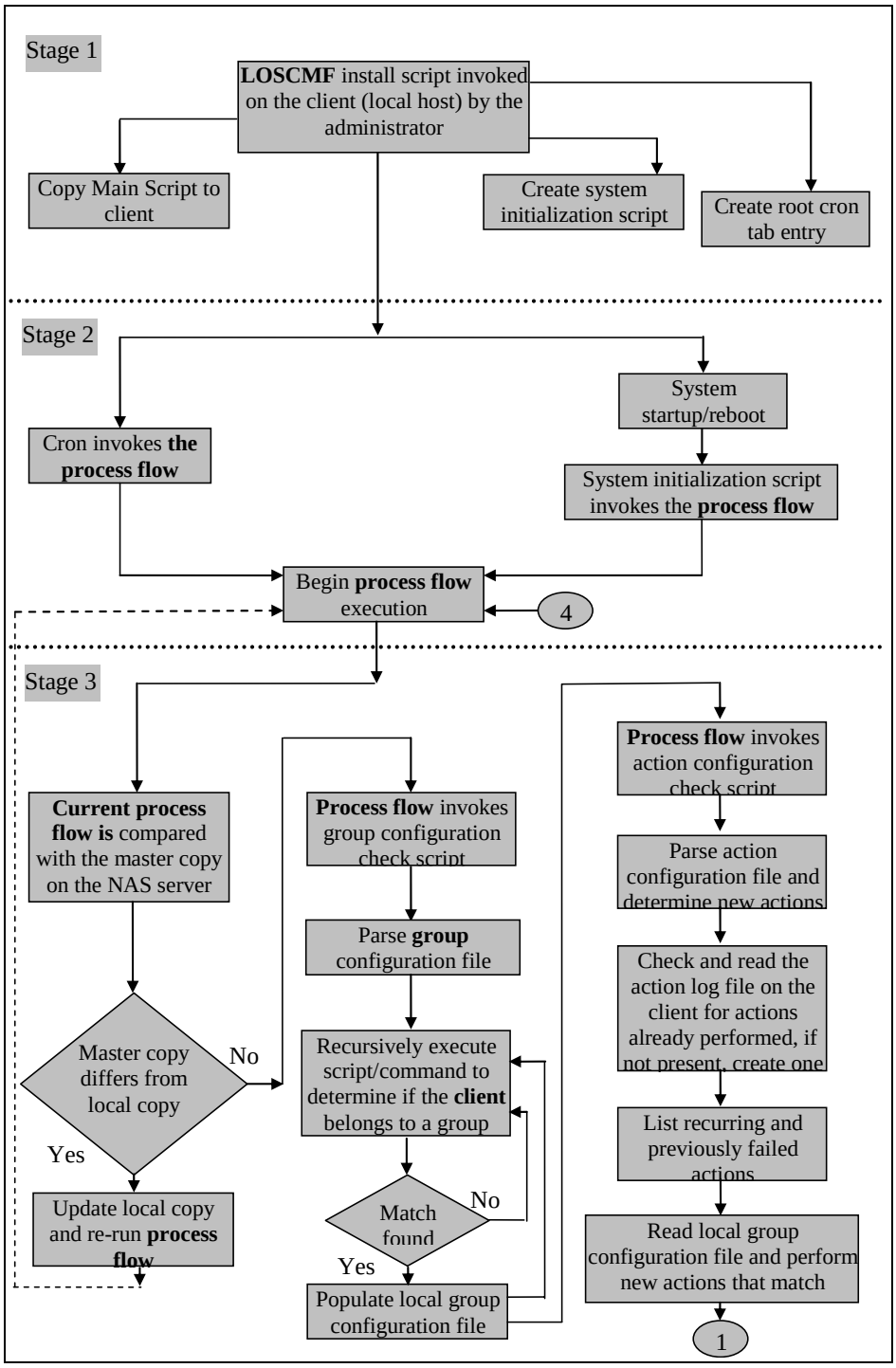

Figure 5-1: Flow chart representation of LOSCMF

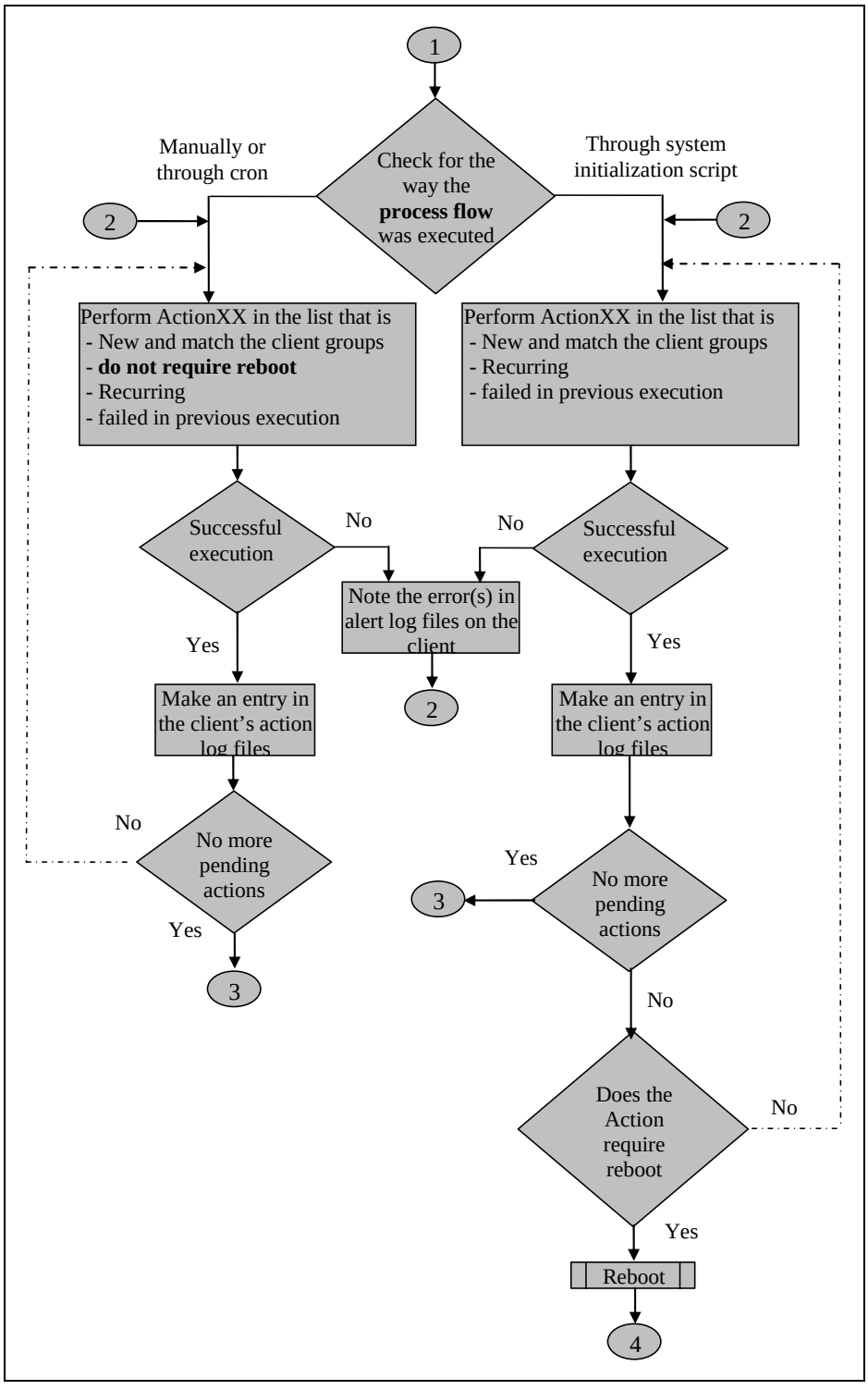

Figure 5-2: Flow chart representation of LOSCMF

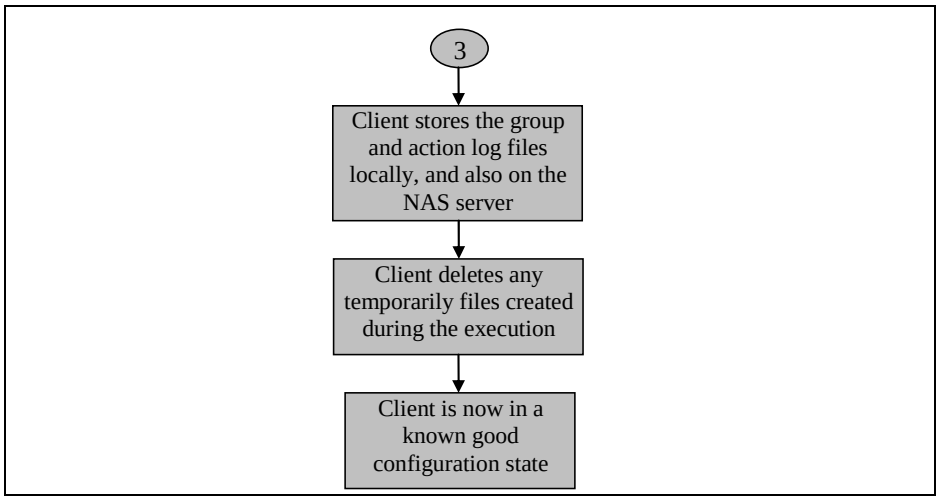

Figure 5-3: Flow Chart Representation of the LOSCMF

5.5 Technical Makeup of the LOSCMF

This section provides details about the technical makeup of the LOSCMF. The initial requirements were provided by a set of five system administrators and IT engineers through their experience of using other open source solutions. This five member team acted as the LOSCMF administration team. Two members within the LOSCMF administration were primarily responsible for the actual development. The rest of the team was part of the planning committee providing valuable feedback through every stage of the development process.

The LOSCMF comprises of about 3000 lines of code, written using object oriented Perl [24], PHP [45], Bourne shell [23] and CGI [46]. Detailed description of my contribution in the development of LOSCMF solution is provided at the end of the document[i].

6 LOSCMF Evaluation

This chapter focuses on applying the LOSCMF solution described in the previous chapters to the Linux environment at the enterprise referred to in section 1.2.3. In section 6.1 we evaluate the implementation of the LOSCMF at the enterprise referred in section 1.2.3, to test our approach. Section 6.2 provides a comparison of our approach with the rest of the available open source as well as commercial solutions as described in chapter 2 and 3. Section 6.3 provides a description of the application of the LOSCMF to a real world example within the enterprise.

6.1 Evaluation of the LOSCMF

In this section we have evaluated the LOSCMF implementation at the enterprise to test our approach and mentioned our results. We have conducted two surveys regarding the LOSCMF approach amongst the system administrators and IT engineers within the enterprise. The purpose of the first survey is to determine their awareness about configuration management utilities as well as their programming and scripting abilities. In addition it has a few questions about the current approach being followed within the enterprise. The purpose of the second survey is to evaluate the ease of use and efficiency of the LOSCMF approach. In addition it has a few questions about the LOSCMF approach. The survey respondent's include both experienced system administrators as well as people with limited experience and novice skill set level. We can thus ensure that the survey results depicted across the various sub sections are consistent in evaluating the LOSCMF solution. The LOSCMF has been evaluated on the following criterion:

- Ease of integration – Procedure and pre-requisites required for integration of the LOSCMF (section 6.1.1)
- Ease of use – Usage characteristics and learning experience of the system administrators (section 6.1.2)
- Modularity – Modular design of the LOSCMF (section 6.1.3)
- Scalability – Scalability of the LOSCMF (section 6.1.4)

- Customization - User customization level (section 6.1.5)
- Overhead – Associated overhead's with the LOSCMF approach (section 6.1.6)
- Efficiency – Efficiency with regards to a system administrator's tasks. The effectiveness of the LOSCMF approach in comparison with the old existing technique (section 6.1.7)
- Ownership cost – Total cost to develop, deploy and maintain the LOSCMF (section 6.1.8)
- Support call statistics – Reduction in support call volume related to issues with Linux operating system configuration management (section 6.1.9)

6.1.1 Ease of Integration

One of the primary requirements while developing the LOSCMF was to provide the ability to seamlessly deploy the framework within the given environment. The following features enable this functionality; the LOSCMF solution does not require dedicated additional hardware and any additional system daemons to be operational, the installation procedure is automated, highly transparent and fail safe.

The only prerequisite for the LOSCMF solution to be functional was to set-up a NAS location or other centralized storage location that is accessible to the clients being installed with the framework. Each of the independent physical sites could have their own individual storage location defined, and the logs from these sites would periodically be collected at one central location. These logs could be processed further to generate customized reports as per the administrators needs.

After the initial knowledge sharing session about the LOSCMF approach with the system administrators within the enterprise, the following steps were performed for integrating the LOSCMF within the enterprise.

- Identifying a central storage location within each of the enterprise's sites (as it had its computing resources spread across multiple physical locations)
- Gathering a master host list of the available systems from a authoritative source

- Communicate with the individual site administrators and provide the relevant installation instructions
- Automatically installing the LOSCMF on to systems, which were being installed periodically

A central NAS storage location was already available within each of the sites within the enterprise and hence the LOSCMF integration in to the environment was extremely easy. An authoritative source such as the existing LDAP [8] and NIS [72] framework was used to gather the master host list of systems for each of the enterprise sites seamlessly. The automated install procedure in itself is fail safe and highly transparent was able to be easily integrated in to the system installation procedure without any issues. As a result of these minimal requirements, the integration process was completed fairly quickly across all sites within the enterprise.

Two surveys were conducted with the administrators and IT engineers within the enterprise regarding the LOSCMF. We have recorded a table of all the quotes received in survey 2 for question 11 as part of the LOSCMF integration process. The general consensus was that the integration of the LOSCMF approach within a small to a mid-size organization is a seamless process.

Table 6-1: Administrator Responses for the LOSCMF Integration Exercise

User ID	Remarks
1	“LOSCMF can be easily installed within a network without many changes to the environment”
2	“Very easy and intuitive process”
3	“Easy to install compared to other tools such as Cfengine, PIKT”
4	“Install script is self explanatory and outputs useful verbose information”
5	“Easy and self explanatory process”
6	“As no additional dedicated hardware is required for installation, it is extremely easy to configure”
7	“Easier compared to other approaches”
8	“Installation on a machine is transparent and does not need any system services to be restarted”
9	“The best part is the installation can be undone when required”

6.1.2 Ease of Use

This section describes the user responses regarding the overall understanding and ease of use of the LOSCMF approach. In addition, we gather information about the programming and scripting abilities of the system administrators as well as their awareness level on the open source solutions described in chapter 3.

Two surveys were conducted with the administrators and IT engineers within the enterprise regarding the LOSCMF approach. These were conducted after the initial knowledge sharing session about the LOSCMF approach. Ten users participated in both the surveys. The purpose of the first survey is to determine the programming and scripting skills of the administrators and their experience using configuration management utilities prior to the LOSCMF approach. The purpose of the second survey is to get the user feedback regarding the deployment, maintenance and the dependence of prior knowledge of a system administrator for using the LOSCMF.

6.1.2.1 Programming and Scripting Skill Set Level of the Users

In survey 1, questions 13 to 16 evaluate the programming and scripting knowledge of the users with respect to Perl [24], PHP [45] and shell [23] scripting (survey 1 is attached in Appendix A.1).

The cumulative scores for each user have been added up. As there are four survey related questions being considered for this evaluation, the lowest score for a user would be ‘4’ and similarly the highest score for a user would be ‘20’. For a user to have minimum level of programming/scripting skill set, we have considered the score to be at least ‘12’, which is about 60%. We can see from the graph that almost 60 % of the users (6 out of 10 users) have the required score level. Thus we can say that 60 % of the users possess programming/scripting skill set.

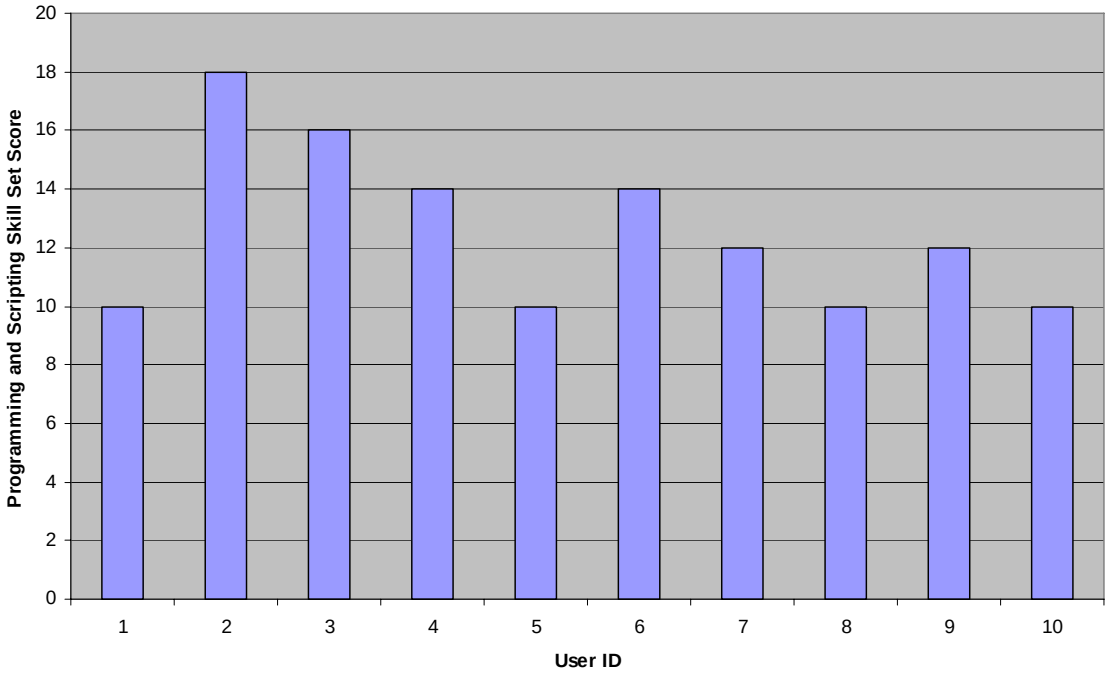

Figure 6-1: Programming and Scripting Skill Set Score by User ID

Question 2 and question 4 in survey 1 provide us with the following information. Of all the users (10 users) who participated in the survey, 80% of the users (8 out of 10 users) have maintained a configuration management solution before. None of the users actually designed and developed their own configuration management solution. Thus we could conclude that although none of the users ever developed their own solution, almost all of them have earlier used and maintained a configuration management solution.

6.1.2.2 Familiarity with the Current Available Configuration Management Solutions

We found from questions 2 and 4 in survey 1 that most of the users have either used or maintained a configuration management solution before. From questions 7 through 12 in survey 1 we can gauge the familiarity of the users particularly with respect to the open source solutions described in chapter 3. Each of the questions could have a maximum score of 5 and a minimum score of 1. As there are six questions in total, the maximum score a user could achieve is 30 and the minimum score is 6. For a user to have

the minimum level of understanding of most of the open source solutions described in chapter 2, we have considered the score to be 18, which is about 60%. We can see from the graph that almost 60 % of the users (6 out of 10 users) have the required score level. Thus we can say that 60% of the users possess at least a minimal understanding of the open source configuration management solutions currently available and described in chapter 3.

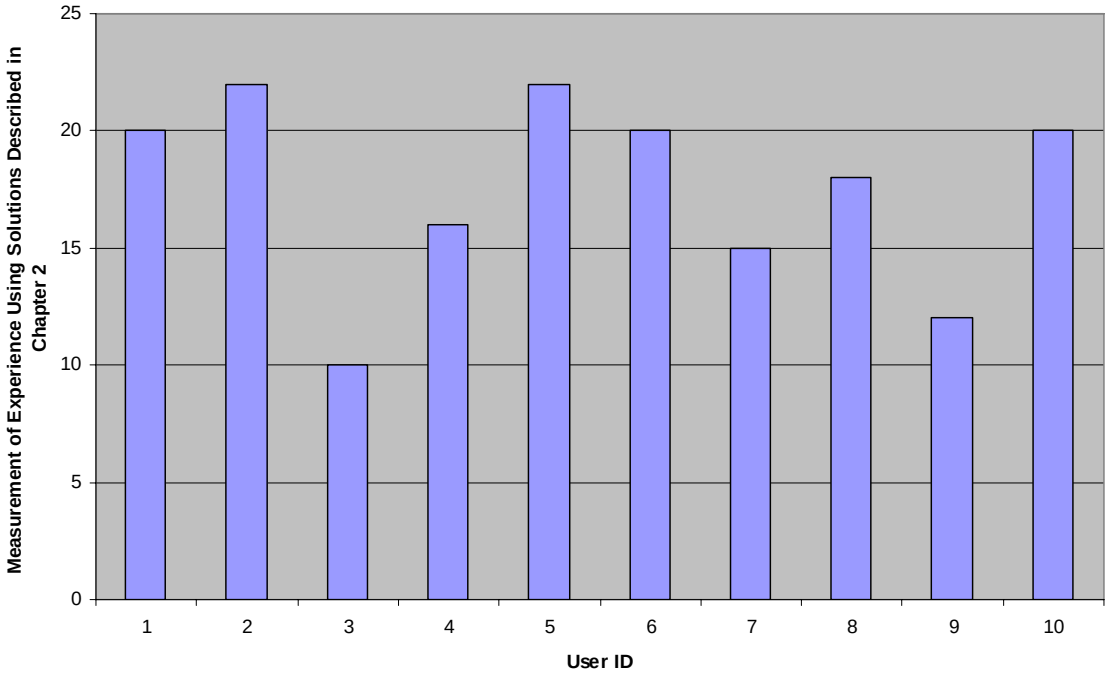

Figure 6-2: Experience Using Existing Open Source Solutions

6.1.2.3 Summary of User Friendliness of the LOSCMF Reporting Interfaces

This section provides a summary of the user friendliness of the reporting interfaces. Questions 9 and 10 of survey 2 require a user to rate the difficulty of using both the command line and web based reporting interfaces on a scale of 1 to 5, with 1 being least difficulty level and 5 being the maximum difficulty level. Statistical analysis of the data was performed and the results are shown below:

Table 6-2: Summary of User Friendliness of Reporting Interfaces

Reporting interface	Number of participants	Minimum score	Maximum score	Mean score
Web based reporting interface	10	1	3	2.00
Command line reporting interface	10	1	3	2.00

The first row in the table presents the scores for the system administrators using web based reporting interface. As the maximum score is at 3, it can be seen that none of the users found it difficult to use the web based reporting interface. The mean of 2.00 suggests that the system administrators were easily able to use the web based reporting interface.

The second row in the table presents the scores for the system administrators using command line reporting interface. As the maximum score is at 3, it can be seen that none of the users found it difficult to use the command line reporting interface. The mean of 2.00 suggests that the system administrators were easily able to use the command line reporting interface.

6.1.2.4 Summary of Ease of Use

In this section we present a summary of ease of use of the LOSCMF approach as a whole from the responses to questions 13, 14 and 15 in survey 2. Section 6.1.2.4.1 presents the overall user responses from deploying and using the LOSCMF approach. Section 6.1.2.4.2 presents the user responses regarding the time taken to push a change through the LOSCMF. Section 6.1.2.4.3 presents the use and learning experience of the web based and command line reporting mechanism.

6.1.2.4.1 Overall Response for the LOSCMF Deployment and Usage

Table 6-3 below presents the user responses regarding the overall deployment and usage of the LOSCMF (question 13 in survey 2). The general consensus was that it is easy to deploy and use the LOSCMF approach.

Table 6-3: Overall Deployment and Usage Experience of the LOSCMF Approach

User ID	Response
1	"LOSCMF while compared to other solutions such as Cfengine and PIKT is both ease to deploy and maintain. The overall design of LOSCMF could be easily understood even by a novice system administrator without any prior configuration management skills"
2	"The design is very modular and usage is relatively easy compared to other solutions available in the open source space"
3	"Rsync and MySQL database server set-up were probably the only difficult parts, but the administrator's guide with step by step instructions helps achieve this task"
4	"Pretty easy to understand and deploy"
5	"Fairly easy"
6	"Easy compared to current approaches being used within the enterprise"
7	"Because of its dynamic nature, it is a very handy configuration management tool"

6.1.2.4.2 Time Taken to Push a Change through the LOSCMF

Table 6-4 below presents the user responses regarding the overall time taken to push out a change through the LOSCMF (question 14 in survey 2). The general consensus was that it is extremely fast compared to the current approaches being used within the enterprise.

Table 6-4: Time Taken to Push a Change through the LOSCMF

User ID	User Response
1	“We have seen that any change could be pushed through within 24 hours from the time when the master configuration file set is updated”
2	“Compared to the old process that required communication between each of the field site system administrators, this process is extremely efficient”
3	“Usually within 24 hours”
4	“Very less when compared to the existing approach of deployment and resultant data collection from each of the field sites”
5	“Relatively quick compared to the current process”
6	“Turn around time less than a day in most cases”

6.1.2.4.3 Usage Experience of Command Line and Web Based Reporting Interface

Table 6-5 below presents the user responses regarding the overall usage experience of the command line and web based reporting interface (question 15 in survey 2). The general consensus was that it is extremely easy to use and self explanatory.

Table 6-5: Usage Experience of Command Line and Web Based Reporting Interface

User ID	Response
1	Easy and self explanatory
2	Adding fields is very easy with simple changes
3	Yes
4	Yes
5	Yes
6	Very Easy
7	Yes
8	Yes

6.1.3 Modularity

As the design of the LOSCMF is modular in nature, additional functionality to the existing mechanism could be integrated seamlessly. After the initial knowledge sharing session about the LOSCMF approach within the enterprise, most of the administrators were able to thoroughly understand the framework and propose additions to the existing functionality. A thorough review of the feedback resulted in additions to the reporting mechanism. These additions were incorporated easily within a short period of time by the members of the enterprise themselves.

6.1.4 Scalability

The LOSCMF architecture has been designed with intent to have it scalable for managing operating system configuration on thousands of computer systems as well as support multiple discrete configurations seamlessly. Currently, within the enterprise, it has been deployed on about six thousand linux based computer systems that are spread across multiple physical locations across the globe. Each of the physical locations is interconnected through secure wide area network (WAN). Synchronization of master configuration file set and log data information between the locations takes place through Rsync [13] running periodically (usually once a day). All of the changes that are made to the master configuration file set are thus propagated through the Rsync [13] mechanism to other field sites. Since the master configuration files are only a few Kilobytes in size, the network bandwidth utilization is low.

We can ascertain that the LOSCMF approach is highly scalable in nature as it is currently installed within the enterprise on about 6000 Linux machines without any issues. In addition it currently supports about 1000 discrete configuration parameters spread across various flavors of Linux operating system such as Red Hat 7.2, Red Hat Enterprise 3.0, Red Hat Enterprise 4.0, SUSE 9.0 [32], SUSE 10.0 [32], and Mandrake 9.0. These numbers are far higher compared to the deployment of each of the approaches mentioned in the related work (See section 6.2.1, section 6.2.2 and section 6.2.3).

6.1.5 Customization

In the LOSCMF, the underlying architecture is independent of the nature of the actual program or command, which is used to make the necessary changes. It (LOSCMF)

provides the flexibility to the IT engineers to use any administrative language such as Shell [23], Perl [24] or PHP [45] to carry out the desired functionality. It only looks for a defined return value from each of the scripts (a return value of 0 indicates a successful completion, whereas a return value of 1 signifies a failure). For further customization, the actual list of machines on which a particular change is to be made can be categorized in to including machines belonging to a particular group and a satisfying sub criterion. The LOSCMF reporting mechanism allows the administrators to generate custom reports based on the desired options. The resulting output can be further processed by using simple command line tools such as GAWK [53] for further customization.

Two surveys were conducted with the system administrators and IT engineers within the enterprise regarding the LOSCMF. We have recorded a table of all the quotes received in survey 2 for question 12 regarding the customization abilities of the LOSCMF within an enterprise. The general consensus was that the LOSCMF approach provides the ability to have a high degree of user customization.

Table 6-6: Administrator Responses for the LOSCMF Customization Experience

User ID	Response
1	“LOSCMF can be molded easily as desired with minimal changes and additions unlike others such as Cfengine”
2	“Easy to modify the output from the reporting tool”
3	“Unlike PIKT which requires the knowledge of its own scripting language, LOSCMF provides the flexibility to use any administrative language such as Perl or Bash”
4	“Easiest part is the independence to use any administrative language”
5	“Even an inexperienced personnel could effectively customize LOSCMF”
6	“Since the main execution script itself can be modified even after deploying it on all clients within the network, this tool offers a high degree of customization when compared to other tools being used”

6.1.6 Overhead

The LOSCMF solution’s design objective is to cause minimal overheads to both the environment and the actual computer system. As mentioned earlier in section 5.1, the

LOSCMF does not require additional system daemons to establish a communication channel between the actual local system and the NAS server which holds the master configuration file set. Within each host, the execution is triggered either by the existing system cron [30] daemon or through the system initialization script (during system shutdown and reboot). It can be triggered manually by the administrator upon requirement. This approach provides the following advantages

- As the client and the server communicate through NFS protocol, there is no requirement of additional system daemons to establish a communication channel between the client and the server components.
- Eliminates the need to periodically check for proper running of any associated system daemons.
- Configuration language does not require additional system resource consumption, as it does not require the use of any binaries.
- As the communication is directed between all the components (including the individual system and the NAS storage location), additional network traffic such as broadcast traffic is eliminated.
- The LOSCMF installation script can be passed with an option to install the cron [30] entry for invoking the LOSCMF execution between specific time intervals. This would help decrease the Input Output (IO) as well as the network load on the NAS server which holds the master configuration file set and the log files information for each of the installed hosts. If all the hosts within the environment were to invoke the LOSCMF at the same time, the communications overhead on the NAS server could render it unusable, as each installed host needs to obtain the master configuration file set as well as replicate the current execution log information.

6.1.7 Efficiency

Section 6.1.7.1 provides a brief summary of the current configuration management process being followed within the enterprise. Section 6.1.7.2 provides the configuration management process through the LOSCMF within the enterprise. When compared, it can be clearly seen that the LOSCMF approach is highly efficient and more

automated than the current existing approach in terms of the overall turn around time, the number of administrators required as well as the amount of the required manual intervention.

6.1.7.1 Summary of Current Process being followed

In this section we summarize the current process flow that is being followed within the enterprise with respect to Linux operating system configuration management. As the enterprise has its computing resources spread across various field offices globally, the task of maintaining a homogenous environment had become a mammoth one. Although each of the individual sites had its own implementation of one of the open source approaches, there was no standard approach being used because of the drawbacks associated with each of them. The system administrators and IT engineers were constantly interacting with each other for exchange of relevant information. The end summary of the changes made, needed to be exchanged manually and processed between these various sites. Some of the approaches already in to production were Cfengine [25], YAST [26], Linuxconf [21] and Webmin [57].

Cfengine [25] was being used in some of the field locations within the enterprise. Owing to its complex nature, it required a seasoned system administrator for deployment and maintenance. Each of the changes being made was to be thoroughly tested across all the platforms before being released in to the production environment. Although Cfengine [25] was robust in functionality, it still needed manual intervention for certain administration tasks such as network configuration changes, which require reboot of the system for being effective. YAST [26] is a very effective way of administering Linux operating system, but its usage is limited only to SUSE [32] platform. Webmin [57] was being used in very small sites with a few computer systems.

In the current process model, any change, which needed to be made was initially tested on all existing operating system versions and later communicated to the field offices. The field offices would later test the same within the local environment and deploy the same in to production. The field offices would later use a mechanism to collect the results of the changes made in terms of percentage of successful completion. The information about the computer systems that failed to carry out the changes was

communicated and was analyzed for any discrepancies. As the number of machines with hardware issues increases, the task of pushing the changes becomes more time consuming.

6.1.7.1.1 Time Taken to Push Configuration Changes

As the current process is not an optimal one, it would take many man hours to propagate even a small change. One of the responses to question 21 in survey 1 is as follows

"The current approach although used various configuration management solutions across the different field sites, involved a lot of communication overhead and dependence of the individual system administrator's availability. Sometimes, when a person was out of office for a prolonged period of time, that particular field site would have to wait for the changes until another person was assigned the task"

The entire process required many hand shakes, and it was particularly affected when a field site administrator was not available for a prolonged period of time.

6.1.7.1.2 Convenience to Push Configuration Changes

As each site had its own version of the solution being used, a central configuration as well as data collection framework was missing. Each site has to be dependent on the individual site administrator's skill set level for programming or scripting the change. One of the responses to question 22 in survey 1 is as follows

"As each site had its own version of the utility such as Cfengine [25], Webmin [57], YAST [26], and Linuxconf [21], the consistency of the changes being made was entirely dependent on the field site system administrator's skill set level."

From the response above, it could be learnt that the current process being followed was extremely inconvenient and time consuming.

6.1.7.2 Configuration Management Using the LOSCMF

Section 6.1.7.2.1 describes the survey results of the need for the system administrators to have a prior knowledge of the programming and scripting languages being used in the LOSCMF for its deployment and maintenance. Section 6.1.7.2.2

describes the flow of events that need to be followed for pushing a change through the LOSCMF approach.

6.1.7.2.1 Requirement of Prior Knowledge of Programming and Scripting Languages

Survey 2 has some relevant questions about Linux operating system configuration management using the LOSCMF approach. Questions 1 through 5 in survey 2 were regarding the requirement of programming and scripting skills to deploy and maintain the LOSCMF, the results of the survey have been tabulated below

Table 6-7: Prior Programming and Scripting Knowledge Requirement

Languages / Utilities	Prior knowledge required	Prior knowledge not required
Perl	1	9
Shell	1	9
MySQL	1	9
PHP	1	9
phpMyadmin	1	9

90% of the survey respondents said ‘No’ to the requirement of having prior knowledge about any of the programming and scripting languages being used. From the above results it can be concluded that prior knowledge of any of the programming languages being used in the LOSCMF is not required for deployment and maintenance.

The field system administrator would only need to ensure that the periodic synchronization of files between the field sites and the master copy is being successfully performed. The LOSCMF approach has a built-in notification system wherein the system administrator could receive notifications about the success or failure of the synchronization process.

6.1.7.2.2 Flow Control of a Change Made Through the LOSCMF

Once the LOSCMF solution has been deployed on the required Linux computer systems within the environment, the system administrator has access to all the relevant

data through the reporting mechanism. In order to push out a required configuration change through the LOSCMF, a system administrator needs to initially decide on the following criterion; what are the effects of the change being pushed out and what group of machines does the change need to be pushed out to.

As the LOSCMF classifies the computer systems in to groups, a high level of granularity can be achieved while trying to decide upon the groups of systems that the change is going to be effective on. It allows for the system administrators to define custom groups as per the requirement. A ‘group’ of machines can be termed as a collection of computer systems that satisfy a common property (e.g. all Linux systems currently running Red Hat Enterprise Linux 3.0 operating system could belong to one group, or all Linux systems running Red Hat Enterprise Linux 3.0 operating system and 64 bit architecture type could belong to one group). Once the groups have been decided, the actual script or command being used to make the change on the computer system would need to be tested on each of the groups that it is being applied on. Once tested, the master set of the group and action configuration files needs to be updated with the changes. The field sites would automatically receive the changes in their respective configuration file set through Rsync [13] mechanism. The LOSCMF has the functionality to notify a pre-defined set of administrators about the changes being pushed through the Rsync [13] mechanism.

A computer system, which when invokes the LOSCMF executable either through cron [30] daemon, or system startup script identifies the changes made in the configuration files. The LOSCMF executable script detects if the change is relevant for the current computer system by comparing the groups for the local system and the groups that the proposed change is to be applied. Upon matching the criterion in the group configuration file, the system automatically invokes the script or command, which performs the change and logs either the success or failure of the action executed. The backend framework of the LOSCMF collects the log information for each of the client systems for further processing.

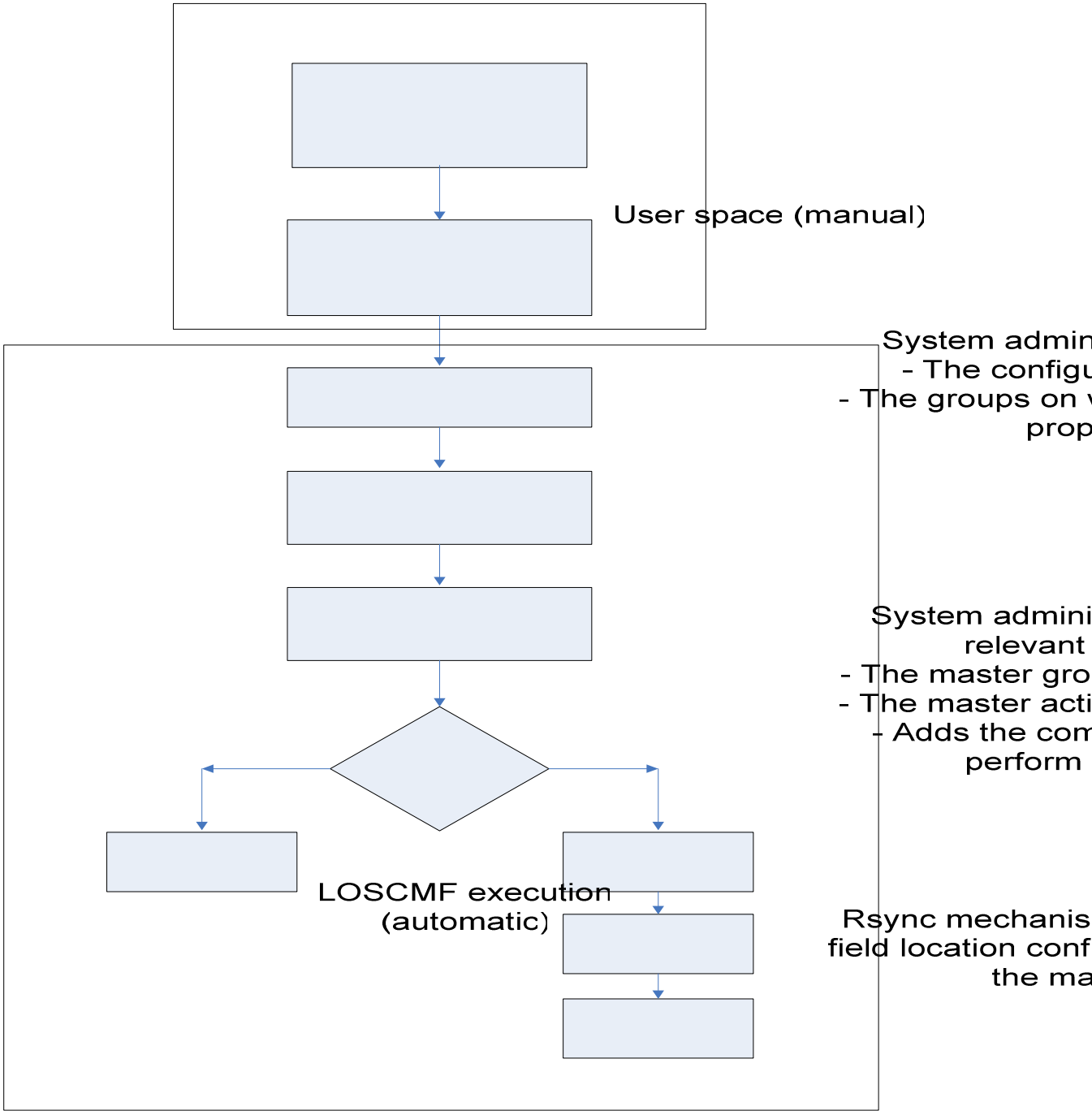

Figure 6-3: Flow Control of a Change Being Made Through the LOSCMF

From above it can be seen that the system administrator only needs to define the required changes and the propagation to field sites is performed automatically without any manual intervention. Thus it can be concluded that the LOSCMF is a more efficient

approach that provides a quick turn around time for propagating configuration management changes with minimal or no manual intervention.

6.1.8 Ownership Cost

If the enterprise wanted to use one of the existing open source utilities as a configuration management solution, it would have to spend a great deal of time evaluating each of the existing utilities, develop workarounds to customize the chosen solution according to the requirements, as well as provide additional training sessions to the members of the IT organization for periodic maintenance. Bugs discovered during the usage of any of the solutions described in section 3.1 would have to be fixed internally within the organization with little or no help.

On the other hand if the enterprise had opted for a commercial configuration management solution such as Red Hat Network (section 3.2.1) or HP open view OS manager (section 3.2.2), it would have spent money on the purchase of the software, yearly license renewal, maintenance and technical support. These costs tend to range between a few thousand to hundreds of thousands of dollars. In addition, the enterprise would have to rely on the service level agreements set forward by the service provider(s).

Using the LOSCMF approach, the enterprise has only the one time cost associated with training its system administrators with the usage and maintenance characteristics. There are no additional purchase and licensing costs associated with the LOSCMF approach. As the LOSCMF only requires minimum manual intervention, maintaining it does not require multiple system administrators. The scalability of the LOSCMF has been thoroughly tested by successfully deploying it across the enterprise on about 6000 linux based computer systems and supporting about 1000 discrete configuration parameters. The result is that the enterprise has to invest very little amount of time and money when compared to customizing, developing and maintaining either a commercial or open source Linux operating system configuration management system.

6.1.9 Support Call Statistics

Figure 6-4 below provides a bar graph of support call volume related to Linux operating system configuration issues across the environment within the enterprise. The figures have been extracted from the tracking system currently being used. The Months

Feb-06 through May-06 (depicted in yellow) represents the time frame when the enterprise was using a mixture of the existing approaches that included Cfengine [25], YAST [26], and other open source utilities. The Months Jun-06 through Aug-06 (depicted in green) represents the time frame after deploying the LOSCMF solution within the enterprise.

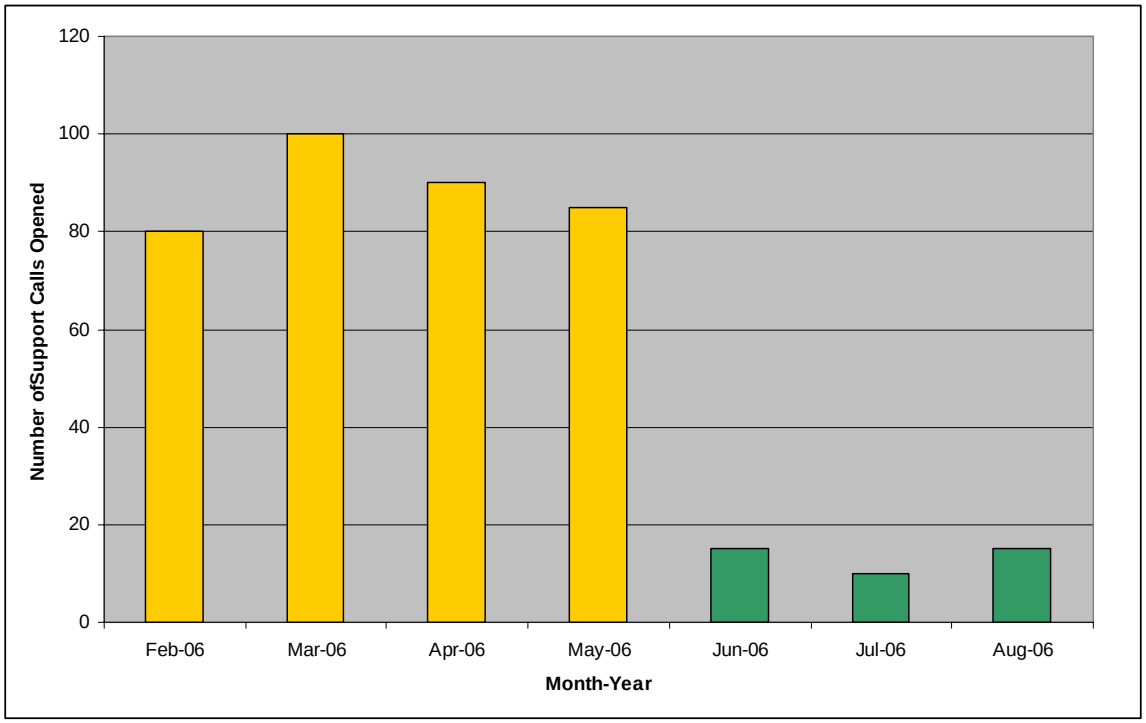

Figure 6-4: Linux Operating System Related Support Call Volume

6.2 Comparison of the LOSCMF with Related Academic Work

This section provides a detailed comparison of the LOSCMF with each of the related work discussed in chapter 3. Section 6.2.1 provides a comparison of the LOSCMF with Local configuration system (LCFG) [48]. Section 6.2.2 provides a comparison of the LOSCMF with Bundle configuration system (BCFG) [15]. Section 6.2.3 provides a comparison of the LOSCMF with Xml based configuration management system (X-CONF) [17]. Section 6.2.4 provides a tabular representation of the comparison between the LOSCMF and the related academic work discussed in chapter 2.

6.2.1 LOSCMF Compared to LCFG

Local configuration system (LCFG) [48] is a large scale UNIX configuration system [2]. It uses a central configuration repository and stores the configuration parameters in an abstract representation [2]. The configuration files are maintained on a centralized storage server under RCS [52] control. Upon change, the parameters are pre processed in to a single table, which is then distributed among the clients using NIS [72] map [2]. Our approach (LOSCMF) uses a centralized location for storing the configuration settings in simple ASCII [73] text files, and these configuration files are under CVS [47] control. The client periodically contacts the storage location (usually a NAS location) through the native NFS [51] protocol and gets the information about a change in the configuration settings. It relies on the pull model instead of the push model being used by LCFG [48].

Current version of LCFG [48] clients run approximately 50 components for configuration management and each of these components upon receiving a change notification from the LCFG server fetches the resources from the server in an XML format [19]. Due to the high number of client components, the associated learning curve is very steep. Our approach the LOSCMF on the other hand does not require individual client components; a central group determination script usually is executed on each of the client to dynamically determine the groups that apply to the client. Basing on the determined groups the corresponding actions are applied to the client. As seen in Table 6-7, 90% of the users have agreed that prior knowledge of any of the programming and scripting languages is not required for deploying the LOSCMF.

Currently LCFG [48] is being used as a production tool on about thousand machines as well as in research testbed [19]. Our approach the LOSCMF is being used within the enterprise on about 6000 Linux computer systems.

6.2.2 LOSCMF Compared to BCFG

Bundle configuration system (BCFG) [15] is a metadata based configuration system, which was primarily developed as a configuration management tool for large heterogeneous clusters. The client configuration within BCFG is represented as multiple configuration fragments [15]. In addition to checking the system installed packages,

BCFG provides the functionality for change detection as well as verification in the configuration management process. Currently it supports verification of */etc* and */var* file systems. BCFG provides the implementation of generators, which are programs to generate configuration fragments on the clients, based on task specific logic [15]. Our approach (LOSCMF) is a metadata based configuration management system, but unlike BCFG, it does not require a complex configuration language. The client configuration is stored in terms of groups, which can further be used to perform actions as defined in the 'action' configuration file.

Currently, BCFG has been deployed on a 320 node test bed cluster within the Argonne National Laboratory [19], and configuration description reusability has been verified by producing a Red Hat 9.0 image from an existing Red Hat 7.3 image [15]. The LOSCMF is currently being used within the enterprise on about 6000 Linux computer systems, and on various distributions of Linux operating system such as Red Hat 7.2, Red Hat Enterprise Linux 3.0, Red Hat Enterprise Linux 4.0, Mandrake 9.0, SUSE 9.0 [32] and SUSE 10 [32].

BCFG currently lacks the automatic change integration process as well as the change reporting interface. The LOSCMF can automatically apply the changes on a client based upon the groups that a client is part off. It (LOSCMF) provides a web based as well as command line based reporting interface that can be used to track changes as well as generate useful information for administrative purposes.

6.2.3 LOSCMF Compared to X-CONF

Xml based configuration management system (X-CONF) [17] is an Extensible markup language (XML) [27] based configuration management system for distributed systems. Its architecture uses XML [27] based managers and XML [27] based configuration agents. All the configuration information is stored in the form of XMLDB [74], which is a database for XML file format. It uses SOAP [28] as the transport mechanism between the manager and the agent components. Our approach (LOSCMF) is a configuration management system for Linux operating system. It (LOSCMF) does not require an additional transport mechanism between the client and the centralized storage location (server). Communication is performed using the native NFS [51] protocol.

Unlike X-CONF, it (LOSCMF) does not require additional daemons to be running on the client.

X-CONFhas been deployed on NG-MON [18], which is a distributed real time internet traffic monitoring and analysis system. The LOSCMF has currently been deployed within the enterprise on about 6000 computer systems running various flavors of Linux operating system.

6.2.4 Tabular Representation of the LOSCMF with Related Work

This section provides a tabular representation (see Table 6-8 below) of the contribution of the LOSCMF in comparison with the related academic work described in chapter 2 (see Table 2-1).

Table 6-8: LOSCMF Comparison with Related Academic Work

Author	Work	Comparison with the LOSCMF
Anderson et al	Paul Anderson, Alastair Scobe, "Large Scale Linux Configuration with LCFG" Division of Informatics, University of Edinburgh [2]	LCFG[48] uses a configuration language and a central repository of configuration settings. Our work (LOSCMF) uses a centralized storage location for the configuration settings, but does not require any special configuration language to be used.
Desai et al	Desai, N.; Lusk, A.; Bradshaw, R.; Evard, R., "BCFG: a configuration management tool for heterogeneous environments," Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER'03) Page(s):500 – 503, 2003	BCFG [15] is a metadata based configuration management system and uses a configuration language that allows for extensive configuration reusability and verification. The LOSCMF on the other hand does not require the use of any special configuration language.
Choi et al	Hyoun-Mi Choi; Mi-Jung Choi; Hong, J.W., "Design and implementation of XML-based configuration management system for distributed systems," IEEE/IFIP Network Operations and Management Symposium, Volume 1, 19-23 April 2004 Page(s):831 - 844 Vol.1, 2004.	X-CONF [17] is an XML [27] based configuration management system for distributed systems. It uses SOAP [28] as the transport mechanism between the manager and the agent components. The LOSCMF does not involve any additional transport mechanism and the client and the centralized storage location (server) can communicate using the native NFS [51] protocol.
Chaudry	Raheel A Chaudry, "User survey of practices using Cfengine", Master's Thesis, University of Oslo, 2005	Raheel A Chaudry [4] has performed a thorough and detailed analysis of Cfengine [25] in the Master's thesis. He developed comprehensive user surveys relating to all aspects of Cfengine [25] usage. Our work is directed towards the evaluation of the current available solutions, which would help in the development of an ideal solution

6.3 LOSCMF Comparison with Other Open Source and Commercial Solutions

In section 3.1 and section 3.2 we have evaluated the available open source and commercial configuration management systems. In this section we provide a comparison of the LOSCMF with each of the solutions described in section 3.1 and section 3.2

Cfengine, [25] although one of the most widely used configuration management tools, is highly complex in nature especially for small scale deployments [4]. Its configuration files are difficult to understand and maintain [4], and it is not a customized approach for Linux operating system. Novice system administrators usually find it difficult to administer. It requires additional system daemons to be running to authenticate the client and server communication [4]. Our work on the other hand is extremely easy to deploy and maintain (see survey response in Table 6-3). The associated configuration files are in plain text format and easy to understand and maintain. The LOSCMF does not require any specialized skill set (see survey response in Table 6-7) and is a customized solution for Linux operating system. It (LOSCMF) does not require additional system daemons to be running for communication between each of the local computer systems (clients) and the centralized configuration repository (server).

COAS [6] documentation does not provide detailed information about the overall design and available documentation. It has limited public mailing lists and any impending issues with its functionality are left to be solved by the administrators themselves without much external help. Our design is modular and self explanatory and provides enough documentation in terms of an administrator guide and a man page that includes various reporting options. The architecture is extremely easy to understand (see survey response in Table 6-3) and provides the administrators the flexibility to use any administrative language to perform the required change

YAST [26] is limited to SUSE Linux [32] distribution It is local to the installed system and cannot be used to remotely administer systems. It is not network aware, and hence the administrator has to log into each system that needs to be rectified. Our approach on the other hand can be used on any variant of Linux operating system and currently has been successfully tested on Red Hat Enterprise Linux [75], SUSE [32], and

Mandrake 9 within the enterprise. Although it is local to the machine, each machine independently contacts the centralized storage location (server) and executes the actions required based upon the groups it belongs to. It does not have the concept of specific user accounts and actions are only performed as the Super User.

Webmin [57] lacks in security in terms of maintaining the local Webmin user accounts. The username and password information are communicated in clear text format across the network. Webmin defaults to running on port 10000 and needs access control restrictions to be set-up through firewall. Our approach on the other hand does need special user accounts to be set-up. An explicit firewall set-up is not required between the client and the server, as long as the native firewall on the client is configured appropriately.

Linuxconf [21] does not support encryption and hence requires external security measures such as IPsec [76] or other forms of IP level security to be able to connect with an installed host in the network. It needs to compile a parser for each of the package using Linuxconf for its configuration management. The LOSCMF does not require encryption mechanism as there is no authentication involved and the client only needs to communicate with the server to read the master configuration file set. This is achieved through the native NFS [51] protocol. The LOSCMF does not need additional compilation of packages and is independent of the programming or scripting language being used to make the required change.

PIKT [11] configuration files are complex in nature. It uses its own scripting language and the end user is expected to have a thorough knowledge about its scripting language for customization. In our approach, the underlying framework is independent of the actual scripting language used to perform the change. The installation procedure is extremely easy to use (see survey response in Table 6-3) and the configuration files are easy to understand and maintain.

Red Hat Network (see section 3.2.1) involves provisioning additional hardware as either the satellite or proxy servers. It involves software purchase costs, as well licensing costs that need to be periodically renewed. Any changes that need to be made must follow the service level agreement set forth by the vendor. The LOSCMF on the other hand does not require additional hardware provisioning other than a central storage

location. It does not involve purchasing cost as well as licensing cost. Any changes that need to be made can be scheduled upon the convenience of the local system administrator

HP OpenView Configuration Management OS manager (see section 3.2.2) involves purchasing as well as licensing costs. These costs increase with increase in the number of computer nodes. The LOSCMF on the other hand does not involve purchasing as well as licensing costs. As it can be easily integrated in to an existing environment (see survey response in Table 6-3), the overhead involved is minimal.

6.4 An Example for the Use of the LOSCMF Approach

This section describes the application of the LOSCMF to a real world example: The recent changes in Daylight Saving Time (DST).

Daylight Saving Time is the convention of advancing clocks such that afternoons have more daylight when compared to the evenings [77]. Starting 2007, Daylight Saving Time (DST) will start on the second Sunday of March (Mar 11th 2:00 AM local time) and end on the first Sunday of November (Nov 4th 2:00 AM local time). Prior to 2007, the corresponding dates were the first Sunday of April and the last Sunday of October. The difference in dates is a result of the Energy Policy Act of 2005. This change may result in inaccurate date if TIMEZONE definitions on the local Linux system do not conform to this change.

This example provides a comparison between the approaches that could possibly be adopted by the enterprise for compliance to the new DST standards.

6.4.1 Scope of the Fixes Required

Each of the operating system vendor provided DST related patches that needed to be applied on the local systems for compliance with the new DST standards. Although most of the operating system version patches only dealt with updating the local time zone data, SUSE [32] Linux required updating the native Glibc [78] packages as well. As a result of the Glibc upgrade process, the systems need to be rebooted for the changes to be effective.

6.4.2 Deployment Approach

This section describes the change deployment approach both through the LOSCMF approach and the old existing approach within the enterprise. Section 6.4.2.1 describes the deployment through the LOSCMF, whereas section 6.4.2.2 describes the deployment through the old existing approach. Comparing the below two approaches, it can be seen that the LOSCMF approach is highly efficient and less time consuming.

6.4.2.1 Deployment through the LOSCMF

- Proposed deployment approach for Linux operating system is through an LOSCMF action
- The LOSCMF action can be put in place well before the actual deadline wherein the changes are effective
- Expected turn around time for all the affected machines to receive the updated packages is about ‘48’ hours
- Machines that required reboot for the changes to be effective, could be rebooted seamlessly through the LOSCMF
- Log information collection for the LOSCMF action can be performed automatically in ‘48’ hours
- Number of man hours required for developing the LOSCMF action is about ‘8’ hours

Number of resources (system administrators) required for verifying the changes being made is ‘3’

6.4.2.2 Deployment through Old Approach

- Each field location would have to develop its own approach to push out the changes as there is no standardization
- Expected turn around time in this case is well above the time limit mentioned in the LOSCMF approach
- Each individual location has to spend time on developing a script or program to automate the changes that need to be propagated

- Since there is no centralized framework available to collect log information from each of the sites, the log information communication and processing time is well above the time limit mentioned in the LOSCMF approach
- Each individual field location would require either one or two administrators for verification of the changes being propagated

7 Conclusion and Future Work

In this report, through the analysis, evaluation and survey results, we have shown that the LOSCMF is a better and more efficient approach for Linux operating system configuration management when compared to both the open source as well as the commercial solutions described in chapter 3, as it does not require the usage of additional system daemons for communication, usage of a complex configuration language or usage of special scripting and programming languages. We have ascertained this by implementing the LOSCMF approach and performing a quantitative analysis within the enterprise, which has about six thousand linux based computer systems.

Section 7.1 provides a summary of how the LOSCMF is a better approach when compared to the current available open source solutions and section 7.2 provides a summary of the LOSCMF comparison with the commercial solutions available.

7.1 LOSCMF Compared to Other Open Source Solutions

The LOSCMF design addresses all the three limitations (usage of additional system daemons for communication, usage of complex configuration language and usage of special scripting and programming language) that are found in one or more of the open source solutions described in chapter 3 (see comparison in section 6.3). We have evaluated the LOSCMF within the enterprise, which earlier had a mixture of one or more of the open source solutions across each of its field location. We have presented a real world example (section 6.4) to compare the efficiency of the LOSCMF approach with that of the existing approach being followed within the enterprise. The LOSCMF provides a user friendly web based as well as command line reporting interface.

Hence we can conclude that the LOSCMF is a better and more efficient approach when compared to any of the existing open source solutions described in chapter 3.

7.2 LOSCMF Compared to Other Commercial Solutions

The LOSCMF is a cost effective solution when compared to any of the commercial configuration management solutions described in chapter 3. It does not involve purchasing costs, licensing costs as well as the need for provisioning of

additional hardware. Additionally it requires very few resources (system administrators) for deployment and maintenance. On the other hand, the LOSCMF solution when compared to the commercial solutions described in chapter 3 (Red Hat Network and HP Open View configuration OS manager) lacks in receiving updated packages directly from the operating system vendor and an enterprise level support contract. The above limitations when compared to the purchasing and licensing costs of the commercial solutions can be ignored.

Hence we can conclude that the LOSCMF is a better and more efficient approach when compared to any of the existing approaches described in chapter 3.

7.3 Future Work

We have evaluated the LOSCMF approach within the enterprise, which is a mid-size organization with a few thousand computer systems. We would like to test how the LOSCMF approach fares when used within a large organization with hundreds of thousands of computer systems.

Additional work that could be performed to further enhance this project is

- Gather more feedback on the reporting interfaces, and provide more options accordingly
- Further increase the scope of compounding groups for executing actions
- Gather more feedback and further enhance the overall flow process to make it more optimal

Bibliography

[1] Paul Anderson, Alastair Scobe, “LCFG – The Next Generation,” UKUUG Winter Conference, 2002

[2] Paul Anderson, Alastair Scobe, “Large Scale Linux Configuration with LCFG,” Proceedings of the Atlanta Linux Showcase, 2000

[3] Paul Anderson, Patrick Goldsack, Jim Patterson, “SmartFrog meets LCFG, Autonomous Reconfiguration with Central Policy Control”, Proceedings of the 2003 Large Installations Systems Administration (LISA) Conference, 2003

[4] Raheel A Chaudry, “User Survey of Practices Using Cfengine”, Master’s Dissertation, University of Oslo, 2005

[5] M. Burgess, “A Brief Overview of the Implementation of Principles of System Administration in Cfengine”, University of Oslo, 2004

[6] Olaf Kirch, “COAS: A Flexible Approach to System Administration Tools”, Linux Journal, Volume 1999, Issue 58es (February 1999)

[7] Webmin Administrator Guide. URL: http://www.swelltech.com/support/webminguide/

[8] LDAP. URL: http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol

[9] Java. URL: http://java.sun.com/

[10] PIKT user survey. URL: http://www.openchannelfoundation.org/survey/pikt_survey.php

[11] PIKT. Official Homepage. http://pikt.org

[12] MySQL 3.23, 4.0, 4.1 Reference Manual. URL: http://dev.mysql.com/doc/refman/4.1/en/index.html

[13] Rsync. URL : http://samba.anu.edu.au/rsync/

[14] CGI on wikipedia. URL: http://en.wikipedia.org/wiki/Common_Gateway_Interface

[15] Desai, N.; Lusk, A.; Bradshaw, R.; Evard, R., "BCFG: A Configuration Management Tool for Heterogeneous Environments," Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER'03) Page(s):500 – 503, 2003

[16] Begnum, K.M.; Burgess, M., "A Scaled, Immunological Approach to Anomaly Countermeasures: Combining pH with Cfengine," IFIP/IEEE Eighth International Symposium on integrated Network Management, Page(s):31 - 42 2003.

[17] Hyoun-Mi Choi; Mi-Jung Choi; Hong, J.W., "Design and implementation of XML-Based Configuration Management System for Distributed Systems," IEEE/IFIP Network Operations and Management Symposium, Volume 1, 19-23 April 2004 Page(s):831 - 844 Vol.1, 2004.

[18] Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju and James W. Hong, "The Architecture of NG-MON: A Passive Network Monitoring System", 13th IFIP/IEEE lnternational Workshop on Distributed Systems: Operations and Management (DSOM 2002). Montreal, Oct. 2002, pp. 16-27.

[19] LCFG workshop at Edinburgh University. URL: http://www.lcfg.org/workshop/lcfg06.pdf

[20] COAS. Official Homepage. URL: http://linux.davecentral.com/-3724 sysutiladmin.html.

[21] Linuxconf. Official Homepage. URL: http://www.solucorp.qc.ca/-linuxconf/.

[22] IDC Official Homepage. URL: http://www.idc.com/

[23] Shell. URL: http://en.wikipedia.org/wiki/Shell_(computing)

[24] Perl. URL: http://perl.org

[25] Cfengine Official Homepage. URL: http://www.cfengine.org

[26] YaST Official Homepage. URL: http://en.opensuse.org/YaST

[27] XML. URL: http://www.w3.org/XML

[28] SOAP. URL: http://www.w3.org/TR/soap

[29] UNIX. URL: http://www.unix.org

[30] cron. URL: http://en.wikipedia.org/wiki/Cron

[31] MySQL. URL: http://mysql.com

[32] SUSE. Official Homepage. URL: http://www.novell.com/linux/

[33] Solaris. URL: http://www.sun.com/software/**solaris**/

[34] Windows. URL: http:// www.microsoft.com/**windows**/default.mspx

[35] SNMP. URL: http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

[36] MIB. URL: http://en.wikipedia.org/wiki/Management_information_base

[37] Python. Official Homepage. URL: http://python.org

[38] Novell. Official Homepage. URL: http://www.novell.com/

[39] TCP. URL: http://en.wikipedia.org/wiki/Transmission_Control_Protocol

[40] OpenSSL. Official Homepage. URL: http://www.openssl.org

[41] Mozilla. Official Homepage. URL: http://www.mozilla.org

[42] Netscape. Official Homepage. URL: http://www.netscape.com

[43] Lynx. URL: http://lynx.browser.org/

[44] Kickstart. URL: http://www.redhat.com/docs/manuals/linux/RHL-7.3-Manual/custom-guide/ch-kickstart2.html

[45] PHP. URL: http://www.php.net

[46] CGI. URL: http://www.w3.org/CGI/

[47] CVS. URL: http://en.wikipedia.org/wiki/Concurrent_Versions_System

[48] LCFG. Official HomePage. URL: http://www.lcfg.org

[49] Sed. URL: http://www.gnu.org/software/sed/

[50] Clearcase. Official Homepage. URL: http://www-306.ibm.com/software/awdtools/clearcase/

[51] NFS. URL: http://nfs.sourceforge.net/

[52] RCS. URL: http://www.gnu.org/software/rcs/rcs.html

[53] GAWK. URL: http://www.gnu.org/software/gawk/gawk.html

[54] LISP. URL: http://en.wikipedia.org/wiki/Lisp_programming_language

[55] GNU. URL: http://www.gnu.org/

[56] HP OpenView Configuration Management OS manager features and overview, URL : http://h20229.www2.hp.com/products/radia_osm/index.html

[57] Webmin. URL : http://www.webmin.com/

[58] Wall, L., Christiansen, T, Schwartz, L. R., Programming Perl, 2d ed. California: O'Reilly & Associates, 1996, pp. 378-427

[59] Hall, J. N., Schwartz, L. R., Effective Perl Programming, Massachusetts: Addison Wesley Longman Inc, 1998, pp. 197-217

[60] IEEE Glossary of Software Engineering Terminology: URL: http://standards.ieee.org/reading/ieee/std_public/description/

[61] Red Hat Network: URL: https://www.redhat.com/wapps/sso/rhn/login.html?redirect=https%3A%2F%2Frhn.redhat.com%2Frhn%2FYourRhn.do

[62] HP Openview configuration management OS manager: URL: http://h20229.www2.hp.com/products/radia_osm/index.html

[63] OpenLinux. URL: http://en.wikipedia.org/wiki/Caldera_OpenLinux

[64] GNU General Public License. URL: http://www.gnu.org/copyleft/gpl.html

[65] ncurses. URL: http://www.gnu.org/software/ncurses/

[66] Artistic License. URL: http://en.wikipedia.org/wiki/Artistic_License

[67] Eiffel. URL: http://dev.mysql.com/doc/refman/5.0/en/eiffel.html

[68] Ruby. URL: http://www.ruby-lang.org/en/

[69] Tcl. URL: http://www.tcl.tk/

[70] phpMyadmin. URL: http://www.phpmyadmin.net/home_page/index.php

[71] LSF. URL: http://www-cdf.fnal.gov/offline/runii/fcdfsgi2/lsf_batch.html

[72] NIS. URL: http://en.wikipedia.org/wiki/Network_Information_Service

[73] ASCII. URL: http://en.wikipedia.org/wiki/ASCII

[74] XMLDB. URL: http://www.oracle.com/technology/tech/xml/xmldb/index.html

[75] Red Hat Enterprise Linux. URL: http://www.redhat.com/rhel/

[76] IPsec. URL: http://en.wikipedia.org/wiki/IPsec

[77] DST. URL: http://en.wikipedia.org/wiki/Daylight_saving_time

[78] Glibc. URL: http://www.gnu.org/software/libc/

[79] GTK+. URL: http://www.gtk.org/

[80] BSD license. URL: http://en.wikipedia.org/wiki/BSD_license

[81] GNU Automake. URL: http://www.gnu.org/software/automake/

[82] GNU Autoconf. URL: http://www.gnu.org/software/autoconf/

[83] GNU Libtool. URL: http://www.gnu.org/software/libtool/

[84] Cost of Red Hat Network. URL: https://www.redhat.com/apps/store/systems/

Appendix A: LOSCMF Surveys

A.1 Survey 1

Survey 1 for the Linux operating system configuration management framework (LOSCMF) project:

This survey is for informational purpose (for Master's thesis). Participation is voluntary. There is no risk of participation. Data from this survey will help evaluate the LOSCMF project. Completion and return of the survey implies the consent to use this data for educational research purposes. Estimated time for the completion of this survey is 15 minutes. Please circle the alternative of your choice of answer. Please provide feedback for other applicable questions.

1) Have you used a configuration management tool/utility?

   Yes No

2) Have you designed or developed a configuration management solution before?

   Yes No

3) If yes, how would you rate your experience?

   VeryEasy Easy OK Difficult VeryDifficult

4) Have you maintained a configuration management solution before?

   Yes No

5) If yes, how would you rate your experience?

   VeryEasy Easy OK Difficult VeryDifficult

6) How would you rate your knowledge about Linux configuration management in general?

   VeryPoor Poor Fair Good Excellent

7) On a scale of one to five, please rate your knowledge about PIKT?

   1 2 3 4 5

8) On a scale of one to five, please rate your knowledge about Cfengine?

   1 2 3 4 5

9) On a scale of one to five, please rate your knowledge about YAST?

   1 2 3 4 5

10) On a scale of one to five, please rate your knowledge about COAS?

    1 2 3 4 5

11) On a scale of one to five, please rate your knowledge about Webmin?

1 2 3 4 5

12) On a scale of one to five, please rate your knowledge about Linuxconf?

1 2 3 4 5

13) On a scale of one to five, please rate your knowledge about Perl?

1 2 3 4 5

14) On a scale of one to five, please rate your knowledge about PHP?

1 2 3 4 5

15) On a scale of one to five, please rate your knowledge about Shell scripting?

1 2 3 4 5

16) On a scale of one to five, please rate your knowledge about CGI?

1 2 3 4 5

17) On a scale of one to five, please rate your knowledge about MySQL?

1 2 3 4 5

18) On a scale of one to five, please rate your knowledge about phpMyadmin?

1 2 3 4 5

19) On a scale of one to five, please rate your knowledge about Rsync?

1 2 3 4 5

20) On a scale of one to five, please rate your knowledge about CVS?

1 2 3 4 5

21) Please describe the time taken to push configuration changes through the old existing process? (If applicable)

22) Please describe the convenience of using the old existing approach for linux operating system configuration management? (If applicable)

A.2 Survey 2

Survey 2 for the Linux operating system configuration management framework (LOSCMF) project:

This survey is for informational purpose (for Master's thesis). Participation is voluntary. There is no risk of participation. Data from this survey will help evaluate the LOSCMF project. Completion and return of the survey implies the consent to use this data for educational research purposes. Estimated time for the completion of this survey is 15 minutes. Please circle the alternative of your choice of answer. Please provide feedback for other applicable questions.

1) Is prior knowledge of Perl programming required to deploy and maintain the LOSCMF approach

   Yes No

2) Is prior knowledge of PHP required to deploy and maintain the LOSCMF approach

   Yes No

3) Is prior knowledge of MySQL required to deploy and maintain the LOSCMF approach

   Yes No

4) Is prior knowledge of phpMyadmin required to deploy and maintain the LOSCMF approach

   Yes No

5) Is prior knowledge of shell scripting required to deploy and maintain the LOSCMF approach

   Yes No

6) Is prior knowledge of Rsync required to deploy and maintain the LOSCMF approach

   Yes No

7) Is prior knowledge of CVS required to deploy and maintain the LOSCMF approach

   Yes No

8) Is prior knowledge of HTML required to deploy and maintain the LOSCMF approach

   Yes No

9) Please rate the difficulty level of using the web based reporting interface, with 1 being the least difficult and 5 being most difficult

   1 2 3 4 5

10) Please rate the difficulty level of using the command line based reporting interface, with 1 being the least difficult and 5 being most difficult

   1 2 3 4 5

11) Please describe your experience of integrating the LOSCMF within the enterprise (If applicable)

12) Please describe your experience of customizing the LOSCMF within the enterprise (If applicable)

13) Please describe your overall experience of deploying and using the LOSCMF within the enterprise (If applicable)

14) Please describe the time taken to push a change through the LOSCMF approach (If applicable)

15) Was it easy to learn and use the LOSCMF reporting interfaces?

Appendix B: Description of the LOSCMF Associated Files

This section provides a brief description of the files associated with each of the computer systems installed with the LOSCMF. For each of the files, the following information such as its location, functionality and dependencies are described, and any associated input and output files are mentioned.

B.1 LOSCMF Process Flow Script

Location:

- Client: The local copy of the LOSCMF process flow script is stored locally on the client
- NAS Server: The master copy of the LOSCMF process flows script is stored on the NAS server location

Functionality:

- Invocation: The LOSCMF process flow script can either be invoked through system initialization script or by cron [30]. It can be invoked manually upon requirement by the system administrator
- Sequence: The LOSCMF process flow invokes all the other related modules.
- Self update: The LOSCMF process flow script updates itself by comparing with the master copy on the NAS server location.

Dependencies:

- Group configuration check script: The LOSCMF process flow script is dependent on the group configuration check script to determine the groups for a particular client
- Action configuration check script: The LOSCMF process flows script is dependent on the action configuration check script to determine the actions that need to be performed for the client

Input:

- Options: The LOSCMF process flow can be used to execute in either verbose, check only (actual change is not performed) or normal mode.

Output:

- hostname.group: The LOSCMF process flow stores information about the current groups that a client belongs to in this file.
- hostname.action: The LOSCMF process flow stores information about the actions that have been executed in this file.
- hostname.alert: The LOSCMF process flow stores information about the errors encountered during the entire process flow in this file.

B.2 Group Configuration File

Location:

- NAS server: The master copy of the group configuration file is stored on the NAS server location

Functionality:

- Group determination: The group configuration file defines the groups, and commands or scripts needed to determine if a particular host belongs to a particular group.

Dependencies:

- Group configuration check script: The group configuration file is parsed by the group configuration check script to determine the groups that a client belongs to

Input:

- none

Output:

- none

B.3 Group Configuration Check Script

Location:

- NAS server: The master copy of the group configuration check script is stored on the NAS server location.

Functionality:

- Group determination: The group configuration check script parses group configuration file and determines the groups that a particular client belongs to.
- Script execution: The group configuration check script executes the script or command defined in the group configuration file to determine if the host belongs to a particular group

Dependencies:

- Group configuration file: The group configuration check script is dependent on the group configuration file

Input:

- Group configuration file: The group configuration check script parses the group configuration file

Output:

- Group information: The group configuration check script logs the group information for a particular host

B.4 Group Information File

Location:

- Local: The group information file for each of installed client is initially stored locally and later replicated on the NAS server during the end of the process flow

Functionality:

- Group information: The group information file is used to determine the appropriate actions for a particular client

Dependencies:

- Group configuration file: The group information file is created by parsing the group configuration file using the group configuration check script

Input:

- Group configuration file: The group information is determined using the group configuration check script

Output:

- Group definition: The group information for each of the clients is defined in the group information file

B.5 Action Configuration File

Location:

- NAS server: The master copy of the action configuration file is stored on the NAS server location

Functionality:

- Action determination: The action configuration file defines the actions, and commands or scripts that are needed to be executed

Dependencies:

- Action configuration check script: The action configuration file is parsed by the action configuration check script to determine the actions that need to be executed by the client

Input:

- none

Output:

- none

B.6 Action Configuration Check Script

Location:

- NAS server: The master copy of the action configuration check script is stored on the NAS server location.

Functionality:

- Action determination: The action configuration check script parses action configuration file and determines the groups that a particular client belongs to.
- Script execution: The action configuration check script executes the script or command defined in the action configuration file

Dependencies:

- Action configuration file: The action configuration check script is dependent on the action configuration file

Input:

- Action configuration file: The action configuration check script parses the action configuration file

Output:

- Action information: The action configuration check script logs the actions executed by a particular host

B.7 Alert Log File

Location:

- Alert log: The alert log information is stored local on each of the installed client

Functionality:

- Log error information: The alert log file contains the log information about the errors encountered during action execution

Dependencies:

- Action configuration: The alert log information is created by storing the errors from each of the action execution for a client

Input:

- Action configuration check script: The alert log information is created by the action configuration check script

Output:

- Alert log: The error information for each client is logged in this file

Appendix C: Description of the LOSCMF Configuration Files

The main configuration files associated with the LOSCMF are the group configuration file and action configuration file. These configuration files are stored on the centralized storage location (NAS server), which can be accessed by every installed client.

C.1 Group Configuration File

Group configuration file defines a particular group and contains the commands and scripts required to determine if a particular client belongs to a group. Each of the groups is defined separated by a new line character in the following format.

<group_name>::<script|command>::<operator>::<value to be evaluated>

A group definition is a collection of four different variables each separated by "::" symbol. Description of each of the fields is as follows

- Group name: It is the actual name of the group as defined by the administrator
- Script or command: It is the utility, which when executed on a particular system produces an output that is later used for matching with the expected output
- Operator: It is the mechanism to compare the output of the script or command with the expected output. Currently the supported operators are as follows
    - **== (double equal to symbol)**
    - **!= not equal**
    - **=~ string like (for string comparison operations)**
    - **>= greater than equals (for date comparison operations only)**
    - **<= less than equals (for date comparison operations only)**
    - **< > range operator (for date comparison operations only)**

- Value: It is the value that needs to be compared with the output string from the script or command using the specified operator

Sample entries in a group configuration file are as follows

MONDAY::date +%A::==::MONDAY

SGE::./scriptgroup/detect_farm_info | awk '{print $2}' | cut -c1-3::==::sge

Explanation of line 1

- The defined group name is “MONDAY”
- The command to be run on the client is “date +%A”
- The comparison operator is “==”
- The values to be evaluated is “MONDAY”

If the output from the client machine after running the command “date +%A” matches “MONDAY”, then the client is said to belong to group “MONDAY”.

Explanation of line 2

- The defined group name is SGE
- The script to be run on the client is “./scriptgroup/detect_farm_info | awk ‘{print $2}’ | cut –c1-3”
- The comparison operator is ==
- The value to be evaluated is “sge”

If the output from the client machine after executing the above script matches “sge”, then the client is said to belong to group “SGE”.

The group configuration file was designed in the above format so as to allow a high degree of flexibility to the system administrators for defining custom groups as per their requirement. Unlike many other solutions, as the LOSCMF does not involve the use of a complex configuration language, the associated learning curve is very much flat.

C.2 Action Configuration File

Action configuration file defines the appropriate actions that need to be performed on a client depending on the groups that it belongs to. Each of the action is defined in a separate line in the following format.

Action<XX>::s|r::single|reboot::<action_script> <arguments>::<group_names>::<description>::<optional log file>

An action definition is a collection of seven variables each separated by "::" symbol. Description of each of the fields is as follows.

- Action Name: The first field, action name is the name of the action as defined by the administrator. Usually it is of the convention ActionXX (e.g. Action1, Action2, Action3)
- single (s) or recurring (r): The second field specifies if the action needs to be performed only once or on a recurring basis. If the value specified is "o", the action is performed only once, if the value specified is "r", the action is performed on a recurring basis. This is useful for executing periodic checks on the machine such as IO, network, memory and CPU utilization.
- single or reboot: The third field specifies if the action requires a reboot or not. If the value specified is "single", the action does not require a reboot operation. If the value specified is "reboot", the action requires a reboot operation for the changes to be effective.
- Action script or command: The fourth field, action script, is the script or command that is executed locally on the machine as a result of this action. The LOSCMF provides the feature of providing optional arguments to the action script (verbose mode, check only mode)
- Group names: The fifth field, group names, is either a particular group or a logical combination of multiple groups in the defined format. The action in consideration is only executed on a client that satisfies the requirements of this field. Currently the following combination of groups is supported. In the operators shown below, "&" indicates logical and operation between groups, "!" indicates negation of a group and "|" indicates logical or operation between the groups.
    - group1&group2 (and)
    - !group1 (negation)
    - group1|group2 (or)

  - group1&group2!group3 (exception groups)
  - !group1!group2!group3 (exception groups)
  - group1|group2&group3
  - group1&group2|group2&group3|group3&group4!group5
- description: The sixth field, description is used to provide a concise description of the change being propagated through the action script
- Optional log file: The seventh field, optional log file can be used to log the output from the action script. Some action scripts require that certain information be stored locally on the client, and this option is useful in such cases (e.g.,Action1::s::single::touch tmp/lock::group1&group2!group3::Sample Action Script::Log_file_Action1).

Appendix D: Sample LOSCMF Implementation

This section presents the configuration files and the change detection and propagation scripts associated with the real world implementation of the LOSCMF as described in section 6.4. The recent changes in the Daylight Saving Time due to the Energy Policy Act of 2005 may result in inaccurate date if TIMEZONE definitions on the local Linux system do not conform to this change.

The DST patch management required updating the TIMEZONE definitions on various flavors of the Linux Operating System such as Update 1, Update2, Update3, Update 4, and Update5 within Red Hat Enterprise Linux 3.0 and Service Pack 1, Service Pack 2 within SUSE Linux 9.0 distributions. Updating the TIMEZONE definitions also included updating the native Glibc packages on SUSE platform. The LOSCMF configuration files are stored on the centralized storage location (NAS Server), which can be accessed by every installed client.

D.1 LOSCMF Group Configuration

LOSCMF execution dynamically groups the machines basing on their current configuration level. Following is a section of the group configuration file which determines the Operating System version installed on each of the computer systems.

“RHEL3.0::./group_config/osversion.sh::==::RHEL3.0

RHEL3.0_U2::./group_config/osversion.sh::==::RHEL3.0_U2

RHEL3.0_U3::./group_config/osversion.sh::==::RHEL3.0_U3

RHEL3.0_U4::./group_config/osversion.sh::==::RHEL3.0_U4

RHEL3.0_U5::./group_config/osversion.sh::==::RHEL3.0_U5

RHEL3.0_U6::./group_config/osversion.sh::==::RHEL3.0_U6

RHEL3.0_U7::./group_config/osversion.sh::==::RHEL3.0_U7

RHEL4.0::./group_config/osversion.sh::==::RHEL4.0

RHEL4.0_U1::./group_config/osversion.sh::==::RHEL4.0_U1

RHEL4.0_U2::./group_config/osversion.sh::==::RHEL4.0_U2

RHEL4.0_U3::./group_config/osversion.sh::==::RHEL4.0_U3

RHEL4.0_U4::./group_config/osversion.sh::==::RHEL4.0_U4

RHEL5.0::./group_config/osversion.sh::==::RHEL5.0

RHEL_WS2.1::./group_config/osversion.sh::==::WS2.1

RHEL_AS2.1::./group_config/osversion.sh::==::AS2.1

RH7.0::./group_config/osversion.sh::==::RH7.0

RH7.1::./group_config/osversion.sh::==::RH7.1

RH7.2::./group_config/osversion.sh::==::RH7.2

RH7.3::./group_config/osversion.sh::==::RH7.3

RH8.0::./group_config/osversion.sh::==::RH8.0

RH9.0::./group_config/osversion.sh::==::RH9.0

SLES9::./group_config/osversion.sh::==::SLES9

SLES9_SP1::./group_config/osversion.sh::==::SLES9_SP1

SLES9_SP2::./group_config/osversion.sh::==::SLES9_SP2”

Following is the group determination script ‘osversion.sh’ that is referenced in the group configuration file above. It is a Bourne Shell script and runs on all installed clients. Its output is compared to the expected output as specified in the group configuration file. If incase the script output matches the expected output, the client belongs to that particular group.

```
#!/bin/sh
# Name: osversion.sh
# Description: Describes OS vendor
# Date: 12/06/2006
# Author: Srinivas Kalidindi
# Dependencies: /etc/redhat-release for Redhat & uname -r for SUSE
# Arguments: none
#Values: RHEL5.0, RHEL4.0, RHEL4.0_U1, RHEL4.0_U2, RHEL4.0_U3,
RHEL4.0_U4, RH3.0, RHEL3.0_U7, RHEL3.0_U6, RH3.0_U5, RHEL3.0_U4,
RHEL3.0_U3, RHEL3.0_U2, WS2.1, AS2.1, RH6.0, RH6.2, RH7.0, RH7.1, RH7.2,
RH7.3, RH8.0, RH9.0, SLES9, SLES9_SP1, SLES9_SP2, SLES9_SP3,  SLES10
test -f /etc/redhat-release && OS_VER=`cat /etc/redhat-release | cut -d"(" -f2 | sed 's/)//' |
awk '{print $1}'`
case "x$OS_VER" in
        xTikanga )
                echo "RHEL5.0" && exit 0 ;;
        xNahant )
                grep 'Update 4' /etc/redhat-release > /dev/null && echo "RHEL4.0_U4"
                && exit 0 ;
                grep 'Update 3' /etc/redhat-release > /dev/null && echo "RHEL4.0_U3"
                && exit 0 ;
                grep 'Update 2' /etc/redhat-release > /dev/null && echo "RHEL4.0_U2"
                && exit 0 ;
                grep 'Update 1' /etc/redhat-release > /dev/null && echo "RHEL4.0_U1"
                && exit 0 ;
                echo RHEL4.0 && exit 0;;
        xTaroon )
                grep 'Update 7' /etc/redhat-release > /dev/null && echo "RHEL3.0_U7"
                && exit 0 ;
                grep 'Update 6' /etc/redhat-release > /dev/null && echo "RHEL3.0_U6"
                && exit 0 ;
                grep 'Update 5' /etc/redhat-release > /dev/null && echo "RHEL3.0_U5"
                && exit 0 ;
                grep 'Update 4' /etc/redhat-release > /dev/null && echo "RHEL3.0_U4"
                && exit 0 ;
                grep 'Update 3' /etc/redhat-release > /dev/null && echo "RHEL3.0_U3"
                && exit 0 ;
                grep 'Update 2' /etc/redhat-release > /dev/null && echo "RHEL3.0_U2"
                && exit 0 ;
                echo "RHEL3.0"  && exit 0 ;;
         xTampa | xDerry ) echo "WS2.1" && exit 0 ;;
```

```
42              xPensacola ) echo "AS2.1" && exit 0 ;;
43              xHedwig ) echo "RH6.0" && exit 0 ;;
44              xZoot | xPiglet ) echo "RH6.2" && exit 0 ;;
45              xGuinness ) echo "RH7.0" && exit 0 ;;
46              xSeawolf ) echo "RH7.1"  && exit 0 ;;
47              xEnigma ) echo "RH7.2"  && exit 0 ;;
48              xValhalla ) echo "RH7.3" && exit 0 ;;
49              xPsyche ) echo "RH8.0" && exit 0 ;;
50              xShrike ) echo "RH9.0" && exit 0 ;;
51              * )
52                      case `uname -r` in
53                          2.6.5-7.97* ) echo "SLES9" && exit 0 ;;
54                          2.6.5-7.139* ) echo "SLES9_SP1" && exit 0 ;;
55                          2.6.5-7.191* | 2.6.5-7.193* ) echo "SLES9_SP2" && exit 0 ;;
56                          2.6.5-7.244* )
57                                  grep "PATCHLEVEL[           ]*=[     ]*3" /etc/SuSE-
58                                  release >/dev/null 2>/dev/null
59                                  if [ $? -eq 0 ]; then
60                                          echo "SLES9_SP3" && exit 0
61                                  else
62                                          echo "SLES9_SP2" && exit 0
63                                  fi ;;
64                          2.6.16.21* )
65                                  grep "SUSE Linux Enterprise Server 10" /etc/SuSE-release
66                                  >/dev/null 2> /dev/null
67                                  if [ $? -eq 0 ]; then
68                                          echo "SLES10" && exit 0
69                                  fi ;;
70                      esac
71  esac
```

D.2 LOSCMF Action Configuration

Following are the LOSCMF actions for applying the DST related patches within the enterprise. ‘Action32’ describes the action for applying the patches on all clients running Red Hat Enterprise Linux Operating System, and ‘Action33’ describes the action for applying the patches on all clients running SUSE Linux Operating System. Since SUSE platform requires updating the native Glibc packages, it has been defined as a separate action which requires the client to reboot (specified by the third field in the defined action). The reboot process is handled internally by the LOSCMF execution. Each of the actions is executed if the client belongs to any one of the mentioned groups (specified in the fifth field).

Action32::s::single::./action_config/dst_fix.sh::RHEL3.0|RHEL3.0_U1|RHEL3.0_U2|RHEL3.0_U3|RHEL3.0_U4|RHEL3.0_U5::"Applying DST patches on affected Red Hat Enterprise Linux versions "::-

Action33::s::reboot::./action_config/dst_fix.sh::SLES9_SP1|SLES9_SP2::"Applying DST patches on affected SUSE Linux versions"::-

Following is the action execution script ‘dst_fix.sh’ that is referenced in the above defined actions. It is a Bourne Shell script that is executed on the clients which belong to either one of the defined groups. The script exits with a value of zero if the execution is successful and one incase of any errors. The return value of the script is recorded by LOSCMF execution to determine if the action needs to be executed once again on the client.

```
#!/bin/sh
#Name : dst_fix.sh
#Description : Script to patch for DST
#Date : 02/14/07
#Author : Srinivas Kalidindi
#Dependencies : RPMs to be present
#Arguments : c,v,n
#Values : returns zero on succesful completion
#Limitations : Service restart might fail.

#Declaring variables
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin
OUTPUT=/dev/null
OPTION=normal
LOGGING_ENABLED=no
EXIT_VAL=0
compatible_os=0
installed=0
HOSTNAME=`/bin/hostname`
already_patched() {
        echo "The OS version running on $HOSTNAME is already patched for DST" >
        $OUTPUT && exit 0
}
hupdaemons() {
        for daemon in xinetd cron crond sendmail
        do
                test ! -f /etc/init.d/$daemon && continue
                /etc/init.d/$daemon  stop > /dev/null
                if [ $? -eq 0 ]; then
                        echo "Stopped $daemon" > $OUTPUT
                else
                echo "Error: could not stop $daemon for Host :: $HOSTNAME" >
                $OUTPUT && EXIT_VAL=1
                fi
                sleep 1
                /etc/init.d/$daemon start > /dev/null
                if [ $? -eq 0 ]; then
                        echo "Started $daemon" > $OUTPUT
                else
```

```
40                      echo "Error: could not start $daemon for Host :: $HOSTNAME" >
41                      $OUTPUT && EXIT_VAL=1
42                      fi
43              done
44      }
45      if [ $# -gt 3 ]; then
46         exit 1
47      fi
48      #Defining the output and other options depending on options passed
49      while getopts crnvf: args
50      do      case $args in
51         c ) OUTPUT=/dev/tty
52            OPTION=checkonly
53            ;;
54         r ) OUTPUT=/dev/tty
55            OPTION=revert
56            ;;
57         n ) OUTPUT=/dev/null
58            OPTION=normal
59            ;;
60         v ) OUTPUT=/dev/tty
61            OPTION=verbose
62            ;;
63         f ) LOGFILE="$OPTARG"
64            if [ "$OUTPUT" != "/dev/null" ]; then
65               OUTPUT=/dev/null
66            fi
67            LOGGING_ENABLED=yes
68            ;;
69         * ) echo "Warning: Invalid args specified, exiting the program "
70            exit 1
71            ;;
72            esac
73      done
74      MY_DIR=`dirname $0`
75      cd ${MY_DIR}
76      cd ../..
77      RPMSRC="`pwd`/global_config/RPMs/DST"
```

```
78  test ! -d ${RPMSRC} && echo "Error. cannot locate RPM repository for DST patches
79  on Host :: $HOSTNAME" > $OUTPUT && exit 1
80  if [ $OPTION == checkonly ]; then
81          echo > $OUTPUT
82          echo "This script is an LOSCMF Action to patch machines for DST compliance"
83          > $OUTPUT
84          echo > $OUTPUT
85  fi
86  if [ $OPTION == normal ] || [ $OPTION == verbose ]; then
87  test -f /etc/redhat-release && OS_VER=`cat /etc/redhat-release | cut -d"(" -f2 | sed 's/)//' |
88  awk '{print $1}'`
89          case "x$OS_VER" in
90                  xTikanga ) already_patched ;;
91                  xNahant )
92                          grep 'Update [2|3|4]' /etc/redhat-release > /dev/null &&
93                          already_patched
94                          OS=RHEL4 ;;
95                  xTaroon )
96                          grep 'Update [6|7|8]' /etc/redhat-release > /dev/null &&
97                          already_patched
98                          OS=RHEL3 ;;
99                  xTampa | xDerry | xPensacola ) OS=RH2.1 ;;
100                 xHedwig | xZoot | xPiglet | xGuinness | xSeawolf | xEnigma | xValhalla |
101                 xPsyche | xShrike ) OS=RH7 ;;
102                 * )
103                         case `uname -r` in
104                                 2.6.5-7* ) OS=SLES9
105                                 grep "PATCHLEVEL = 3" /etc/SuSE-release > /dev/null
106                                 && already_patched ;;
107                                 * ) echo "Warning! cannot determine OS version ::
108                                 $HOSTNAME" > $OUTPUT && exit 1 ;;
109                         esac
110                 ;;
111         esac
112         case `arch -k` in
113                 i686 ) ARCH=32bit ;;
114                 x86_64 ) ARCH=64bit ;;
115         esac
116         case $OS in
117                 SLES9 ) cd ${RPMSRC}/SLES9/${ARCH}
```

```
                        rpm -Fvh *.rpm > /dev/null
                        test $? -ne 0 && EXIT_VAL=1 ;;
                RH7 ) test ! -x /usr/sbin/zic && echo "Error: time zone compiler missing
                on Host :: $HOSTNAME" > $OUTPUT && exit 1
                        mkdir -p /usr/share/zoneinfo/src
                        cd /usr/share/zoneinfo/src
                        tar -zxf ${RPMSRC}/RH/7.2/tzcode2007a.tar.gz > /dev/null
                        tar -zxf ${RPMSRC}/RH/7.2/tzdata2007a.tar.gz > /dev/null
                        /usr/sbin/zic africa asia australasia europe northamerica
                        southamerica
                        /usr/sbin/zic backward
                        test $? -ne 0 && EXIT_VAL=1 ;;
                RH2.1 ) rpm -Uvh ${RPMSRC}/RH/2.1/*.rpm > /dev/null
                        test $? -ne 0 && EXIT_VAL=1 ;;
                RHEL3 ) rpm -Uvh ${RPMSRC}/RH/3.0/tzdata-2006m-3.el3.noarch.rpm
                > /dev/null
                        test $? -ne 0 && EXIT_VAL=1 ;;
                RHEL4 ) rpm -Uvh ${RPMSRC}/RH/4.0/tzdata-2006m-3.el4.noarch.rpm
                > /dev/null
                        test $? -ne 0 && EXIT_VAL=1 ;;
        esac
        case $OS in
                RH* )
                test ! -f /etc/sysconfig/clock && echo "Error: cannot determine timezone
                on Host :: $HOSTNAME" > $OUTPUT && exit 1
                TZONE=`grep ZONE /etc/sysconfig/clock | awk -F= '{print $2}' | sed
                's/"//g'`
                cp -f /usr/share/zoneinfo/${TZONE} /etc/localtime
                hupdaemons ;;
        esac
    exit $EXIT_VAL
fi
if [ $OPTION == revert ]; then
        echo "No revert option at this time!"
        exit $EXIT_VAL
fi
exit $EXIT_V
```

[i] In addition to attending the periodic planning meetings of the LOSCMF, the following are the specific details about my contribution in the development of the LOSCMF.

- Analyzed Cfengine, PIKT, YaST, BCFG & LCFG solutions
- Contributed significantly in the overall LOSCMF process flow design
- Contributed significantly in the design of group and action configuration file formats
- Created the format to be followed for developing group determination and action execution scripts
- Enhanced the existing group determination script to include additional options such as classifying hosts in to implicit groups and error groups if they fail to run the script or the command (hosts that are part of the error or implicit groups do not execute any actions to avoid any unexpected behavior)
- Enhanced the action determination script and provided additional functionality such as group compounding, and exception lists. Sample compound group options are as follows
    - Using ‘And’ operator: group1&group2 (actions specified with the ‘&’ symbol will only be executed if the host belongs to both group1 and group2)
    - Using ‘Not’ operator: !group1 (actions specified with the ‘!’ symbol will only be executed if the host does not belong to group1)
    - Using ‘Or’ operator: group1|group2 (actions specified with the ‘or’ symbol will be executed if the host belongs to either group1 or group2)
    - Combining ‘And’ and ‘Not’ operator: group1&group2!group3 (actions specified in this format will only be executed on hosts that belong to group1 and group2 but not group3)

- Combining ‘Not’ operator: !group1!group2!group3 (actions specified in this format will only be executed on hosts that do not belong to either group1, group2 and group3)
- Combining ‘Or’ and ‘And’ operator: group1|group2&group3 (actions specified in this format will only be executed on hosts that belong to either group1 or a combination of group2 and group3)

- Designed the MySQL tables to store log information.
- Enhanced the reporting mechanism, which has either web based or command line interface as the front-end and MySQL [31] database tables as the back-end.
- Performed detailed evaluation of the LOSCMF solution