Design and Implementation of a Tool for Automating Cluster Configuration
For a Software Defined Storage System

Submitted by:
Sindhusha Marakani
This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author:
Sindhusha Marakani
E-mail: sima14@student.bth.se

External advisor:
Stefan Bernbo
CEO, Compuverde AB
Email: stefan@compuverde.com

University advisor:
Prof. Lars Lundberg
Department of Computer Science and Computer Engineering
Email: lars.lundberg@bth.se

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57
ABSTRACT

Context Traditional storage systems are proving to be inefficient to handle the growing storage need of a modern IT organization. The need for a cost effective and scalable storage framework has led to the development of a Software Defined Storage (SDS) solution. SDS can be defined as an enterprise class distributed storage solution that uses standard hardware, with all the important storage and management functions performed by an intelligent software. Configuring and maintenance of these storage clusters require converting an SDS from any unknown state to a predefined, known state. This configuration of the SDS is best done with minimal human intervention, to ensure minimal errors and save the man hours spent in the configuration process.

Objectives A tool for automatic configuration of a SDS storage cluster has been designed and implemented. The tool has later been used to study the man hours saved in the configuration of the SDS cluster. The study also involves a cost-benefit analysis to estimate the break-even point for such a tool to motivate the automation of a SDS cluster configuration process.

Methodology In this study, experts from the field of Software Defined Storage have been interviewed to identify interesting and most common states of a SDS cluster. Later a tool was build such that it communicates with the underlying SDS storage cluster to configure it into one of the identified final states. This tool built was later used to conduct experiments wherein the amount of man hours saved by automating the process of cluster configuration was calculated.

Results The tool built was validated through results obtained from the experiments which show that the work time involved in the process of cluster configuration is reduced by 90% - 96% (based on the complexity of the cluster configuration). Also, the lead times of the configuration process are similar when configuring simple states but is greatly reduced by automation when performing complex configurations.

Conclusions Similar to any other software automation, the process of automating the configuration of a distributed storage cluster has proven to be beneficial. Automating the process of cluster configuration saves time, reduces human errors induced in the configuration process and improves repeatability of the configuration process. Through the cost-benefit analysis of the complete process, the use of the tool beyond 20 days is deemed profitable for the organization.

Keywords: Software Defined Storage, Automation, Cluster Configuration, Cost-Benefit Analysis.
ACKNOWLEDGEMENT

Firstly, I would like to express my sincere gratitude to my advisor Prof. Lars Lundberg for his continuous support in my master thesis and related research, for his patience, motivation, and immense knowledge. I could not have imagined having a better advisor and mentor.

Besides my University advisor, I would like to thank my external advisor Stefan Bernbo and Mikael Blomqvist (Executive Chairman) for giving me the opportunity to conduct this thesis along with Compuverde AB. I would also like to thank Christian Melander and Yovko Petrov for their insightful comments and their encouragement. Without their precious support it would not be possible to conduct this research.

I thank the employees of Compuverde AB and my test subjects for their valuable time and for participating in my experiments. Also, I thank my friends at Blekinge Institute of Technology for their stimulating discussions and for all their support.

Last but never the least, I would like extend my gratitude to my dad Satyanath Marakani and my mother Rajarathnam Marakani for encouraging me all through my masters.
CONTENTS

ABSTRACT ..................................................................................................................1

ACKNOWLEDGEMENT .................................................................................................II

CONTENTS ..................................................................................................................3

LIST OF FIGURES .......................................................................................................5

LIST OF TABLES ..........................................................................................................6

LIST OF ABBREVIATIONS .........................................................................................7

1 INTRODUCTION ......................................................................................................8
  1.1 WHY IS AUTOMATION OF CLUSTER CONFIGURATION INTERESTING? ..........8
  1.2 BACKGROUND ....................................................................................................9
    1.2.1 Software Defined Storage ...........................................................................9
    1.2.2 File system ..................................................................................................9
    1.2.3 Erasure coding ............................................................................................10
    1.2.4 Storage cluster ............................................................................................10
    1.2.5 Configuring a storage cluster .......................................................................10
  1.3 AIM AND OBJECTIVES ......................................................................................11
  1.4 RESEARCH QUESTIONS .....................................................................................11
  1.5 THESIS OUTLINE ...............................................................................................12

2 RELATED WORK .....................................................................................................13

3 METHODOLOGY .....................................................................................................15
  3.1 LITERATURE REVIEW .......................................................................................15
    3.1.1 Inclusion and Exclusion criteria ...................................................................17
  3.2 EXPERT INTERVIEWS .......................................................................................17
  3.3 DESIGN AND DEVELOPMENT OF THE AUTOMATION TOOL .........................18
  3.4 EXPERIMENT ....................................................................................................19
    3.4.1 Test Environment .......................................................................................19
    3.4.2 Test subjects ...............................................................................................20
    3.4.3 Experimental results .................................................................................20
    3.4.4 Run-time errors ..........................................................................................22

4 RESULTS AND ANALYSIS ..................................................................................24
  4.1 IDENTIFIED CLUSTER STATES .......................................................................24
  4.2 COST TO BUILD THE TOOL .............................................................................25
    4.2.1 Planning cost ...............................................................................................25
    4.2.2 Tool development cost ..............................................................................25
    4.2.3 Total Cost of development .........................................................................25
  4.3 COST-BENEFIT ANALYSIS ...............................................................................25
    4.3.1 Calculating the break-even point .................................................................26

5 DISCUSSIONS .........................................................................................................28
  5.1 LIMITATIONS ....................................................................................................29
  5.2 VALIDITY THREATS ..........................................................................................29
    5.2.1 Internal validity ...........................................................................................29
    5.2.2 External validity ..........................................................................................29

6 CONCLUSION AND FUTURE WORK ....................................................................30
  6.1 FUTURE WORK ..................................................................................................30

REFERENCES .............................................................................................................31
APPENDIX A: LITERATURE REVIEW - KEYWORDS AND RESULTS OBTAINED............ 33
APPENDIX B: INDIVIDUAL EXPERIMENTAL RESULTS........................................ 34
APPENDIX C: SOURCE CODE OF THE AUTOMATION TOOL............................... 38
APPENDIX D: INTERVIEW QUESTIONNAIRE TO CALCULATE THE NUMBER OF TIMES EACH CLUSTER CONFIGURATION IS PERFORMED PER DAY............................ 45
LIST OF FIGURES

Figure 1.1: An overview of SDS architecture.........................................................9
Figure 1.2: An SDS setup and topology.................................................................10
Figure 2.1: SDStore system architecture..............................................................11
Figure 3.1: Overview of methodology.................................................................14
Figure 3.2: A screenshot of the automation tool that was developed....................16
Figure 3.3: Description of the test environment....................................................17
Figure 2.4: Solution for run-time error.................................................................19
Figure 4.1: Cluster configurations identified during expert interviews....................21
LIST OF TABLES

Table 3.1: Search results of literature review.........................................................15
Table 3.3: Average time for cluster configuration (for end users).................................18
Table 3.4: Average time for cluster configuration (for a company).................................18
Table 4.1: Benefits of the automation tool developed.................................................22
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDS</td>
<td>Software Defined Storage</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
</tr>
<tr>
<td>SPEC</td>
<td>Standard Performance Evaluation Corporation</td>
</tr>
<tr>
<td>NFS</td>
<td>Network File System</td>
</tr>
<tr>
<td>CIFS</td>
<td>Common Internet File System</td>
</tr>
<tr>
<td>LT</td>
<td>Lead Time</td>
</tr>
<tr>
<td>WT</td>
<td>Work Time</td>
</tr>
<tr>
<td>CBA</td>
<td>Cost Benefit Analysis</td>
</tr>
<tr>
<td>BEP</td>
<td>Break Even Point</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>SSHD</td>
<td>Solid State Hard Disk</td>
</tr>
<tr>
<td>SDDC</td>
<td>Software Defined Data Centre</td>
</tr>
<tr>
<td>ROI</td>
<td>Return Of Investment</td>
</tr>
<tr>
<td>DFS</td>
<td>Distributed File System</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

Everyday large amounts of data is being generated as a result of various activities that take place in several organizations. In order to reuse this data, it has to be stored into a data storage system.

Traditionally, all the data generated in an organization was stored in large central storage systems. Later, due to single point failures and performance bottlenecks, storage technologies have moved towards distributed storage solutions. Distributed storage offers increased data availability and bandwidth. However, in the current scenario, distributed storage solutions have reached their breaking point and are unable to cope with the tremendous growth in storage-related demands for capacity, performance, functionality, and flexibility [1].

A Software Defined Storage (SDS) comes as a solution for this issue by abstracting the storage control operations from the storage devices and setting it inside a software layer [2]. As the storage control operations are abstracted from the storage devices, the SDS solution is not limited by the capacity of the underlying hardware and has the ability to scale limitlessly. The software adaptation also means that reliability and accessibility features, such as data redundancy and accessibility of storage (through protocols) have to be implemented in the central software. This allows the end users to customize their solution according to their needs. Each of these variations that result from the customization of an SDS solution is called as a final state.

An SDS cluster is defined as a collection of storage nodes, wherein each node is an independent computer system [3]. A virtual file system spans over all the nodes, abstracting the combined storage capacity of all the nodes into a single virtual disk and providing the end users with a shared view of the files stored on it. With a wide variety of parameters to an SDS solution, it is safe to say that an SDS solution is a customizable, complex distributed storage solution that has been designed to provide flexibility, scalability and cost-efficiency.

As storage now advances towards SDS solutions that can be customized, currently there exists no tools that automate the process of configuring a cluster. Also, since the field of SDS is pretty new, there exists no standard principle for implementing a SDS solution. This has resulted in various native implementations of SDS that are incompatible with one another. Also, there have not been any studies that substantiate the importance and effectiveness of automation in SDS systems.

Through this study, I intend to begin research in the area of SDS automation. Since, the current research area is nascent and keeping my time constraints in mind, I intend to limit this research scope to the process of automating cluster configurations in a SDS solution. Through this research, I demonstrate the effectiveness of automation in the field of SDS and encourage further studies that deal with automation of various processes in a SDS system.

1.1 Why is automation of cluster configuration interesting?

A SDS cluster can be configured into various final states by varying its features and parameters, these end states differ in their reliability, accessibility and robustness. From the consumers’ point of view, a company moving to a SDS technology will be interested to measure the differences in the performance of the different final states of an SDS solution.
From the perspective of a company that offers SDS technology, they would need an automation tool to configure their clusters into different final states for running test cases and benchmark programs. Meaning, they can automatically configure their cluster into a predefined state before performing functionality, performance test or running benchmarking tools on their product.

Also, automating cluster configuration will ensure minimal human intervention, reduce the man-hours spent on the configuration of each final state and minimize human induced errors in the configuration process. This also, improves the repeatability of the complete process.

1.2 Background

The following study has been conducted in collaboration with Compuverde AB [17], a Swedish IT company in Karlskrona that deals with data storage solutions and cloud computing technology. This Compuverde storage solution [17] was used for automating the process of cluster configuration and running experiments as a part of the cost-benefit analysis.

The rest of this section provides the reader with a brief introduction to the field of SDS, the state-of-art in this field and basic information about the various features of a SDS solution.

1.2.1 Software Defined Storage

The theme of a SDS cluster is to separate the functionality of data management from the underlying hardware [5, 6]. As a result the data storage policy is no longer dependent on the hardware that has been used, giving user the flexibility of using cheap commodity hardware. The separated control is now placed in a “Control layer” which is a software that deals with management of storage operations along with providing data resilience and security.

![Figure 1.1: An overview of SDS architecture](image)

As the management of data is now abstracted from the storage layer, data management functions such as load management, availability, resilience and security are converted into their software counterparts. For example, data resilience that was earlier implemented by having a redundant array of storage disks (RAID) is now implemented in the software through replication or erasure coding.

This software adaptation of various data management functions of the distributed storage, gives it flexibility and the possibility to scale the storage solution according to ones need [17].

1.2.2 File system

A file system is traditionally used to control the process of storing/retrieving data in the underlying storage. A distributed file system (DFS) allows users of physically distributed
computers, connected by a local area network (LAN), to share data and storage resources by using a common file system [22].

In the Compuverde storage solution, the file system is implemented as a single namespace that spans over all the nodes in the storage cluster and all the nodes have a consistent view of the files that are stored in it. This file system is generally accessed to read/write data through various protocols such as Common Internet File System (CIFS), Network File System (NFS), Cloud Data Management Interface (CDMI), etc.

1.2.3 Erasure coding
Files stored in a distributed storage solution can either be replicated or encoded using erasure codes to improve availability and long-term durability of the storage solution [30]. In the SDS solution used for this thesis, Compuverde uses erasure codes to encode the data stored in their system and exploit the gains in network bandwidth, storage capacity and computational resources.

Erasure codes divide the files stored into ‘m’ fragments and recode them into ‘n’ fragments (with padding) such that ‘n’ greater than ‘m’. The key property of erasure codes is that even if the storage cluster loses (n-m) fragments of the file due to node failure, the original data file can still be reconstructed from any of the ‘m’ fragments that remain in the cluster.

1.2.4 Storage cluster
A storage node is referred to a storage device that is connected to a LAN network. This network enables data communication within the storage devices leading to sharing of information and resources [3].

A storage cluster is defined as a large scale data processing system that is made up of a collection of storage nodes [3]. The various nodes that are present in a cluster have a single namespace spanning over them and share a consistent view of data stored in the cluster. Through their shared resources the nodes in the cluster ensure that data is distributed across all the nodes in the cluster. This distribution of data along with the implementation of Erasure codes, ensures the protection of data stored in the cluster.

1.2.5 Configuring a storage cluster
As mentioned earlier, a SDS cluster would contain numerous storage nodes that are configured to be a part of a single cluster. These nodes are connected through a LAN or a network switch that would enable sharing of resources between all the nodes in the cluster. All the nodes of the cluster would share a single namespace (a shared file system) that can be accessed through various access protocols.

All the storage nodes in the cluster are installed with a software (Compuverde in this case) that forms the ‘control layer’ of the SDS solution. This intelligent software would take over all the data management functions and enforce reliability of data through erasure coding and user access control(s). Also, all the nodes of the cluster communicate through the control layer to ensure that the nodes are running and are synchronized in terms of data at all times.

Figure 1.2, gives an overview of a SDS setup. The clients of the storage solution that wish to access data connect to the underlying storage through the network. These clients use various access protocols to access the data stored in the SDS solution and to view the file share. Here, a management tool is installed on one of the client systems and can be used to manage the SDS cluster.
The process of cluster configuration would involve configuring the shared file system and its parameters such as: erasure coding, access protocols and users access and setting up folders on the file share with a particular format.

1.3 Aim and objectives

Aim: To design and implement a tool for automating the process of cluster configuration in a SDS system and conducting a cost benefit analysis of the same.

Objectives:

- Design and implement a tool that can automate cluster configuration.
- Measure the time spent in designing and implementing the tool.
- Conduct experiments to measure the time saved by using the tool versus configuring the cluster manually.
- Figure out the factors that may be considered as costs and benefits of the automated tool.
- Conduct cost-benefit analysis.
- Calculate the break-even point for the analysis.

1.4 Research Questions

RQ1: What are the different states of a distributed cluster that are interesting to automate?

Motivation: Since only few studies talk about the automation and cluster configurations in a SDS solution, we first need to identify the different cluster states that should be automated.
Interesting states of a cluster, can be defined as the final states that are either frequently used, states that are prone to human induced errors or the states that involve very complex configurations thus taking up a lot of man-hours in the configuration process.

**RQ2:** How much time does it take to develop a tool that automates the process of configuring a cluster?

**Motivation:** Calculate the total amount of time that has been spent in building the automation tool will denote the cost of the automation process. This time measured will include the time spent in planning, designing, coding and testing the automation tool. The time measured would help determine if it is advised to automate procedures in a SDS solution. Also, this time measured can later be used for the cost-benefit analysis to motivate the automation of the configuration process.

**RQ3:** What would be the break-even point, from a cost-benefit perspective, of the automation process?

**Motivation:** Calculating the break-even point of the automation process would determine the time period after which the use of the tool will be profitable to the organization and will further inspire studies to automate and to integrate the field of SDS.

### 1.5 Thesis Outline

The following thesis is outlined as follows: The “Introduction” provides a short description about the context of the study that will be dealt with in the thesis work. The “Related work” section of this document describes in short, other studies that have been conducted in the areas of automation, software defined storage and cluster configuration. The “Methodology” section then explains details about how the thesis has been conducted followed by the “Results and Analysis” section that demonstrates the findings of the study and analyses the findings using statistics. The “Discussions” section validates the study and finally the “Conclusions” sections demonstrate the knowledge gained through the thesis.
2 RELATED WORK

The need for agility and optimization in enterprises has led to the trend of software defined computing, where in the entire computing infrastructure - compute, storage and network is becoming software defined and dynamically programmable [4].

In terms of storage, converting traditional distributed storage into a Software Defined Storage would mean to abstract the control of data from the underlying hardware into a control layer [5, 6]. This separation of control from the storage devices, is the essence of software defined storage and equips the solution to be scale-out, customizable, automatic and manage policies through masking [6].

![Figure 2.3: SDStore system architecture from [2].](image)

SDstore [2] (shown in Figure 2.1) is an experimental framework that uses an emulator to check the feasibility of developing a software defined storage solution. This study draws inspirations from software defined networking and tries to dissociate storage policies related to data flows from the underlying storage layer. IOFlow [7] by Microsoft is another study that implements high level data flow policies in their control layer. Through the control plane, complex yet useful policies are implemented which would otherwise be difficult to implement in a traditional distributed storage environment.

While SDstore and IOFlow, deal with abstracting dataflow policies from the storage layer, flexstore [8] implements a policy engine module that monitors the energy supply to the storage system. Based on the analysis of the energy consumption and availability for the storage solution, the policy engine enforces power tradeoffs in the system. Another study [9] described how the implementation of different file system features of the SDS can affect the utilization of sparse disks in the underlying storage layer.

Syndicate [10, 5] is an attempt at unifying multiple technologies to provide a stronger control layer in their SDS implementation. Here, the control plane in Syndicate decreases the coupling of a storage solution to its constituent provider by addressing cross provider consistency, security and fault tolerance [10]. By such a decoupling, the functional benefits
of each component of the storage solution can be exploited without worrying about their interoperability and common storage concerns.

In [5], the authors integrate storage resources such as Hadoop HDFS [11], Ceph [12] and Swift [13] with OpenStack [13] to achieve the concept of SDS. In this implementation, OpenStack acts as the control layer and manages cloud services that make use of this SDS. Through the use of HDFS, Ceph and Swift, the SDS architecture was able to provide support for file, block or object storage and a software named ViPR was used to control the different storage devices and manage storage spaces for OpenStack [5].

Several other commercial implementation of SDS, such as EMC ViPR [14], IBM Storewize [15], IBM Open platform [16] and Compuverde [17] have been developed to improve the agility, efficiency, resiliency and security of a traditional distributed storage system.

Within these implementations of SDS, Ceph [12], an open-source SDS solutions also provides a deployment tool named Ceph-deploy that helps a user to configure a storage cluster to begin using Ceph. Other such tools available are Chef and Puppet that deal with version dependency relations while updating the software’s on various servers of the SDS solution [31]. However, these softwares does not deal with configuration of the Ceph clusters or any sort of automation relating to cluster configuration by tweaking the parameters of the cluster or the file system that is deployed on the Ceph solution.

After gaining an insight into various implementations of an SDS, studies that dealt with automation of an SDS solution and configurations of clusters in a SDS were investigated. Reference [18] claims that the lack of intelligence, robustness and self-adjustment are hinder the use of a SDS and proposes a Quality of Service (QoS)-Aware cluster that schedules its I/O resources accordingly. The authors aim to build data centers that are highly available, scalable and QoS. Similarly, [19] aim to develop intelligence for a Software Defined Data Centre (SDDC). In this study, the SDS performs run-time analysis of user activity and estimates future user activity by using K-means clustering and Baum-Welch algorithm.

Finally, Google’s Borg [20] is a cluster manager that abstracts the underlying storage to run application tasks based on task priority. The underlying storage is completely abstracted from the applications on Borg and may consist of a number of storage clusters called “borglets” which ensure efficient utilization of the storage resources. Borg admits, schedules, starts, restarts, and monitors the full range of applications that Google runs [20].
3 METHODOLOGY

This section describes the steps that were involved in the study. Initially, a basic literature review was conducted to gain knowledge in the field of SDS and to discover tools that can be used to configure a SDS cluster. During this study, the lack of tools that configured a SDS solution into different configurations was observed and this research gap was used to formulate the three research questions that were focused as a part of this thesis work.

After the formulation of the three research questions, there was a need for a deeper and detailed review of the existing literature to gain insight into areas of SDS technologies. The findings of this literature review were presented in Section 2 as ‘Related Work’. Also, experts from the field of SDS technologies were interviewed to gain this knowledge. These interviews were conducted as the field of SDS is rather new and dynamic and expert opinions would help us identify the cluster configurations that were currently interesting.

The results from the literature review and expert interviews were used as inputs while designing the automated tool that can be used for SDS cluster configuration. The inputs influenced the number of cluster configurations that were automated, i.e. the states that were identified in the previous step were automated using the tool. Once the design for the tool was completed, the tool was then developed and tested following a cyclic software development process [27]. The time that was spent in designing, implementing and testing the automation tool was measured to be used in the cost-benefit analysis during the later stages.

After the tool was developed, it was used to conduct experiments and determine if the process of automating cluster configuration is beneficial and to determine the break-even point of the process. For this, experiments were conducted in collaboration with Compuverde AB, where employees of the company and few other test subjects were asked to configure a Compuverde cluster (an SDS cluster) into the states identified earlier. They were asked to perform the configurations both manually and by using the automation tool to measure the time difference in both the procedures. The results of these experiments along with the time measurements that were made during the development of the automation tool were later used to perform a Cost-Benefit Analysis (CBA).

Based on the results from the CBA, the conclusions for this study were drawn and are described in sections 4 and 5 of this thesis report. Figure 3.1 shown below, gives a brief overview of the complete methodology followed during this study.

3.1 Literature Review

For the literature review, studies related to SDS, current tools that deal with automating SDS technologies and finally cluster configuration tools in the field of SDS were considered. The research papers pertaining to the different areas were obtained from the IEEE, Inspec, Science Direct and ACM databases.

The search was performed in six stages using different sets of keywords each time. The keywords were chosen such that they initially targeted a narrow field of SDS. Later the keywords were modified to broaden the search and look for different studies that reported software defined technologies, how these technologies were configured and studies that deal with automating them. The list of keywords used for searching the databases and the results obtained in each search is specified in Table 3.1
A total of 1020 Journal and Conference papers were obtained as results to the literature review conducted. These results obtained were then filtered to only contain articles whose full text was available in the database for access, this exclusion criteria reduced the search result to 756 articles. Later, these results were further scrutinized to contain articles from the last 15 years, causing the search results to contain 520 research papers. The titles followed by abstracts of all the 520 results were reviewed to see if they seem relevant or interesting for the thesis. Finally, 12 papers were selected from the search results. In spite of obtaining a large number of papers during the literature review, lot of them discussed software defined networking and hence had to be excluded due to which only 12 papers were selected from the 520 obtained.

Furthermore, the literature review was deepened by following the forward and backward snowballing technique [28] and looking into references of the selected papers.
<table>
<thead>
<tr>
<th>Keywords</th>
<th>Results Obtained</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IEEE</td>
</tr>
<tr>
<td>“Software defined storage”</td>
<td>11</td>
</tr>
<tr>
<td>“software define” and storage</td>
<td>137</td>
</tr>
<tr>
<td>“software defined storage” and automation</td>
<td>1</td>
</tr>
<tr>
<td>“software defined” and storage and automation</td>
<td>5</td>
</tr>
<tr>
<td>“software defined storage” and cluster configuration</td>
<td>0</td>
</tr>
<tr>
<td>“software defined” and storage and cluster configuration</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.1: Search results of literature review

3.1.1 Inclusion and Exclusion criteria

Inclusion Criteria:
- Studies that discussed SDS, its various features and automation tools were considered.
- Journal articles and conference papers that were published in the last 15 years (2000-2015) were considered.

Exclusion Criteria:
- Only articles published in English were considered.
- Articles that were not available as “full text” in the search databases were excluded. Meaning, searches that only had abstracts or citations available and that needed to be purchased to access the complete paper were excluded.
- Articles other than Journal and Conference Papers were excluded keeping in mind the quality constraints.

3.2 Expert Interviews

As an initial step for the design of an automation tool, the number of cluster configurations that would be dealt as a part of this study had to be limited to abide by the time constraints and company needs. Hence, the most interesting or commonly used cluster configurations of a storage cluster had to be determined. This was done by interviewing experts that are currently working in the field of SDS.

Expert interviews were chosen as the field of SDS is nascent and I intend to gain knowledge about the various features of SDS along with their implementations. An alternative to using interviews, was to use surveys. Surveys were rejected as each SDS vendor has their own implementation of SDS solution which will only provide confusion about the features and implementation methods for each feature of the SDS solution.

Thus, the interviews conducted were unstructured, so that I could communicate with the experts to obtain comprehensive knowledge in the implementations and functioning of an SDS solution. The questions asked were open-ended and started from basic questions like
“What is the main theme of an SDS solution to?” and ranged to deeper questions that provided insights of how each feature mentioned in Section 1.2 were implemented.

A System Architect, a Senior Developer and the Chief Executive Officer & Founder from Compuverde AB were interviewed to gain insights into the SDS technology and cluster configurations that were interesting to automate. The first expert, System Architect, provided me with an insight into the internal architecture of a SDS solution and gave a brief idea of how the different features of the SDS solution were implemented in Compuverde. Next, the Senior Developer from the company provided me with information about compliance testing and how their clusters were configured. Finally, the Chief Executive Officer and the founder of Compuverde AB, help me validate the various final states that were identified.

Interviewing the experts in this domain, gave me enough insights into how a SDS solution is designed to function and how it was implemented. This knowledge was then generalized to describe the basic features of any SDS solution and eight final states were identified as interesting to automate. Interesting states were defined as the states that are either frequently used in a SDS company, involve complicated configurations or are prone to human-induced errors as they involve numerous and minute variations in cluster parameters. Through this generalization of the knowledge gained, the threat of only using inputs for one organization were tackled. Finally, through conducting interviews, it was ensured that the information about interesting cluster states is up-to-date and was derived from the people currently working in this field.

3.3 Design and development of the automation tool

Based on the states that were identified during interviews with experts, a tool for automating cluster configuration was designed. The tool was designed to work on Windows environment and was developed using C# .NET and the Visual Studio IDE. The tool developed had no Graphical User Interface (GUI) to keep it as simple as possible and was built using an existing Compuverde Command Line Interface (CLI). The Compuverde CLI was used to interact with the underlying storage solution, issue commands to it and to control the nodes in the storage cluster.

Based on the user choice, the automation tool issues different commands, through the CLI, to the nodes in the storage cluster. These commands operate on the storage nodes to convert them from any unknown state to one of the final states that were identified during the expert interviews.

Figure 3.2, shown below is a screenshot of final automation tool that was developed. The tool requires user inputs such as: the cluster’s IP address, the IP address of the public network configured to the cluster and the IP address of the network interface. After the relevant information has been provided, the tool calculates the number of nodes that are present in the cluster and displays a list of options. These options represent the various final states of a cluster that were identified during the expert interviews. The user now chooses an option and the rest of the cluster configuration is executed automatically.
3.4 Experiment

The SDS solution used for the following experiments is from Compuverde AB, wherein Compuverde is a software defined distributed storage solution. The Compuverde solution is made up of two parts: The Compuverde software that is installed on a server, separating the control of data from the hardware; and the data storage layer consisting of cache and storage disks.

The Compuverde cluster would contain numerous servers that were installed with the Compuverde software and are configured to be a part of a single storage cluster. The servers (or storage nodes) can now share resources and communicate through the network using the installed software. A virtual file system which is a single namespace, with a consistent view of all the files stored on it, spans over all the nodes of the storage cluster. The storage cluster is configured to look similar to the Figure 1.2.

The Compuverde storage solution requires three or more storage nodes in your distributed storage setup in order to function as a SDS and thus four nodes were used for my experiments.

3.4.1 Test Environment

The experiment was conducted on a four node Compuverde cluster, where each node was a Compuverde Virtual Machine (VM) running on a Vmware ESXi host. Each node in the cluster had one cache, a 20GB disk and one storage disk, a 500GB Solid State Hard Drive (SSHD) attached to it. Figure 3.3, shows the setup of the Compuverde cluster that was used for the experiment.
3.4.2 Test subjects
For the experiment, test subjects of different levels of experience and user groups were considered. This was done to ensure that the tool would be beneficial for company personnel and the end users with different levels of experience.

Eight test subjects were a part of this experiment, four of which were employees of Compuverde AB and the rest represented end users. The end users had no domain knowledge and thus were provided with information about what was meant by cluster configuration and were allowed to interact with the Compuverde Management Tool before starting the experiment. This was done to ensure that the measurement of cluster configuration through the manual process would not be affected by the test subject’s inability to operate the Compuverde Management Tool.

The four employees of Compuverde that were chosen were testers. All the testers working at the company were chosen as they are the people who generally perform cluster configurations at the organization. The four end users were selected randomly from the university.

3.4.3 Experimental results
The experiment was conducted in a controlled environment, wherein the test subjects interacted with a desktop connected to the aforementioned test environment through a network. The standalone system ran on Windows 7 with standard configurations. Also, Compuverde Management Tool was installed on the desktop, as the test subjects would need it to configure the storage cluster manually.

Each subject was first given a list of cluster configurations that were identified during expert interviews. They were then asked to configure each scenario using the regular, manual process and the lead and work times of this process was noted using a standard stopwatch. A stop watch was used instead of tick-tocks in the code as, the time calculations measure the human hours saved by the use of the automation tool and not the speed of tool itself.

Next, the test subjects repeated the same scenarios using the automation tool that has been developed. The work time and lead times in this case was also noted. Furthermore, test
subjects from the company were given the flexibility of performing the cluster configuration as they do so in their daily work style to capture the time saved in real-time cluster configuration process that will be done at the company every day.

Table 3.3 and Table 3.4 gives the average times of all test subjects that were a part of this experiment.

<table>
<thead>
<tr>
<th>Scenarios for cluster configuration</th>
<th>Manual Process</th>
<th>Using Automation tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>A clean cluster</td>
<td>386 sec</td>
<td>386 sec</td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td>408 sec</td>
<td>408 sec</td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td>422 sec</td>
<td>422 sec</td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td>497 sec</td>
<td>497 sec</td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td>439 sec</td>
<td>439 sec</td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td>476 sec</td>
<td>476 sec</td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td>680 sec</td>
<td>680 sec</td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td>1212 sec</td>
<td>1212 sec</td>
</tr>
</tbody>
</table>

**Table 3.3: Average time for cluster configuration (for end users)**

<table>
<thead>
<tr>
<th>Scenarios for cluster configuration</th>
<th>Manual Process</th>
<th>Using Automation tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>A clean cluster</td>
<td>394 sec</td>
<td>394 sec</td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td>416 sec</td>
<td>416 sec</td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td>432 sec</td>
<td>432 sec</td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td>491 sec</td>
<td>491 sec</td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td>433 sec</td>
<td>433 sec</td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td>445 sec</td>
<td>445 sec</td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td>618 sec</td>
<td>618 sec</td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td>1025 sec</td>
<td>1025 sec</td>
</tr>
</tbody>
</table>

**Table 3.4: Average time for cluster configuration (for a company)**
From the above results it is observed that, the lead times for configuring a cluster into simpler states is almost equal in both the manual and the automated process. However, one could save a lot of lead time by automating complex cluster configurations. Also, the work time for each cluster configuration is roughly 51 seconds in the automated process while it ranges from 386 sec to 1212 sec in the manual procedure. Meaning that it has been reduced by an average of 93%.

A statistical analysis for the time calculation involved in the experiment was made by calculating the standard deviations. It is observed that the time calculations differ on an average by 28.10 sec for company personnel and 4.35 sec for end users while using the automation tool. This means that the variance in the work times involved in cluster configuration through the automation tool is negligible when compared to their lead times. The average standard deviation also motivates the fact that, carrying out the experiment with more number of test subjects will not affect the results of the experiment.

It is interesting to note that the average time for cluster configurations for end users is lower than that for company personnel in some cases. This is because the test subjects from the company spent some time in configuring the cluster judiciously and ensuring that all the parameters are configured correctly while the end users followed the stated instructions. This cautiousness from the company personnel is a result of their daily experiences in cluster configuration process and their understanding of the importance of tuning each parameter of the SDS solution correctly.

3.4.4 Run-time errors

Error handling mechanisms have been incorporated in the automation tool that has been developed. These mechanisms prompt the users for corrections, in case there are errors in the inputs provided. However, there exists one run time error that seldom occurs.

The automation tool rarely fails to connect to the file server present on the storage cluster while trying to create files on it. This error occurs because of Windows caching and when this happens, the automation tool gives the error message: “Error connecting to the file share”.

![Diagram](Figure 4.4: Solution for run-time error)
In such cases, open the windows command prompt and check for open connections using the command “net use”. If there already exists an open connection to the file share that we are trying to connect to, close this connection manually, using the command “net use (connection name) /delete”. Once the connection is closed, the choice of final state into which the cluster should be configured is re-entered into the tool and the tool successfully configures the storage cluster as the Windows caching problem is now solved.
4 RESULTS AND ANALYSIS

4.1 Identified Cluster states

During expert interviews, eight cluster configurations were identified, which are described below:

- **A clean cluster**: A clean cluster is a cluster with no data or file system, this configuration is required during maintenance or prior to performing tests.

- **A cluster with file system**: The underlying virtual storage is abstracted to the user as a single file system, with file hierarchies. The file system, represents a root file that is shared between all the nodes of a cluster. The file shared in between all the nodes may have variations in the access protocols and file erasure encodings that are configured.

- **A Cluster to test file encoding**: File encodings (erasure codes) are configured on the file system and determine how the files are stored in the underlying storage. The different types of erasure codes that are configured on the share file system provide varied levels of data reliability and performance of the cluster. Hence testing the implementation of the various erasure codes would be interesting as well as essential. Such a cluster has to be cleaned of any prior data and configured with different erasure encodings on the share file system.

- **A Cluster to test snapshots**: Through snapshots, a user can periodically save the state of the file system to protect it against involuntary human-induced errors. A cluster to test snapshots should be cleared of any prior data and be configured to have three folders, each with a different type of snapshot on it.

- **A Cluster to test access rights**: Access rights are used to provide access privileges to different users of the shared storage. It is essential to test a cluster and check if the SDS accurately safeguards data. This would require having a clean cluster with configurations of different scenarios in which the users have access restrictions.

- **A Cluster to test access protocols**: A cluster to test the accessibility of the SDS solution should be cleaned and later configured with a combination of access protocols based on the requirements of the user.

- **A Cluster for SPEC performance testing**: An interesting aspect for the organizations investing in SDS is the I/O speeds it can deal with. SPEC's benchmark is designed to evaluate the speed and request handling capabilities of file servers using the NFSv3 and CIFS protocols [29].

- **A Cluster for PyNFS and MSTestSuite testing**: PyNFS and MSTestSuite are benchmarks that are used for testing protocol compliance to NFS and CIFS protocols. These tools need the cluster to be configured in a specific method in order to run the benchmarks on the distributed SDS cluster.

Figure 4.1 given below represents the various states identified during the literature review and expert interviews.
4.2 Cost to build the tool

The cluster automation tool that was built involved three main stages: planning, development and testing. The cost involved in the development of the tool may be calculated by measuring the time spent in each of the aforementioned stages.

4.2.1 Planning cost
Planning for the tool involved discussions with experts, scrutinize the different scenarios identified and validating the final states that were identified. An approximate of 24 hours were spent in planning for the tool.

4.2.2 Tool development cost
The development phase involved coding and testing the code to check if the tool is successful in configuring the storage cluster. The time that was spent during the coding and testing phases was measured as 41 hours and 25 minutes.

4.2.3 Total Cost of development
The total cost of building the tool can simply be calculated as the time spent in both planning and developing the tool and can be equated to 65 hours 25 minutes.

4.3 Cost-Benefit Analysis

Cost-Benefit Analysis (CBA) is the most comprehensive method for assessing any project. It sets out all the costs and benefits associated with a given project, in order to weigh up if a project brings a net gain to the society [25].

Costs involved in the project:
- Cost of planning and designing the tool
- Cost of developing the tool
- Cost of testing the automation tool
Benefits from the project:
- Time saved by use of tool
- Time saved from repeating the configuration process
- Time saved through reduced human induced errors

4.3.1 Calculating the break-even point
In order to simplify the calculation involved in the cost-benefit analysis, the costs and benefits of the study are conducted in terms of time. As calculated in Section 4.2, the costs involved in the project can be summed up to 65 hours and 25 minutes.

Similarly, for calculating the benefits of the project on a time scale, the time saved by the reuse of the automation tool and the time saved by alleviating human errors in the configuration process is calculated. To perform the aforementioned calculations, all the testers from Compuverde AB was interviewed (a structured interview) to calculate the number of times each final state is configured during their typical day of work. Compuverde AB employs three testers who were given a questionnaire to answer the number of times each cluster configuration the tester performed per day. An average of all these results was calculated and described in Table 4.1 such that it depicts the configuration performed by one tester on an average day. Appendix D, provides the questionnaire that was provided to each tester as a part of the interview.

Table 4.1, shows the calculations involved in measuring the number of times each configuration is carried out at Compuverde AB and the total time saved by using the tool.

<table>
<thead>
<tr>
<th>Option</th>
<th>Number of times performed per day</th>
<th>Time saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>A clean cluster</td>
<td>5</td>
<td>5 x 346 sec = 1730 sec</td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td>5</td>
<td>5 x 370 sec = 1850 sec</td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td>1</td>
<td>1 x 371 sec = 371 sec</td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td>1</td>
<td>1 x 446 sec = 446 sec</td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td>2</td>
<td>2 x 363 sec = 726 sec</td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td>1</td>
<td>1 x 372 sec = 372 sec</td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td>2</td>
<td>2 x 568 sec = 1136 sec</td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td>5</td>
<td>5 x 963 sec = 4815 sec</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>11446 sec (3hours 12min)</strong></td>
</tr>
</tbody>
</table>

*Table 4.1: Net benefit of the automation tool developed (per day)*

In the table above, the amount of time saved during the configuration of every state has been calculated from Table 3.3. The time saved is obtained by subtracting the work time involved in configuring the cluster using the tool from the time taken to do so manually.

Based on the number of times each cluster configuration is carried out at the company, three hours and twelve minutes is saved everyday by the use of the automation tool that has been developed.

The Return Of Investment (ROI) for the tool that was developed is:

\[ \text{ROI} = \frac{\text{net benefit}}{\text{total cost}} \]

\[ \text{ROI} = \frac{\text{(Time saved by use of tool)}}{\text{Total time involved in developing the tool}} \]

\[ \text{ROI} = \frac{3.2 \text{ hours}}{65.4 \text{ hours}} \]

\[ \text{ROI} = 4.9 \% \text{ per day} \]
The ROI of the tool has been measured to be equal to 4.9% per day, meaning that the testers save 4.9% of the time invested in building the tool, by its use. To make more sense out of the above calculation, the ROI has been converted to the time that a tester saves at Compuverde AB, which equates to 3.2 hours each day (in the 65.4 hours that have been invested) by the use of the tool.

Since it has taken 65.4 hours to develop the tool, the company would need to use the tool for more than 20 days in order to reap benefits from the use of the tool. The 20th day is considered the break-even point for the tool and any use of it beyond the 20th day is considered a profit for the investment that the organization has made in the development of the automation tool.

The aforementioned calculation has been made considering that there exists only one tester in the company. If there exists more than one tester, the break-even point for the analysis will be lower than 20 days. For example, for two testers, the break-even point would be at the 10th day. Also, if the company provides this automation tool to its customers, for configuring their storage clusters, the break-even point would be lower than 20 days.
5 DISCUSSIONS

Software defined Storage is a new term that has been coined in the field of distributed storage systems. It has been conceptualized during the last decade and is yet to find a standard definition to what it stands for. It is due to this that various indigenous SDS solutions have surfaced that merely share the abstract concept of a SDS storage with one another. This project was performed along with Compuverde AB, an IT company that provides a SDS solutions.

Due to the recent nature of this field, there has been a deficit of studies and tools that deal with automating the various functionalities of a distributed SDS solution. Identifying this research gap, this project was proposed to automate the process of cluster configuration. In order to gain further insight into the working of the control layer in a SDS and to gain information about the cluster states that are interesting to automate, a literature survey was carried out. Also, experts currently working on an SDS solution were interviewed to determine final states of a distributed cluster that are useful to automate. The experts were selected to provide extensive knowledge about the features and concepts of an SDS solution and to provide a detailed knowledge about their implementations and shortcomings.

Eight cluster states were identified after expert interviews. I have tried to generalize the identification of the final states such that the final states identified can pertain to any SDS solution and not only to Compuverde AB. The identified final states of a storage cluster were useful to automate as they are either repeated numerous times at the company or consume a lot of time to configure. Also, the cluster configuration process is very susceptible to errors while configuring the various parameters of the SDS solution. By automating the process, the whole cluster configuration process becomes repeatable and will now not involve any human induced errors. As there is very less human-input that goes into the automated process.

After, identifying interesting final states of a cluster, a tool for automating the configuration of these states was designed and developed. Experiments were then conducted to determine the amount of time that has been saved by the use of the automation tool. It is interesting to note that the results obtain show that the work time involved in the process is now reduced by almost 90% for a simple cluster states and by 96% for states that are complex, involving a lot of steps.

The above mentioned time experiments were conducted on a four node Compuverde cluster. In cases where the cluster is larger and involves more number of storage nodes, the time saved due to automation process becomes more profound and evident. Also, the time spent on configuration of each final state depends on the hardware that the SDS solution is made up of. Since one of the main feature of an SDS is the ability to use commodity hardware, the user has the choice for the hardware of the cluster. If the user chooses to have faster / slower hardware that that mentioned in Section 3.4.1, the time measurements would be vary proportionally.

A cost-benefit analysis of the complete automation process was made to determine the break-even point of the process after which the use of the tool would be considered profitable for the organization. Based on the results obtained from the cost-benefit analysis, we can infer that the cost involved in automating the process of cluster configuration is quite insubstantial as the use of the automated tool beyond 20 days is deemed profitable to the company.
5.1 Limitations

The current automation tool that has been developed is limited by its use to the provider that it has been designed for. Meaning, this current tool can only be used for automating distributed clusters that have Compuverde installed on them. This limitation is due to the fact that there exists no universal standard yet, that dictates the functioning of a SDS solution, causing each vendor to develop a customized software defined solution.

However, we can overcome this limitation by adapting the current tool to configure clusters operated by a different vendor. This is done by simply replacing the commands that are used to communicate with the underlying distributed storage.

5.2 Validity Threats

5.2.1 Internal validity

Internal validity refers to the extent to which the researcher could claim that the independent variable caused the dependent variable [26].

Validity Threats:

- **Only eight test subjects were considered for experiments:** Although only eight test subjects were considered for performing experiments, the work time that was measured depended on the speed of each individual in providing inputs for the tool. Also, having a larger number of test subjects in the experiment will make no difference as only an average standard deviation of 28.10 sec for company personnel and 4.35 sec for end users was observed for all the eight final states. This can be seen in the Section 3.4.3.

- **Only three experts were interviewed to identify interesting states:** The three experts that were interviewed were a System Architect, a Senior Developer and the Chief Executive Officer & Founder of Compuverde AB. This ensured that I had inputs from both the conceptual design and practical implementation of the SDS solution. Also, it was impossible to collaborate with experts from different organizations due to the absence of a universal standard for an SDS implementation.

5.2.2 External validity

The external validity is the extent to which the findings of this study can be generalized legitimately [26].

Validity Threats:

- **The test environment is only made up of four nodes:** The experiments that were conducted as a part of this study used a cluster containing four nodes, which is a basic cluster that can be expanded to contain thousands of nodes. In cases where the cluster is made up of more nodes, the only difference that would be observed is that in the manual process both the work time and the lead time would increase proportionately. However, when using the tool, the work time would remain the same while the lead time will increase accordingly.

- **Time differences upon using different hardware for your cluster:** As in a distributed SDS solution, the end user has the possibility to customize his hardware according to his needs, the underlying hardware might differ from the one that is used in the study. However, in this case, the lead time for completing the cluster configuration using the tool may differ but the work time would remain the same.
6 CONCLUSION AND FUTURE WORK

Based on the literature review, interviews from experts and experiments conducted as a part of this study, the answers for the research questions that formulated can be answered as follows:

**RQ1: What are the different states of a distributed cluster that are interesting to automate?**
Eight distinct, final states for a storage cluster were identified during the literature survey and the expert interviews as states that are interesting to automate. These states ranged from a clean cluster to a much more complicated configurations that would later be used to perform testing. Section 4.1 gives a detailed view of all the different states that were identified in the study.

**RQ2: How much time does it take to develop a tool that automates the process of configuring a cluster?**
The development of the tool, including planning, coding and testing the tool was completed in approximately 65 work hours. These 65 hours include the time spent on planning for the tool and developing it using C#.NET in the Visual Studio IDE. Also, one should note that these 65 hours exclude the time spent in planning the experiments, setting up the test environment, conducting experiments, making the cost benefit analysis and documenting the thesis. The time spent on these activities was excluded as it does not pertain to the actual planning and building of the tool and would distort the cost benefit analysis [25].

**RQ3: What would be the break-even point, from a cost-benefit perspective, of the automation process?**
The Return Of Investment (ROI) of the automation process was calculated as 3.2 hours per day for a single tester. As around 65 hours were spent in developing the automation tool, the company would need to use it for at least 20 days to obtain benefits from its use. Use of the tool beyond the 20th day is deemed cost-effective for the organization.

6.1 Future work
This study tries to automate the process of configuring a software defined storage cluster. As a part of this attempt, eight final states for a cluster were identified and automated, due to time constraints. As extension to this work, more number of cluster states can be automated. The cluster states that have been automated using the tool developed include all the interesting states that were identified. However, as the field of SDS is still in its nascent stage and will develop further, more states can be added later based on the future developments in this field. Also, the tool can be extended to function in both Windows and Linux environment as the current tool is limited to Windows environments.
REFERENCES


## Appendix A: Literature Review - Keywords and Results Obtained

<table>
<thead>
<tr>
<th>Keywords</th>
<th>IEEE</th>
<th>Inspec</th>
<th>Science Direct</th>
<th>ACM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
<td>A</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>“Software defined storage”</td>
<td>11</td>
<td>11</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>13</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>“software defined” and storage</td>
<td>137</td>
<td>136</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>435</td>
<td>35</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>440</td>
<td>11</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>“software defined storage” and automation</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>“software defined storage” and cluster configuration</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>9</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>107</td>
<td>64</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>122</td>
<td>119</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>“software defined” and storage and cluster configuration</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>“software defined” and storage and cluster configuration</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>108</td>
<td>104</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Table 0.1: Results obtained during the literature review

**Key:**

B = Results obtained before applying exclusion criteria  
A = Results obtained after applying exclusion criteria  
S = Selected studies
### APPENDIX B: INDIVIDUAL EXPERIMENTAL RESULTS

**Test Subject 1**  
User group: End user

<table>
<thead>
<tr>
<th>Scenarios for cluster configuration</th>
<th>Manual Process</th>
<th>Using Automation tool</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lead Time</td>
<td>Work Time</td>
</tr>
<tr>
<td>A clean cluster</td>
<td>374 sec</td>
<td>374 sec</td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td>402 sec</td>
<td>402 sec</td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td>430 sec</td>
<td>430 sec</td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td>430 sec</td>
<td>430 sec</td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td>498 sec</td>
<td>498 sec</td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td>458 sec</td>
<td>458 sec</td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td>737 sec</td>
<td>737 sec</td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td>1164 sec</td>
<td>1164 sec</td>
</tr>
</tbody>
</table>

**Test Subject 2**  
User group: End user

<table>
<thead>
<tr>
<th>Scenarios for cluster configuration</th>
<th>Manual Process</th>
<th>Using Automation tool</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lead Time</td>
<td>Work Time</td>
</tr>
<tr>
<td>A clean cluster</td>
<td>366 sec</td>
<td>366 sec</td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td>409 sec</td>
<td>409 sec</td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td>418 sec</td>
<td>418 sec</td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td>567 sec</td>
<td>567 sec</td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td>417 sec</td>
<td>417 sec</td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td>491 sec</td>
<td>491 sec</td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td>695 sec</td>
<td>695 sec</td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td>1323 sec</td>
<td>1323 sec</td>
</tr>
</tbody>
</table>
### Test Subject 3
User group: End user

<table>
<thead>
<tr>
<th>Scenarios for cluster configuration</th>
<th>Manual Process</th>
<th>Using Automation tool</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Manual Process</td>
<td>Using Automation tool</td>
</tr>
<tr>
<td></td>
<td>Lead Time</td>
<td>Work Time</td>
</tr>
<tr>
<td>A clean cluster</td>
<td>373 sec</td>
<td>373 sec</td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td>412 sec</td>
<td>412 sec</td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td>418 sec</td>
<td>418 sec</td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td>515 sec</td>
<td>515 sec</td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td>422 sec</td>
<td>422 sec</td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td>459 sec</td>
<td>459 sec</td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td>621 sec</td>
<td>621 sec</td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td>1212 sec</td>
<td>1212 sec</td>
</tr>
</tbody>
</table>

### Test Subject 4
User group: End user

<table>
<thead>
<tr>
<th>Scenarios for cluster configuration</th>
<th>Manual Process</th>
<th>Using Automation tool</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Manual Process</td>
<td>Using Automation tool</td>
</tr>
<tr>
<td></td>
<td>Lead Time</td>
<td>Work Time</td>
</tr>
<tr>
<td>A clean cluster</td>
<td>432 sec</td>
<td>432 sec</td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td>408 sec</td>
<td>408 sec</td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td>422 sec</td>
<td>422 sec</td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td>474 sec</td>
<td>474 sec</td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td>418 sec</td>
<td>418 sec</td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td>497 sec</td>
<td>497 sec</td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td>667 sec</td>
<td>667 sec</td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td>1147 sec</td>
<td>1147 sec</td>
</tr>
</tbody>
</table>
### Test Subject 5
User group: Company Personnel

<table>
<thead>
<tr>
<th>Scenarios for cluster configuration</th>
<th>Manual Process</th>
<th>Using Automation tool</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lead Time</td>
<td>Work Time</td>
</tr>
<tr>
<td>A clean cluster</td>
<td>378 sec</td>
<td>378 sec</td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td>420 sec</td>
<td>420 sec</td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td>485 sec</td>
<td>485 sec</td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td>581 sec</td>
<td>581 sec</td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td>482 sec</td>
<td>482 sec</td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td>487 sec</td>
<td>487 sec</td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td>644 sec</td>
<td>644 sec</td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td>1324 sec</td>
<td>1324 sec</td>
</tr>
</tbody>
</table>

### Test Subject 6
User group: Company Personnel

<table>
<thead>
<tr>
<th>Scenarios for cluster configuration</th>
<th>Manual Process</th>
<th>Using Automation tool</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lead Time</td>
<td>Work Time</td>
</tr>
<tr>
<td>A clean cluster</td>
<td>402 sec</td>
<td>402 sec</td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td>421 sec</td>
<td>421 sec</td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td>431 sec</td>
<td>431 sec</td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td>463 sec</td>
<td>463 sec</td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td>429 sec</td>
<td>429 sec</td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td>421 sec</td>
<td>421 sec</td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td>595 sec</td>
<td>595 sec</td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td>1145 sec</td>
<td>1145 sec</td>
</tr>
</tbody>
</table>
### Test Subject 7
User group: Company Personnel

<table>
<thead>
<tr>
<th>Scenarios for cluster configuration</th>
<th>Manual Process</th>
<th>Using Automation tool</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lead Time</td>
<td>Work Time</td>
</tr>
<tr>
<td>A clean cluster</td>
<td>393 sec</td>
<td>393 sec</td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td>417 sec</td>
<td>417 sec</td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td>400 sec</td>
<td>400 sec</td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td>477 sec</td>
<td>477 sec</td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td>404 sec</td>
<td>404 sec</td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td>423 sec</td>
<td>423 sec</td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td>565 sec</td>
<td>565 sec</td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td>868 sec</td>
<td>868 sec</td>
</tr>
</tbody>
</table>

### Test Subject 8
User group: Company Personnel

<table>
<thead>
<tr>
<th>Scenarios for cluster configuration</th>
<th>Manual Process</th>
<th>Using Automation tool</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lead Time</td>
<td>Work Time</td>
</tr>
<tr>
<td>A clean cluster</td>
<td>401 sec</td>
<td>401 sec</td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td>407 sec</td>
<td>407 sec</td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td>410 sec</td>
<td>410 sec</td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td>443 sec</td>
<td>443 sec</td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td>417 sec</td>
<td>417 sec</td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td>449 sec</td>
<td>449 sec</td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td>667 sec</td>
<td>667 sec</td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td>764 sec</td>
<td>764 sec</td>
</tr>
</tbody>
</table>
APPENDIX C: SOURCE CODE OF THE AUTOMATION TOOL

Code for cleaning the cluster

```csharp
public void CleanCluster(string ClusterIP, string NetworkInterface, List<string> NodeIP) {
    //The thread is set to sleep for 1 second to avoid overlap of commands
    string output = "";
    ExecuteCommandPromptCommand cmd = new ExecuteCommandPromptCommand();
    Console.WriteLine("Your cluster is being cleaned....");
    output = cmd.ExecuteCommand("cman.exe nodeoff -c=\" + ClusterIP + \" -a -l=" + NetworkInterface);
    System.Threading.Thread.Sleep(2000);
    for (int i = 0; i < NodeIP.Count; i++)
    {
        output = cmd.ExecuteCommand("cman.exe diskinit -c=\" + ClusterIP + \" -n=" + NodeIP[i].ToString() + \" -a -l=" + NetworkInterface);
    }
    System.Threading.Thread.Sleep(NodeIP.Count * 78500);
    output = cmd.ExecuteCommand("cman.exe nodeon -c=\" + ClusterIP + \" -a -l=" + NetworkInterface);
    System.Threading.Thread.Sleep(3000);
    Console.WriteLine("Your cluster has been cleaned.");
}
```

Code to configure a basic file share

```csharp
public void BasicCluster(string ClusterIP, string NetworkInterface, List<string> NodeIP) {
    string output = "";
    CleanCluster(ClusterIP, NetworkInterface, NodeIP);
    System.Threading.Thread.Sleep(2000);
    Console.WriteLine("File share id being created...");
    ExecuteCommandPromptCommand cmd = new ExecuteCommandPromptCommand();
    output = cmd.ExecuteCommand("cman.exe fscreate -c=" + ClusterIP + \" -name=domain -coding=raid5_2_1 -smb1 -smb2 -smb21 -smb3 -nfs3 -nfs4 -l=" + NetworkInterface);
    output = cmd.ExecuteCommand("cman.exe shareadd -name=share -path=\"/\" -nfs -cifs -domain=domain -c=" + ClusterIP + \" -l=" + NetworkInterface);
    output = cmd.ExecuteCommand("cman.exe groupadd -name=Group -domain=domain -c=" + ClusterIP + \" -l=" + NetworkInterface);
    output = cmd.ExecuteCommand("cman.exe useradd -name=user -password=password -main_group=Group -domain=domain -c=" + ClusterIP + \" -l=" + NetworkInterface);
    Console.WriteLine("The File share has been created with default settings.");
    System.Threading.Thread.Sleep(2000);
    return;
}
```
public void UserAccess(string ClusterIP, string NetworkInterface, List<string> NodeIP)
{
    string option, output, choice;
    ExecuteCommandPromptCommand cmd = new ExecuteCommandPromptCommand();
    Console.WriteLine("Would you like to use our existing file share? [Y/N]:");
    choice = Console.ReadLine();

    Console.WriteLine("Choose the type of user access you would like to test:");
    Console.WriteLine("1. Two users belonging to the same group");
    Console.WriteLine("2. Two users belonging to two different groups each");
    option = Console.ReadLine();

    switch (option)
    {
        case "1":
        {
            if (choice == "N" || choice == "n")
                BasicCluster(ClusterIP, NetworkInterface, NodeIP);
            output = cmd.ExecuteCommand("cman.exe groupadd -name=Access_Group -domain=domain -c=" + ClusterIP + " -l=" + NetworkInterface);
            output = cmd.ExecuteCommand("cman.exe useradd -name=access_user1 -password=password -main_group=Access_Group -domain=domain -c=" + ClusterIP + " -l=" + NetworkInterface);
            output = cmd.ExecuteCommand("cman.exe useradd -name=access_user2 -password=password -main_group=Access_Group -domain=domain -c=" + ClusterIP + " -l=" + NetworkInterface);
            Console.WriteLine("Two users belonging to the same group have been created.");
        }
        break;
        case "2":
        {
            if (choice == "N" || choice == "n")
                BasicCluster(ClusterIP, NetworkInterface, NodeIP);
            output = cmd.ExecuteCommand("cman.exe groupadd -name=Access_Group1 -domain=domain -c=" + ClusterIP + " -l=" + NetworkInterface);
            output = cmd.ExecuteCommand("cman.exe useradd -name=access_user1 -password=password -main_group=Access_Group1 -domain=domain -c=" + ClusterIP + " -l=" + NetworkInterface);
            output = cmd.ExecuteCommand("cman.exe groupadd -name=Access_Group2 -domain=domain -c=" + ClusterIP + " -l=" + NetworkInterface);
            output = cmd.ExecuteCommand("cman.exe useradd -name=access_user2 -password=password -main_group=Access_Group2 -domain=domain -c=" + ClusterIP + " -l=" + NetworkInterface);
            Console.WriteLine("Two users belonging to the different group have been created.");
        }
        break;
    }
}
Code for testing snapshots

```csharp
public void TestSnapshots(List<string> NodeIP, string ClusterIP, string NetworkInterface)
{
    string output = "", Node, temp, temp2;
    string[] onlineNodeList = new string[NodeIP.Count];
    onlineNodeList = getOnlineNodeList(ClusterIP, NetworkInterface, NodeIP);
    Node = onlineNodeList[0];
    ExecuteCommandPromptCommand cmd = new ExecuteCommandPromptCommand();
    BasicCluster(ClusterIP, NetworkInterface, NodeIP);
    System.Threading.Thread.Sleep(2000);

    temp = Node.Remove(Node.LastIndexOf("."));
    temp2 = temp.Remove(temp.LastIndexOf("."));
    temp = Node.Replace(temp2, "");
    Fileshare = Fileshare + temp;

    try
    {
        NetworkConnection networkConnection = new NetworkConnection(@"\" + Fileshare + "\share", new NetworkCredential("user", "password", "domain"));
        myNetworkConnections.Add(networkConnection);
        if (Directory.Exists(networkConnection.NetWorkName + @"\Hourly Snapshot\") == false)
        {
            Directory.CreateDirectory(networkConnection.NetWorkName + @"\Hourly Snapshot\");
        }
        if (Directory.Exists(networkConnection.NetWorkName + @"\Daily Snapshot\") == false)
        {
            Directory.CreateDirectory(networkConnection.NetWorkName + @"\Daily Snapshot\");
        }
        if (Directory.Exists(networkConnection.NetWorkName + @"\Weekly Snapshot\") == false)
        {
            Directory.CreateDirectory(networkConnection.NetWorkName + @"\Weekly Snapshot\");
        }

        output = cmd.ExecuteCommand("cman.exe snapshotadd -c=" + ClusterIP + " -domain=domain -path=\"/Hourly Snapshot\" -interval=hourly -time=1,3,6,10,15,21,24 -copies=5 -l=" + NetworkInterface);
        Console.WriteLine("The snapshots have been created");
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        Console.WriteLine("Please check the IP address of the public network.");
    }
}
```
Code for testing file share access

```csharp
public void FileShareAccess(string ClusterIP, string NetworkInterface, List<string> NodeIP)
{
    String protocol_list, command_string, output;
    ExecuteCommandPromptCommand cmd = new ExecuteCommandPromptCommand();
    Console.WriteLine("Please provide a list of options (in the same line): ");
    Console.WriteLine("1. Enable CIFS on the file share ");
    Console.WriteLine("2. Enable NFS on the file share ");
    Console.WriteLine("3. Enable OpenStack on the file share ");
    Console.WriteLine("4. Enable Amazon S3 on the file share ");
    protocol_list = Console.ReadLine();
    BasicCluster(ClusterIP, NetworkInterface, NodeIP);
    System.Threading.Thread.Sleep(20000);
    command_string = "cman.exe sharemodify -c=" + ClusterIP + " -domain=domain -name=share -path=\"/\"";
    if (protocol_list.Contains("1"))
        command_string = command_string + " -cifs";
    if (protocol_list.Contains("2"))
        command_string = command_string + " -nfs";
    if (protocol_list.Contains("3"))
        command_string = command_string + " -openstack";
    if (protocol_list.Contains("4"))
        command_string = command_string + " -s3";
    command_string = command_string + " -l=" + NetworkInterface;
    output = cmd.ExecuteCommand(command_string);
    Console.WriteLine(output);
    Console.WriteLine("Your file share is now ready.");
}
```

Code for performance testing

```csharp
public void PerformanceSPEC(List<string> NodeIP, string ClusterIP, string NetworkInterface)
{
    Common CommonFunction = new Common();
    ExecuteCommandPromptCommand cmd = new ExecuteCommandPromptCommand();
    string output, Node, temp, temp2;
    string[] onlineNodeList = new string[NodeIP.Count];
    onlineNodeList = getOnlineNodeList(ClusterIP, NetworkInterface, NodeIP);
    Node = onlineNodeList[0];
    CleanCluster(ClusterIP, NetworkInterface, NodeIP);
    output = cmd.ExecuteCommand("cman.exe fscreate -c=" + ClusterIP + " -name=domain -coding=raid5_4_1 -smb1 -smb2 -smb21 -smb3 -nfs3 -nfs4 -l=" + NetworkInterface);
    output = cmd.ExecuteCommand("cman.exe shareadd -name=share -path=\"/\" -nfs -cifs -domain=domain -c=" + ClusterIP + " -l=" + NetworkInterface);
    System.Threading.Thread.Sleep(2000);
    output = cmd.ExecuteCommand("cman.exe groupadd -name=Group -domain=domain -c=" + ClusterIP + " -l=" + NetworkInterface);
    output = cmd.ExecuteCommand("cman.exe useradd -name=user -password=password -main_group=Group -domain=domain -c=" + ClusterIP + " -l=" + NetworkInterface);
    //create 15 folders on the file system
    temp = Node.Remove(Node.LastIndexOf(".")));"
temp2 = temp.Remove(temp.LastIndexOf("."));
temp = Node.Replace(temp2, "");
Fileshare = Fileshare + temp;
try {
    System.Threading.Thread.Sleep(200);
    NetworkConnection networkConnection = new NetworkConnection(@"\" + Fileshare + ":share", new NetworkCredential("user", "password", "domain"));
    myNetworkConnections.Add(networkConnection);
    for (int i = 1; i < 15; i++)
    {
        if ((Directory.Exists(networkConnection.NetWorkName + @"\"Folder_" + i.ToString() + "\")) == false)
            Directory.CreateDirectory(networkConnection.NetWorkName + @"\"Folder_" + i.ToString() + "\");
        output = cmd.ExecuteCommand("cman.exe shareadd -name=Folder_" + i.ToString() + "\" -path=/Folder_" + i.ToString() + "\" -nfs -cifs -domain=domain -c=" + ClusterIP + " -l=" + NetworkInterface);
    }
    Console.WriteLine("The cluster is ready for the performance test");
} catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}

Code for PyNFS and TestSuite testing
public void PyNFSandTestSuite_Old(string ClusterIP, string NetworkInterface, List<string> NodeIP)
{
    string output, encoding, temp, temp2, Node;
    string[] onlineNodeList = new string[NodeIP.Count];
    onlineNodeList = getOnlineNodeList(ClusterIP, NetworkInterface, NodeIP);
    Node = onlineNodeList[0];

    ExecuteCommandPromptCommand cmd = new ExecuteCommandPromptCommand();
    do{
        Console.WriteLine("Select a file encoding for your new cluster:");
        Console.WriteLine("1. Erasure(2+1)");
        Console.WriteLine("2. Erasure(2+2)");
        encoding = Console.ReadLine();
        if(Convert.ToInt32(encoding) < 1 || Convert.ToInt32(encoding) > 2)
            Console.WriteLine("Please select a valid option for file encoding.");
    } while (Convert.ToInt32(encoding) < 1 || Convert.ToInt32(encoding) > 2);
    CleanCluster(ClusterIP, NetworkInterface, NodeIP);
    if (encoding == "1")
        output = cmd.ExecuteCommand("cman.exe fscreate -c=" + ClusterIP + " -name=daily -coding=raid5_2_1 -smb1 -smb2 -smb21 -smb3 -nfs3 -nfs4 -l=" + NetworkInterface);
    else if (encoding == "2")
        output = cmd.ExecuteCommand("cman.exe fscreate -c=" + ClusterIP + " -name=daily -coding=raid6_2_2 -smb1 -smb2 -smb21 -smb3 -nfs3 -nfs4 -l=" + NetworkInterface);
    else Console.WriteLine("Please select a valid option for file encoding.");
}
output = cmd.ExecuteCommand("cman.exe shareadd -c=" + ClusterIP + " -d=+NetworkInterface");
System.Threading.Thread.Sleep(2000);
Console.WriteLine("The file share has been created.");

//groups
output = cmd.ExecuteCommand("cman.exe groupadd -name=users -domain=daily -uid=1000 -l=" + NetworkInterafce);
output = cmd.ExecuteCommand("cman.exe groupadd -name=nfs -domain=daily -uid=1 -l=" + NetworkInterafce);

//users
output = cmd.ExecuteCommand("cman.exe useradd -name=testadmin -password=Password1 -main_group=users -domain=daily -c=" + ClusterIP + " -l=" + NetworkInterafce);
output = cmd.ExecuteCommand("cman.exe useradd -name=testnfs -password=Password1 -main_group=nfs -domain=daily -uid=1 -l=" + NetworkInterafce);

//add file shares
try 
{
System.Threading.Thread.Sleep(2000);
NetworkConnection networkConnection = new NetworkConnection(@"\" + Fileshare + "share", new NetworkCredential("user", "Password1", "daily"));
myNetworkConnections.Add(networkConnection);
for (int i = 0; i < ShareNames.Length; i++)
{
    if (Directory.Exists(networkConnection.NetWorkName + @"\" + ShareNames[i].ToString() + "\") == false)
        Directory.CreateDirectory(networkConnection.NetWorkName + @"\" + ShareNames[i].ToString() + "\");
    output = cmd.ExecuteCommand("cman.exe shareadd -c=" + ShareNames[i].ToString() + " -path="/ + ShareNames[i].ToString() + " -nfs -cifs -l=" + NetworkInterafce);
}
}
if (Directory.Exists(networkConnection.NetWorkName + @"\BasicFileShare\Sub\"))
    == false)
    Directory.CreateDirectory(networkConnection.NetWorkName +
    @"\BasicFileShare\Sub\");
Console.WriteLine("Folders in your file share have been created.");

    //create symboliclink
    output = cmd.CreateSimulink(Fileshare);
    Console.WriteLine("Your cluster is ready for pNFS and TestSuite testing.");
} catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
### APPENDIX D: INTERVIEW QUESTIONNAIRE TO CALCULATE THE NUMBER OF TIMES EACH CLUSTER CONFIGURATION IS PERFORMED PER DAY

<table>
<thead>
<tr>
<th>Option</th>
<th>Number of times performed per day</th>
</tr>
</thead>
<tbody>
<tr>
<td>A clean cluster</td>
<td></td>
</tr>
<tr>
<td>Cluster with basic file share</td>
<td></td>
</tr>
<tr>
<td>Cluster to test erasure coding</td>
<td></td>
</tr>
<tr>
<td>Cluster to test snapshots</td>
<td></td>
</tr>
<tr>
<td>Cluster to test user access rights</td>
<td></td>
</tr>
<tr>
<td>Cluster to test access protocols</td>
<td></td>
</tr>
<tr>
<td>Cluster for SPEC performance Testing</td>
<td></td>
</tr>
<tr>
<td>Cluster for PyNFS and TestSuite Testing</td>
<td></td>
</tr>
</tbody>
</table>

The above questionnaire was provided to all the testers in Compuverde AB and the average of the results obtained have been used in Section 4.3.1.