Open Source Tools for Remote Incremental Backups on Linux: An Experimental Evaluation

Aurélio Santos, Jorge Bernardino

Instituto Superior de Engenharia de Coimbra, Portugal

aurhes@gmail.com, jorge@isec.pt

Abstract: Computer data has become one of the most valuable assets that individuals, organizations and enterprises own today. The majority of people agree that losing their data (programs, data sets, documentation files, email addresses, photos, customer data, etc.) would be a disaster. The reason most individuals avoid performing data backups though, is because they feel the process is complicated, tedious and expensive. This is not always true. In fact, with the right tool, it's very easy and affordable. In this paper we compare the performance and system resources usage of five remote incremental backup open source tools for Linux: Rsync, Rdiff-backup, Duplicity, Areca and Link-Backup. These tools are tested using three distinct remote backup operations: full backup, incremental backup and data restoration. The advantages of each tool are described and we select the most efficient backup tool for simple replication operations and the overall best backup tool.

Key words: open source tools, remote incremental backups, GNU/Linux, data backup, full backup, data restore

1. Introduction

Computer data has become one of the most valuable asset that individuals, organizations and enterprises own today. However data loss has become a common problem, originating a large array of data protection and recuperation techniques. People know that they need to regularly back up their programs, data sets, and documentation files, but it's not done as frequently or systematically as it should be particularly if there are changes in files from hundreds of folders or subfolders every day or week. The best way to ensure routine file backups is to make it easy to do and easy to document. Everyone knows that it is always important to regularly back up our files, because they can be accidentally deleted or overwritten (Hutchinson, N. C. et al. 1999).

Unfortunately, a vast majority of people and small businesses do not adequately protect and back up their data. Many individuals do nothing at all, while some think they are doing enough simply by copying their files to an external hard drive. While this method seems logical in theory, it has many disadvantages. Not only is it inefficient, since the backups are not automated and the simple copy of files does not protect them from viruses or corruption, and overall, it does not protect the entire operating system. Backup programs are specifically designed to offer easy, simple and comprehensive ways to protect the data. A good backup solution involves backup software. Backup software will make the data protection process much simpler and using backup software is much faster than copying all files.

The simplest way to protect a file system against disk failures or file corruption is to copy the entire contents of the file system to a backup device. The resulting archive is called a full backup. If a file system is later lost due to a disk failure, it can be reconstructed from the full backup onto a replacement disk. Individual lost files can also be retrieved. Full backups have two disadvantages: reading and writing the entire file system is slow, and storing a copy of the file system consumes significant capacity on the backup medium. Faster and smaller backups can be achieved using an incremental backup scheme, which copies only those files that have been created or modified since a previous backup. Incremental schemes reduce the size of backups, since only a small percentage of files change on a given day. A typical incremental scheme performs occasional full backups supplemented by frequent incremental backups. Restoring a deleted file or an entire file system is slower in an incremental backup system; recovery may require consulting a chain of backup files, beginning with the last full backup and applying changes recorded in one or more incremental backups (Chervenak, A. L. et al. 1998).

Linux as one of the most popular operating system has received a large number of backup tools. Those tools differ in the number of functionalities and techniques used to compress, encrypt and

transfer data to other systems. Most system administrators have been faced with the challenge of choosing a backup tool that suits their particular needs because many of the tools available, offer the same functionalities required by most people, making the process of choosing one, hard and normally being done solely based on the popularity of that tool. Beside the feature set, one important factor to choose a backup tool is the system resources usage, however this aspect is generally not tested and documented.

In order to find which tool has the best characteristics, we use different scenarios to test five popular remote incremental open source backup tools for Linux. In summary, the main contributions and achievements of this paper are:

- We present experimental results of system resources usage, execution time, and disk usage of five backup tools;
- We evaluate the differences of backup tools under three operation modes: full backup, incremental backup and data restoration;
- We evaluate the impact of using data compression and encryption;
- We choose the best tool depending on the particular needs of a user.

The remainder of this paper is organized as follows. Next section presents an overview of backup tools along with their advantages and disadvantages. In the following sections we introduce details about the five remote incremental open source backup tools. Next, we present our test methodology for the comparison. Next sections present the main results of our experimental evaluation and the other makes an overview of the related work. Finally, last section presents our conclusions and point out some future work.

2. Backup Tools

The process of backing up data has historically being a field of interest only to system administrators, concerned with the safety of user and corporation data under their responsibility (Crossler, R. E. 2010). With the exponential growing of computer users data loss incidents also have increased significantly. As such, more and more people are beginning to understand the importance of a good backup plan.

Over the years several tools have emerged, using different ways of doing backups. In this paper only remote and incremental backup tools will be considered due to the following reasons:

- Remote backups are considered the correct way of keeping data safe, because it guarantees that even in case of hardware failure of the backed up system, data can be restored (Zahed, K. S. et al. 2009).
- Using removable media to store backups of files is generally not a practical solution in terms of cost and because it could imply a less frequent backup of data.
- Incremental backup's tries to solve the problems associated with high network bandwidth required to transfer all the information necessary to replicate files on remote systems, transferring and writing only the modified parts of files (Qian, C. et al. 2010).

Backups are generally done not only to protect data from being lost, but also to allow the restoration of old copies of files, preserving the file modification history. As such, many backup tools can store different instances of the same file having each one an associated timestamp. This dramatically increases the storage space required, being normally necessary to limit the number of instances of each file, by deleting older versions beyond a certain number of days or only storing a determinate number of recent modifications done to a file.

Another way of reducing the storage space usage is by compressing files before transmitting them (Kothiyal, R. et al. 2009; Platoš, J. et al. 2008). The advantage of doing this is not only less space occupied but also less data that needs to be transmitted over the network. However, compression is typically a heavy operation to the system and, as such, it dramatically increase the system resources required to do backups. Also the process of applying modifications to files that have been compressed is typically more difficult due to the fact that they need to be decompressed and recompressed. Many tools prefer to store modifications on separate compressed files to avoid this additional system resource usage, being that, modifications are applied to files only when they are restored because that operation also implies their decompression (Platoš, J. et al. 2008).

Many times the storage used to place the backed up files does not belongs to the owner of the files, such as rented remote servers, cloud storage or peer-to-peer backup networks. This means that personal or confidential data can potentially be accessed without the consent of the owner (Machanavajjhala, A. & Reiter, J. P. 2012). A normal solution to this problem is file encryption (Ludwig, S. & Kalfa, W. 2001; Chakraborty, D. & Mancillas-López, C. 2012; Shih-Yu Lu. 2013), which is already supported by many backup tools. However, this also suffers from the same problems as compression of files. There is a system overhead during the encryption and decryption and the process of applying modifications also requires the decryption of the files.

Another important characteristic that is also sometimes required is to protect the transmission of files on shared networks such as the Internet (Shakhanova, M. V. 2013). Encryption of files is generally a good solution to that problem. If the files are not encrypted a secure transmission protocol can be used, being SSH the most widely used and supported by all the tested tools. In this paper our focus is on evaluation of remote and incremental backup tools for Linux that will be described in the next section.

3. Incremental Backup Tools for Linux

Due to the large number of backup solutions available for Linux it would be impossible to include the all on this paper. As such we selected a set of tools to test based on their popularity and on their capacity to be used from the command line and for its efficiency and practicality for use on servers. Based on the extensive list present on the Arch Linux wiki ("Backup Programs - ArchWiki" 2013) we choose the following five remote incremental backup open source tools for Linux: Rsync, Rdiff-backup, Duplicity, Areca and Link-Backup. In next sections we describe each one and present the reasons behind their inclusion in this paper.

3.1 Rsync

Rsync (Tridgell, A. 1999; "Rsync" 2013) is an open source utility that provides fast incremental file synchronization. It is one of the most popular backup tools and is installed by default in most Linux distributions. It is licensed under the GNU General Public License Version 3 (Kaminski, H. & Perry, M. 2007). One of the main characteristics of the tool is its flexibility that permits its use in several scenarios to do data transfers and file synchronization while minimizing the amount of data transferred. Rsync algorithm (Tridgell, A. & Mackerras, P. 1996) generates delta files containing only the modifications that a file received. Therefore, only those modifications are transmitted over the network and are applied to the replicated file. Rsync has been extended to create automated snapshot backups on Linux ("Easy Automated Snapshot-Style" 2013). The snapshot backup works as follows: Rsync has three processes that work together - the sender, generator and receiver. The generator, takes the basis file, i.e., the previous version, divides it into blocks and sends the signatures of the blocks to the sender. The sender returns only tokens for the matching blocks and literal data for non-matching parts. The receiver uses the tokens to copy blocks from the basis file into the right position and copies the literal data sent to construct the new snapshot backup version.

By being a quite efficient file synchronization tool many others have emerged, using Rsync algorithm as the basis for file synchronization while adding some others features such as file modification history and scheduled executions. Rsync is a quite focused tool for data synchronization and as such it does not offer much more functionalities beyond that. Although it is typically combined with other standard Unix tools to provide many optional features for data backup such as compression and encryption. In this paper we do not explore those possibilities because of the sheer number of combinations possible.

Rsync was chosen because it is considered the standard for backups in Linux and as such it serves as a basis of comparison to all other tools.

3.2 Rdiff-backup

Rdiff-backup ("Rdiff-backup" 2013) uses the Rsync algorithm to generate delta files with the modifications done in the file after the last backup. Those delta files can be applied to the original file to transform it into a more recent version of it.

Unlike Rsync, it only stores the changes to a file, not creating a new copy of it with those modifications applied, being more space efficient to keep the history of these file and allowing to restore any

previous versions of it. As such, it tries to combine the simplicity of a full copy of the data like Rsync, while keeping the modifications history. Rdiff-backup by default also compresses modifications of files but not the actual file. This tool is licensed under the GNU General Public License Version 2.

Rdiff-backup was chosen to compare how a specialized tool can improve the performance of backups in comparison with the more generic Rsync tool and to see the impact of keeping the data modifications history separated from the actual copy of the files.

3.3 Duplicity

Duplicity ("Duplicity" 2013) is a backup tool that uses the Rsync algorithm for file modification and is licensed under the GNU General Public License Version 2.

Unlike Rdiff-Backup, it supports file history and fully compresses stored data. Furthermore it adds file encryption capabilities using GnuPG ("The GNU Privacy Guard" 2013; Jallad, K. et al. 2002) to keep files safe in non-trusted storage. It can use more storage types than Rsync and Rdiff-backup, such as cloud storage. The files data and its modifications are stored in the GNU-tar format to save space. Even though the data is not stored with the simplicity of Rsync and Rdiff-backup, if needed, a manual restoration could still be achieved with the regular Unix tools: GnuPG, Rdiff and Tar.

Duplicity was chosen to this comparison because it supports modern storage types, encryption and is more recent than Rsync and Rdiff-backup, being quickly increasing in popularity recently.

3.4 Areca Backup

Areca Backup ("Areca Backup" 2013) is an easy to use backup solution supporting files compression and encryption. It is licensed under the GNU General Public License Version 2.

Like Rsync, Areca can use the delta algorithm to transmit only the modified parts of files. It is composed of several tools forming a versatile suite which includes a graphical user interface to generate backup plans stored in a XML file, although the execution of backups can also be done on the command line. It is a more configurable and complete tool than the others, allowing to perform full or incremental backups and differential backups in which it stores only the files that were modified since the last backup.

Areca was chosen because it is a fully featured backup suite, and as such, it is interesting to compare it against the others more lightweight backup tools.

3.5 Link-Backup

Link-Backup ("Link-Backup" 2013) is a Python based backup tool that uses an interesting method of replicating a file structure on a remote host.

This method tracks all unique file instances on a directory and builds a catalog of the files. The directory structure is preserved using hard-links to the catalog. This assures that no file is doubly stored even if it is duplicated on the source directory. Also renaming, moving or duplicating files will not cause a significant increase in storage space usage or transferred data, because it is just a question of changing links. This tool is licensed under the MIT License (Kaminski, H. & Perry, M. 2007).

It was because of this different way of storing backups that this tool was chosen for this test, so we can see how it compares to the others more traditional tools.

4. Test Methodology

In this section we present the test methodology used to compare the selected tools. Typically there are two distinct modes of operation for a backup tool: backup the data and restore the files. After the initial backup is performed, subsequent backups can be optimized in terms of used disk space and transferred data, by storing only the modifications the files suffered. This mode of operation is designated incremental backup and will be analyzed in this paper too. This is actually the most relevant mode for a real life scenario because it is the one that will be most widely used.

Backup tools are generally used to replicate files on remote machines, because storing files on the same machine do not solve most of the potential dangers that files suffer. As such it is relevant to

analyze the behavior of those tools in remote transfer operation, in order to see the bandwidth efficiency of each one. SSH was the chosen protocol for encrypted communication on this test due to being the most widely used (Ylönen, T. 1996) and also supported by all of the analyzed tools.

There are two major features that a backup tool could offer: compression and encryption. Compression of files is used to reduce the large amount of space required to store all the modifications that files have received and encryption to improve security of data stored into non trusted storage. Not all analyzed tools support those two functionalities and as they potentially impact severely the system resources usage. Therefore, the test set was separated into three main groups: tools can use compression and encryption, tools can use only compression and finally tools can´t use neither compression nor encryption.

A dataset of 10 GB was created to test the backup operations. In order to make it similar to real world backup's files it was generated with 50,000 files of 200KB each. To test the compression capabilities of the analyzed tools it was observed that the compression ratio of real world files was of approximately 50%. As such the files used on the tests were generated with equal proportions of linear data from /dev/zero interface and random data from /dev/urandom interface. Linear data achieves extremely high levels of compression due to the fact that compression works by identifying patterns in files and a linear file always has the same pattern. Random data generated files achieves the inverse by not being able to reduce its size due to not having recognizable patterns. The commands used to generate the files were the following:

```
dd if=/dev/urandom of=file1 count=1024 bs=107;
dd if=/dev/zero count=1024 bs=107 >> file1;
```

To simulate the directory structure of a real data set, the files were put in several randomly directories in which every folder contains a maximum of 10 folders inside, and so on, until a maximum of 10 levels of sub-folders in the hierarchy. A full mirror of the data is performed on the first test run. Then the following modifications are applied to the files: a) 10% of the files are deleted, b) 10% of new files are created and c) another 10% of files are modified by regenerating half of the file content. In the end the data has the same size but 15% of files contents were modified. Then the backup tool is run again in order to transmit only the file contents. Finally, to test the recovery of data, the original dataset is removed and the backup tool is run in recovery mode.

In order to analyze the system resources usage, it was used the dstat applications ("Dstat: Versatile resource" 2013). It combines vmstat, iostat and ifstat functionalities in only one tool while adding some other features. The arguments used on dstat where "--Tcmnd" in order to receive the UNIX epoch time and statistics from the CPU, memory, network and disk usage. The results were output to a CSV file. Furthermore, execution time of commands and backup storage space usage are logged into a file. Those statistics were only registered on the client side because it is there that the processing is done.

All tools that support compression are tested with it enabled and disabled, and the tools that support encryption are also tested with it enabled and disabled. Four tools can operate with compression and encryption disabled: Rsync, Rdiff-backup, Areca and Link-Backup. Just three can operate only with compression (Rdiff-backup, Duplicity and Areca) and only two can operate with both enabled (Duplicity and Areca). Consequently, there are three different tests (without compression and encryption; only with compression; with compression and encryption) that can be executed under three different scenarios (initial/full backup, incremental backup and restore).

In order to avoid possible interference on the results by the delayed reads and writes optimizations that Linux does, the "sync" command is executed between every run in order to flush all data buffered in memory to disk.

Two computers have been used, one as a client and the other as a server that stores the backups. The two systems where connected by a Gigabit Ethernet and tests where run with the two systems without any GUI server initialized and with the minimum necessary programs running.

The client specifications are:

- ArchLinux 64bits (17-01-2012 snapshot)
- CPU: Intel Core 2 Duo SU7300 at 1.3 GHz
- Memory: 4Gb of DDR3 at 1066 MHz
- Hard drive: SATA3 500GB, 8Mb cache, 5400rpm

The server specifications are:

- Ubuntu Server 11.04 64bits
- CPU: AMD A64 3500+ Winchester at 2.2 GHz
- Memory: 512Mb of DDR2 at 533 MHz
- Hard drive: SATA2 80GB, 8Mb cache, 7200rpm

The versions of the software tools are:

- Rsync 3.0.9-1
- Rdiff-backup 1.2.8-5
- Duplicity 0.6.17-3
- Areca 7.2.4-2
- Link-Backup 0.8-5

5. Experimental Evaluation

In this section we present the results of experimental evaluation with the three possible situations: without compression and encryption, only with compression, and with compression and encryption. From the data generated by the tests we record eight parameters considered the most relevant: i) the duration of the test (Time); ii) the average CPU usage (CPU); iii) the average consumed memory (Mem); iv) the average network received (Recv) and v) sent data volume (Sent); vi) the average disk read (Read) and vii) written data (Write); and viii) the final backup data size (Size). The following sections show the results for each test set.

5.1 Experiments without Compression nor Encryption

In Table 1 we show the test results for a backup of a dataset with 10GB for all the tools that support working without compression nor encryption. The tool that is not tested is Duplicity because it does not allow disabling compression of data. Some of the presented results have an average value of less than 1 MB, which explains the 0 values in the table. The tests are executed under three different scenarios: Initial/full backup; Incremental backup; and Restore.

Tab. 1: Results without using compression nor encryption

		Time (min)	CPU (%)	Mem (MB)	Network		Disk		Size (GB)
					Recv (MB)	Send (MB)	Read (MB)	Write (MB)	
Full	Rsync	12	28	143	0	15	13	0	10.37
	Rdiff-B.	25	25	7	0	3	7	0	10.45
	Areca	22	31	96	0	8	7	0	10.37
	Link-B.	44	7	26	0	4	3	0	10.39
Incr	Rsync	3	19	132	0	7	0	0	11.40
	Rdiff-B.	22	5	47	0	0	0	0	11.51
	Areca	4	39	124	0	9	0	0	12.46
	Link-B.	16	6	32	0	2	9	0	12.53
Rest	Rsync	10	38	112	20	0	0	19	
	Rdiff-B.	20	13	74	4	0	0	8	
	Areca	27	35	202	6	0	0	6	
	Link-B.	115	4	252	1	0	0	2	

The results of Table 1 show that Rsync is always the fastest tool in all scenarios, with 12, 3, and 10 minutes (to Full, Increment and Restore, respectively) and Link-Backup show the worst results in almost all modes being only surpassed by Rdiff-backup in incremental mode. CPU usage is similar for all tools with the exception of Link-Backup due to the fact that it also takes longer to execute. Memory usage show some mixed results with no clear winner. In terms of network usage Rsync shows to be

the less optimized tool and in terms of disk usage there is no clear winner. Backup size is quite lower with Rsync in full and incremental backup operations. The last four lines of the table represent the Restore operation, which obviously has no backup files.

The results show that Rsync is the overall best backup tool in all scenarios of operation. Remembering that the initial size of data set is 10 GB, Link-Backup has shown not being suitable for this kind of dataset with many files. The necessity of looking for duplicated files causes an overhead particularly in the full backup operation. The restoration time for Link-Backup is the highest, being for instance more than 10 times the time of Rsync showing a big inefficiency of the backup strategy.

Although Link-Backup takes more time to execute than the other tools, it has also the lowest usage of CPU, being that long operation not so noticeable on the overall system performance. The memory usage shows that Rsync requires a larger portion of memory to operate than other tools.

5.2 Experiments Only with Compression

In this section we show the test results of the three tools (Rdiff-backup, Duplicity and Areca) that can be configured to only use compression. Table 2 presents the results for the three scenarios: Full, Incremental and Restore.

Tab. 2: Results using only compression

		Time (min)	CPU (%)	Mem (MB)	Network		Disk		Size (GB)
					Recv (MB)	Send (MB)	Read (MB)	Write (MB)	
Full	Rdiff-B.	24	25	6	0	3	7	0	10.45
	Duplicity	24	30	90	0	3	7	0	5.27
	Areca	83	43	99	0	2	2	0	5.30
Incr	Rdiff-B.	22	5	44	0	0	0	0	10.99
	Duplicity	4	39	122	0	3	0	0	6.07
	Areca	11	44	128	0	1	0	0	6.12
Rest	Rdiff-B.	20	13	24	4	0	0	8	
	Duplicity	11	34	76	9	0	0	15	
	Areca	19	43	143	1	0	0	3	

Duplicity is the overall fastest tool with 24, 4, and 11 minutes for Full, Increment and Restore, respectively. Rdiff-backup is the best in terms of CPU usage, Duplicity is the best in terms of memory usage and Link-Backup the worse in both tests. Both network and disk usages have no clear winners.

Rdiff-backup does not compress files on the full backup operation, only the increments. As storage space is the most important aspect of compressing data, Rdiff-backup is not a really suitable tool for this kind of operation being its advantages only noticeable after large amounts of files modifications which this set of tests do not explore. The other two tools achieve the expected level of compression, but Duplicity is clearly the fastest of them all, being somewhat impressive that it can be faster the Rdiff-backup which has a simpler compression operation.

The overall best tool for usage with data compression is Duplicity due to its speed and to achieve the lowest storage space usage.

5.3 Experiments with Compression and Encryption

To test tools that allow compression and encryption we only have two tools: Duplicity and Areca. Table 3 shows the results of these two tools under the three different scenarios.

Tab. 3: Results using compression and encryption

		Time (min)	CPU (%)	Mem (MB)	Network		Disk		Size (GB)
					Recv (MB)	Send (MB)	Read (MB)	Write (MB)	
Full	Duplicity	24	36	95	0	3	7	0	5.30
	Areca	89	44	99	0	1	1	0	5.30
Incr	Duplicity	3	48	142	0	3	0	0	6.10
	Areca	12	45	120	0	1	0	0	6.12
Rest	Duplicity	16	45	80	6	0	0	10	
	Areca	21	44	140	1	0	0	2	

The results of Duplicity and Areca using encryption and compression are very similar to the tests results without encryption. Even so in CPU, memory, disk and network usage showing that due to the optimization used by these tools, encrypting of data is barely noticeable in terms of performance. As such the result is approximately the same as the tests with only compression. Duplicity is the best overall tool due to its speed and capabilities.

5.4 Comments on Methodology Approach

We know there are a couple of problems with the methodology used. Firstly, the data set used does not represent real world typical backed up files, neither the volume of data is similar. It was considered to show the results of real files instead of generated ones, but it was an objective of this test that the results could be reproduced by anyone. Therefore the BASH script used to run these tests is publicly available at ("Backup tools performance" 2014). The creation of the data set mimics several characteristics of real world data, such as number and size of files, their distribution on several directories being top level directories more likely of containing a larger number of sub-directories, and finally the file compression ratio of approximately 50% over a data set constituted by distinct file types. We also have carried out tests over real files and the results have similar characteristics to the ones shown in sections before, as such there is some degree of confidence that the generated data set is acceptable.

The tests were also realized in sequence with little time between each other (approximately 10 seconds). This poses a risk of the system being still processing the previous requests even after the program has finished, due to the use of many types of caches by modern hardware and operating systems. To mitigate this it was used the "sync" command to flush memory data into disk. Also the tests were executed in inverse order and there were no significant differences on test results.

Even though both client and server where running with the minimum of system processes possible, there is some interference done by those processes working on the background. The collected data show the overall system statistics and not the programs specific ones. A way to improve this test set would be to run the tests several times and calculate the average of the registered values.

6. Related Work

Although data backup is an important topic in computer industry, there are not many scientific papers in the area. For example, Chervenak et al. (1998) presents a good survey of backup techniques for protecting file systems and explain several of their characteristics. These include such choices as device-based or file-based backup schemes, full vs. incremental backups, and optional data compression. They also discuss techniques for on-line backup (backups performed while users continue to access the file system) and technique for protecting data from site disasters and media deterioration.

Most incremental backup methods do not support encryption because as encryption protects the file content, it may not allow the incremental backup algorithm to detect the parts of a file that have been changed since the last backup. Shih-Yu (2013) proposes a backup mechanism that supports encryption and incremental backup. After the encryption of a file, the encrypted data will be treated as normal data in the storage server, and the storage server will need to have the ability to perform only a few basic functions. When a user wants to restore a file, the system can use the checksum file to

retrieve the encrypted file from the server and decrypt it on the client side. In this way, the server does not need additional software for backup; it becomes a simple storage space because most of the computing is performed on the client side. In the above mentioned paper is presented the encrypted incremental backup method and the system.

The performance of the Rsync tool has already been analyzed by Mayhew (2001) on a comparison with the cfengine for file distribution for the purposes of system maintenance. This study focuses on two aspects of file distribution: the transfer during an initial copy from a master server to a client, and the file verification of the client's files against the server's files. It is concluded that Rsync is more efficient for larger files, while cfengine manages smaller transfers better.

The PRUN backup system (Won, Y. et al. 2008) shows an alternative way to reduce data storage requirements after several backups performed by reducing data redundancy. PRUN is a backup system for massive scale data storage with the aim to improve the backup latency and storage overhead of backup via effectively eliminating information redundancy in the files. PRUN consists of client module and server module with three key technical ingredients: redundancy detection, fingerprint manager, and chunk manager. However, the tests presented in this paper do not explore the performance of the tools after several backups performed, so this is an interesting subject to be studied as future work.

Zahed et al propose (Zahed, K. S. 2009) a technique on top of the Rsync algorithm, which can increase the storage capacity significantly by eliminating duplicate data in a file and avoiding duplication of information in storing file system backed up data. The proposed technique uses the signatures generated by Rsync to eliminate duplicate data in the file across backups thus helping in lesser storage costs. The technique has the potential to be extended across the entire data blocks of the file system for better efficiencies and can be made as the cheapest de-dup available in the field. The storage savings are very high for loads that do less percentage of overwrites and higher percentage of append writes. The simulation experiments on a developer environment and random number of create, delete and append operations on large directory structures show a minimum of 39% savings in storage space required.

However, and in contrast to the present paper, none of the cited authors make an effective comparison of several tools already being used in real scenarios.

7. Conclusions and Future Work

In this paper we evaluate the current popular solutions for backup of data. We demonstrate that there were more efficient tools that were not really used due to the fact that they appeared too late or for some reason they did not captivate the attention of users. Even if this paper does not exhaustively tests all the existent solutions it uses a group of tools with different characteristics from the simplest data replication tool to a fully featured backup suite. Rsync has shown to be an excellent tool if compression and encryption is not required, while Rdiff-backup has disappointed in the sense of not being able to improve the results of Rsync by being more focused backups. Link-backup has also shown that even using an interesting way of structuring data, in practice it is not able to be better than the traditional full mirror backup. It is possible that on a dataset with many repeated files it can show storage space gains, but that rarely happens on real world backed up data. Areca is also not able of doing better than simpler solutions, such as. Duplicity which has justified is current rapid increase of popularity being the best analyzed tool when compression or encryption is necessary.

We concluded that Rsync is the most efficient backup tool for simple data remote replication operations and Duplicity is the overall best of the backup tools capable of compression and encryption.

As future work we pretend to test the tools exhaustively in a real world environment using real data, in particular the tests should also be performed after several incremental backups to find the performance of these tools over continuous usage. We also pretend to improve the system resources usage method to get more accurate results. The chosen tools are only a small subset of current backup solutions available, as such, in the future we intend to also test more backup tools.

References

Areca Backup, 2013: [Online] Available at: http://www.areca-backup.org [Accessed 28 December 2013]

Backup Programs, 2013: ArchWiki [Online] Available at: https://wiki.archlinux.org/index.php/Backup_Programs [Accessed 17 January 2013]

Backup tools performance test script, 2014: [Online] Available at: https://github.com/aurhe/backup-tools-performance-test [Accessed 12 January 2014]

Chakraborty, D. & Mancillas-López, C., 2012: Double Ciphertext Mode: A Proposal for Secure Backup. *International Journal of Applied Cryptography*, Volume 2, Issue 3, pp. 271-287

Chervenak, A. L., Vellanki, V. & Kurmas, Z., 1998: Protecting File Systems: A Survey of Backup Techniques. In: *Joint NASA and IEEE Mass Storage Conference*

Crossler, R. E., 2010: Protection Motivation Theory: Understanding Determinants to Backing Up Personal Data. In: *43rd Hawaii International Conference on System Sciences*, pp. 1-10

Dstat: Versatile resource statistics tool, 2013: [Online] Available at: http://dag.wieers.com/home-made/dstat [Accessed 28 December 2013]

Duplicity, 2012: [Online] Available at: http://duplicity.nongnu.org [Accessed 28 December 2013]

Easy Automated Snapshot-Style Backups with Linux and Rsync, 2012: [Online] Available at: http://www.mikerubel.org/computers/rsync_snapshots/index.html [Accessed 28 December 2013]

Hutchinson, N. C., Manley, S., Federwisch, M., Harris, G., Hitz, D., Kleiman, S. & O'Malley, S., 1999: Logical vs. Physical File System Backup. In: *3rd Symposium on Operating Systems Design and Implementation*, pp. 239-249

Jallad, K., Katz, J. & Schneier, B., 2002. Implementation of Chosen-Ciphertext Attacks against PGP and GnuPG". In: *5th International Conference on Information Security (ISC '02)*, pp. 90-101

Kaminski, H. & Perry, M., 2007. *Open Source Software Licensing Patterns*. Computer Science Publications. Paper 10. Available at: http://ir.lib.uwo.ca/csdpub/10 [Accessed 28 December 2013]

Kothiyal, R., Tarasov, V., Sehgal, P. & Zadok E., 2009. Energy and Performance Evaluation of Lossless File Data Compression on Server Systems. In: *Israeli Experimental Systems Conference (ACM SYSTOR '09)*

Link-Backup, 2013: [Online] Available at: http://www.scottlu.com/Content/Link-Backup.html [Accessed 28 December 2013]

Ludwig, S. & Kalfa, W., 2001: File System Encryption with Integrated User Management. *ACM SIGOPS Operating Systems Review*, Volume 35, Issue 4, pp. 88-93

Machanavajjhala, A. & Reiter, J. P., 2012: Big privacy: protecting confidentiality in big data. *ACM Crossroads XRDS*, Volume 19, Issue 1, pp. 20-23

Mayhew, A., 2001: File Distribution Efficiencies: cfengine vs. Rsync. In: *15th USENIX conference on System administration (LISA '01*), pp. 273-276

Platoš, J., Snášel, V. & El-Qawasmeh, E., 2008: Compression of small text files. *Advanced Engineering Informatics*, Volume 22, Issue 3, pp. 410-417

Qian, C., Huang, Y., Zhao, X. & Nakagawa, T., 2010: Optimal Backup Interval for a Database System with Full and Periodic Incremental Backup. *Journal Of Computers*, Vol. 5, No. 4, April 2010, pp. 557-564

Rdiff-backup, 2013: [Online] Available at: http://rdiff-backup.nongnu.org [Accessed 28 December 2013]

Rsync, 2013: [Online] Available at: http://rsync.samba.org [Accessed 28 December 2013]

Shakhanova, M. V., 2013: Optimization of Protection in Automated Data Transmission and Processing Systems. *Automatic Control and Computer Sciences*, Volume 47, No. 3, pp. 139-146

Shih-Yu Lu., 2013. Encrypted Incremental Backup without Server-Side Software. In: 27th International Conference on Advanced Information Networking and Applications Workshops, pp. 1467-1472

The GNU Privacy Guard, 2013: [Online] Available at: http://www.gnupg.org [Accessed 28 December 2013]

Tridgell, A., 1999: *Efficient Algorithms for Sorting and Synchronization*. Available at: http://www.samba.org/~tridge/phd_thesis.pdf [Accessed 28 December 2013]

Tridgell, A. & Mackerras, P., 1996: The rsync algorithm. *Joint Computer Science Technical Report Series*

Won, Y., Kim, R., Ban, J., Hur, J., Oh, S. & Lee, J., 2008: PRUN : Eliminating Information Redundancy for Large Scale Data Backup System. In: *2008 International Conference on Computational Sciences and Its Applications (ICCSA '08)*, pp. 139-144

Ylönen, T., 1996: SSH - Secure Login Connections over the Internet. In: 6th *conference on USENIX Security Symposium, Focusing on Applications of Cryptography* - Volume 6 (SSYM'96)

Zahed, K. S., Rani, P. S., Saradhi, U. V. & Potluri, A., 2009: Reducing Storage Requirements of Snapshot Backups based on rsync utility. *Communication Systems and Networks and Workshops*, 2009

JEL Classification: M15