GIT Server Performance Optimization Using Git-Annex

Meor Nur Hasyim Meor Aziz, Aznida Hayati Zakaria @ Mohamad, Mohd Fadzil Abdul Kadir, Hasni Hassan, Rohana Ismail

Faculty of Informatics & Computing, Universiti Sultan Zainal Abidin, Besut Campus, 22200 Besut, Terengganu, Malaysia

ABSTRACT

This project is to implement git-annex to improve the performance optimization of a git server. When a git server have too many files in the repository, it performance will be affected. In response, this will lead to slow productivity of developer's team that are utilizing the git server. By studying concepts of version control software and building one, it could help understanding the complexity of the version control software environment. This project wish could help the users that are utilizing version control software to implement git annex to improve their server performance. The result show that git-annex drastically improves performance not only the server, but also to the local itself.

Keywords : *Git-Annex, Git Server, Optimization.*

I. INTRODUCTION

Git is a type of version control software that has the ability to manage the modification and configuration of an application and also keep tracks of them [1]. Version control is important to software developers because developers can compare files, identify differences, and merge the changes if needed. When the troubleshooting of the code has an issue, developers can compare the last working file with the faulty one, decreasing the time spent identifying the cause of the issue. There are many types of version control software such as Mercurial, Git and Apache Subversion [2].

Git-annex is a distributed file synchronization that aims to solve the problem of sharing and synchronization collections of large files [3][19]. It is installed in a git server to help and improve performance slowdown that is caused by large file storage stored in the server. Git-annex runs in the background to automate the synchronization of repositories.

II. RELATED WORKS

A. WordPress Based on Git Implementation

This article presents preparations and good practices for independent web development [4]. The article does not have a client, but the end product of the article can be used for assignments by actual clients. The aim of the article is to prepare two computers as web servers, and to present version control software Git. With Git, the article should cover most basic usage and operations, mainl relating to web development. At the end of the article, web servers meant to be used in development are ready for use and Git version control is in use. Git version control includes clean installation of content management system WordPress. GitHub is used as the service provider for the Git server. Version controlled WordPress base can be moved on any Debian based web server, which fulfils the requirements of WordPress. By following the instructions of this article, creation of a web server should be possible, as well as use of Git. This article leaves room for further development. The article could be continued by creating the actual website, for which the preparations are made for, and by automating some of the actions presented in the article

B. Version Control Software using Virtual Machine

The objective of this article is to design and implement an improved control version software server [5]. A virtual machine is installed as the basic setup for the project. Also mention that using virtual machine, it is possible to create technically difficult and advanced setups without interfering with the current server environment. This is because the virtual machine allows to develop the server environment in stages and it can be tested outside the server environment. This article also mentions that virtual machine use plenty of system resources which may result in problems later. Alternatively, one may use application container to optimize system resources aside from virtual machine

C. Types of Version Control Software

This article discusses three type of version control software. The version control software involves in the article is Mercurial, Git, and Apache Subversion (SVN) [2]. The article further discusses commands used in each type of version control software. Also mentioned in the article that Git is a distributed revision control and source code management system with an emphasis on

speed. Also there are explanation to Git basic terminologies like trees, commits, branches, clone, pull, push and revision. Regularly individuals get confounded thinking Git and Github are one and the same; however, there exists real contrasts amongst them. In spite of the fact that you can run your own particular Git server locally, Github is a remote server, a group of designers, and a graphical web interface for dealing with your Git project. It's free to use for up to 5 public repositories. GitHub has as of late settled Git as a great version control system, giving a wonderful front end to numerous substantial projects, for example, Rails and Prototype.

D. Git Server Practices

This article research mentioned examples of best practices for a git server to run efficiently [6]. The best practices are some general advice for effective software development using version control. One of them is to explain commits completely. Every version control tool provides a way to include a log message (a comment) when committing changes to the repository. This comment is important. If we consistently use good comments when we commit, our repository's history contains not only every change we have ever made, but it also contains an explanation of why those changes happened. These kinds of records can be invaluable later as we forget things. Other practices include to build and test the code after every commit

E. Git-Annex

This article mentioned tool called git-annex that was developed to address some problems of storing large files in git repositories and allow for controlled transfers of restricted datasets [7]. This tool uses a storage format similar to git, but it keeps the large object store separately from the regular git-object store. The storage format of git-annex has been modified to allow handling of very large files on the order of multiple terabytes. However, the git-annex tools still maintain full metadata related to the large files in the main git repository so git still can record all manipulations done to these large filefiless as well as maintain the integrity of the repository through the use of checksums. This is achieved by git-annex checking-in the links to the hashes of the large files into the git repository. Such scheme also allows research groups at different organizations to share the code used for processing the datasets without sharing the datasets themselves, and at the same time both parties can obtain the same dataset from the proper authority and independently inject it into their repository. All of this could be achieved through standard git commands without tedious and error prone verifying of long checksums

III. METHODOLOGY

A. Project Framework

The project will have to undergo different processes so that the analysis at the end of the project can be done. So framework need to be designed in order to see and understand the flow of the process ensuring it can be done as proposed.

The methodology framework is divided into 3 main phase:

a) Phase 1: Raspberry Pi Setup
b) Phase 2: Git Server Performance Testing Process
c) Phase 3: Result Comparison and Analysis

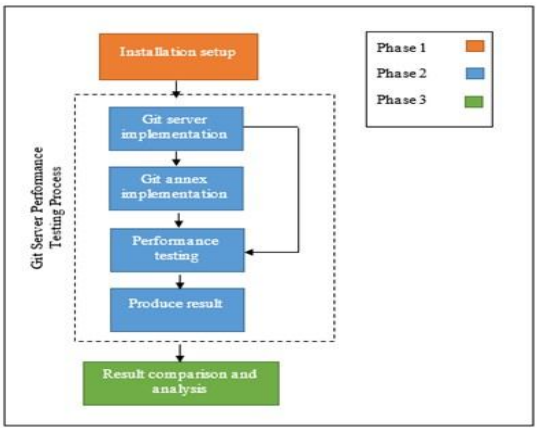

Figure 1 : Project Framework

Phase 1: Installation Setup

Installation setup is the first phase of the project. Raspbian OS will be installed as the Raspberry Pi operating system. Git will be installed in the server and local machine. Git-annex also will be installed in the server and the local machine.

Phase 2: Git Server Performance Testing Process

The phase is proceeded with testing the performance test. There are two varied performance test for the git server. First performance test is to look at the git server without implementing git-annex. The performance test is based on the time taken of the standard git and also the git-annex setup to upload and download files from the server.

The result of the performance test will be kept and recorded. The second test is to look at performance of the git server with implemented git-annex. Git-annex is installed while keeping the same Raspbian operating system, and system setting like previous test. The git-annex implemented git server will undergo the same performance test. Result will be kept and recorded for the use of Phase 3. Both test will be run.

5 times with 5 different file sizes until average values is obtained. This is to ensure the consistency of the result of the performance test.

Phase 3: Result Comparison and Analysis

In this phase, result of the performance of git-annex implemented and without it will be collected for the project analysis. In the analysis phase,both result of the performance test will be compared. In will be the deciding factor to conclude that whether git-annex can improve the performance of the git server or not

B. Test Environment Setup

The setup of the project consists of only one Raspberry Pi 3 Model B and a laptop. The Raspberry Pi will become the server while the laptop will act as the local machine. For the server, there are two interchangeable mode (with and without git-annex). The git repository in the server needs to be deleted in order to execute the next test.

For the local machine, two variants of git repository, one without the git-annex, and another with the git-annex will be installed in two separate Linux Ubuntu in the same VMware Pro Workstation.

The two Linux Ubuntu on the VMware Pro Workstation need the same parameter of setup to ensure the consistency of both results from the same type of test.

The parameter setup of each of Linux Ubuntu is fixed as below:

1. 3GB of RAM
2. 20GB of single disk storage
3. Same location of data storage in the SSD of the laptop.
4. 4 numbers of processors allocated.
5. 1 core of processor of the laptop allocated to the VMware

Pro Workstation.

Most of the parameter of the Linux Ubuntu needs to be considered since performance of the git system is heavily depended on the hardware of it. The results of both test may be varying if we use different parameter of hardware for Linux Ubuntu. So, it is worth noting the parameter setup since the results will be referred back to them.

Since we only use one Raspberry Pi with two interchangeable usages, we do not need to consider the hardware parameter. The most important thing for the server is delete all git repository data related on the server whenever the next test will be performed

C. Performance Test Workflow

There will be two types of test involving this project. The two main types of test are:

1. Pushing test - where files will be added into the git repository and then upload them to the git repository server.
2. Pulling test - where files from the server that have been uploaded will be downloaded back to the local git repository

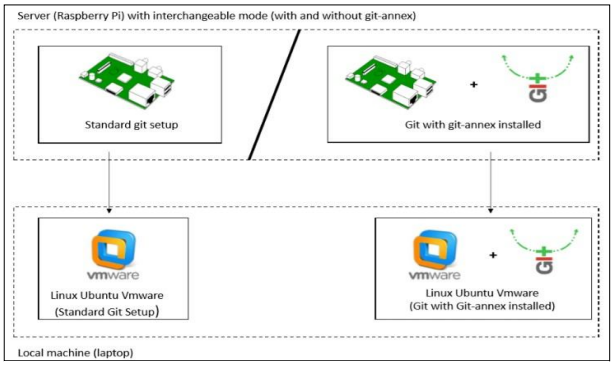

Figure 2: Performance test workflow

Both test will examine how fast a file will be written from the source to the destination. The time taken for each test to be completed will be recorded for the results analysis and comparison. Since the first results of pushing may be varying from the second one, we will perform the test five times to get an average value of the time taken to complete each test.

Each test will use different sets of command. Both variant standard git setup and git-annex setup will undergo these two test. This means there are four types of performance test specifically.

The four specific test are:

1. Standard git pushing test
2. Standard git pulling test
3. Git-annex pushing test
4. Git-annex pulling test

The project needs a specified fixed sets of files that will undergo the same progress across all the test performed. For the consistency of the results, five different sizes of files will be chosen for the test. The five different sizes of files are needed to show how different variant sizes of files will give impact on the performance of the git repository.

The five different sizes of files are:

1. 10MB
2. 20MB
3. 50MB
4. 100MB
5. 200MB

The five different sizes of files will undergo the same test whether it be the standard git setup or the git annex setup. So, a workflow framework will be developed to give an overview how the test for the files to be carried out across the project

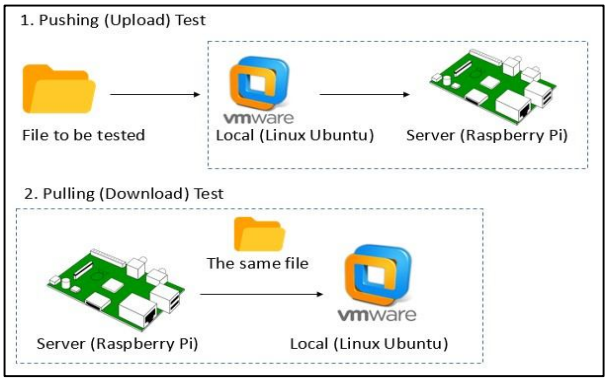

Figure 3: Test workflow

IV. RESULTS

Figure 4 shows git-annex setup takes shorter time to complete the pushing test compared to standard git setup.

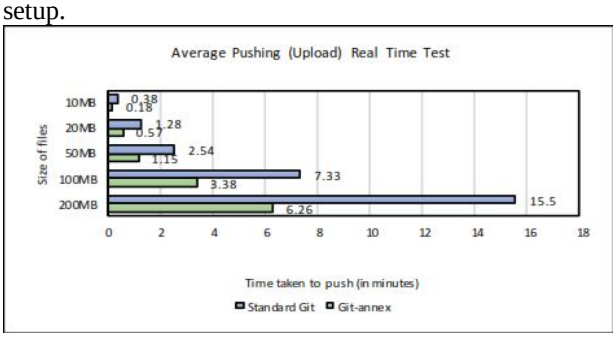

Figure 4: Average pushing real time test

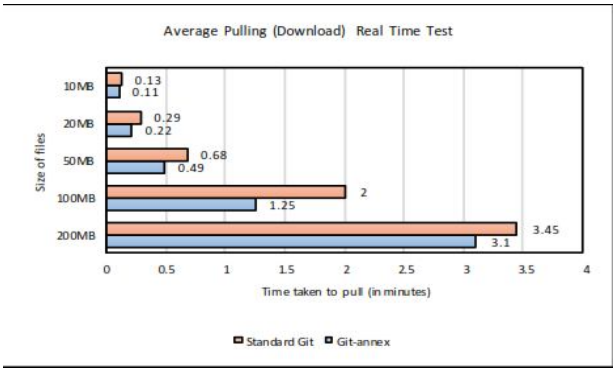

Figure 5: Average pulling CPU time test

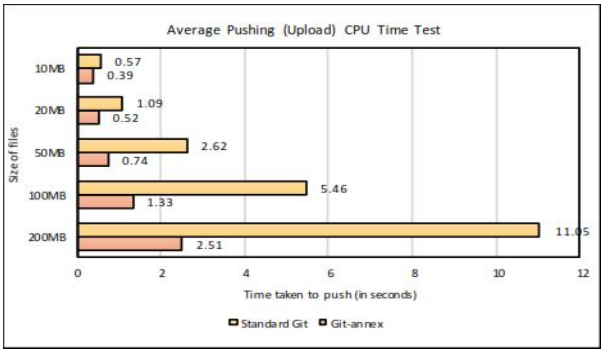

Figure 6: Average pushing CPU time test

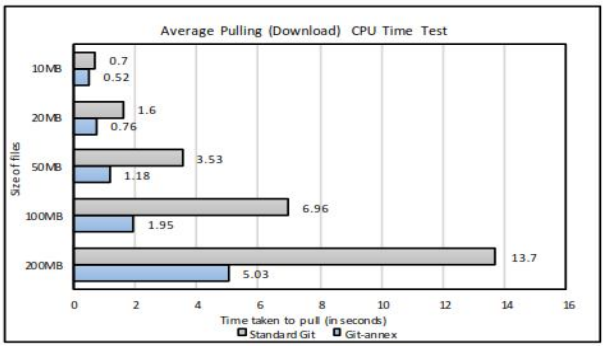

Figure 7: Average pulling real time test

Figure 5 shows graph comparison where git-annex setup takes shorter time to complete the pulling test compared to standard git setup. Figure 6 shows the average pushing CPU time test using git-annex setup and standard git setup. The figure shows the comparison where git-annex setup takes shorter time to complete the pushing test compared to standard git setup in CPU time test. The CPU time is the value of user and system are summed up to have a total of actual CPU time.

The results show that git-annex drastically improves performance not only the server, but to the local itself. The way git-annex consistently take less time to complete the test proving that git-annex is the answer of improving performance of the git environment. The impact of git-annex implementation can be seen in a bigger files or repository, meaning that small size of files or repository have small negligible improvement. This is proven by test from the 10MB file size which only speedup the time in small amount of time. Logically, most of the projects developed by a team of developer are a big one and worth hundreds gigabyte of size. By implementing git-annex, not only the git environment can be improved, the developers also can increase more productivity to work on new code since less time spend in pushing and pulling from the server.

Git-annex also reduce the workload of CPU processing. Looking at the graph result, we can clearly see CPU spend less time in executing command when git-annex is implemented. There are many advantages in this matter. Firstly, if only less CPU time spend for the processing, the probabilities of the CPU to go very hot and throttle is very unlikely. This leads to a better maintenance of hardware that hosts the CPU. A throttle CPU will only pissing the users off since the CPU will have to lower its clock speed so that the temperature will cool down. This will make the CPU have to work in a slower than it should be.

V. CONCLUSION

Finally, even though the same file size is used to push and. pull across the test, pulling test always have a

quicker time to be completed compared to pushing test. The big factor contributing to this anomaly is that there is a vast different of write and speed between the server and local. In the server, when the file is being pushed, the speed of writing the file is based on the USB drive on the Raspberry Pi [18]. In contrary, when the file is being pulled, the file is being written to an SSD of the local machine. Since we are comparing the speed of standard git setup versus git-annex setup; not a pushing versus pulling test, this information is less relevant for the overall project. It is worth noted since the time taken for both pushing and pulling test is biased towards pulling test.

This project has more room to be improved and more potential to be polished for future work and research. Firstly, a working git server should have good specification from the hardware perspective. Even though this project only uses a Raspberry Pi to host the git server and working just fine, it still has problems. This includes the storage speed issue. Raspberry Pi is fine when is being used for personal use and also to simulate projects at a smaller scale. A git server needs proper physical host to store all the data and also to make sure the temperature stays in control every time.

Next, simulation of this test can be done using a real Linux machine. This project only uses a virtual machine to mimic the git repository in a local machine. The disadvantages of using virtual machine includes less accurate of measuring time taken during the test and also slower reaction of time when using the virtual machine. As virtual machine needs hypervisor that acts as a bridge between the operating system of the physical hardware and the operating system of the virtual machine, the CPU has to consider the hypervisor every time making the operating system in the virtual machine can be felt a little slower.

Another recommendation is to monitor properly the temperature of the git server's physical hardware. One of the thing that plays a big part in the test results is the temperature of the USB drive in the Raspberry Pi. The time taken to complete the test is noticeably longer when the USB drive is a little hotter than usual. This can be felt while doing the pushing test since we are writing the files onto the USB drive

ACKNOWLEDGEMENT

We would like to thank Universiti Sultan Zainal Abidin for the financial support to this project under the University Research Fund Grant (UNISZA/2017/DPU/75).

REFERENCES

[1] What is Version Control [Online]. Available: https://www.atlassian.com/git/tutorials/what-is-version-control

[2] Aishwarya Nair, Meenakshi Dhanani, Rupesh Gangwani, Prof. Hema Gaikwad. "*Comparison of Software Configuration Management Tools*".

[3] Git-annex [Online]. Available: https://en.wikipedia.org/wiki/Git-annex

[4] Toni Vaakanainen. (2016). "*Preparation of Services to Essential Web Development.*

[5] *Riku Ojala. (2016). Version Control System, Designing and Implementing Server Infrastructure*".

[6] Eric Sink. (2011). "*Version Control by Example*".

[7] Vlad Korolev, Anupam Joshi. (2014). "*PROB: A tool for Tracking Provenance and Reproducibility of Big Data Experiments*".

[8] Git-annex Walkthrough [Online]. Available: https://git-annex.branchable.com/walkthrough/#index3h2

[9] Git Commands Tutorial [Online]. Available: https://www.siteground.com/tutorials/git/commands/

[10] Lee Hinman. (2016). "*Getting Started with Git-annex.*"

[11] Git-annex On Your Own Server [Online]. Available: https://gitannex.branchable.com/tips/centralized_git_repository_tutorial/on_your_own_server/

[12] Git-annex, How it Works [Online]. Available: https://git-annex.branchable.com/how_it_works/

[13] Scott Killdall. (2015) Gitpi: A Private Git Server on Raspberry Pi. [Online]. Available: https://www.instructables.com/id/GitPi-A-Private-Git-Server-on-Raspberry-Pi

[14] Thomas Loughlin. (2012). GitPi: Using your Raspberry Pi as a Dedicated Gi Repository [Online]. Available: http://thomasloughlin.com/gitpi-using-your-raspberry-pi-as-a-dedicated-git-repository/

[15] Perforce.com. (2017). "*Storing Large Binary Files in Git Repositories*" [Online]. Available: https://www.perforce.com/blog/storing-large-binary-files-in-git-repositories

[16] Vitaly Emporopulo(2017).Git Performance Benchmark. [Online]. Available: https://open-amdocs.github.io/git-performance-benchmark

[17] Harlo Holm. (2014). So, I'm Excited about Git-Annex. [Online].Available:http://harloholm.es/2014/04/16/so_im_excited_about_git-annex.html

[18] Nur Haziq Mohd Safri, M., Nor Shuhadah Wan Nik, W., Mohamad, Z., & Mohamad, M. (2018). "*Wireless Network Traffic Analysis and Troubleshooting using Raspberry Pi.*" International Journal of Engineering & Technology, 7 (2.15), 58-60.

[19] Awang, W. S. W., Deris, M. M., Rana, O. F., Zarina, M., Rose, A.N.M. (2019). "*Affinity replica selection in distributed systems*". Lecture Notes in Computer Science, 11657 LNCS, pp. *385-399*.

[20] Mjlae, S. A., Mohamad, Z., & Suryani, W. (2019). "*Impact factors of IT flexibility within cloud technology on various aspects of IT effectiveness*". International Journal of Advanced Computer Science and Applications, 10(4), 479 – 489.