Performance Evaluation of Netsync Protocol in Monotone, A Software Configuration Management Tool

Hu Kwong Liik  
Faculty of Computer Science and Information Technology  
Universiti Malaysia Sarawak  
94300 Kota Samarahan  
Sarawak, Malaysia  
E-mail: kwongliik@yahoo.com

Johari Abdullah  
Faculty of Computer Science and Information Technology  
Universiti Malaysia Sarawak  
94300 Kota Samarahan  
Sarawak, Malaysia  
E-mail: ajohari@fit.unimas.my

Abstract: Netsync protocol is a bi-directional pipeline protocol used in Monotone for synchronizing distributed repositories in a peer-to-peer architecture. The objectives of this paper are to evaluate netsync performance and to identify its performance evaluation metrics. From the analysis, netsync’s performance is dependent on synchronization entities such as manifests, certificates, and file data/deltas that need to be exchanged during the synchronization process and the time taken to search for missing pieces. Besides, protocol design is another important factors that determined how fast netsync is able to search for missing pieces (how many communication bytes need to be performed) and how it is going to build the merkle tree (from scratch or incremental). Netsync’s performance degrades when there are more manifests and certificates involved (meaning larger merkle tree). This increases the searching time for missing pieces. BitKeeper performs better than Monotone due to the different synchronization entity and networking protocol used. The influence of changes in LAN traffic has posed no significant effect on the results of Monotone and BitKeeper. As a conclusion, there is a possibility of improvement for netsync protocol design and it can become a potential networking protocol used by distributed SCM tools especially where developers need to update their repositories frequently.

Keywords: netsync, peer-to-peer (P2P), software configuration management (SCM), merkle tree, manifest, certificate, Monotone, BitKeeper.

As a development support tool, it helps developers in maintaining consistency between independent software components, record the composition of versioned software products evolving into many revisions, build derived objects (compiled from source code), and constructing a new configuration based on the software properties [7]. SCM started in the late of 1960s. In 1970s, the American government developed a number of military standards, which included SCM. Later, especially in 1990s, many other standards and publications discussing SCM have emerged. In recent years, SCM gained more attention followed by the success of the open source project, such as Linux operating system, Apache web server, and Mozilla browser.

A current de-facto SCM is the concurrent versions system (CVS) [17]. It has become a preferred configuration management (CM) used by open source developers to version their source code and maintain differences between revisions (called deltas). CVS helps to coordinate the development of software product along its life cycle by reporting its status, statistic of the changes made, audit the product, and ensure the organization’s policy is followed [9]. It allows read-only access to a repository by anonymous users [13].

We organized this paper by discussing the literature reviews in Section 2, problem statements in Section 3, objectives in Section 4, methodology in Section 5, results and discussions in Section 6, and conclusions in Section 7.

1. Introduction

Monotone is an open source software configuration management (SCM) tool. SCM is used as management support and development support tool. As a management support tool, it helps to identify product component and their versions, change control (follow a strict procedure when performing a change), status accounting (recording and reporting the status of components and change requests), audit and review (quality assurance functions to preserve product consistency) during the life cycle of a system [7,15].

CVS is a centralized SCM based on client-server architecture [6]. Hence, it is categorized as network-oriented SCM. The advantages of CVS are that it provided network-transparent access of version control and allow us to access the source code from anywhere as long as there is Internet connectivity. The disadvantages of CVS are there is a possibility of single failure, which will cause developers to be unable to access the repository, and also bottlenecks occur when there is a lot of requests from developers. CVS is