Data-Driven Storage Optimization for Digital Repositories Using Big Data Techniques

Dr. Mariana Oliveira, Dr. Stefan Keller, Dr. Lucia Fernandez
Department of Environmental Science, University of São Paulo, Brazil;
Institute of Geophysics, ETH Zurich, Switzerland;
Faculty of Earth Sciences, National Autonomous University of Mexico (UNAM), Mexico

ABSTRACT

Software Configuration Management (SCM) deals with various changes and evolution in the software. Each software comprises of thousands of versions. Individual versions need to be stored again and again. Every software keeps on evolving so we need to keep track on each evolution. Software engineer uses mining techniques to store and retrieve these kinds of data's. This research paper deals with the design, and implementation of an efficient storage management for SCM repositories that facilitates a developer's to store revisions of software changes using Map reduce Techniques. The main objective of this research work is at making the storing and retrieval process of SCM repositories easier. Storage of SCM repository keeps on increasing. SCM repository needs a processing technique to process those data before storing. Map reduce Technique is used to process those data in Divide and Conquer manner. It stores source code in the format of the graph so storing and retrieval process is much easier. It, in turn, reduces the storage whole SCM repository. Thus, an efficient storage system for SCM repositories is achieved and a prototype is discussed in this paper.

Key words: *SCM Mining, Repository, Big data, Divide and Conquer.*

I. INTRODUCTION

Storage management system processes the data and stored for future retrieval. This system evolves through various stages, now developers are using software mining techniques for effective storage [1]. It is effective but it can't handle large number of data. In order to make process effective we use map reduce techniques which process data in divide and conquer manner. The existing system of storing and retrieval in SCM is performed by Versioning and Base lining it needs lot of storage to store the coding.

Daniel Rodriguez, et al [2] presented a survey of the publicly available repositories and classify the most common ones as well as discussing the problems faced by researchers when applying machine learning or statistical techniques to them. Micheal Muller, et al [3] presented thesis is to design and implement a front-end plug-in for an existing software comprehension tool, the VIZZANALYZER, providing the capability to extract and analyze multiple versions and evolutional information of software systems from SCM repositories and to store the results. Thereby, the implemented solution provides the infrastructure for software evolution research.

Filip Van Rysselberghe et al [4] makes an attempt to turn the software maintenance craft into a more disciplined activity, by mining for frequently applied changes in a version control system.

II. BACKGROUND STUDY

The Research and approaches that primarily examine a single version or release of a software system are excluded from this survey, as they typically do not directly address the issues of software evolution and change. For example, we felt work that focused on analyzing a single version, and just happened to use a data-mining technique for analysis, is not within the scope of this project. This type of investigation is research on analysis methods to support testing (or some other software engineering task) [6]. In other words, this is not a project for investigations applying data-mining techniques to software engineering problems, but rather a survey of investigations that examine the changes and evolution of software and use data mining and other similar techniques. In a very few cases, we have included work that presented techniques that could readily be applied to multiple versions but was only applied to a single version.

These are included for completeness and typically represent important contributions to the study of software repositories. Rather than data mining for storing and retrieval purpose we able to new techniques such as Map-Reduce Technique.

III. PROPOSED METHODOLOGY

System architecture is the conceptual model that defines the structure and/or behavior of the system. It provides a way in which products can be procured, systems can be developed an architectural overview of the overall system. The architecture diagram tells about the rough design of the implementation it helps us know how the system will react to the inputs, it shows concept how the system behaves for each and every input.

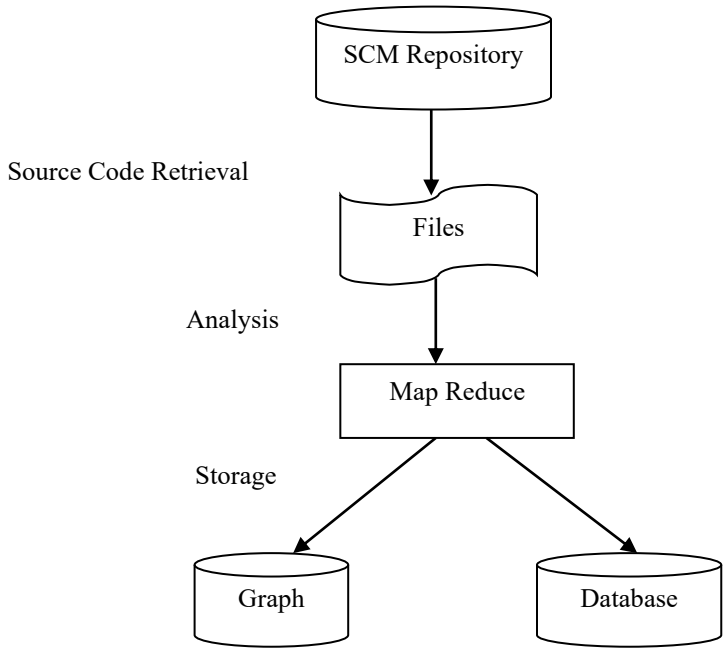

Figure 1. Proposed System Architecture

Map Reduce

MapReduce is a framework used for implementation of processing and generating large data sets with a parallel, distributed clustering algorithm on a cluster [5].

- Search through large datasets
- Map: extract the associative data
- Shuffle the data and clusters
- Reduce the output
- Generate results

Prototypical Map reduce Example

Input: a set of large data sets
Apply two functions:
Map (s,p) → list (s1,p1)
Reduce (s1, list (p1)) → p2
(s1, p1) is an intermediate result
Output is the set of (s1, p2) sets

IV. CONCLUSION

The primary goal of this research work is to reduce the size and storage process needed to store source code in the SCM repository. In this phase for each versions code have been appended to produce new ones. This process enables developer to store without repetition. Map- Reduce is the best programming model for processing large datasets. It has different level of optimization it makes storing and retrieving process efficient.

REFERENCE

1. *Van Rysselberghe F, Demeyer S. Mining version control systems for FACs (frequently applied changes). Proceedings 1stInternational Workshop on Mining Software Repositories (MSR'04). IEE: Stevenage, 2004; pp 48–52.*
2. *Zimmermann T, Zeller A,Weigerber P, Diehl S. Mining version histories to guide software changes. IEEE Transactions on Software Engineering 2005; pp 429–445.*
3. *Ying ATT, Murphy GC, Ng R, Chu-Carroll MC. Predicting source code changes by mining change history. IEEE Transactions on Software Engineering 2004; pp 574–586.*
4. *Livshits B, Zimmermann T. DynaMine: Finding common error patterns by mining software revision histories. Proceedings 13th International Symposium on Foundations of Software Engineering (ESEC/FSE'05). ACM Press: New York NY, 2005; pp 296–305.*
5. *Kagdi H, Yusuf S, Maletic JI. Mining sequences of changed-files from version histories. Proceedings 3rd International Workshop on Changes of softwares*
6. *Robles G, Gonz´alez-Barahona JM, Michlmayr M, Amor JJ. Mining large software compilations over time: Another perspective of software evolution. Proceedings 3rd International Workshop on Mining Software Repositories (MSR'06). ACM Press: New York NY, 2006; pp 3–9.*