<u>**CS550 ADVANCED OPERATING SYSTEMS**</u>

PROJECT PROPOSAL

Comparison of various Indexing techniques for distributed file system/object storage

Vaibhav Uday Hongal
MS Computer Science
A20378220

Vivek Bajpai
MS Computer Science
A20361204

INTRODUCTION

Background

Keeping the unstructured data available safe and integrated, as the data sets continue to grow in today's world, is a key challenge for today's data centers. With a massive increase in the data to be stored, a distributed approach of storing data using distributed file systems (HDFS, Microsoft DFS, Novell Netware, Microsystem's NFS etc.) or distributed object storage (IBM COS, Amazon S3, Microsoft's Azure storage etc.) became more and more popular. Only raw data distributed among multiple machines is of no use and one need to perform different operations like search, listing, prefixed listing, ranged reads, ranged writes etc. on it. Generally, an index is maintained (either centralized or distributed) for the data which provides quick access to it and reduces the time required for these operations drastically. The design of this index is very crucial to the overall performance of the system and hence there are various well known data structures used for maintaining it. We plan to compare the performance of different commonly used data structures and propose the use of B-link trees [1] for the same.

Motivation

There are many readymade solutions which are used for indexing of the file data in the DFS or OS, but most of them are not tailor made for this purpose. Using a relational database is one of the option. Although it is best suited for searching and gives very good read performance, the updates are pretty expensive, as relational databases are designed using OLTP in mind, which reduces performance. Distributed NOSQL databases like Mongo DB and Couch preferred to support availability, partition tolerance, and performance over consistency, which can't be tolerated for our use case. Traditional distributed hash tables are another common method for such an index, but does not make optimum use of principal of data locality. B-trees helps us overcome that, but it again follows a "lock and write" approach and hence does not allow concurrent operations, eventually reducing the performance. B-link[1] tree is a newly introduced data structure which can be used for this purpose. It provides us best of both worlds (concurrent operations and cheap contiguous access). In this paper we will analyze the performance of different already available data structure for indexing and compare them with a custom implementation of B-link tree, on various parameters.

Project Goals

The project aims to achieve the following:

1. Implementation of B-link trees for indexing in a Distributed file system or Object storage, using Java.
2. In-depth comparison of different techniques used for indexing with each other and B-link trees on various parameters. We are planning to use either a distributed file system or Object storage for our testing, depending on whichever is feasible.
3. Concluding on whether B-link trees are a good alternative for indexing, and in which cases. Complete performance analysis of it on various parameters using real world workflows.

The performance will be measured for following operations:

a. Searching
b. Reads
c. Contiguous reads (reading files present in contiguous locations)
d. writes
e. Contiguous writes (writing to files present in contiguous locations)
f. Prefixed listing
g. Adding a new file (object)
h. removing a file (object)

Performance will be tested for different file/object sizes ranging from very small to very large.

PROJECT EXECUTION PLAN

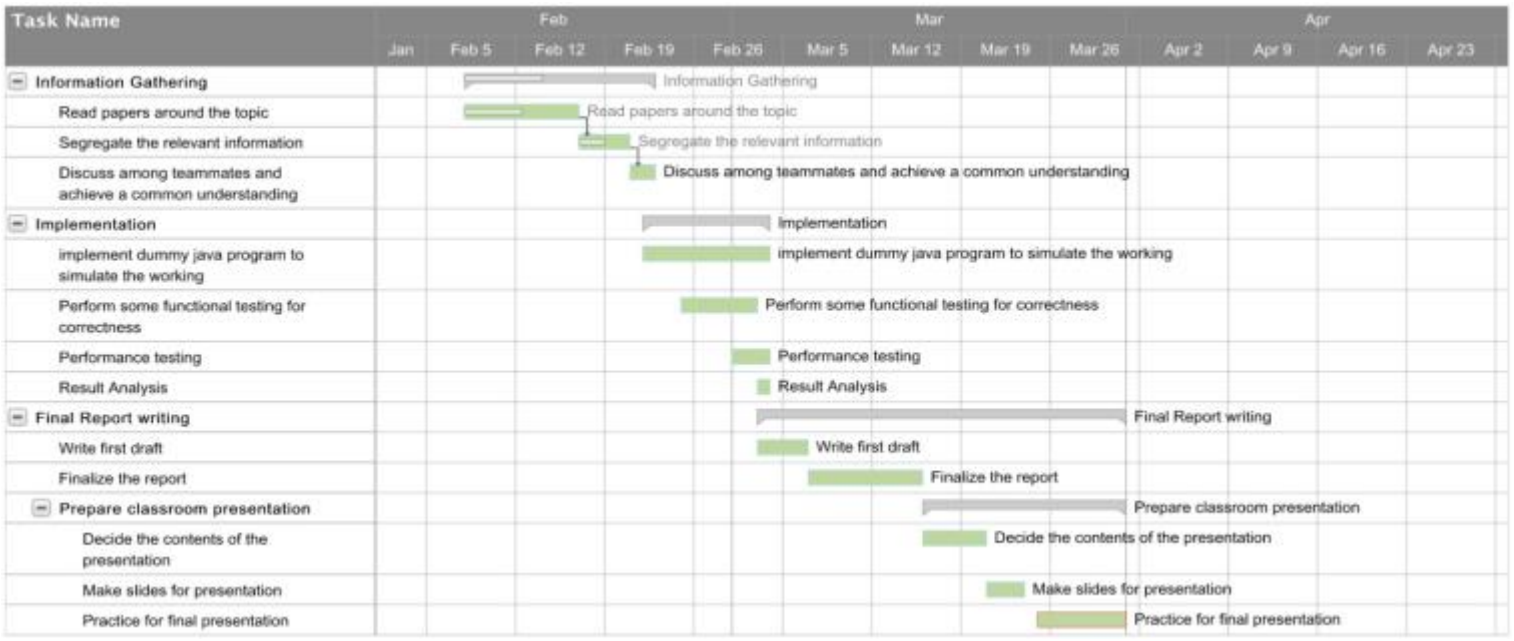

Fig. 1: Project plan chart

The above figure display only a rough draft of the execution plan. As the project progresses, certain external factor such as exams and other subject projects might influence the above schedule. Constant reviews and planning will be done to make sure the project proceeds as planned and achieve the project goals to the maximum possible extent.

REFERENCES

[1] Philip L Lehman and S. Bing Yao, **Efficient Locking for Concurrent Operations on B-Trees,** ACM Transactions on Database Systems, Dec 1981.

[2] Itua Ijagbone, **Scalable Indexing and Searching on Distributed File Systems***,* May 2016.

[3] Andrew W Leung, **Organizing, Indexing and Searching Large scale file systems,** Technical report UCSC-SSRC 09-09 December 2009.

[4] Jerome H. Saltzer, **File System Indexing and Backup,** Laboratory Of Computer Science, MIT.