CS87 Midway Progress Report: Deduplication on Distributed File System

Keonwoo Oh, Sam Rothstein, Ford Johnstone
Computer Science Department, Swarthmore College, Swarthmore, PA 19081

July 26, 2020

1 Project Schedule/Milestones

- Week 1 (04/06-04/12)

  Original Plan: Finish implementing the client side API. Thoroughly test the functions using "dummy" master node and data node.

  Updated Progress: We finished implementing and incrementally tested the ClientNode class and the necessary data structure classes (FileDescriptor and Node) and protocols (Client) for coordination and communication between the client application and the distributed file system. The system supports four basic functions, including open, close, read, and write. We implemented and tested the essential data structure classes (SecureMap, Inode, and FileDirectory) for the MasterNode class, which is the process that executes on the master node of the distributed file system. Then, we finished implementing the master node functionality for open, close, read and write operations. The master node process consists of Client objects, which are threads that each handles requests of each client node, and data structures for the file system metadata, including the FileDirectory, BlockMap, and GroupMap.

- Week 2 (04/13-04/19)

  Original Plan: Finish implementing the master node functionality. Thoroughly test the functions using already implemented client side API and dummy data node.

  Updated Progress: We implemented the library (ClientProg) the client application has to include in order to access the distributed file system, including the FileObject, which handles the requests of the client application by communicating with ClientNode process through TCP socket and shared memory (for buffer) and FileDirectory, a data structure to keep track of files the client application is accessing. Then, we tested and debugged open and close functions by executing sample applications on top of the system built so far, including ClientNode, ClientProg, and MasterNode. Most of the protocols were already implemented and by now, we had a pretty good idea of how they should operate. So, we simply proceeded to complete building the entire system by implementing the DataNode class, which is the process that executes on each data node and establishes read and write data streams with clients and stores or retrieves data blocks upon request. We have not yet implemented a hash algorithm for generating blockIDs so we tested and debugged the whole system using a dummy hash function that returns the same string for any input data block. After implementing SHA1

hash function, we incorporated it into the system and extensively tested and debugged the entire file system using sample programs. We, then, discussed some logistics for setting up experiments on Swarthmore CS lab machines with Professor Knerr.

- Week 3 (04/20-04/26)

  Original Plan: Finish implementing the data node functionality. Thoroughly test the the fully functioning system.

  Updated Progress and Plan: We spent some time preparing for the midway presentation, which took place on Tuesday. Meanwhile, we built some sample scripts to start conducting experiments at a relatively small scale. We used the scripts to further test and debug the system. After making sure that there were neither bugs nor memory leaks, we designed and started conducting two simple types of experiments to measure deduplication rate, read, and write throughput for different block sizes. One type of test involves each client writing a sequence of repeated characters while the other involves each client writing a completely random sequence of characters. The experiments were conducted with 4 clients, 4 data nodes, and 1 master node. So far, based on the result of the first type of experiment, we observed a significant effect of block size on the read and write speeds and deduplication ratio where both speeds increased while the ratio decreased with increasing block size. We plan to finish conducting the two types of tests and start designing and conducting experiments with more sophisticated patterns of data stream that may involve writing and reading files of different sizes with certain duplicate ratios that may model real world data more closely.

- Week 4 (04/27-05/03)

  Original Plan: Conduct experiments. Write the final report and prepare for presentation.

  Updated Plan: We plan to finish conducting the experiments and start analyzing the results. As we analyze the data, we plan to start writing the final report, which we expect to take quite some time. Large part of the project involved implementation of a sophisticated system, so we expect to spend a significant amount of time describing and explaining the mechanism of the system with appropriate figures. This process will probably happen in conjunction with planning for the final presentation. We would be able to evaluate our work based on the experimental results and discuss our hypothesis, actual outcome and their implications in detail in the report.

- Week 5 (05/04-05/10)

  Original Plan: Finish writing the report, clean up the source code for submission.

  Updated Plan: We will spend more time polishing the final report and start preparing for the final presentation. Given longer presentation time, we should be able to explain our project in greater detail, but we should pay enough attention to the audiences' understanding of the material taking their familiarity with the subject and the scope of our project into account. We should especially put in effort in making good figures to aid the audiences' understanding of the material. We will prepare for the presentation through rehearsals. Before and after the final presentation, we will clean up the source code for submission. Also, we will probably schedule a meeting for a demo.

2 Difficulties

Designing and implementing the entire system was very challenging, given the complexity of the system. The important thing was to simplify the model and narrow down the scope of the project to a feasible level. Even after a few week long process of simplification and careful planning, we still had to make many decisions with regards to the implementation details as we were building the system. Choices, however, could not be made lightly as they could potentially have major impact on the system. It was our first time doing a project of this size using C++, so learning how to use various tools and libraries, ranging from TCP socket programming, shared memory, openssl, Makefile to multithreading, and object oriented programming was very challenging. But, we checked various resources on the internet, referred to past work in other computer science courses, and asked for help when necessary and eventually learned how to use them. Another major challenge was debugging the system, which was extremely difficult given its distributed nature. We resorted to execution of numerous sample programs, and debug print statements to locate and fix bugs. There were still some bugs even when we started executing some test scripts, but, as for now, all major bugs are fixed and we have not encountered any problem conducting the main experiments so far. The current challenge we have is designing an experiment that could accurately test the performance of the system, given that we will not be able to run standard benchmark tests. We plan to discuss this in more detail with Professor Newhall and collaborate to come up with some good experimental designs.