Mumfie DFS

Time Report

Rasmus Holm, carho647@student.liu.se
(Samuel Trennelius, samtr396@student.liu.se)
Last modified: 2012-04-25

2012-02-22 Wednesday

This meeting was about the requirement specification and it mainly was about the width of the project. What is a must-have and what is a nice-to-have. In the end we both agreed that since the project is quite complex, it is hard to tell this early. We tried to estimate how many hours each function is but as of today we lack knowledge of how to build parts of the project so we might have overestimated our own capability but we are certain that we should have a product somewhat like the Requirement specification.

Time used:

Rasmus: meeting and writing Requirement specification, 4h
Samuel: meeting and writing Requirement specification, 4h

2012-02-29 Wednesday

The width of writing the Analysis and Design is debatable. We chose to go in the middle of the road and make both description and graphs. At first it felt hard to write about design in such a non-interactive and non-graphical software this one. At the end of the meeting we felt that the graphs and descriptions was good but we left out the class descriptions out, since we thought that we needed some time to think about how they best should be described at this state.

Time used:

Rasmus: meeting and writing Analysis and Design, 5h
Samuel: meeting and writing Analysis and Design, 5h

2012-03-01 Wednesday

After a good nights sleep we agreed on how to describe the classes and ended up with a very general description since many things are very unsure at the detailed level. We checked grammar and spelling and was happy about our Analysis and Design document.

Time used:

Rasmus: meeting and writing Analysis and Design, 0.5h
Samuel: meeting and writing Analysis and Design, 0.5h

2012-03-13 Tuesday

Atlast, we started working and hack some code for MDFS. I begun working on the Name Node with the Meta-Data and User credential. This moved along quite nicely and no big problems were encountered. Next on the agenda is to implement request parsing and IO

Time used:

Rasmus: 4h

2012-03-15 Thursday

- I started working on the NameNode IO part. I decided on a protocol for how the communication is suppose to be handled between the different atomic parts.
- MDFS Communication Protocol, MDFSCP consists of a JSON-header and a optional binary file trailing it.
- The communication takes place in a manner where 1st a Request is sent and then 2ed a Response is sent, much like the HTTP. This means that a only one request and response per socket and session. However, a client can open more then 1 socket if asking for ex. 10 files, 10 sockets can be opened.
- Making MDFS generic I relized that we need a usfull way on making configurations of the software and seeing that it not a core function I decided we use Apache Commons Configuration (http://commons.apache.org/configuration/) for parsing the configuration and letting it be in a standard .ini formant (it supports XML as well if our need expand).
- I started to write the code for NetIO, implemented Configuration support and started to figure out how to use JSON in a right manner. I also wrote a couple of tests for different parts

I am now starting to realize how big this project will be and that a lot of time will be used just to make it bare minimum functional.

Time used:

Rasmus: 6h.

2012-03-16 Friday

We had a meeting today discussing parts of MDFS and how we should work. We look at the different parts of the software and decided that Rasmus handles MDFS core and Samuel will create a client application much like Dropbox. This due to the different levels of programming complexity and so on.

Time used:

Samuel: 2h
Rasmus: 2h

I started to look into how to parse the JSON-header and handle it and quickly realize that there are many cases that has to be handled. I decided to build a parsing factory due to the different needs of requests. I am trying to write the parsing as generically as possible making the NameNode, DataNode and ClientAPI parsing as similar as possible.(Still only working on the NameNode)

Time used:

Rasmus: 4h

2012-03-19 Monday

Today i finished up the most basic cases request parsing, executing the request and building a response. I am intentionally not spending to much time trying to optimize and build a optimal structure. This since mush is changing as we go along.
Test are encouraging as i now can se that i via MDFSCP can read and write meta-data information to the MetaDataRepository that stores it all in memory.

Time used:

Rasmus: 5h

2012-03-20 Tuesday

The core functions of the NameNode is now starting to be satisfied in a manner that i now can do basic operations. Much is still to be done.
So today i started to work on permanent storage of the meta-data from the Name Node. I wrote a couple of MySQL classes that simply stores the data in five (5) different tables. Since RAM is fast and HDD:s are slow i decided to do a threaded queue implementation, when a change in meta-data is made a SQL statement are created and sent to a queue. This queue is then handled by a separate thread that actually makes the SQL queries.
Loading data from the MySQL database into the MetaData- and UserDataRepositorys are still to be implemented.
Since i don't want to over work anything too much yet i will soon be moving on from the NameNode to the ClientAPI. My idea is to implement and make the simplest case of reading and writing first and then start adding other features and making optimizations and so on.
Time is getting away from me in this project but it's fun and i truly believe in the usefulness of the finished software so i don't mind. We are at 1523 lines of code.

Time used:

Rasmus: 6h

2012-03-23 Friday

I have started writing the Interfaces for the ClientAPI that will enable third party softwares to communicate with MDFS. This interface contain generic and logical file operations such as cd, ls, rm, get, put, chown, chmod and so on. The operations utilize both streams and logical names for files that are to be used.

Time used:

Rasmus: 1,5h

2012-03-25 Sunday

Today i Continued writing on the ClientAPI side handling parsing and IO operations, I have only implemented the simple case of writing data so far and done testing together with the NameNode. This works well at the moment but more things have to be coded. Later i started working on the DataNode portion with handling file name translation and mapping them to the local file system together with some general file operations. So far the Client and the DataNode are not yet communicating but that is what remains in the basic implementation. I am today at 2191 lines of code in 18 different package and divided in 3 different softwares and 2 help packages. I am soon starting to work on actually implementing all the different cases as soon i have got the basic one to work so the size will surely grow. What i have been thinking of lately is instead of using a protocol to communicate between the different parts (Client, NameNode, DataNode) to use Java RMI, which would radically reduce complexity of the software. This would mean to rewrite bug chunks but i will look more into it.

Time used:

Rasmus: 7h

2012-03-27 Tuesday

Time used:

Rasmus: 5h

2012-03-28 Wednesday

Simple case is finished with writing a file. However there are some problems handling inputstream. A BufferedReader steals bytes from the file portion of the protocol as text. This has to be corrected and a solution might simply be writing a own buffer.

Time used:

Rasmus: 4h

2012-03-28 Thursday

Atlast, the simple case of writing a file to MDFS now works. It has taken about 5h to solve the problem from yesterday. It was, in a nutshell, that Java's abstraction of reading from a inputstream (like BufferedReader or InputStreamReader) all buffers a bunch of bytes before

they are requested. The problem arose when a first a text header was sent and then a binary file when trying to read it in the same order. The InputStreamReader buffered a number of bytes from the input stream that was never read so when the binary file was read pieces of it was missing due to that the bytes had already been read. I tried a bunch of different solutions but at last solved it by abandoning java's input and output streams abstractions and wrote code to handle the raw-byte data directly.

Time used:

Rasmus: 5h

2012-03-29 Friday

I was quite pleased with myself after yesterday and spent most time today looking and making small changes to the ClientAPI and some time commenting written code. Next on the agenda is to

- Start doing the real work in parsing requests both on the NameNode and DataNode.
- Decide on what data structures to use in the NameNode for storage of userdata(I'm thinking a modified splay tree to maximize performance for many repeat access) as well as metadata(im thinking a multiple skiplist in the form of a file-tree, B+-tree would be preferable but are going to take far too long to implement)
- Do the final implementation of the MySQL backup, restore and save.

I'm at 3049 lines of code, 52 classes in 19 packages and 3 different softwares, including comments, tests, interfaces, enums and so on.

Time used:

Rasmus: 4h

2012-04-10 Tuesday

After easter break i started working on building a better data storage model for storing metadata and so on. It all got rather complex and tok a lot of time but i ended up with a decent data structure. I also did some work on the parsing part

Time used:

Rasmus: 5h

2012-04-11 Wednesday

Today i started working on the parsing of our protocol as well as wrote the data node segment that communicates with the NameNode och the local file system changes. I am finished with the

case of writing files to the filesystem where the feedback form the datanode to the namenode is working. This means that there are not much left to read enable reading as well. I spent some extra time optimizing the data transfer which turned out quite nicely. I am today up at 3946 lines of code.

Time used:

Rasmus: 9h

2012-04-12 Thursday

I have today been working on the parsing of the protocol and making the different softwares working together in a proper way. Things are starting to come together, on can now store files and directories, navigate them and retrieve meta-data from them. The main part left to implement is retrieving files, which is not much, and then main functionality are done and what remains are permanent storage of meta.data on the name node, building a raw data replication manager, and some general tweaks and more extensive testing aside from the continuous one.
Today: 4309 lines of code

Time used:

Rasmus: 8h

2012-04-13 Friday

Have done some general work on the parsing part and today **It Lives**. I have gotten all basic parts to work and i can now write, read, overwrite and traverse the file system. I also spent a great deal of time commenting the code and creating a small CLI to test the functionality which helped me to straighten out some kinks. I am quite pleased and not its time to start “painting the corners”, fixe out permanent storage, data replication and optimize.
Today: 4989 lines of code

Time used:

Rasmus: 10h

2012-04-15 Sunday

Today i have been working on a couple of things. Some optimizations when it comes to the queuing of MySQL queries and storing the Meta-data permanently and recover/loading it when bootstrapping the NameNode. I also fixed a bug in the DataNode regarding name translations that was introduced by mistake last week. I feel like i'm nearing the finish line.

To benchmark how fast the system is i simply wrote 10.000, 4 byte small, files MDFS in succession. Since the bottleneck in file transfer is just that, writing a large amount of small files since each file is handled separately. It took 61 seconds when each file was written in succession, but only 6.785 seconds when having 10 threads working and 6.406 second with 100 threads in parallel to write the 10.000 files. I did the same operation with a FTP server which took 8 minutes and 24 seconds. After the benchmarking tests i had about 75.000 file in my MDFS session and it takes the name node about 10 seconds to restart and load all the data from the SQL database into memory and build necessary relations.
I am pleased with the performance.

Today: 5423 lines of code

Time used:

Rasmus 9h

2012-04-16 Monday

Today have worked all parts of MDFS implementing recursive support in writing and removing file as well as cascading file changes and removals out to the different data nodes. What now remains is to introduce the feature of cascading a new file as well as changes to available data nodes. I also implemented a SplayList to handle user data since it i like that the last user who made a request to the name node soon does it again. I borrowed a pre existing implementation that i change for my purpose, to store a key and a object separately.

Today: 5688 lines of code.

Time used:

Rasmus: 4h

2012-04-17 Tuesday

Today i added support for file replication over the DataNodes. It works which makes all the Must-have requirements fulfilled. Now what remains is to write some more documentation, clean up the code a bit and optimize.

Today: 5876 lines of code

Time used:

Rasmus: 3h

2012-04-19 Thursday

Today i have been working on the UML and are done with the class diagrams representing the NameNode, DataNode and ClientAPI. The UML for the NameNode has gotten quite large so Im not sure how to fit it on a page. Next on the agenda is going through the code and do extensive commenting. When working through the code and having the experiences from hacking this project i find that I, if doing it again, would do some things differently regarding parsing. The experience that you way of writing code also changes over time is also present.

Time Used:

Rasmus: 4h

2012-04-24 Tuesday

Today i have been commenting the code which takes a while. I have been commenting continuously throughout the project but it really eats time doing it well. And i still see some areas that i could work on. What is good i that one do find some bugs working through the code and i find areas that could really use a makeover in both regard to optimization and nice looking code. I find that once style of coding evolves over just the course of a minor project like this and there are some major part i would like to rewrite but time does not really permit. Optimization and optimal code is hard to produce in the setting i been working in. Since i have no more time for iteration and many parts are not yet implemented, not the requirements though.
As well as commenting i wrote a new small benchmarking program, and i'm a little sceptical to the result i gained on 2012-04-15, today i manage to write the 10000 files in 20sec. However, with some tweaks on how i close sockets i have gotten it down to about 15 sec, this shows that much could be done to better performance. What is interesting to see is that MySQL concurrency is lagging comparing to the speed that MDFS is working which clearly show that my solution is preferable since MDFS outperforms MySQL and it would have been a bottleneck.

Time Used:

Rasmus 7h

2012-04-25 Wendsday

I have today been working on a few thing. Building some junit tests for single classes, modules and for MDFS as a whole. Drawing some diagrams for the documentation, writing the MDFS Transfer Protocol description, updating Design and analysis(more to be done), doing some more commenting(more could be done), fixing a few bugs, implemented a new feature (not a needed one). It is not much left now before hand in.

Today: 7218 lines of code

Time used:

Rasmus: 9h

2012-04-26 Thuseday

I have been finishing up, writing the last of documentation, doing some commeting, som last testing, organizing and some more. Takes a long time but not much to tell

Today: 7224 lines of code

Time used:

Rasmus: 6h