PROJECT

Marmalade

Project: Software Architecture for Distributed Systems
Software Engineering and Management Bachelor Program
Gothenburgs University IT-Section

Group members: Fredric Ola Eidsvik, Markus Erlach, David Lindenberg, David M Szabo, Flutra Tahiraj, Mengjiao Wei

Github repository: https://github.com/Marmalade1
Website and Twitter_Miner are the repositories with the working program.

Table of Contents:

- Table of Contents:
  - 1. Introduction
  - 2. Purpose
  - 3. Projects Purpose
  - 4. Project Management
    - 4.1 Scrum
    - 4.2. Group Name and Members
    - 4.3 Responsibilities
    - 4.4 Group Risk Management
    - 4.5 Other reasons for cutbacks in productivity
  - 5. Work Environment
  - 6. Project Backlog
  - 7. Burn Down Chart
  - 8. Frontend
    - 8.1 Website:
    - 8.2 Web Hosting & Web Domain:
    - 8.3 Design:
  - 9. Riak:
  - 10. Backend:
    - 10.1 Handler, Extractor and Main
    - 10.2 Important functions:

1. Introduction

This documentation will be about a group projects process through how the creation went of a web based application. The final project will let the user check twitters today, last 7 days and last 30 most popular hashtags.

2. Purpose

The purpose of the this documentation is simply for keeping reports of the whole projects process, and will not act solely on the project but also about the members cooperation between each other. There will be many causes that places a weight on the project.

3. Projects Purpose

We are a group of seven members who got an assignment where we had to create an application with mapreduce which will use real data from social networks. The main idea is the learning experience of how a mapreduce works, then it was up to us how we wanted to use this information and if we preferred a more entertaining application or more serious where other company would have an interest in.
With a vote it led to the entertaining purpose, and the attitude that if we thought it was something funny then our motivation would be there. Although as the project went along this project could be used as an static research purpose as well.

4. Project Management

4.1 Scrum

The development method that we have been using is called Scrum, and the scrum tool Pivotal Tracker. With this method and tool have been giving us a good overview of our tasks and milestones and provide us with enough time to make changes and improvements of the projects components.

4.2. Group Name and Members

The website and the project itself is called Swedishgeotweets.com a the website we display the most top10 popular hashtags from the twitter world.

4.3 Responsibilities

In a “real life” project the each one of the members would have theirs specific roles in the project but due to the small size of development team and project in whole, the responsibilities were spread out the group equally, but further in the project time there were members in better experience and understanding in some sections and therefore were the “leader” in that section.
Every member had the responsibly to finish their sprints in time, and if there were struggles, this had to be reported to the rest of the group so someone could help.
By the end of the project the goal will be that everyone leaves with the same knowledge.

4.4 Group Risk Management

This group have came across many barriers which has slowed down the projects process. The lack of communication within the group caused many irritations in the group and could have caused big problems but have been taken cared of in early stage so it never got to the extreme level. But when we went from seven members down to six, that was something that we learned a lot form. It took weeks till we realized that we really lost a member, and have to move one without that person, need to take over that person's work and sprints. This left us with confusions, where should we start, what do we have, what do we not have. Should we start all over? The answer to the last question was yes. We had an experienced person on the front end, that experienced that it was enough with only one person, so when we no longer had this members we also didn't have a anyone else close to the same knowledge. Too much time had already been lost and the stress was just around the corner.
Lesson learned form this was to always have someone as a backup. From now one there were always two members working together in case there would be a reason for absence for one of the members, the project would still be standing on two feet.

4.5 Other reasons for cutbacks in productivity

The project were taken care of in parallel with other assignments and exams studies, most of the time the project were not a priority for the members and therefore more time for the project got lost. This could have been scheduled better perhaps but because everyone has their own study techniques there was hard to find a common time where everyone could have time for the project. So it had to wait. Consequences were stress and again overlapping with other studies, and this caused frustration and lost motivation among members.

5. Work Environment

To be able to even start working on the project we needed to have some specific tools. To use Riak as our database were a requirement but the information about the need to have Linux installed in your computer were something many members had a problem with. One computer had pre installed windows 8.1 with specified Bios/UEFI for acer computers. This computer caused a great deal of trouble. The owner of the computer had to spend 30h researching why his windows installation would not work. He had to call support and after talking with several supporters at Acer and Windows, finally

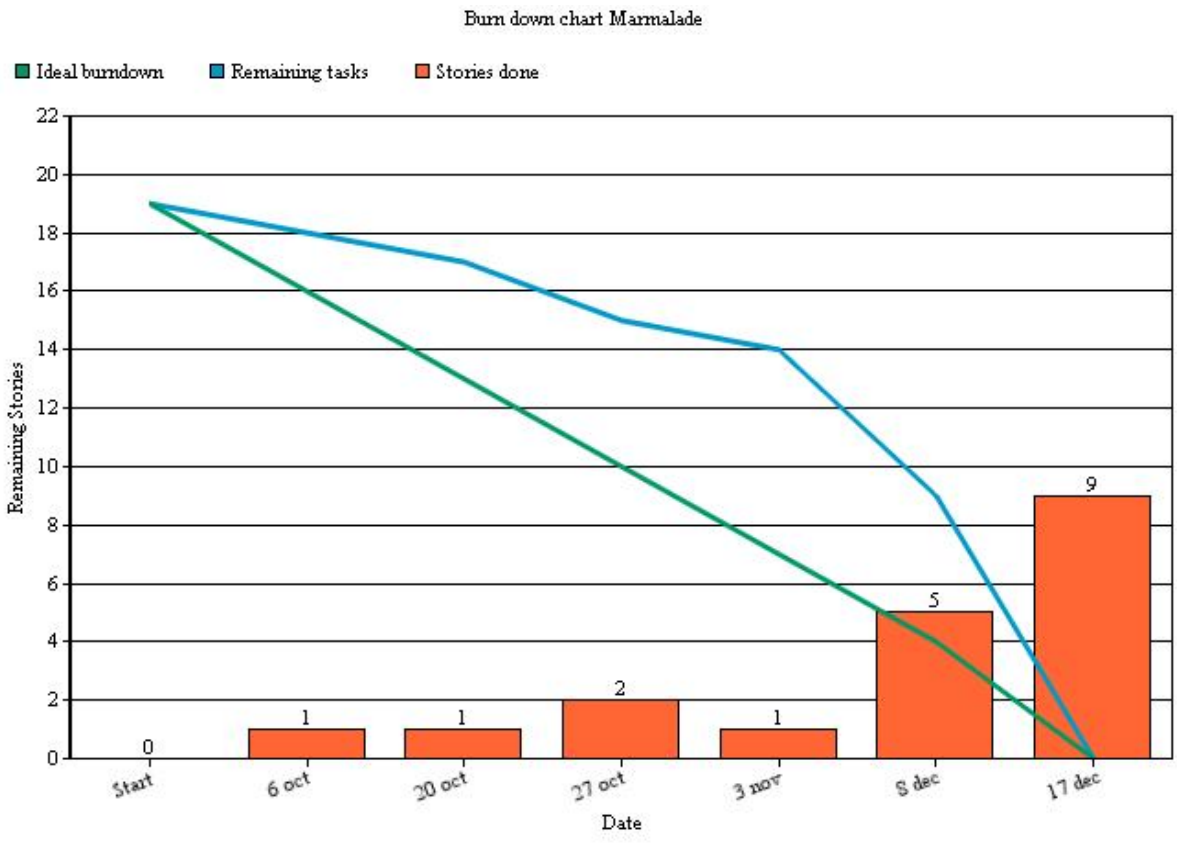
one supporter said that he needed to change some stuff in bios. After he made the changes he could install windows 7 and after that he had no problem double booting with Ubuntu.

6. Project Backlog

See attached link.

https://docs.google.com/spreadsheets/d/1EjhwDg4SM1LV9uB6gQQggEmARa6EKRBQAgBoDLTpQow/edit?usp=sharing

7. Burn Down Chart



We unfortunately used the backlog on pivotal tracker badly and have therefore a burndown chart that does not accurately show that actual progress and time spent on each sprint. But what it does show combined with the backlog is how we spent time in the beginning slowly researching and acquiring knowledge about what needed to be accomplished and have then ramped up our work as well as progress drastically to deliver a working product as well as documentation for the product.

8. Frontend

8.1 Website:

From the start, members were working separate to be able to “play around” with the code and try out functions that might suit the project. This way we would not interrupt the other members work. Like mentioned before, we were not that experienced so this type of testing were needed and helpful. Eventually we decided to put the website in to our server, there we could do “permanent” changes. Everyone could then be up to date and wouldn't be missing any code. Testing could be done long aside and would not be an effect on the real code-files.

8.2 Web Hosting & Web Domain:

As mentioned before, the person with the most experience with web design left the group. This caused that another person had to do some research how to create an website. After some tips from another classmate we were tipped to use Godaddy as an web domain and INIZ as a web host.
Godaddy worked perfectly fine and was up and running straight away, INIZ however caused a great deal of complications due to not being able to link with Godaddy since they used some other design. After a lot of talking with support the domain and web hotel were finally connected.
This is where we found the second problem with INIZ, we were not able to upload any code to INIZ and therefore that made INIZ unsuitable for this project. We canceled the subscription and started a subscription to Hostgator instead. This Webhotel worked perfectly fine straight away and worked perfectly with Filezilla as well.

8.3 Design:

The web page that we have created for the project is of a simple design, and this is the third version of it. the first two were more of draft of what we thought the webpage would be.
Why have we chosen a simple design? Well we want people to quickly be able to go on to the web page and check what's trending in the twitter world. It can’t be something that takes too much time. you want the answer and that’s about it, to see what are people tweeting today, the last 7 days and the last 30 days. And we think that we have accomplished this and also kept the colors that we wanted for our project that are suppose to symbolize Marmalade.

9. Riak:

In the beginning, we tried to use different servers for the website and riak, but we can’t make the connection works when we tried to connect the Riak by using same ip address and port number for riak-erlang-client. After talking with other groups, we decided to put the website on the same server with riak, so that we could use the default ip address and port number for connection.
Getting the value shown: Our website was written in html, php,css and javascript, so we use riak-php-client to fetch the value and then we use json_encode to convert and show on the website. And we also used this value in the javascript chart.

Everything that is stored to riak is created as a riak object where we store our objects as an erlang tuple that includes the hashtag, an ID and a timestamp. When we store this using our extractor we also add two secondary indexes for the hashtag itself and the timestamp. These two values let us exact match query something so that we can easily find a particular hashtag. The timestamp index is added in order to enable range querying where we can search between two particular dates in order to created queries for last the 7 days and the last 30 days.

In order to store things to riak as secondary index, we had to change the database type from BitCask to LevelDB that has secondary indexing enabled. Without it, we couldn't search for particular things in general. It all had to be done by Key Search which meant we needed to know every single key. This would be fine if we had maybe 100 keys, but when we store a few hundred each time we run the extractor, its start to get a little too much information.

10. Backend:

10.1 Handler, Extractor and Main

While working with the backend we ran into a lot of difficulties early. Everything from trying to install ubuntu on windows 8 computers to issues with just installing riak.
These issues caused a lot of trouble since we couldn't get started on the backend until we had it fully running.

Once Linux Ubuntu and Riak were fully operational we could get started on using Michals skeleton and retrieving a twitter feed from twitters services. We started out by using Michals my_print function and started adapting it to our needs. We first thought we should take a lot of information and store it in riak. We began by using Michals extracting function in my_print and with that we retrieved information such as ID, Location, Created_at, Language as well as the Hashtag. To do this we needed to print out a full list of information that the tweets contained and then use the extract upon the specific part we wished to retrieve something out of. This gave gave us a lot of unnecessary information in the beginning and we soon realized that we needed less and less of that. Eventually we ended up with only using the hashtag out of all that information. Location for example was way too rare to be used. We got around 3 locations per 400-500 tweets which is less than 1%. So we decided it was not something we could reliably use to show an accurate account for where tweets originated. The created_at timestamp was something we decided to use as a secondary index instead so we created a timestamp function that we use instead to show when WE received and stored each tweet.

After being done with extracting the information we deemed necessary for our project we started creating a query handler to run calculation and format our tweets in a way that we could show it on a website where users can select the information they want to see. So here we made a mapreduce function to run on the key/bucket combination we wanted to run it on. This combination is retrieved by using a few different functions so that we can choose if we want to run mapreduce on 1 day, 7 days or a 30 day interval. While doing this we realized that we wanted to be able to filter out certain keys or

hashtags to show a revised list as well so we had to create additional functions for that. This gave us the option of showing 6 different top 10 lists.
These 6 lists are sorted in order 1-10 with the hashtag most used would be 1.

When all this was done we ran into some difficulties querying our erlang query handler to retrieve these lists and show them on the web page so we had to change around a few things and decided to make sure the lists are updated twice a day after we have run the extractor to retrieve new tweets.

So after being sorted, we store all the top 10 lists in a Result bucket created with the sole purpose of keeping the lists up to date. And that bucket can then be queried from our php web page.

When these parts where done we needed a module to supervise the other modules as well as run them at designated times so we created a module called Main. In the main module we start a scheduler, the handler and a loop that will receive from or send messages to the handler. The scheduler is created so that every 10 seconds it checks the time and if the time is the time we want the extractor to run, it will start the extractor and then send a message to the query handler to start as soon as the extractor is done retrieving and storing tweets to riak. This module will also make sure that we can stop the scheduler or restart it and the handler incase something goes wrong or crashes.

10.2 Important functions:

10.2.1 Filter information: The filter function will take the stream of tweets from twitter and filter out the information we want from it so that we can store it in riak. This it does by using Michals extract function that is a basic keyfind function. When the desired information is found, it looks for the next desired information if any and at the end return the information to be stored.

10.2.2 Save tweet: If the hashtag is not empty, we store the filtered tweet as a value
Value = {Id, {}, Hashtag} with a bucket and the ID as a key as an object to riak. We then update that object and add secondary indexes to it. Those secondary indexes being the hashtag again, and a timestamp. They are added to enable filtering. We then send this updated riak object to riak and store it.

10.2.3 fix hashtag: The extracted hashtag we get from the tweets contain additional data and is not structured in a way that is good and easy to handle if sent to riak. So this function takes out the information we actually want from the retrieved hashtag and returns it.

10.2.4 Show Today: This function is created so that we can run a mapreduce on values by using the secondary index to filter out todays information. The information from today will be run through the mapreduce and sorted into two top 10 lists. One regular list and one revised that removes the filtered out word. Then it is stored into Riak's Result bucket.

10.2.5 Show last 7: This function is created so that we can run a mapreduce on values by using the secondary index to filter out one weeks information. The week will be from 7 days ago to today. The

information from today will be run through the mapreduce and sorted into two top 10 lists One regular list and one revised that removes the filtered out word. Then it is stored into Riak's Result bucket.

10.2.6 Show last 30: This function is created so that we can run a mapreduce on values by using the secondary index to filter out one weeks information. The week will be from 30 days ago to today. The information from today will be run through the mapreduce and sorted into a top 10 lists. One regular list and one revised that removes the filtered out word. Then it is stored into riak's Result bucket.

10.2.7 Go: The go function will run through all 3 functions. Show_today, show_last_7 and show_last_30 to update the Results that can be queried from the web page.

10.2.8 Mapreduce: We have implemented a mapreduce function for our project, that takes care of our algorithm to get the hashtags information from the tweets that we need.
Our mapreduce takes our tweets with the hashtag and checks how many times each hashtag is used from the tweets that we mine, and creates a list with the amount of times each hashtag is used.
With this we create a top 10 list of the most uses hashtags and a last 7 days and a last 30 days list.
The mapreduce that functions takes our PID to connect to riak, we have to assume that we have started a link to Riak and saved the process-id as Pid, and saved our keys. With this we can invoke our mapreduce, this starts the mapreduce on the server with two phases. One map-phase and one reduce-phase. Our map function dose a list comprehension for all the items that we get stored in the RiakObject and puts it in a dict together with the value 1. Once the mapping is complete the output sends it to our reduce function.
Our reduce function uses list:foldl that calls a function for every element in the list, we use an anonymous function which uses dict:merge to merge the input.
this function now merge our input into a list that uses our reduce-function uses to display the reduces list.

Marmalade zipfile: SAD document. Marmalade document pt. 2, Retrospectives, backlog