Music Recommendation System

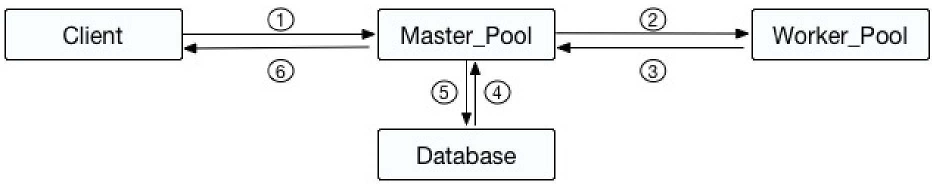
Distributed Crawler Using AKKA

1. Overview of Problem

Some frequently used and powerful Music players like iTunes and spotify may provide a daily recommendation playlist to their users so that they could find more singles with the same or different genre they might like. However, a true music lover may not to be satisfied with only a fixed number of recommended playlists at a day, he or she may also find it hard and relatively time-consuming to try all the songs in the playlist. Here we aim to design a special music recommendation system using distributed crawlers. What we are going to achieve is that whenever the user wants to try some recommended songs in a day, they could just type in the songs they like, then a list of potentially-liked songs will automatically pop out. Users may find good musics by using our recommendation systems at any time in a day, which is more flexible.

To actually deploy and implement our model, the most important thing that we need to do is to extract the songs from the website hierarchically (i.e layer by layer), Though the traditional web crawler may be also OK to solve the problem, it is still not good enough. The fact is that instead of activate only one crawler, we could activate more crawlers at a time, each assigned with different tasks so that they could do the job concurrently. The final result is then aggregated to create a database for further use. By doing so, we minimize the cost and increase the speed. Note that the traditional web crawler also need to concern with the ip address problem, since some website may have some "anti-crawl" techniques to stop it from crawling. If we use more than crawler, or more workers with different ip addresses, a distributed-system based solution is thus proposed to tackle with this problem easily.

2. System Architecture



① A client enters identities of three loved songs to find the potentially-liked songs with the same genre that might be liked by them. Identities will be given at a ".txt" file. Those three identities are sent to Master_Pool to be handled.

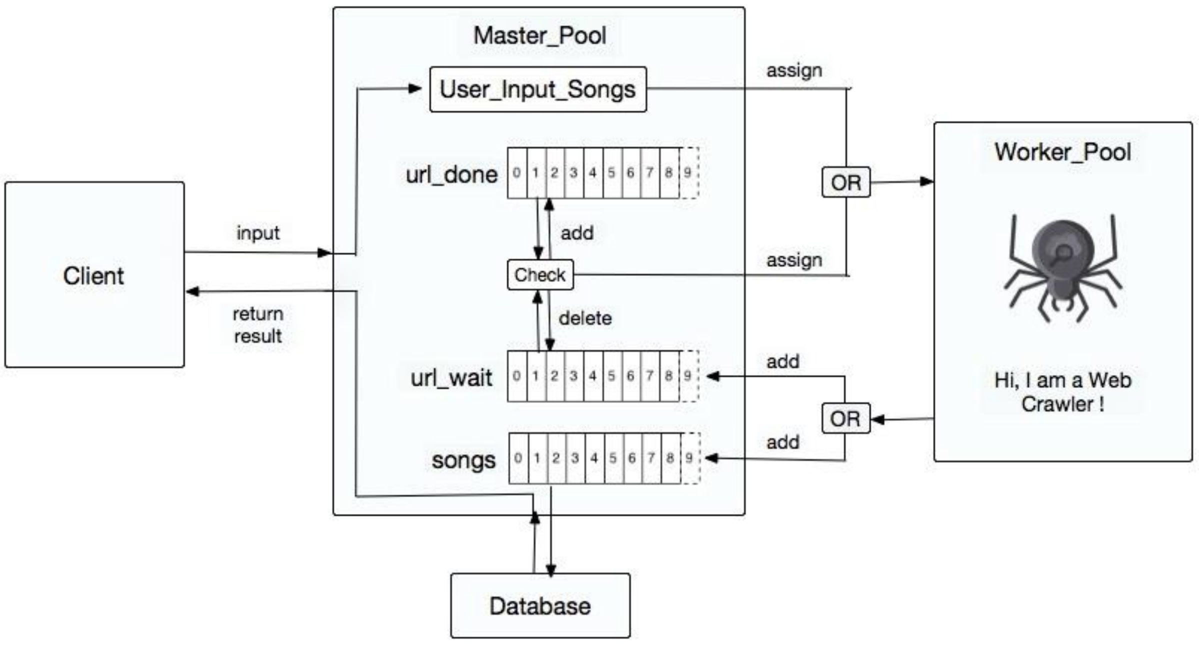
② Masters in Master_Pool send urls of songs and playlists to Worker_Pool many times in order. It will not stop until the threshold is reached.

③ Workers in Worker_Pool return data to be disposed according to the urls received from Master_Pool. For the url of a song, it returns three ids of relevant playlists. For the url of a playlist, it returns a list of songs in the playlist.

④ Identities, names and number of occurrence of songs that be received are sent to the database.

⑤ Songs most frequently occurred like top 10 can be returned to the Master_Pool.

⑥ Master_Pool sends returned final result of database back to client.



The overall process has been described above. This part will introduce the function of Master_Pool and Worker_Pool. The User_Input_Songs, which are obtained from the Client, will be transmitted to the Worker_Pool to crawl the playlists of the songs respectively. These playlists are added to the url_wait list to wait for the check whether this is existed in the url_done. And then the not repeated one can be kept and be sent to the Worker_Pool again to crawl list of songs in the playlist. The songs can be stored at the url_songs and url_wait. The url_wait part need to be check whether they were crawled before by checking repeatability compared with url_done and delete the duplicated one. The behavior of send playlists and songs to the Worker_Pool and Worker_Pool returns results are repeat again and again until the songs' number in url_songs reach the number 500 which can be changed. The data in url_songs is reserved in the database. The database returns top 10 highest frequency songs to Master_Pool which will be backed to the client side as preferences that the client might like.

3. Technology Choices

Technology	Motivation for Use
AKKA	-AKKA actor model inherently supports and automatically enforces the best practices of concurrent programming, it is easy to implement -Focus Shift: it provide a good structure to avoid low level coding.
Web Crawler e.g. Jsoup	The web we intend to crawl is actually a html source code, Jsoup provide functions to easy analyze and extract the useful information.
MySQL	The crawlers return a list of songs, which could be stored in a sql database. SQL provide query languages like select, sort,orderby, thus make it easy to analyze the final result set.

4. The Team

Give details of your team members and their assigned tasks below.

Student Number	Name	Assigned Task(s)
14207127	Li Shuo	- Implement AKKA actor model, and connecting Master and Worker - Implement Web crawler in Worker - Implement Database

		- Try to implement Kafka and RPC structure.
14207131	Jiang Mudi	- Crack the target website and implement web crawler in Worker - Data retrieving in Master - Implement front end - Try to implement Kafka and RPC structure.
14207139	An Yu	- Implement web crawler in Worker - Url assigning, data retrieving and processing and analysing in Master - Implement Database - Try to implement Kafka and RPC structure.
14207127	Mo Tong	- Write the group proposal report

4. Task List

4.1 Task 1: Implement Actor system

This task is considered to be the most important task in the whole project. Since the aim of the project is to simulate multiple crawler workers to retrieve data and further process the data, Actors are basically concurrent processes which communicate through asynchronous message passing (Haller, 2006) , The problem that need to be resolved is that: a master should always know when to get the returned data and when to assign new task to workers, similarly, a worker should always know when to do the jobs. In this project I override onReceive() method, which is a simple abstract method that could be invoked for every processed messages. During this process, I firstly implement the master which is responsible for saving all the songs to the list, and assigning new urls to the worker. Then I implement the worker to actually crawl the url assigned to them and return a list of urls to the master that still need to be crawled, the whole process repeats until the list reaches its threshold.

I found that there is a very tricky situation in the actor model: when there are too much actors existing in the system, AKKA seemed to be automatically choosing a thread in the threadpool to execute the actor, this results in the case that many different actors may be executed by the same thread or one actor may be executed by many threads. This problem adds the complexity in debugging our codes, The solution we come up with is to shrink the execution time in each individual actor so that it won't bothers with the scheduling of other actors.

4.2 Task 2: Url assigning, data retrieving, processing and analysing in Master

In this task, we agree that the the masters should be designed first since it involves how to assign the new task to workers, the process of which does not consider with AKKA actor related technology, so we could only focus on deal with the inputs and outputs of the Master. The functionality of a Master is first get one input from client output a url to Worker and get results from worker and repeatedly assign worker url and get results until the results are reach a limit then using database analysing the results and finally return to client the system final conclusion.

At the beginning stage, we want to use a RPC structure and also use Kafka technique. RPC is a multi-worker structure to reduce time spended. Kafka is a distributed streaming platform, which can give benefits on broadcasting, receiving and giving message in order.

In our case can assign urls to multi-workers and automatically orders the received results. This should give a excellent performance. However, implementing our music system using RPC and Kafka proved to be very tricky because most examples are using scala and Kafka need to installation.

First url assigning

The first thing to deal with is how to transfer the client input songs to a useful url, by analysing the target website url rules, we find out the url of songs are start with fix string and end with different song id. For instance the song id is 517411583 and the corresponding url of this song is: http://music.163.com/#/song?id=517411583.

Data retrieving and Processing

In terms of the received results from worker, there are two types of possible return input, one is a set of urls need to be crawled later or a list of songs details that including the song id, song name and appear times. As a result, based on the return type, two deal method are included. If return type is HashSet type, it means a list of urls' parts (the distinct part of urls) are return. Iterator this set and store in a arraylist called urls_wait. If return type is HashMap, it means the return type is a list of songs along with its details. Add them in a HashMap songs for later use.

Check Repentance

For now one layer is done, the next challenge is assign other urls that needed to be crawled. But before that in order to avoid waste time on visited urls, we need to first check if the url is already crawled or not. This is done by record every url that visited into a hashset and check this hashset before assign new url.

Iterator urls

The next challenge is assign other urls that need to be crawled, which are listed in the arraylist called urls_wait and iterator this process again and again until the results songs number is reach a limit.

The first idea come up is to call onReceive() method in onReceive() method so that the program can iterator by itself. By long time debugging this idea finally worked. And the iterator process worked.

4.3 Task 3: Crack the target website and implement web crawler in Worker

The main job of the task 3 is to implement web crawler in Worker. There is a challenge during this task because the music web, which is called Netease Cloud Music, add some settings of anti-crawler. After getting the HTML, the selection of contents is also a significant task. Technologies of regular expression and JSoup can be used to crawl scripts.

Webpages of Netease Cloud Music encrypt the parameters of obtaining data. After researching the source codes and some corresponding similar assignment, the crack method of the target website was found. There two ways to solve this problem. One is using the algorithm to deal with the HTML. Another is easier and we choose to utilize this one that is deleting the sign “#” in the url. For example, for the webpage: “http://music.163.com/#/song?id=435278010”, removing the “#”.

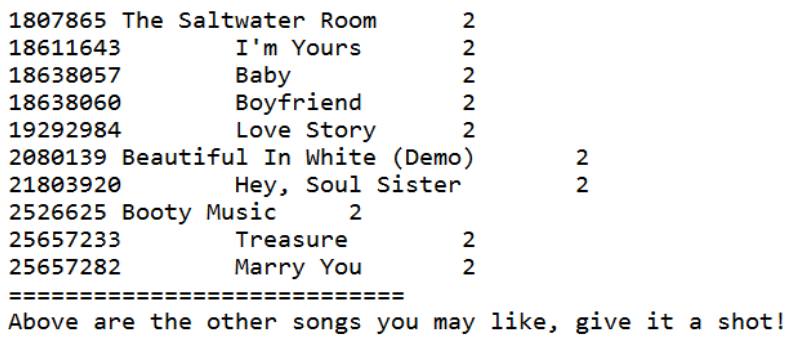
Worker can receive two kinds of urls which will be crawled, including the url of a song and the url of a playlist. Netease Cloud Music provides some playlists which contains the song for a song and this can offer the similar preference. For the url of a song, Worker should seek out non-fixed parts of urls of the playlists and return this to the Master. This can be achieved by the regular expression. And then, using the split() to dispose the matcher part, and non-fixed parts can be gained. For the url of a playlist, Worker should seek out and return identities, names of songs and frequencies which are default 1. Jsoup can help to analysis the HTML, and selection of the element can help to acquire the content.

As for the distinguish of url between songs and playlists, this can be achieved by the judgement of the 21st character of the input url. If it is a character “p”, the url should be the HTML of a playlist. The character “s” means the HTML of a song.

4.4 Task 4: Implement Database

In this task I assume that I have already obtained the songs crawled by the worker, Then I shall only focus on implementing the database to store the songs information, including its id, name and associated frequency. Firstly I create the Mysql music recommendation system database in the command line interface and establish a connection connecting to the database, then I write java code to simulate sql language to insert the data. The last step is to write another query function to get the final result, which contains a sorted list of songs that the users may prefer.

For example a user may blike: Perfect(Ed Sheeran), ready for it(Taylor Swift), As long as you love me (Justin Bieber), our music recommendation system find other songs that the user may also like:



```
1807865 The Saltwater Room      2
18611643        I'm Yours       2
18638057        Baby            2
18638060        Boyfriend       2
19292984        Love Story      2
2080139 Beautiful In White (Demo)       2
21803920        Hey, Soul Sister        2
2526625 Booty Music     2
25657233        Treasure        2
25657282        Marry You       2
============================
Above are the other songs you may like, give it a shot!
```

However there is a limitation in the database, for the total number of concurrent connection is set to be 128 as default, this will give rise to a database exception. To fix this, we could change the settings “max-connections=128” in the my.ini configuration document to bigger number so that it allows more connections at a time.

4.5 Task 5: Improve the Music Recommendation System

After the first success run of out Music Recommendation System, several improvements are made to this system.

First, there is a arraylist in master called urls_wait, every time this list is added new elements. But there should have some remove action after the url in this list assigned to the worker. By add this remove action, the time spend in finding the unassigned url is reduced. Therefore, the performance should be improved. By testing, the results show the response time is slightly improved.

Another improvement is made on the database, after run several time of this system the searchable songs is increased due to the database we build is growed. Another possible improvements also found out that, set the default connection number setting of MySql to be larger, than it can store more data every time.

In the improvement process, we first want to add a front end panel. The front end includes three parts, those are welcome panel, enter panel and ok panel. Welcome panel shows welcome words and “Please enter names of three songs”. There are three enter panels for clients to enter names of songs. The ok panel only has a button. If clients decide which three songs they would like to use, they can click the button and the preference list can be shown. The front end is not used though due to the fact that we think it is not necessary and increase the response time, worse the performance.

5. Reflections

- Were the technologies appropriate?
  There are mainly three types of technologies involved in this project.
  In terms of Akka, as a distributed technology, it can reduce the time which is better than the situation without it. In this project we only use some basic actor models, and a more advanced structure like RPC can be used for a better performance.
  For crawler technologies, the current version is good enough to meet our needs. However, if we want a more intelligent system, some algorithms can be implemented to further boost the crawlers performances, e.g., crawlers with anti-encryption abilities.
  The third technology involved in this project is MySql database, this result shows a relative good performance. By involving a database technology, more information are stored and some queries can be used for further analysing the output result.

- Did distribution cause unexpected problems?
  Distribution do cause some unexpected problems.
  The first one is that the message between the actor system is not the same as we thought about. For instance, a worker gives a master a hashset, the message received is not a hashset. In order to get the hashset a constructor must be used.
  The next unexpected problem is the message returned are not in sequence, which make it hard to debug our code.
- What are there limitations of the technology you used?
  The limitations of the Akka actor system is that: when there are excessive actors existing in the system, AKKA seemed to be automatically choosing a thread in the threadpool to execute the actor, this results in the case that many different actors may be executed by the same thread or one actor may be executed by many threads. This problem adds the complexity in debugging our codes, The solution we come up with is to shrink the execution time in each individual actor so that it won't bothers with the scheduling of other actors. Other solutions involves the use of zookeepers, which provide structures to synchronize the distribution services.
  MySql technology used in this project is using a localhost database, so other client want to use this system must have MySql installed and create the database and table first. Another limitation is the default connection number settings is usually very small limit the size of stored data.
  Crawler that we used are target for this unique website, target problem, it can not apply to other websites. This crawler sometime can be found by the website firewall and block our IPs temporarily due to the fact that the datas stream in short time is too large. Eg: threshold songs number in master code max set to 5000.
- What are the benefits of the technology you used?
  Akka technology is easy to use, it provides lots of powerful wrapped-up method that we can use directly or only need to change a small part. As a result, the code of distribution system using akka technology usually very short. Focus more on the actual problem instead of the distribution system structure.
  In our project, one of the best technology used is to retrieve and further extract the data with the help of jsoup. Despite the fact that there might be other strategies to obtain the data, we all agree that jsoup method is when compared, more easier to achieve.
  MySql technology is also very easy to use. Also, using it provide convenient method to process and analyze the data by using the sql query.
- What did you learn?
  From this project, we not only get the better understanding the basic principles of AKKA distributed model, but also got some hands-on experiences to solve the real world problem with the knowledge learned both in the project and in the lecture. Besides we learned how to implement the web crawlers to retrieve and extract the data. We may prefer to solve the problems by using single thread, and we all try to avoid the painful multithreading and threads synchronization. However some distributed structure like AKKA provide some simple methods and is relatively easy to implement. In the future we might need to cope with different problems associated with distributed system, which we believe would benefit from this early exposure.

References

[1] Collier, R.(2017) 'Network Programming: Actor Programming' [Lecture], *COMP30220: Distributed Systems*. University College Dublin.

[2] Haller, P., Odersky, M. (2006) "Event-Based programming without inversion of control", *Joint Conference on Modular Programming Languages*, 4228, pp. 4-22.