Spring 2015

A SCALABLE SEARCH ENGINE AGGREGATOR

Pooja Mishra
San Jose State University

A SCALABLE SEARCH ENGINE AGGREGATOR

A Project Report

Presented to

The Faculty of Department of Computer Science

San Jose State University

In Partial fulfillment

Of the Requirements for the Degree

Master of Science in Computer Science

by

Pooja Mishra

May 2015

SAN JOSE STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

A SCALABLE SEARCH ENGINE AGGREGATOR

by
Pooja Mishra

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Chris Pollett, Department of Computer Science Date

Dr. Sami Khuri, Department of Computer Science Date

Dr. Robert Chun, Department of Computer Science Date

APPROVED FOR THE UNIVERSITY

Associate Dean Office of Graduate Studies and Research Date

ABSTRACT

A SCALABLE SEARCH ENGINE AGGREGATOR

The ability to display different media sources in an appropriate way is an integral part of search engines such as Google, Yahoo, and Bing, as well as social networking sites like Facebook, etc. This project explores and implements various media-updating features of the open source search engine Yioop [1]. These include news aggregation, video conversion and email distribution. An older, preexisting news update feature of Yioop was modified and scaled so that it can work on many machines. We redesigned and modified the user interface associated with a distributed news updater feature in Yioop. This project also introduced a video updater feature for Yioop. This feature converts uploaded video into formats that are compatible with all browsers. It can quickly convert lengthy videos by splitting and converting them in a parallel fashion. It then merges them back into a single video. In this report, we discuss a solution to off-load the task of sending emails from the Yioop web application to the Yioop media updater by aggregating emails over a period of time. We conclude this report by describing experiments with these developed features on a cluster setup on an AWS platform.

ACKNOWLEDGEMENTS

It gives me immense pleasure to present my acknowledgement, sense of gratitude to one and all, who directly or indirectly, have lent their hand in this project. I am extremely thankful and indebted to my project advisor, Dr. Chris Pollett for sharing expertise, and sincere and valuable guidance and constant encouragement extended to me throughout the project. I am grateful to my committee members, Dr. Sami Khuri and Dr. Robert Chun, for their encouragement and suggestions without which this project would not have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

An important aspect of creating a modern search engine is the ability to display various media sources in an appropriate way. These media sources include news feeds, videos, images, group discussions, blogs, etc. Most of the search engines like Google, Yahoo, and Bing have these features incorporated and customized according to the needs of their users. In this project, the goal was to scale the existing news updater feature and design and implement a video updater feature of an open source search engine Yioop.

Yioop had a news updater feature [3] that re-indexes feed on an hourly basis. Users could also upload videos. Prior to this project, certain search engine tasks, such as updating news feeds, could be carried out by only on a single machine, i.e., a name server. One could upload videos, but there was no mechanism in place to convert these videos to a certain format. We have explored the media updater feature of the Yioop search engine in this project, and have experimented with it in order to scale it and make it distributed. This included thoroughly understanding, working with, and experimenting on the already-existing news updater functionality. We discussed different approaches to how a news updater feature could be scaled and implemented one of them.

To make use of these newly added features, we designed a new user interface to include the distributed architecture so that one can keep tabs on the status of running processes on different machines. Yioop had the ability to upload videos, however, these videos needed to be converted to mp4 format in order to be remain functional on all the browsers. We introduced the video updater feature in Yioop.

The report is divided into seven chapters. Chapter 2 includes summary of background work done. Chapter 3 covers the news updater feature of the media updater and the experimentation done on the existing design. The details of the implementation are also given. Chapter 4 focuses on the web interface that manages machine activity in Yioop, as well as the purpose behind redesigning and modifying it. Chapter 5 deals with the video updater feature, discussing the approaches / design experiments, and covering their pros and cons in-depth. Chapter 6 discusses the mail aggregation solution. Chapter 7 records all the experiments and testing done on these features. It also has integration within Yioop. Chapter 8 concludes the report.

CHAPTER 2

BACKGROUND WORK IN YIOOP

Yioop is a PHP search engine developed by Dr. Pollett. In Yioop the user, have control over the exact sites which are being indexed with Yioop, you have much better control over the kinds of results that a search will return. Yioop provides a traditional web interface to do queries, an rss API, and a function API. The Yioop search engine can be used both as a regular search engine as well as a search engine that performs searches on a pre-defined set of URL's given by the users i.e., users can customize the search results by performing a crawl and setting the URL's that the user wants as the index to be searched in Yioop. Yioop is written using its own model-view-controller framework with a user interface that is simple and user-friendly. The Seekquarry [2] web site contains all the documentation and resources that are related to Yioop.

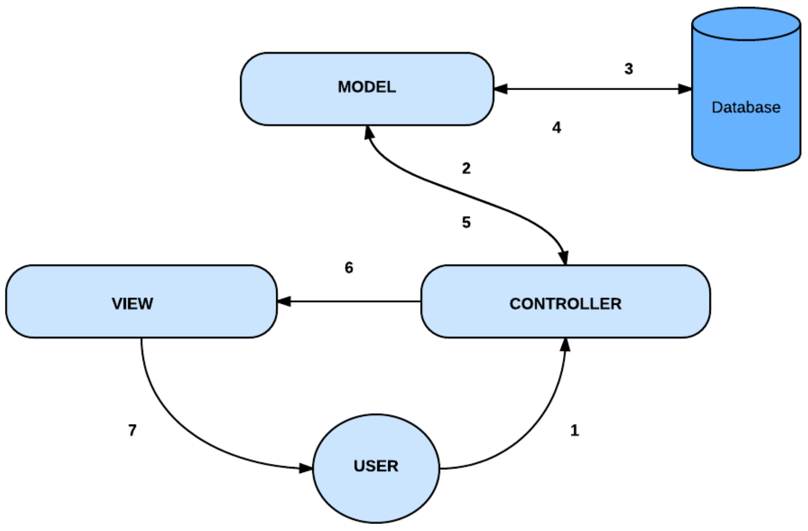


Figure 1: Yioop Architecture Diagram

As mentioned earlier, modern search engines need the ability to display various media sources such as news feeds, videos, and group discussions in an appropriate way. To understand the project better and to gain insights as to how to design and implement the media updater features in Yioop, I went through the updater features of various social networking such as news feeds, and familiarized myself with the high-level working method. Google has a news feeds section which categorizes them nicely. They also offer a 'Suggested for you' section [12]. Yahoo news feeds are popular too. Yahoo has a producer-consumer model [8], which talks about how they fetch feeds and minimize the cost by localizing decisions. Facebook [5] mainly uses links and connections of a person to choose news feed on his/her personal page.

Table 1: Comparison of news feature in Various Search Engines

	Based on Categories	Customized for user	Basis for generating customized news feed
Google news	YES	YES	Google search queries and articles visited
Yahoo news	YES	NO	Trending stories
Facebook news	YES	YES	Connections and likes, post views etc.

Google always maintains secrecy around the algorithms that they use for their popular products like YouTube [13]. I went through these popular social networking sites and found out which video formats these sites support.

Table 2 : Video Formats Supported by YouTube and Facebook

YouTube	Facebook
MOV	3g2 (Mobile Video)3gp (Mobile Video)
.MPEG4	3gpp (Mobile Video)
MP4	asf (Windows Media Video)
AVI	avi (AVI Video)
WMV	dat (MPEG Video)
WebM	divx (DIVX Video)
FLV	dv (DV Video)
3GPP	

In order to introduce the video updater feature, we wanted to select certain video formats for conversion. This included formats that are eligible for conversion, as well as the format they will be converted into. To select the formats that are eligible for conversion, we looked at the video formats that are usually uploaded by users. The AVI video format is standard for most digital video cameras. Mov video formats [15] also are very common, as most Apple devices produce mov format videos. Many different browsers, such as Chrome, Firefox, Safari, etc., are used. We did a survey and found out the video formats that are supported by different browsers.

Table 3 : Video Formats Supported by Popular Browsers

Browsers	**webm**	**Mp4**	**mov**	**avi**
Chrome	✓	✓	✓	✓

Safari	✓	✓	✓	✓
Firefox	✓	✓	✓	
Internet explorer	✓	✓		

Looking at above table, we can conclude that mp4 and webm are the video formats that are supported by the most popular browsers. The rest of the video formats are hit or miss. 50% of them might support some formats, while others don’t. We selected the final conversion format as mp4. We chose mp4 over webm because if you compare same file in two formats, mp4 usually occupies less memory. In Yioop’s video updater feature, we finalized the mp4 format based on the above observations so that they can be viewed later in any browser without disruption.

CHAPTER 3

NEWS UPDATER FEATURE

Yioop has important processes like queue server, fetcher, media updater etc. Each of these processes accomplishes a different task and can be configured on any of the machines running the Yioop search engine. In this chapter, the news updater module of the media updater process is covered in detail.

3.1 Existing Architecture and Flow in Yioop

Yioop has a media updater [3] process that can be used to automatically update news feeds from various news sources. News feeds can either be RSS or Atom feeds, or can be scraped from an HTML page using XPath queries and re-indexed. This more timely information can then be incorporated into Yioop search results in order to improve the quality of the results.

Yioop can be configured to work in a single machine setting or a multiple machine setting. The media updater process was initially implemented in such a way that it was only running on the name server node of the Yioop server machine. I might alternatively refer to the single machine setting as ‘name server mode’, or vice versa. Also, the multiple machine setting is referred to as ‘distributed mode’ at some places.

The below diagram depicts the process flow when the news updater is working in name server mode. To summarize the diagram, the single machine/name server will periodically (once an hour as per current settings in Yioop) fetch the news feeds from the news sources added into the database, then will update the database and rebuild the index shard. These fresh news feeds will then be shown in news section of Yioop, as well as in the search results for any relevant query.

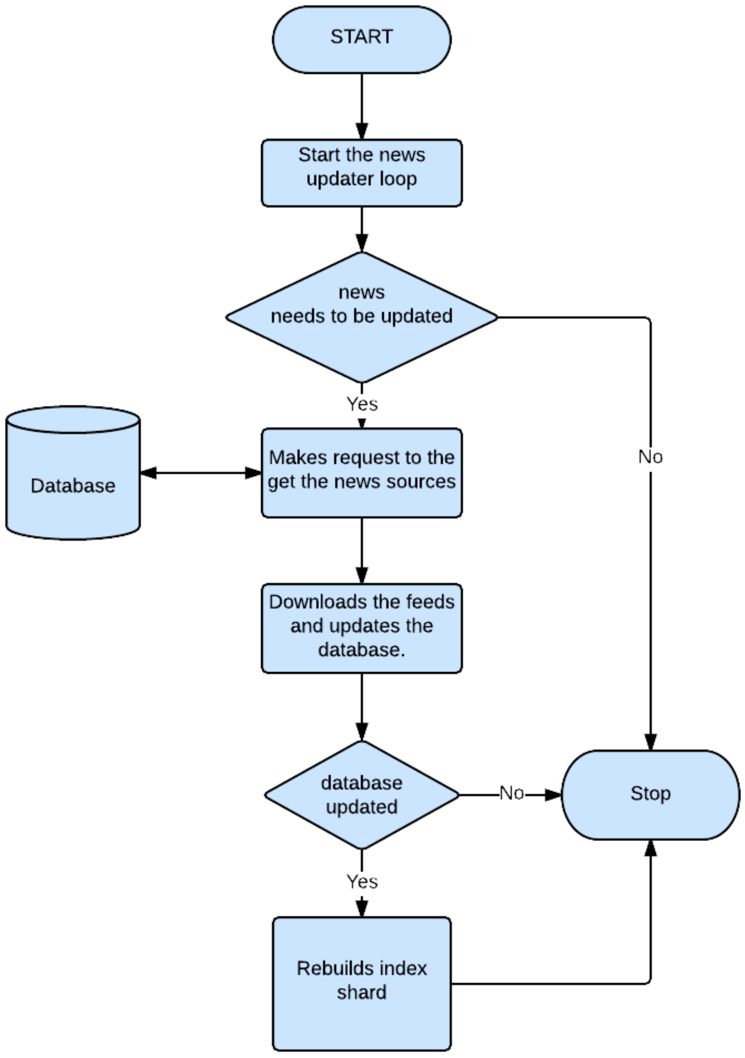

START
Start the news updater loop
news needs to be updated
No
Yes
Database
Makes request to the get the news sources
Downloads the feeds and updates the database.
database updated
No
Stop
Yes
Rebuilds index shard

Figure 2 :A flow diagram for existing news updater feature

A thorough code walkthrough[4] was done in order to understand the news updater activity from the ground up. It was implemented using model, controller classes interacting with each other and making database trips in order to keep it updated .

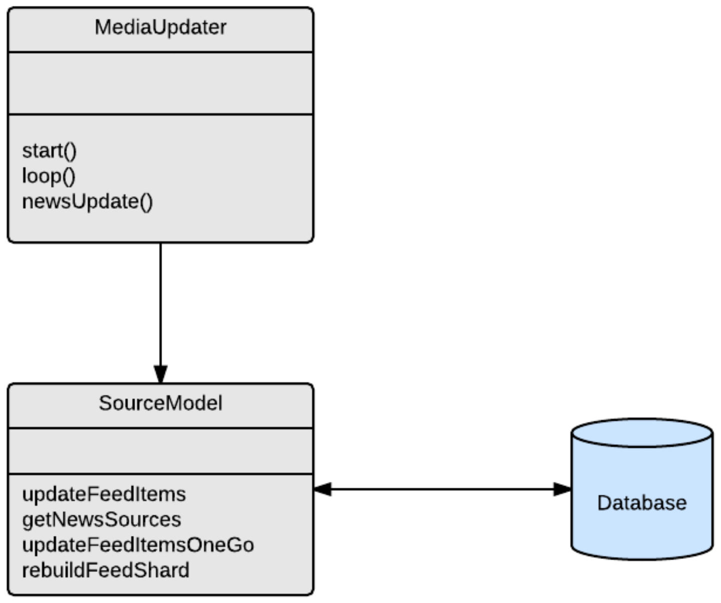


Figure 3: Class diagram for existing architecture

All the model and controller classes extend from the parent class Model and Controller, respectively. MediaUpdater is the main class that interacts with the sourcemodel class, which queries the database to get the news sources, then crawls them to download news feeds, update the database, and rebuild the index shard.

3.2 Experiments Conducted on the Existing Setup

After gaining a thorough understanding of the existing functionality, experiments were conducted to assess the performance of the news updater activity in the single machine setting. I downloaded the search engine, installed on the local machine, did a fresh clone of git, and created the working directory. The configuration used for testing is Intel core i5 with CPU @1.60 GHz , 6GB RAM and windows platform.

In Manage Machines, we added a single machine as local under Add Machine. I then restarted my computer to get Yioop working on the localhost. Through the Yioop web interface, I navigated to search sources and added a few news sources in the media section. Then, I went to manage machines and started the news_updater that is currently working on the name server and let the news_updater crawl and fetch feeds from the news sources.

I added few news sources (rss and html type) to the media sources and ran the news_updater by reducing its update time to 60 seconds (from the initial time of one hour).

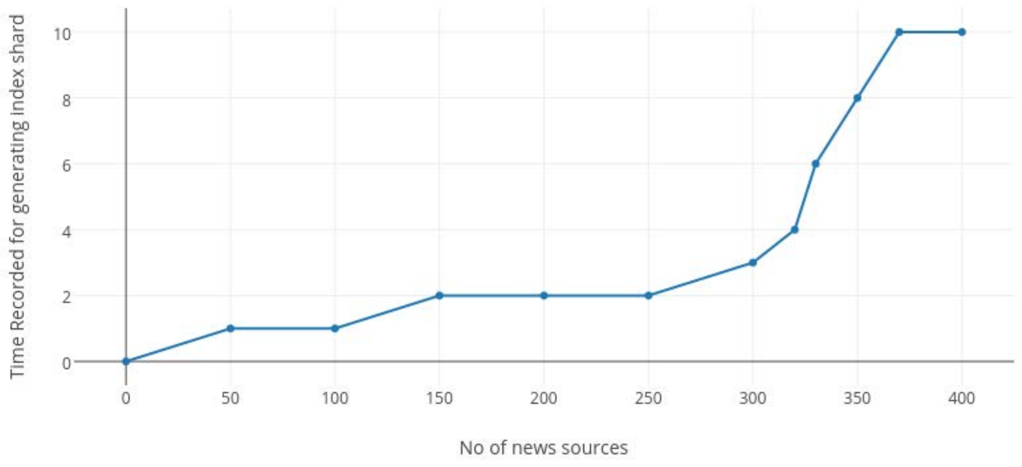


Figure 4: Line Chart for Existing News Updater Performance Testing

After adding few news sources, the news_updater was taking increasingly more time to update. After almost 275 news sources, it took more than 5-6 minutes to build the index file, and after 350 news sources, it took almost 10 minutes to generate the index file. We can easily predict that, as the number of news sources goes on increasing, that there will be an increase in the time it takes to build the index shard. Referring to this chart, we can make a rough estimate that in an hour, the single machine news updater can handle around 3-4 k news sources.

3.3 Scaling the News Updater Feature in Yioop

In order to decrease the time it takes to build the index shard of the news feeds as the number of news sources continue to increase, we implemented the news updater feature in such a way that now it can work on a cluster of machines. We experimented with different approaches and came up with solution that seemed the most feasible in terms of implementation and computing time.

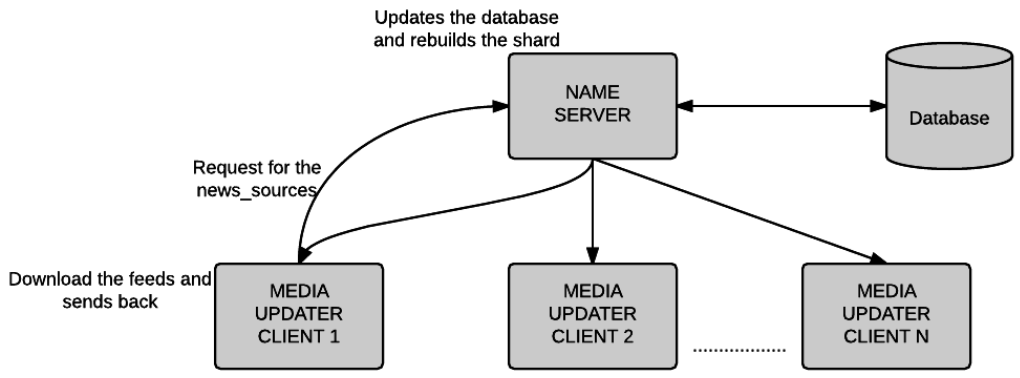

Updates the database and rebuilds the shard
NAME SERVER
Database
Request for the news_sources
Download the feeds and sends back
MEDIA UPDATER CLIENT 1
MEDIA UPDATER CLIENT 2
MEDIA UPDATER CLIENT N
....

Figure 5: Distributed news updater

The multiple machine setting consists of the name server node and slave nodes. All machines will essentially be running the same copy of program, though only the name server would query the database and distribute the news sources among the requesting slave machines. With reference to the above diagram, one can see that name server acts as single point for distributing the requested information, querying the database, etc.

We have used divide and conquer strategy and distributed the work among a number of machines that, at the end, can merge these results in one place. In this cluster, the name server node does all the bookkeeping-related work in order to divide and then merge the pieces of work/info submitted by other machines. Only one machine queries the database to get the news sources from where user-entered news sources can be fetched. It then passes them on to the other machines for the purpose of downloading feeds from them.

According to MVC architecture of Yioop[4], any class can make a simple xmlhttp request to the server to get a certain piece of information. In this case, this request can either be made directly, or one can make use of already available functions written in Yioop. Taking into consideration the already existing code, we implemeted the feature on distributed mode without disrupting the existing code's functioning.
Parallel Model class already had the capabilty to make an xmlhttp request to server . We simply extended from this class and made a request to the crawl_controller of the server for the purpose of retrieving the news sources.

As depicted by the class diagram below, we have added the extra class of crwal controller in order to make a request to the name server for the news sources. Once the requesting machines get the news sources, it just downloads the feeds and returns the feeds back to the server.

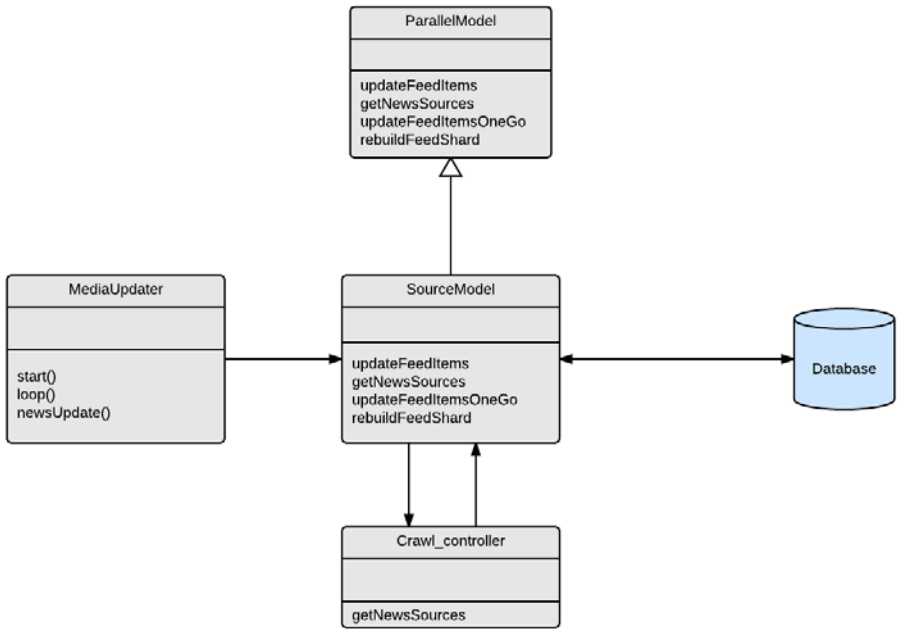


Figure 6: Class Diagram for news updater in distributed environment

3.5 Algorithm

We modified the news_updater functionality to distribute it over multiple machines so that each machine can independently run its own news_updater, make a request to the name server, and get the news sources from where it fetches the news.

News_updater will be running on all the machines except name_server at periodic intervals.

- A variable MULTIPLE_NEWS_UPDATER has been defined in the config.php file. If the value of this Variable is set to true, then the news_updater will work in the distributed fashion. If it is set to false, then it will run in single machine mode. One can also create local_config.php and set the variable from there, instead of setting it from the config.php file.
- If the variable MULTIPLE_NEWS_UPDATER is set to true, then the following events will occur:

- The machine will call the updatefeeditems method from source_model from its news_updater.
- The machine that makes the request to name_server for the news updater will be identified by reading the text file 'current_machine_info.txt' by the updateFeedItems method of source_model.
- updateFeedItems makes an exec call to getNewsSources function in crawl_controller.
    - The exec call is made through parallel_model, and current_machine hash string is appended to the request url.
- The getNewsSources function gets all the news_sources from the database, calculates their md5 hash, and puts in a feeds array that has the same length as the number of machines.
- After the hash values array for all the feeds is calculated, then the feed values corresponding to the requesting machine are returned to the calling function.
- Once the machine receives the feeds_array, it will resume its process and fetch the articles from these news_sources.

While scaling the news updater feature, we had to take into account certain issues, such as uniform distribution of the news sources among the requesting client machines. Google highlights the importance of maintaining such even distribution [7] [11]. This task was accomplished by devising a strategy. The requesting machine produces a text file that has the hash of current machine name. This information is passed when a request is made to the name server for news sources. At the end of name server, certain steps are carried out, which can be seen in the code snippet below. This ensures that all news sources are evenly distributed.

In order to get the news sources for a particular requesting machine, we follow a below strategy.

```
Get the current machine identifier from string
Match it against the array of machine identifiers present in given distributed setting
And save the index as machine_index_match
 For each feed
     Calculate the hash value and store it in hash variable
     then divide this hash variable with total no of machines
          present in order to round it up to at least one of the machine
       if this hash index matches with the machine_index_match then add it to the feeds array
     otherwise continue with next news feed;
```

The above pseudocode depicts how are we evenly distributing the number of news sources between multiple machines.

The flow of the news updater feature more or less remains the same, as per the diagram below. The only difference is that each machine checks if it is running in multiple machine setting. If so, then the requesting server gets the news sources.

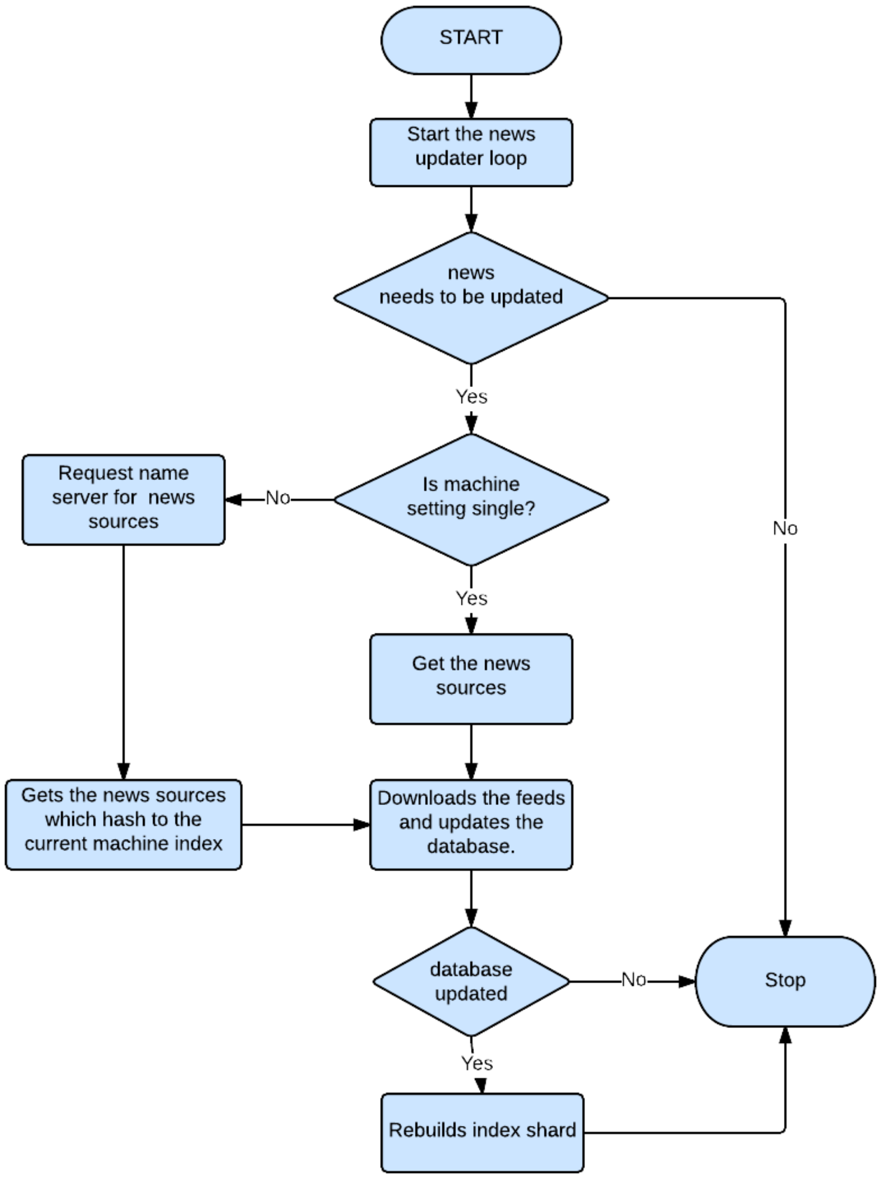

START
Start the news updater loop
news needs to be updated
Yes
No
Is machine setting single?
No
Request name server for news sources
Yes
Get the news sources
Gets the news sources which hash to the current machine index
Downloads the feeds and updates the database.
database updated
No
Stop
Yes
Rebuilds index shard

Figure 7 : A flow diagram for distributed news updater feature

3.4 Comparison with other social networking sites news feed feature

Google News is a computer-generated news site that aggregates headlines from news sources worldwide, groups similar stories together, and displays them according to each reader's personalized interests[12] .Their articles are selected and ranked by computers that evaluate, among other things, how often and on which sites a story appears online. They also rank based on certain characteristics of the news content, such as freshness, location, relevance, and diversity. As a result, stories are sorted without regard to political viewpoint or ideology, and you can choose from a wide variety of perspectives on any given story. Based on your search queries, Google offers a section called ‘suggested for you,’ which contains the news feeds related to your search queries and interests.

The stories that show in your News Feed are influenced by your connections and activity on Facebook. If any of your connections view, post, or like any stories, they will also appear in your news feed. Facebook also provides customized setting options for its news feed, which more or less filter out the news feeds before presenting them to you. Yahoo also provides news feeds based on various categories, but it does not have any customized categories like Google.

Yioop differs from the above engines in the way in which the list of video and news sites can be configured through the GUI. This way, we can keep our news sources from filtering out unwanted content. Scaling the news updater feature on Yioop has the added advantage of being able to add more news sources without worrying about the time it takes to generate the news feed. We can add machines depending on our needs.

3.5 Conclusion

A distributed environment can help speed up operations by running processes in a parallel fashion. One can fully utilize the computing resources of the distributed environment in order to dramatically reduce the time that would otherwise be taken on the single machine setting. In the future, Yioop will be able to switch to the multiple machine setting or the single machine setting depending on the requirements of the news updater feature.

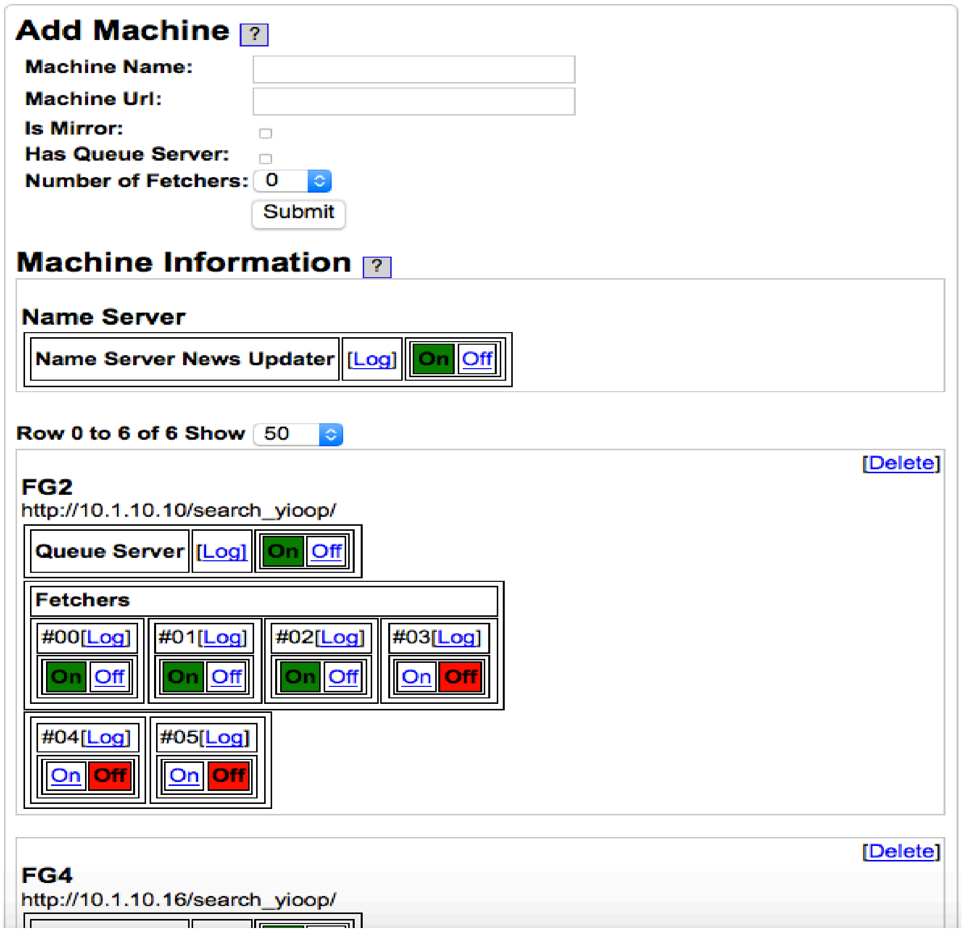
CHAPTER 4

MANAGE MACHINES WEB INTERFACE

A graphical user interface is always an integral part of any application. In today's era of Web applications, web interfaces play a crucial role. Yioop has a web interface that was mainly developed using html embedded within php. It is in line with the MVC architecture of Yioop. In this section, I will cover the existing web interface of Yioop, as well as proposed changes and their implementation details.

4.1 Existing Web Interface in Yioop

In Yioop, machine statuses and the statuses of processes running on these machines can be seen through the web interface. In Yioop, it is possible to start or stop and view the log files of the queue servers and fetchers through the Manage Machines activity. One can configure multiple queue servers and/or fetchers on a machine. The media updater is by default part of every Yioop instance, irrespective of its part in the cluster. One has only the ability to start or stop it like the other processes.


Add Machine ?

Machine Name:	
Machine Url:	
Is Mirror:	☐
Has Queue Server:	☐
Number of Fetchers:	0
	Submit

Machine Information ?

Name Server

Name Server News Updater	[Log]	On	Off

Row 0 to 6 of 6 Show 50

[Delete]

FG2
http://10.1.10.10/search_yioop/

Queue Server	[Log]	On	Off

Fetchers

#00[Log]	#01[Log]	#02[Log]	#03[Log]
On Off	On Off	On Off	On Off
#04[Log]	#05[Log]		
On Off	On Off		

[Delete]

FG4
http://10.1.10.16/search_yioop/

Figure 8: Snapshot of the existing GUI for the Manage Machines activity

4.2 Proposed Changes in Manage machines activity

This deliverable is related to the previous deliverable in which we want the media updater process to be a part of all machines. Media updater is one process which will run on all the instances of Yioop by default. As with the others, one will have the ability to start or stop and view the log files of media updater. It also gives one the ability to see how many machines are in the cluster and which are currently running the media updater. We designed certain mockups using html, then decided on the final look and feel of the Manage Machines activity.

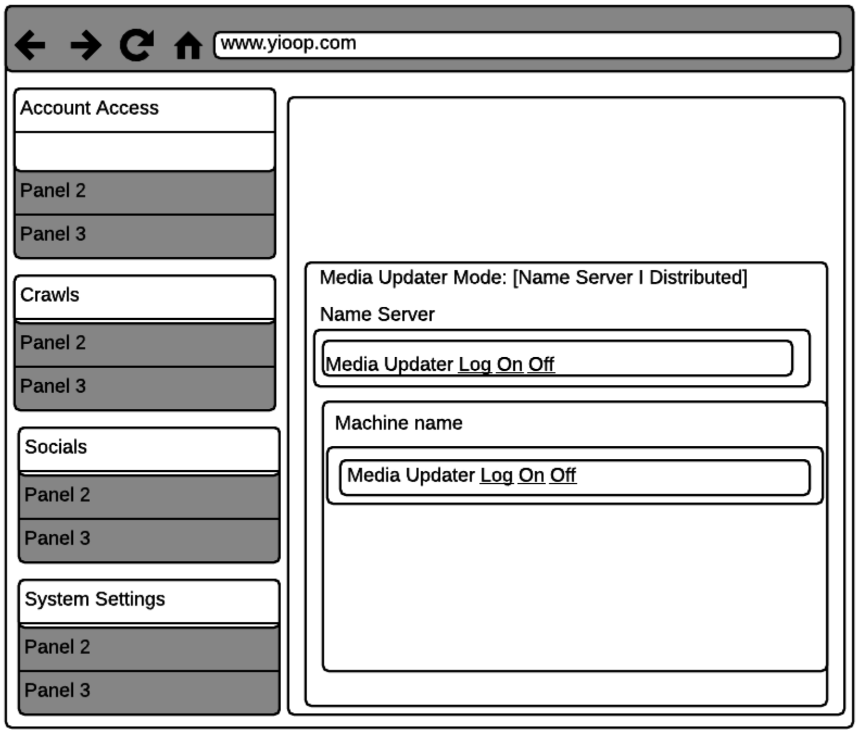


Figure 9: Wireframe for proposed changes

4.3 Changed Web Interface in YIOOP

We modified the existing code to accommodate the changes. Thee manage machine web interface will now be able to show the media updater process status on the Yioop instances. We created the following table in the database with table

MEDIA_UPDATER_PROPERTIES

Table 4 : Database Table Format for Manage Machines Activity

NAME	VALUE
PRESENT_ON_NAME_SERVER	TRUE/FALSE

Based on this value, machine_model will fetch the value from the database and render the view from machinestatus_view. The communication is done throughs admin controller, where machine_model puts fetched data from the database into the data array of admin controller. machinestatus_view then renders view based on this data.

The implementation mainly involved related classes to query the database and put the data in the appropriate storage structure. The view then refers to this data and renders the page.

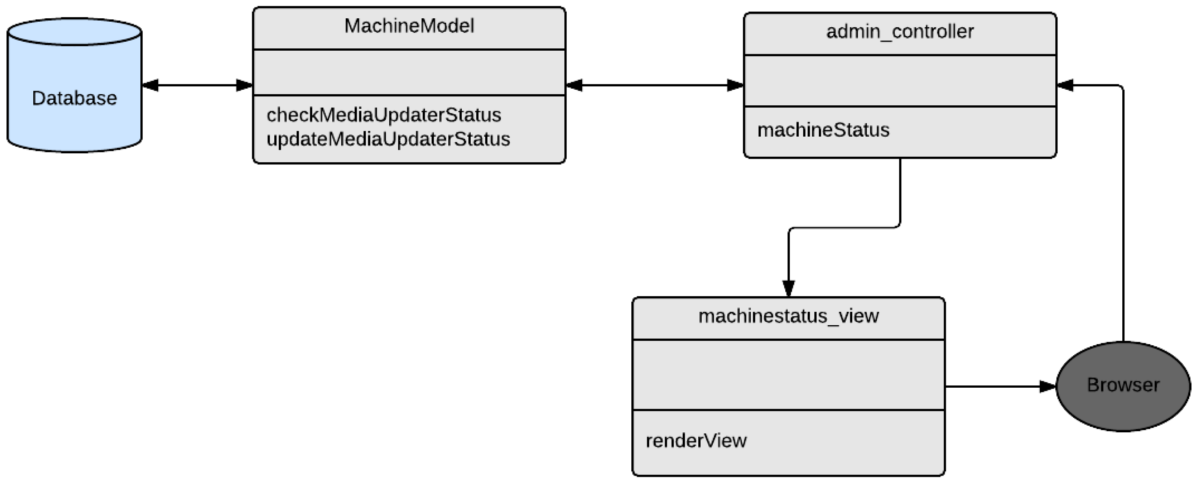


Figure 10: Class Diagram for Proposed Web Interface

One should also be able to switch between the view based on their choice of Media Updater Mode, such as Distributed or Name Server. The flow of the entire manage machines table is depicted in the diagram below. The ability to switch from Name server to Distributed mode gives the user the flexibility to keep the media updater running on as many machines as he/she wants.

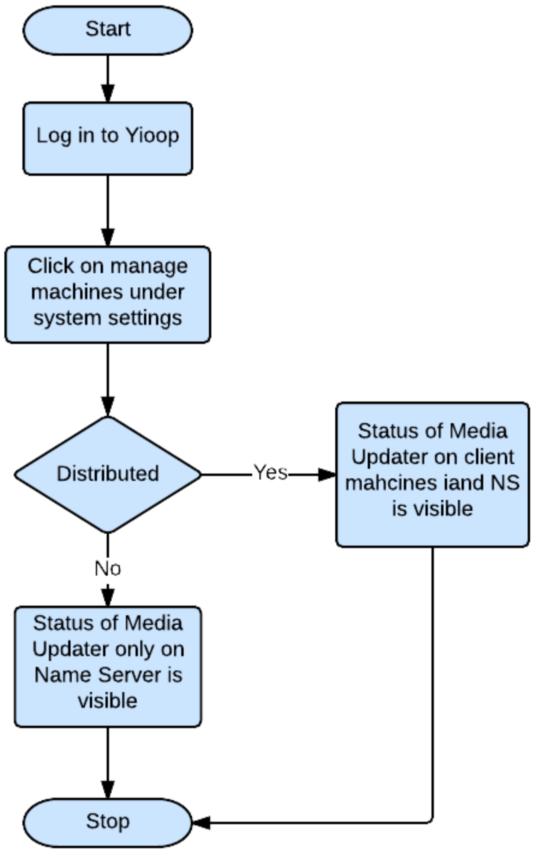

Start
Log in to Yioop
Click on manage machines under system settings
Distributed
Yes
Status of Media Updater on client mahcines iand NS is visible
No
Status of Media Updater only on Name Server is visible
Stop

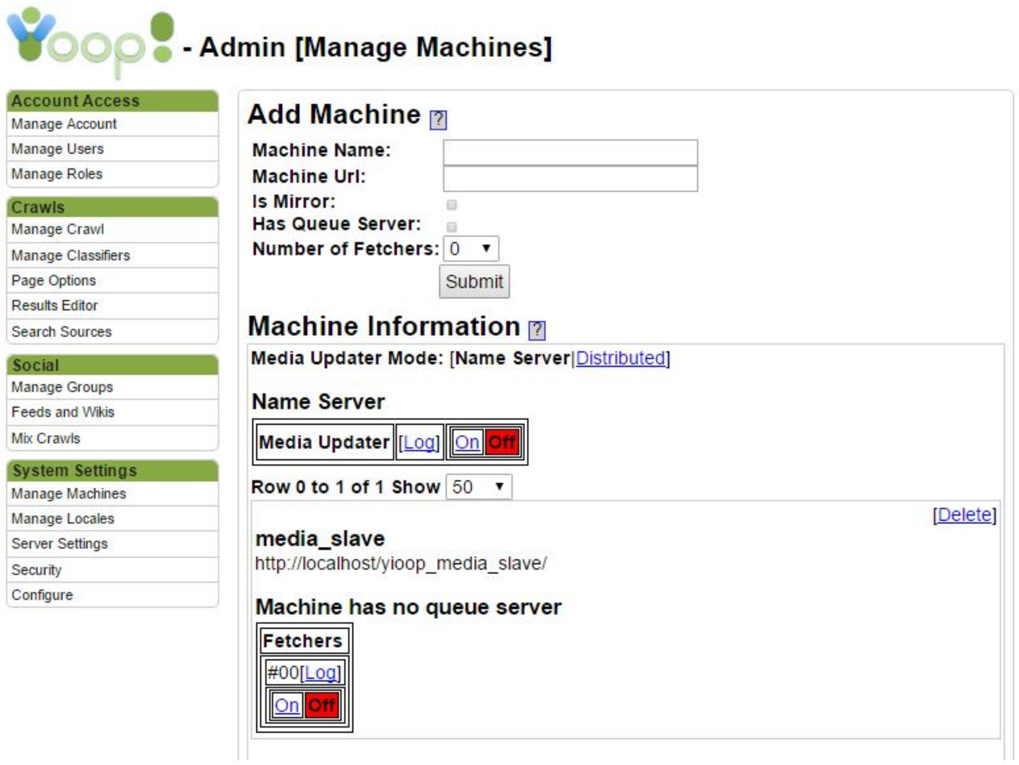
Figure 11 : A flow diagram for the Manage Machines activity

In case of single machine setting, even though the user has both of these options, it really doesn't make any difference. It is worth noting that the media updater is running on name server even in distributed mode, but in name server mode, it is running only on server. The reason behind the above setting is that the media updater process of server has some specialized functions to execute (discussed in chapter 5), so we have to keep it running in either case.

After implementing the new interface, it now looks like the screenshots below.

The new web interface now looks like:

a) Name Server Mode



Add Machine

Machine Name:	
Machine Url:	
Is Mirror:	
Has Queue Server:	
Number of Fetchers:	0

Submit

Machine Information

Media Updater Mode: [Name Server|Distributed]

Name Server

Media Updater [Log] On Off

Row 0 to 1 of 1 Show 50

[Delete]

media_slave

http://localhost/yioop_media_slave/

Machine has no queue server

Fetchers

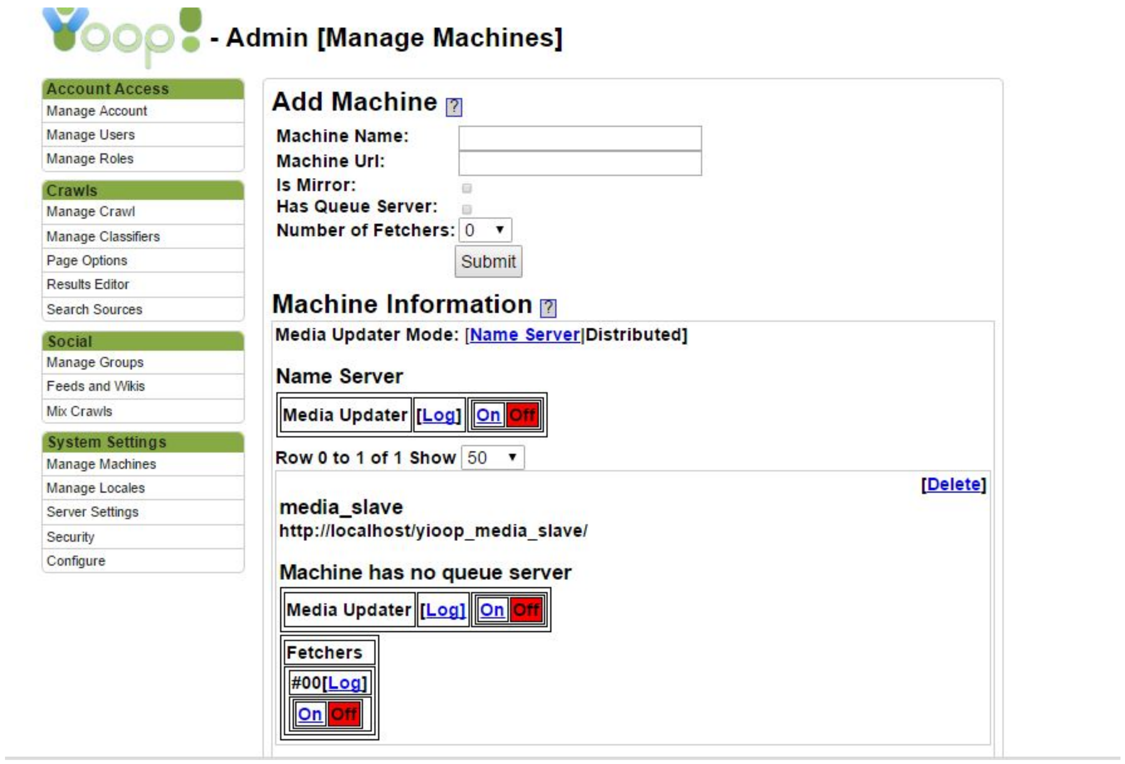
#00[Log]

On Off

Figure 12: Snapshot for Manage Machines activity in name server mode

In this mode, the media updater will only work on the name server. The Yioop user can control the status of the media updater on the name server.

b) Distributed Mode:



Machine Name:
Machine Url:
Is Mirror:
Has Queue Server:
Number of Fetchers: 0

Media Updater Mode: [Name Server|Distributed]

Name Server

Row 0 to 1 of 1 Show 50

media_slave
http://localhost/yioop_media_slave/

Machine has no queue server

Figure 13: Snapshot for Manage Machines activity in Distributed mode

The name server will be visible in both the cases, as the media updater on the name server has to perform some explicit tasks that can’t be interrupted.

4.4 Conclusion

Yioop can now switch between name server mode and name server mode setting for viewing the status of and controlling all running processes.

CHAPTER 5

VIDEO UPDATER FEATURE

Many search engines have the capability to upload videos and view them later. One of the most popular examples is YouTube. In this section, I will cover the purpose, attempted approaches, an evaluation of these approaches, tools used, and the implementation details of the chosen approach.

5.1 Purpose

In the media_updater, there is a provision for uploading videos and different video sources. For uploaded video[15] files below a configurable size limit, videos are automatically converted to web-friendly mp4 formats. We have discussed the reason behind choosing avi and mov file formats to mp4 file format at length in background work.

To convert such video formats to a different format, we can use a tool called ffmpeg, which is described above. Using this Linux utility, a script can be used to convert the uploaded video formats to a particular format, such as mov, and then upload them back on the server for the users to watch. As part of this deliverable, we have designed the architecture for the video uploader, which will be part of the Media uploader only.

5.2 Approaches

5.2.1 Server converting video as a whole

In this approach, only the name server will be working converting the videos and uploading back the converted file.

Pros:

1. Easy to implement.
2. Less bookkeeping
3. No loss of data

Cons:

1. Will take longer in terms of time.
2. Name server's other functions will also be interrupted and delayed during the media conversion.

5.2.2. Distributed setup converting split video as a whole

In this approach, we decided to scale the previous approach so that it can work in a distributed environment. Each video will be split into multiple video files, and then sent to slave media updaters for conversion. Large video files are split into small files primarily to facilitate of the uploading and downloading of the video file for conversion purposes by the slave media updaters.

We decided on the functions/modules which will be assigned to different Yioop instances.

Table 5 : Functions Assigned based on the Role of Machine in Cluster

NAME SERVER	MEDIA UPDATER SLAVE
Look for Videos to Split	Convert
Split	
Looking for converted videos	
Concatenate	
Generate assemble files for conversion	

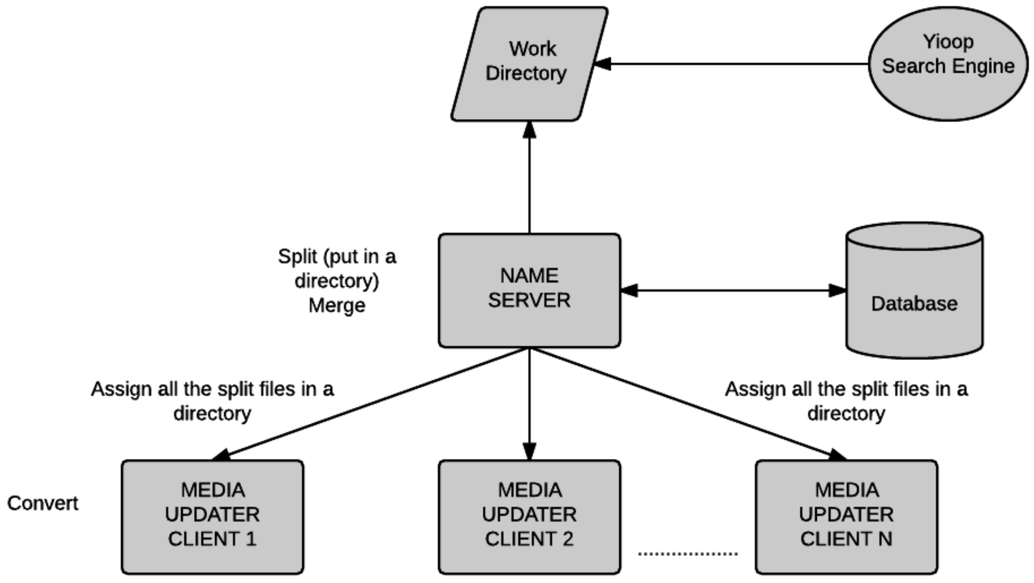


Figure 14 : Distributed setup converting split video as a whole

Pros :

1. Easy to implement
2. Less bookkeeping
3. Greater ease of downloading and uploading the video files for conversion.

Cons :

1. Videos are assigned in the order that they are assigned for conversion.
2. Longer video files assigned to a media updater will take longer even if they were assigned first. A user might end up waiting for the video to be uploaded back, even if it he/she uploaded it before another.

5.2.3 Distributed setup converting split video

We expanded the distributed setup so that instead of assigning all the split files belonging to one video to only one media updater, we now assign any split video file to any media updater based on the video time. This way, there is an overhead of bookkeeping at the end of the name server node in order to put the relevant files back.

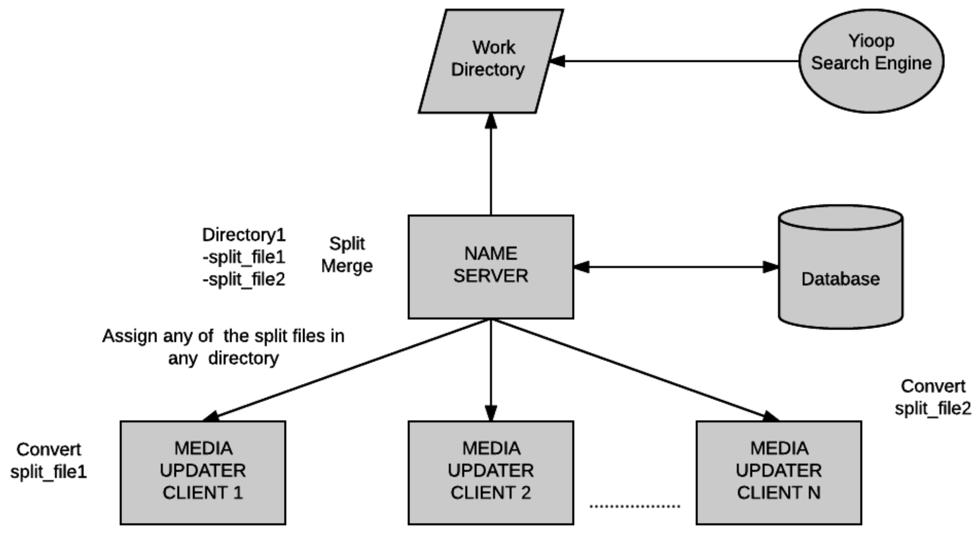


Figure 15 : Architecture diagram for Distributed setup converting split

Pros :

1.Time reduction for the video conversion.
2. Easy to upload and download split video files to and from name server.

Cons :

1. Difficult to implement
2. Lots of bookkeeping on the server side.

5.3 FFmpeg and CURL

FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter, and play pretty much anything that humans and machines have created. It supports everything from the most obscure ancient formats up to the cutting edge.

The object of much research over the past few decades has been the elaboration of new algorithms that manage to reduce the quantity of data needed for audio and video, while also

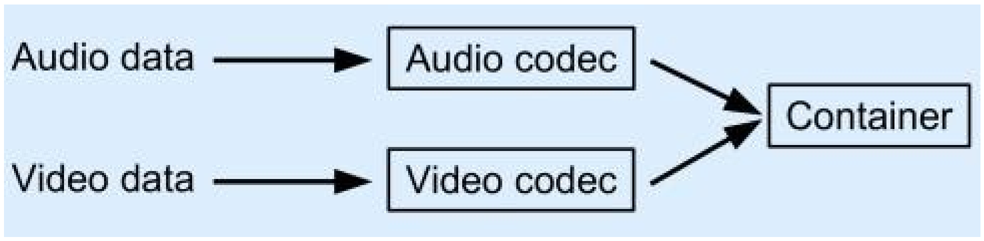


Figure 16: Video Container Format

reducing the damage done to the picture and sound as a result of compression. These algorithms, called codecs, allow us to encode the data in order to transport it and then decode the data at the other end.Several codecs are contained in the libavcodec library supplied with FFmpeg. Others are provided by LAME, libgsm, XviD, AMR and libvorbis.

Once the audio and video streams have been encoded by their respective codecs, this encoded data needs to be encapsulated into a single file. This file is called the container. One particular type of container is AVI (Audio-Video Interleaved). AVI is only a method of mixing the encoded audio and video together in a single file. It contains data that informs the media player what audio and video codecs were used. There are many different possible combinations of codecs that can be used within each type of container, which explains why a system may play back some AVI files and not others. The system can extract the encoded audio and video data from the AVI file, but it can decode that encoded data only if the required codecs are present.

FFmpeg [9] is a free software project that produces libraries and programs for handling multimedia data. FFmpeg includes libavcodec, an audio/video codec library used by several other projects, libavformat, an audio/video container mux and demux library, and the FFmpeg command line program for transcoding multimedia files.

Playback of a multimedia file operates in the exact opposite direction. Once the container used is identified, it tells us which codecs are needed to decode the data. The audio and video streams are then extracted from the container and fed through the appropriate codecs. At the other end, we get raw audio and video data that can be fed to the audio and display subsystems of the computer.

FFmpeg is a tool that, in its simplest form, implements a decoder and then an encoder, thus enabling the user to convert files from one container/codec combo to another. The original container is examined, and the encoded data is extracted and fed through the codecs. The newly-decoded data is then fed through the target codecs into the new container.

FFmpeg can also do a few basic manipulations of the audio and video data just before it is re-encoded by the target codecs. These manipulations include changing the sample rate of the audio and advancing or delaying it with respect to the video.

We studied different ffmpeg commands for converting videos from one format to another.

We wrote a few required modules for the purpose of video manipulation.

Convert command

```
ffmpeg -i sample.mov -vcodec h264 -acodec aac -preset veryfast -crf 28  -strict -2 sample.mp4
```

This command converts mov video to mp4 video format.

Split command:

```
ffmpeg -i sample_iTunes.mp4 -acodec copy -f segment -segment_time 50 -vcodec copy -
```

```
reset_timestamps 1 -map 0 -an sample_iTunes%d.mp4
```

Concatenate command:

```
ffmpeg -f concat -i  list.txt -c copy output.mp4
```

This command simply merges the files listed in the text file into one text file.

cURL:

cURL [10] is a library that lets you make HTTP requests in PHP. PHP supports libcurl, a library created by Daniel Stenberg. libcurl allows you to connect to and communicate with many different types of servers with many different types of protocols. libcurl currently supports the http, https, ftp, gopher, telnet, dict, file, and ldap protocols. libcurl also supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading (this can also be done with PHP's ftp extension), HTTP form-based upload, proxies, cookies, and user+password authentication. We have used curl to download and upload files using http.

5.4 Video Updater in YIOOP

The class diagram explains the classes that have been modfied for the implementation of the Video updater. Primarily, most of the video updater functions are included in the media updater as one video uodater module. These functions then call the other auxillary functions written in model and controller classes as needed.

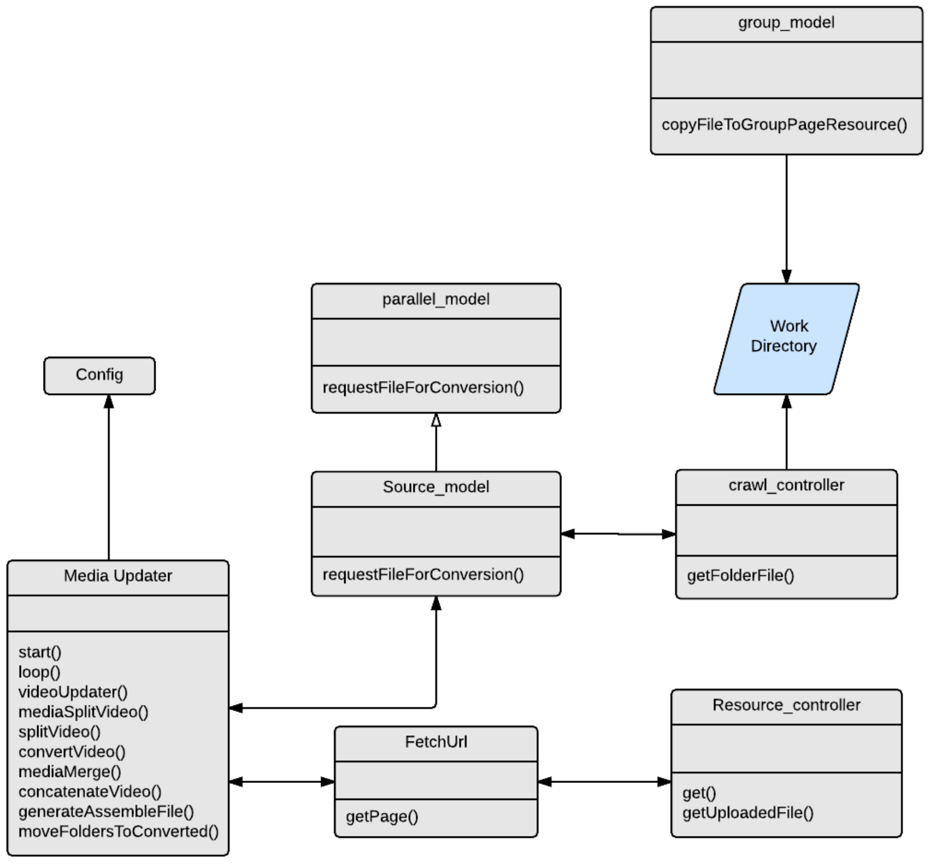


Figure 17 : Class Diagram for video updater future

The media_updater process will run on several yioop instances simultaneously in the distributed environment.

Based on which Yioop instance it is running, it follows different processes.

a) Name server process

   The name server process is only responsible for a few functionalities. The name server does not convert videos. The video updater can only work in distributed mode. This is the reason that the name server has to be kept running in either mode, since it also does bookkeeping.

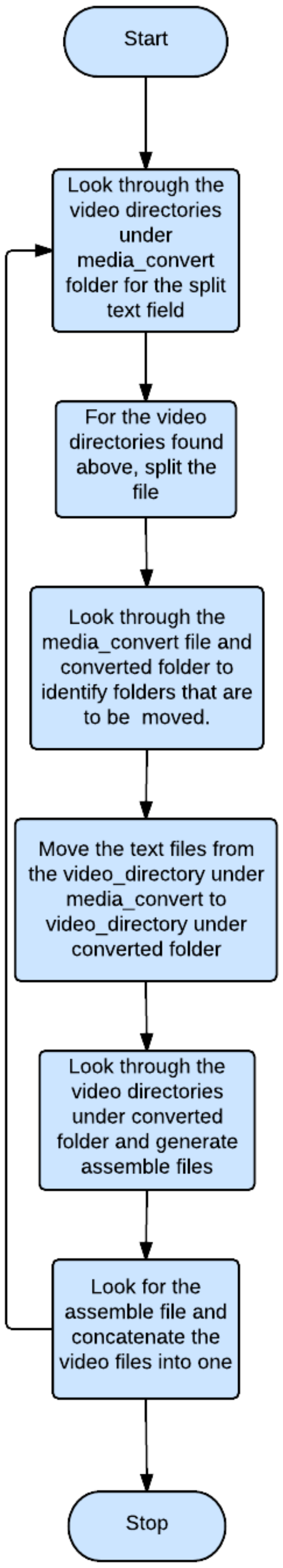

Start
Look through the video directories under media_convert folder for the split text field
For the video directories found above, split the file
Look through the media_convert file and converted folder to identify folders that are to be moved.
Move the text files from the video_directory under media_convert to video_directory under converted folder
Look through the video directories under converted folder and generate assemble files
Look for the assemble file and concatenate the video files into one
Stop

Figure 18: A flow diagram for the name server media updater process

b) Media updater slave process

The slave process is pretty simple. It only converts videos that it receives from the name server. Most of the intelligent work is done by the name server.

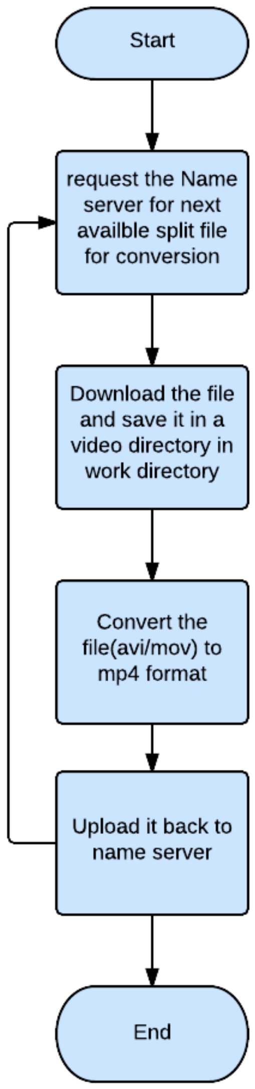

Start
request the Name server for next availble split file for conversion
Download the file and save it in a video directory in work directory
Convert the file(avi/mov) to mp4 format
Upload it back to name server
End

Figure 19: A flow diagram for the media updater slave process

In order to avoid the name server having to go through same video directories looking for certain things, we have devised the below process, where it can decide whether to discard the video directory or continue with it.

Table 6 : Text Files based on the Role of Machine in Cluster

Function task	**Look for**	**Generate**	**Delete**
Media split	Split.txt		Split.txt
Move folders to converted	Count.txt		
Generate ready to assemble file	Concatenated.txt Ready to assemble.txt	Ready to assemble.txt	
Media merge	Concatenated.txt	Concatenated.txt	All text files

The above text files serve as a way to communicate or convey information regarding what could be the next step for the name server, as well as help reduce the time taken by the name server to go through them and perform the required tasks.

Also, in order to convert videos in the order that they have been assigned for split, we implement the below strategy.

1. Video directories are named based on the time they are placed into the work directory of the name server by the group model.
2. When a request is made to the name server for the file conversion, the following steps are followed.

```
For each video_directory placed in video directory
{
     For each file in the video directory
     {
          If the filename contains word ‘taken’
               {
               Check the timestamp
                    If the timestamp is older than 5 minutes
                    {
                         Rename this file with current timestamp
                         Send video directory name and file name
                    }else {continue}
               }else
               {
                         Rename this file with current timestamp
                         Send video directory name and file name
               }
     If result is not empty break out of the loop

     }

If result is not empty break out of the loop

}
```

5.5 Conclusion

Video updater is now introduced in Yioop, which converts video formats from one type (mov and avi) to another (mp4) efficiently using the distributed setup.

CHAPTER 6

MAIL DISTRIBUTION

One can create groups in Yioop and start group discussions. People who are subscribed to the group are notified via email if there is new activity in a group. Currently, Yioop webapp sends out these emails in real time as soon as new activity occurs. If there are too many people subscribed to a group then, there the process could time out resulting in some people not being notified. Webapp sending emails will result in rendering the Yioop pages slowly and might even result in Denial of Service attacks. In this chapter we will discuss how to aggregate these emails and send them periodically in the media updater process.

6.1 Existing Email Functionality in Yioop

In Yioop [4], when a user starts a new thread or comments to an existing thread, several people will be notified via email provided they have their email address configured properly. For a new thread, the groups' owner will be notified if they themselves weren't the ones starting the thread. If it was in fact the group's owner that starts a thread, then everyone who belongs to the group will be notified. For a comment to a thread, excluding the commenter, the group's owner, the person who started the thread, and anyone else who has commented on the thread would be notified via email.

If Email Activation is chosen, then the user can specify the email address that the email address for notification. The checkbox Use PHP mail () function controls whether to use the mail function in PHP to send the mail, this only works if mail can be sent from the local machine.

Alternatively, if this is not checked like, one can configure an outgoing SMTP server and send the email through it. Sending emails immediately can result in traffic congestion if too many people are subscribed to the group. We worked on aggregating such emails and sending them periodically. This solution can be scaled so that it can work on multi machine set up.

6.2 Aggregating the mail distribution

We aggregated these mails to be sent in a text file and scheduled them to be sent by media updater. Media updater periodically keeps on checking if there are any emails to be sent. Based on the current setting (single/ multi machine) these emails are sent. Below diagram explains the proposed solution. This solution is designed to work in both single as well as multi machine settings.

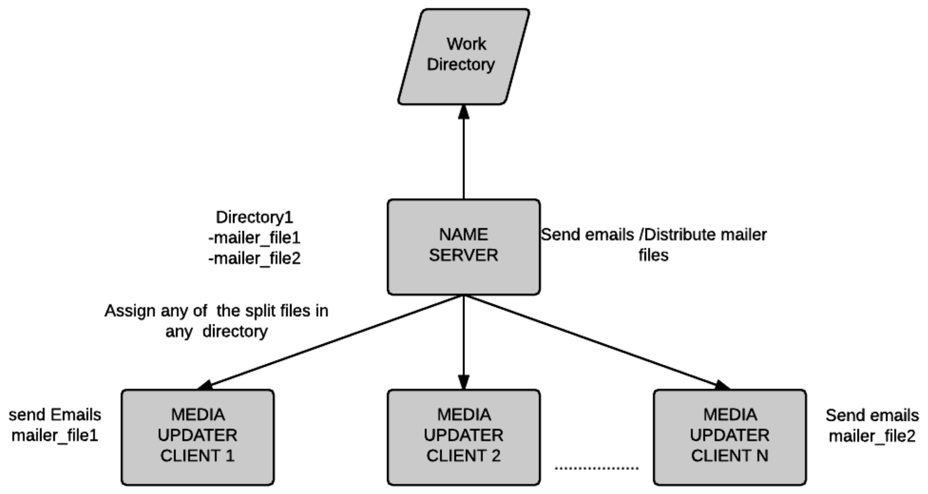


Figure 20: Architecture Diagram for Proposed Solution

We set one global variable (SEND_MAIL_MEDIA_UPDATER) in config file. If this variable is set to true then Mail server writes the email details (to, from, subject, message) into a text file which is created periodically (every 5 minutes as of now and this time can be changed in config file).Below is the class diagram giving implementation details.

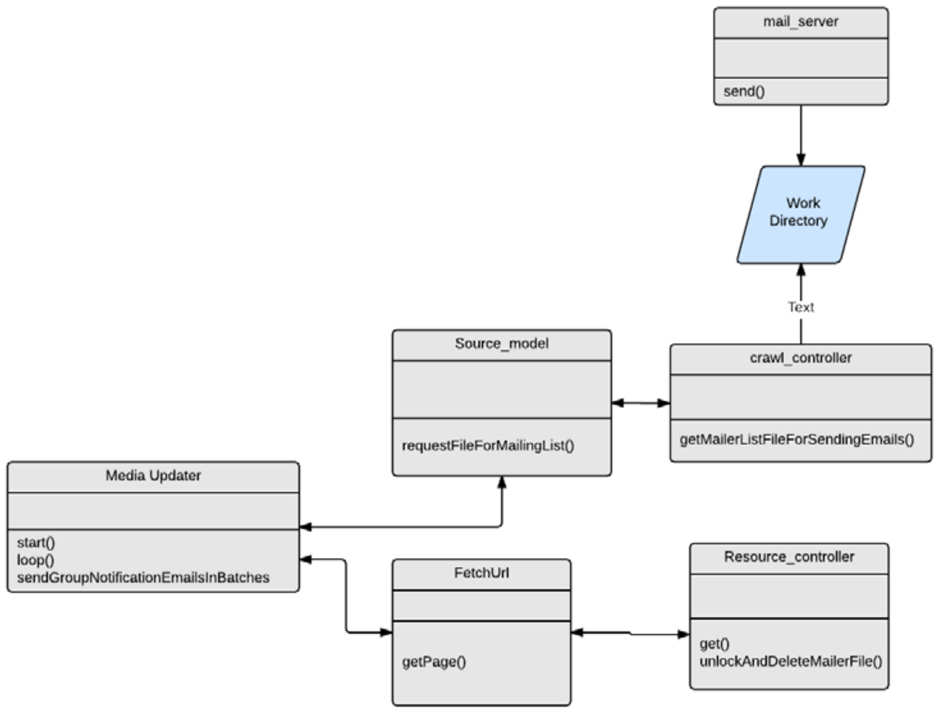


Figure 21: Class Diagram for Mail Distribution

The mailer text files produced by mail_Server are saved into schedules/mail folder of work directory. Media updater when invoked checks for pending emails (if any mailer files are present in the mail folder). If any files are present, based on setting and the Yioop instance, either name server will send the emails or cater to the requested of slave media updaters to distribute the mailer files.

6.3 Locking mechanism to Avoid Race Conditions

Mail server and name server would be accessing the same files from mail folder. To avoid the race conditions, locking mechanism in php is used. When mail server is writing into a file, first a lock is obtained and released only when it is done writing. Similarly, for name server, it establishes lock on a file before reading into it. If any of the process is unable to acquire lock on the file, it will either wait or will move on to the next file.

The flow of mail distribution is given in below flow diagram for both the cases –single as well as multi machine setting.

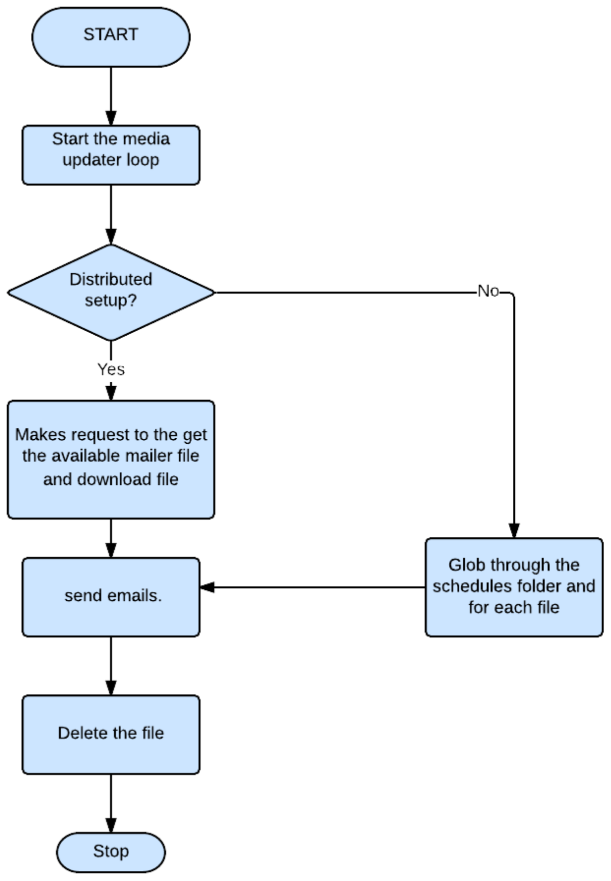

START
Start the media updater loop
Distributed setup?
No
Yes
Makes request to the get the available mailer file and download file
Glob through the schedules folder and for each file
send emails.
Delete the file
Stop

Figure 22: A Flow Diagram for Mail Distribution

6.4. Conclusion

The mail server will be aggregate emails to be sent over a period of time and then media updater will send these emails out periodically. This will serve the purpose of offloading the task of sending emails from the Yioop webapp, so that it can serve webpage requests quickly, thus resulting in improved performance. This will also help in preventing denial of service attack. Furthermore, scaling this task will reduce the burden of sending emails only from the name server.

CHAPTER 7

EXPERIMENTS AND INTEGRATION

After integrating [16] media updater features into Yioop, I conducted some experiments in a real distributed environment.

7.1 Experiment Setup

I created a cluster on AWS of a few machines. The configuration for machines was as follows:

1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory

Platform : Ubuntu

I set up Yioop on one such instance and got Yioop running on one of them. Then I created clones of this instance using the AMI option[14]

7.2 News Updater Performance Testing

I had already done testing in name server mode for the news updater . After distributing the code, I manually added the news sources [6] to the name server, kept adding slave machines to the cluster, and recorded the time it took to build the index shard, allowing me to see how it was being reduced.

Table 7 : Time recorded for News Updater Performance Testing

No of news sources	**No of machines in the cluster**	**Time taken (in seconds)**
100	1	150
	2	90
125	1	150
	2	80
	3	40

200	1	160
	2	80
300	1	160
	2	80
350	1	160
	2	90
400	2	80
	3	30-35
500	2	85
	3	30-35
550	1	170
	2	85
680	1	190
	2	90
	3	65-70
700	2	98-100
	3	50-60

Looking at above table, we can see that as the number of news sources increases, the time required to build the index shard steadily increases. However, this time can be reduced exponentially as a number of machines are added into the cluster. Two machines in the cluster reduce the time almost exactly by 50%, which proves that the news sources are being distributed evenly amongst them.

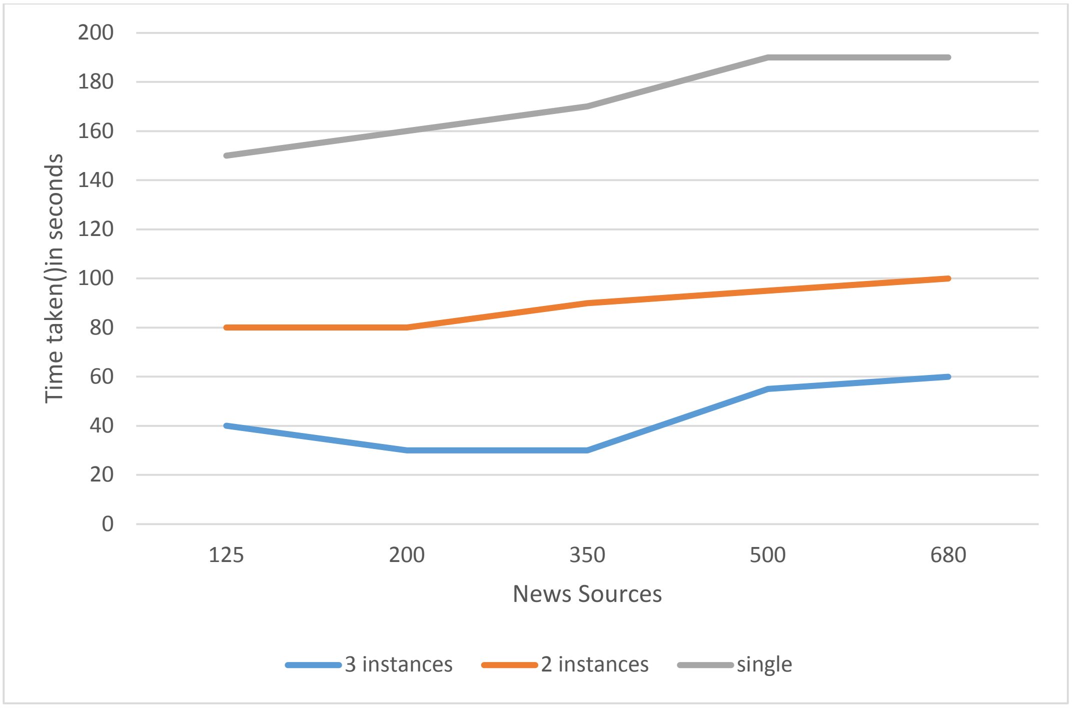


Figure 23: Line chart for Comparison of News Updater Performance Testing

7.3 Video Updater Performance Testing

For video updater testing, I used the above distributed setup and recorded the time it took to convert the videos. We split each video into 5-minute segments. They were then converted by different slave media updaters, then finally merged by the video updater. From the recorded times of the video conversions in the below table, we can say that the video updater converts a video very quickly.

Table 8 : Conversion Time Recorded for Increasing Length of Video

Length of videos	**No of machines**	**Time (in seconds)**
1 min	1	5
7-8 mins	1	15
14-15 mins	1	35
	2	15-20

24 mins	1	90
	2	25
	3	20
50 mins(500 Mb)	1	300
	2	180

CHAPTER 8

CONCLUSION

In this project, we have presented a media updater for Yioop. This updater supports the news updater and video updater features. We scaled these features so that they can work in a multiple machine setting, which effectively reduced the time.

News updater can also cater to large numbers of news feeds without worrying about the time constraint. This will help Yioop to fetch feeds from many sites even more frequently and keep them updated without worrying about the impact on its other functionalities. These fresh news feeds can then be used in Yioop’s search results. The user will also be able to manage media updater processes running on ta number of machines. One can start or stop it when required.

The video updater feature is responsible for converting to the video format (mp4) so that it is visible in all browsers. Since this feature is written with the distributed aspect in mind, the user can upload lengthy videos without worrying about the time Yioop will take to convert and upload back for viewing purposes.

The mail distribution feature will give Yioop ability to aggregate the emails over a period of time and then send them in batches in single or multiple machine setting.

Currently, the video updater converts only the most widely-used video formats, such as mov and avi. In the future, we can add more formats to this list.

REFERENCES

[1] Yioop website. Retrieved September 2, 2014.
From http://www.yioop.com/

[2] Seek Quarry website. Retrieved September 2, 2014.
From http://www.seekquarry.com/

[3] Yioop Documentation. Retrieved September 2, 2014, *Seek Quarry web site*.
From http://www.seekquarry.com/?c=main&p=documentation

[4] Yioop Source code Documentation. Retrieved September 13, 2014, *Yioop web site*. From http://www.seekquarry.com/yioop-docs/

[5] News feeds. (n.d.). Retrieved September 20, 2014, *Facebook web site*. From https://blog.bufferapp.com/facebook-news-feed-algorithm

[6] News Piler. (n.d.). Retrieved April 29, 2015, *Seek Quarry web site*. From http://newspiler.com/

[7] David Karger, & Matthias R. (n.d.). *Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems*. Retrieved October 25, 2014. From https://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/33339.pdf

[8] Adam S. Jeff T., Brian F. C., Raghu R. *Feeding Frenzy: Selectively Materializing Users' Event Feeds* Retrieved October 30, 2014

[9] Ffmpeg Retrieved March 13, 2015, *Ffmpeg web site*.
From https://www.ffmpeg.org/

[10] PHP curl Retrieved March 20, 2015, *PHP Curl Documentation Website*
From http://php.net/manual/en/book.curl.php

[11] Stamp.M, Wiley (2011) Information Security: Principles and Practice, (2nd ed.).

[12] Google news ranking stories Retrieved March 13, 2015
http://searchengineland.com/google-news-ranking-stories-30424

[13] Shumeet B. ,Rohan S., D. Sivakumar,Yushi J. ,Jay Y., Shankar K., Deepak R., Mohamed A. *Video Suggestion and Discovery for YouTube: Taking Random Walks Through the View Graph* Retrieved March 13, 2015

[14] Amazon Web Services EC2 services Retrieved April 30, 2015, *AWS EC2 Website* From http://aws.amazon.com/ec2/

[15] Understanding video file Retrieved March 02, 2015 http://www.dpbestflow.org/Video_Format_Overview

[16] Yioop Coding Guidelines. Retrieved September 2, 2014, *Seek Quarry web site* From http://www.seekquarry.com/?c=static&p=Coding#Issue Tracking/Making Patches/Commit Messages