USING HADOOP TO CLUSTER DATA IN ENERGY SYSTEM

Thesis

Submitted to

The College of Arts and Sciences of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Master of Computer Science

By

Jun Hou

UNIVERSITY OF DAYTON

Dayton, Ohio

May, 2015
USING HADOOP TO CLUSTER DATA IN ENERGY SYSTEM

Name: Hou, Jun

APPROVED BY:

____________________________
Zhongmei Yao, Ph.D.
Advisor Committee Chairman
Associate Professor, Department of
Computer Science

____________________________
Mehdi Zargham, Ph.D.
Committee Member
Professor, Department of Computer
Science

____________________________
Saverio Perugini, Ph.D.
Committee Member
Associate Professor, Department of
Computer Science
ABSTRACT

USING HADOOP TO CLUSTER DATA IN ENERGY SYSTEM

Name: Hou, Jun
University of Dayton

Advisor: Dr. Zhongmei Yao

With the large amount of data generated by various devices, data scientists face big challenges since conditional machine learning algorithms applied on a single computer can no longer be used for processing/analyzing such large data sets. This thesis takes a distributed computing approach built upon Apache Hadoop, which is a distributed data analysis framework running on multiple computers. The main components of this work includes implementation of k-means machine learning algorithms on the Hadoop Map-Reduce framework, processing raw data from real energy systems, classifying the data using k-means algorithms in Hadoop, and improvement on seed selection for k-means algorithms. Finally, this thesis demonstrates the efficiency and effectiveness of our approach using different data sets.
I dedicate this thesis work to my parents and loved ones
ACKNOWLEDGMENTS

I want to give special thanks to my academic adviser Dr. Zhongmei Yao, who have inspired and helped me unreservedly during past few years. I also appreciate the help form the committee members Dr. Medhi Zargham and Dr. Saverio Perugini for the comments on my research.

I also would like to express my sincere thanks to the University of Dayton Computer science department for the founding in purchasing the hardware for the research in my thesis. Moreover, I genuinely appreciate the help from Mr. Thomas Walters on hardware configuration.
# TABLE OF CONTENTS

ABSTRACT ................................................................. iii
DEDICATION ............................................................... iv
ACKNOWLEDGMENTS .................................................... v
LIST OF FIGURES ....................................................... viii

I. INTRODUCTION ....................................................... 1
   1.1 Overview ......................................................... 1
   1.2 Data Mining ..................................................... 2
   1.3 Big Data ......................................................... 3
   1.4 Load Forecast ................................................... 5

II. BACKGROUND ....................................................... 7
   2.1 Hadoop ......................................................... 7
   2.2 Techniques ..................................................... 14
   2.3 Data Mining Algorithms ....................................... 16
   2.4 K-means ......................................................... 18

III. CASE STUDIES ...................................................... 22
   3.1 Case Overview .................................................. 22
   3.2 Platform Establishment ........................................ 23
   3.3 Algorithms ..................................................... 26
   3.4 Similarity Measurement ....................................... 29

IV. EXPERIMENTS ....................................................... 32
   4.1 Result Overview ............................................... 32
      4.1.1 Distinguish family user and home business user ........ 32
# LIST OF FIGURES

1.1 Data Mining Procedure [1] .................................................. 3

2.1 Relationship between Hadoop versions ................................. 9

2.2 Key feature of different Hadoop versions ............................... 9

2.3 Program concept of Map Reduce [2] ................................. 11

2.4 Program concept of Map Reduce [3] ................................. 13

2.5 K-means clustering procedure ........................................... 19

3.1 E-R diagram view of case data .......................................... 23

3.2 Network configuration ..................................................... 24

3.3 DFS Admin status report ................................................ 25

4.1 Clustering result ............................................................ 33

4.2 Map-Reduce data flow [3] ................................................ 35

4.3 10000 Data points sample .............................................. 37

4.4 K-means converge speed with growth of data size .................. 38

4.5 Seed generation algorithm efficiency ................................. 38

4.6 Converge process compare .............................................. 39
CHAPTER I

INTRODUCTION

1.1 Overview

With the rapid growth of data size, data scientists face numerous challenges. These challenges include the techniques used to store and access large amount of data in a fast, secure and cost-effective way, and the techniques for processing and analyzing data concurrently. The owner of the world largest web search engine, Google, met the challenge at the beginning of 21st century, the age of explosive growth of the World Wide Web. As a search service provider, Google has to cache hundreds millions of web pages and respond to search queries promptly. Google is a company famous for its cost control and they deliver their answers in late 2003. The solution that is published in the proceedings of the nineteenth ACM symposium on Operating systems principles is called the Google File System (GFS) [4].

It proposed a distributed computing system with high capability of data storage and parallel processing based on low cost hardware. The details of mechanism will be introduced in following section. The open source implementation of Google File System named Hadoop became available two years later by the team of Doug Cutting, who was inspired by Google’s GFS solution. The Hadoop became popular soon after its announcement, and sustained the rapid development. The Hadoop makes big data analysis services available at affordable cost. The electric power industry that will be used as an example in the thesis is one of the beneficiaries. They collect the large amount
of energy data through the deployment of smart devices to elevate their profits via the data mining approach. In this thesis, we demonstrate that clustering analysis helps energy industry to classify their users for better load forecasting.

1.2 Data Mining

It is wildly accepted that data mining, as an interdisciplinary subject, can be defined in many different aspects. What data mining involves are far more than what we can understand from its literal meaning [5]. However, it is not too difficult to answer what data mining can do for us. Data mining uses intelligent ways to manipulate data that usually is in large scale in order to find out consistent patterns and internal relationship among different features, and then to validate the results by comparing the predicted data with real ones. The term KDD (Knowledge Discovery in Data) is a good way to conclude the behavior of data mining. Business is the first and still the primary field where data mining shows its power. In the business world, data mining is used to get valuable insights and to answer sophisticated questions to help the owner make the well-informed decision. Besides, data mining has been applied almost everywhere like health care, military intelligence, education and scientific research nowadays [1]. In 2014, Apple launched the Health Kit API on Apple products to collect the user behavior data and shared it with cooperative medical organizations for supporting medical research. This would be the largest health data mining program in the history. Data mining applies sophisticated mathematical algorithms to categorize the data and predicts the probability of future events. The general process of data mining usually consists of three major stages: (1) the exploratory analysis, (2) building models or identifying the patterns with verifications, (3) deployment (i.e., to make data driven predications to guide the decision making) [6]. The figure 1.1 shown the procedure of a typical data mining project. Among these steps, the second one is the most complicated and time consuming part. In order to determine the best suitable model or pattern, all the data need to be repeatedly processed for multiple times. This process raises high demands
on performance of data analysis system, including the performance of computing and performance of data access.

1.3 Big Data

The word big data not only describes the size of data, but also implies the method and approach we deal with the data. According to the definition by Viktor and Kenneth (2013), the big data analysis is processing and analysing massive data sets without using shortcut like sampling analysis. They also emphasized the four distinguishing characteristics of big data analysis, which are volume, velocity, variety, and value [7].

So a question we may ask is where the data comes from. In one respect, the technology upgrades in electronic engineering expand the source of data. The data collecting devices consume less power which makes it possible for wildly deployment. Moreover, it makes wireless devices like wearable devices last longer without frequently being charged. The development of the sensor technology enables us to collect a variety of data regarding motion, location, vibration, pressure, micro electricity flow, temperature, humidity, movements, and even the change of air composition.
In the meantime, the low-cost and high volume storage media provides the space to save the massive data. The first low-cost media appeared at 60s, which was a punched card with less than a kilobits storage space. After 50 years, gigabytes of flash memory were sold only for several dollars. These two innovations contributed directly to the rapid growth of data size. The executive chairman of Google Eric Schmidt once said: “Every two days now, we create as much information as we did from the dawn of civilization up until 2003 [9].”

To deal with massive data, the method and approach should also be upgraded. “Big data wont show its substantial value unless a revolutionary analysis schema is taken.” This is what Gartner Group once commented on the analysis approach on big data [10]. In data rich scenario, the traditional approach of data analysis may fail completely. In the past decades, large amount of data has not been analyzed before it is permanently deleted from the hard drive. To deal with massive data sets, the data has to be stored somewhere. The cost of data storage and management (i.e. routine backup) is high if we require zero defects. With the increase of the data size, the traditional analysis approach requires ultra-high performance of processors. In order to improve the capability of the processors and storage systems, scientists have proposed to combine multiple computers together to work as a team instead of building super computers in 80s. This idea was the embryonic form of distributed systems.

The early prototype of distribute systems is called grid computing. Grid computing is a distributed computing model where multiple computers are connected together via a local network to solve a complex problem [11]. This computing model simply allows computers to share storage and do parallel computing. During the runtime, the computers needs a large amount of data exchange via the network. The bandwidth may limit data I/O and also hampers utilization of processors.

The revolutionary distributed computing architecture was first announced by Google in 2003. Google proposed a brand new platform for their implementation of distributed computing, which
has been successfully applied for their search engine and PageRank algorithm [4]. In 2006, the Apache foundation officially introduced the open source implementation of Google’s revolutionary platform named Hadoop. Hadoop attracted great attention from the industry. Nowadays, the Hadoop has already become the standard architecture among any cloud service provider such as Microsoft Azure, Amazon web services, Cloudera and EMC².

1.4 Load Forecast

It is crucial to understand the electric power industry before we can tackle the electric load forecasting problem. The fundamental characteristic that makes the electric power industry distinctive is the limited storage capability of electricity. Since there is no inventory or buffer from generators to end users (customers), ideally, power systems have to be built to meet the maximum demand, i.e., the so-called peak load, to insure that sufficient power can be delivered to the customers whenever they need it. In transmission and distribution systems, the capacity of any equipment has to exceed the peak demand of all the downstream customers it serves. Due to this nature, many utilities are concerned with the accuracy of forecasting of peak loads [12].

In 21 century, the energy shortage has become a tough problem globally. The source of energy decreases while the demands are always growing. Unfortunately, the way of generating energy still depends on burning the fossil fuels that is exactly the same as past centuries, and these fuels are non-renewable. To confront the challenge, scientists first emphasize how to stop the wastage. In the electric industry, they turn to load forecasting.

Accurate load forecasts are critical for either short term, mid-tern or long tern planning for utilities. The load forecast impacts on high level decisions, including generator working schedules in a certain period, and broadly affects the retail electricity market prices.

Load forecasting algorithms often feature prominently in reduced-form hybrid models for energy prices, which are the most effective models for simulating users behavior and modelling energy
derivatives. The model of load forecast can also be applied to other cases. It consists of three factors: intrinsic factor, external factor and stochastic factor. The combination explains the condition of many social behavior, i.e., purchasing, stock market, criminal ... The utility price forecast is also frequently used by market participants in many trading and risk management applications [13].

Load forecasting plays an important role in planning an operational decision for various utility companies. This is even more significant with regard to the deregulation of the energy industries. It could be told in terms of fluctuating relationship of supply and demand, the changing weather condition, and increasing expense on energy by several factors during the peak season. Short-term load forecasting can be used to predict load flows, and it would lead to making decisions without overloading. This implementation helps improve the network reliability and reduce the concurrence of equipment breaks down and blackouts. For cases such as contract and financial products evaluations, load forecasting works well. Compared to non-deregulated economic conditions, decisions on capital expenditures which is based on annual forecasting is especially important under the deregulated economy when increasing rate could be justified through capital expenditure projects [14].

Generally there are two commonly used approaches for load forecasting that are known as statistical techniques and artificial intelligence. Examples includes regression, fuzzy logic, neural networks and expert systems[15]. The project in this thesis will focus on clustering analysis for better supporting load forecasting.
CHAPTER II

BACKGROUND

2.1 Hadoop

Apache Hadoop software family is a framework which allows the distributed computing of large data sets across clusters of computers using unique programming paradigm [3]. Hadoop aims at flexible scales. It can be either as small as a single node or as large as thousands of nodes. Each node has independent integrity so that any accident would not affect others. Instead of relying on the hardware to deliver high-reliability, the library itself is designed for detecting and handling failures at the application layer. This approach reduces the standard and cost of the hardware, while delivering the high reliability storage and analysis tools [3].

Hadoop was first created by Doug Cutting, the same creator of Apache Lucene, the widely used text search library. The core of the search library is called Nutch, which is the predecessor of Apache Hadoop.

Doug's first task is to write a light weight tool to serve as part as his web search engine. His goal is simple: fast and economy. Doug Cutting along with his partner Mike Cafarella intended to build a high performance system which can deal with billions of index within acceptable time based on million dollars hardware and $30,000 monthly maintenance cost. Nevertheless, they believed it was a worthy goal, as it would open up and ultimately democratize search engine algorithms. Soon after the beginning, the project developing smoothly. Nutch was running in 2002, and a working
crawler and search system quickly emerged. However, they realized the platform they were using is hard to meet the needs of quick response when it caching billions of web page. They are struggled at this problem for years. Twilight was seen with the publication of the new distributed computing concept from Google in 2003. In the paper, Google described the concept of Map-Reduce, and the supporting platform GFS (Google File System). Google claim that their new computing platform in the prefect solution for store and access for a large quantity of data. And this solution give Doug inspiration on his problem. However, although Google has already put them into service, the paper didnt reveal many implementation detail of this architecture as its a commercial secret. Doug didnt give up. In the same year, he set about writing an open source implementation: the Nutch Distributed File system (NDFS) [16].

In 2003, Google published a paper to introduce Map-Reduce to the whole world. In the early 2005, Nutch developers had successfully implement Map-Reduce in Nutch. By the middle of 2005, all the major Nutch algorithms had been ported to run based on Map-Reduce and NDFS. Starting from the concept by Google, Doug sketch his software carefully by guess and reasoning. Within two years of developing, the copy of GFS and Map-Reduce has been implemented successfully. And sooner the algorithm in Nutch was ported to their new platform. And the increment of performance is amazing. Nowadays, Hadoop is supplemented by an ecosystem of Apache open-source projects that extend the value of Hadoop and improve its usability.[3] The first stable release of Hadoop came to us in early 2006, and after near decade of development, it has many sub version distributed. The difference between them can be conclude in three aspects: I/O Format, calculation model and security. In 2.1 we can find the relation of each distribution and in chart 2.2 has shown the difference between the major release version.
Figure 2.1: Relationship between Hadoop versions

<table>
<thead>
<tr>
<th>Time</th>
<th>Version</th>
<th>Append</th>
<th>RAID</th>
<th>Symlink</th>
<th>Security</th>
<th>H-R v1</th>
<th>YARN</th>
<th>Name/Node Federation</th>
<th>Name/Node HA</th>
<th>Is Stable</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>0.20.2</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>0.21.0</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2011</td>
<td>0.20.200</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>1.0.0</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>0.22.0</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2012</td>
<td>1.2</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>2.1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 2.2: Key feature of different Hadoop versions
Although the feature of Hadoop always varying in the process of development, the key components has never changed, which are Hadoop Common, HDFS and Map Reduce. Hadoop Common is a module which provides the base level utilities that support the other Hadoop components. HDFS, also known as Hadoop Distributed File System, is a file system that provides reliable data storage by access redundant copy in a Hadoop cluster. It connects the file systems on many local nodes to create a single file system. As mentioned before, the most advanced feature of Hadoop architecture is data localization and computing localization. The Map-Reduce computing model is designed to achieve that feature. The first generation of Map-Reduce is released alone with the first version of Hadoop in 1996. It is a framework as well as a programming method that process large amounts of formatted and unformatted data in parallel across all the nodes in cluster in a reliable, fault-tolerant manner. The second generation of Map-Reduce which is also called YARN came to us in late 2011. It assigns ‘resources’ like CPU, memory and storage to maximize the performance for applications running on a Hadoop cluster. It enables application frameworks other than Map-Reduce to run on Hadoop, opening up a wealth of possibilities. In a nut shell, the updates of YARN comparing to the old one is comprehensive control of Hadoop cluster resource. Map - Reduce is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster [17].

The programming paradigm of Map-Reduce is easy to understand. The term Map-Reduce represent the two stage of task. A map-reduce program may include one or multiple map-reduce loops, and yet map task can be done separately. The map task is the first stage of job. It read from data source line by line in steam. Then it reorganize the data into key-value pair. The key should be the index which is designed to pass into reduce task. The second stage is reduce task. The reduce task accept the key-value pair generated by map task and handle them by reducer function. As the
sequence of the name Map - Reduce implies, the reduce job is always performed after the map job. The schema is illustrate in figure 2.3.

Hadoop changes the economics as well as the dynamics of large-scale computing. Its impact can be concluded to four salient features. Using Hadoop will bring us some unique advantage comparing with other solution. The first advantage is scalable. Hadoop can be as small as one node, and can be as large as thousand of nodes which depend on your needs. When it need to be expand, the process is as simple as plug-and-play, all the data will be reallocated automatically if it is necessary. The second is cost effective, which is critical to those enterprise lack of fund. Hadoop brings massively parallel computing to commodity servers. It will bring to enterprise the decrease of hardware cost meanwhile enhance the productivity. Whats more, Hadoop is also a fault tolerant system. When there is a node lost contact, the system will redirects work to another location of the data and continues processing without missing a beat. Last but not least, Hadoop is flexible than others. Hadoop is schema-less and can absorb any type of data, formatted or not, from any number of sources. Data from multiple sources can be joined and aggregated in arbitrary ways enabling deeper analyses than any one system can provide[18].
If we considered Hadoop is a revolution of data analysis platform, it is quite necessary to compare it with our traditional platform: the database system. Because in the recent years, the technical leader of RMDBS like Oracle and Microsoft, has already make temptation to upgrade their product to keep up on times. Some feature, like distribute storage and data caching, have been applied on the new version of their product. Although the Hadoop itself is born to meet the needs of big data era, it didn’t take the place of the database rapidly, even 10 years passed. So what is the weakness of the Hadoop? Firstly, the function of the calculation model Map-Reduce has limitation. Because of the distributed calculation model, each worker node can only access part of data. Even the reduce stage will schedule data transfer to another node, the reducer node still can only get a part of data. In another word, the Map-Reduce module can not implement all common use algorithm. The second part is missing most of the features that are routinely included in current DBMS and incompatible with all of the tools DBMS users have come to depend on [19]. Modern DBMS hides the most implementation detail in order to provide an easy-to-use interface. Due to many constrains and rule, data stored in database is formatted and integrity. And Hadoop didn’t provide an API to other data analysis and visualization tool like R and Matlab at the time of release. In many cases, data scientist need use a brunch of software as Hadoop can not provide a end to end solution. In the year 2011, the Apache foundation release the second version of Map-Reduce called YARN. And also update its derivatives. The Hadoop product family is shown in figure 2.4 And a list of introduction from Hadoop official website is followed.

**Ambari**: A web-based tool for administration, managing, and monitoring Apache Hadoop clusters which includes support for Hadoop family products. Ambari also provides a visual interface to monitor and administrate the status of products in Hadoop family’ [3].

**Avro**: A data serialization system’ [3].
**Cassandra:** ‘A scalable database with no single points of failure’ [3].

**Chukwa:** ‘A data collecting software for managing large distributed systems’ [3].

**HBase:** ‘A scalable, distributed database that supports structured data storage for large tables like DBMS’ [3].

**Mahout:** ‘A Scalable machine learning and data mining library’ [3].

**Pig:** A high-level data-flow language and execution framework for parallel computation’ [3].

**Spark:** ‘A fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model that supports variety of applications, including ETL, machine learning, stream processing, and graph computation[3].

**Tez:** A generalized data-flow programming framework, built on Hadoop YARN, which provides a powerful and flexible engine to execute a random DAG of tasks to process data for both batch and interactive use-cases’ [3].

**ZooKeeper:** ‘A high-performance coordination service for distributed applications in Hadoop family’ [3].
2.2 Techniques

Data scientist usually apply different techniques and tools to help them to interpreted the data, so as to take the advantage of every tools. For this case in thesis, no doubt that Map-Reduce is the most important technique, meanwhile other technique like statistic modelling, machine learning, and visualization are indispensable. In this section, I will discuss some technique I may use in the analysis according to the process of traditional data analysis procedure. The sequence of a typical data analysis project is as follows: Raise the question, find the accessible data, obtain data, clean data, exploratory analysis, data manipulation, statistical analysis, interpret result, synthesize answer and finally distribute answer. This project is cooperated with National Energy Partner, and the company provide all the data they have. So my job actually start from obtain data and clean data. In any formatted data storage such as database, there will be some rule or restrain to guarantee the data meet the needs to be process and analysis. Hadoop doesn’t, infect, Hadoop’s data storage system is more like the file system before the database is available. So its important to check the data before upload to HDFS. The check work includes data integrity, data format and data range. After it pass the check, all data need to be normalize which will make result look nice. This operation is operational. For numeric value, normalization can be easily calculate as:

\[
X_n = \frac{X - X_{\text{avg}}}{X_{\text{max}} - X_{\text{min}}}
\]  

(2.1)

After data normalize, the data will map to an area from -1 to +1. This step is very useful for some multiple property cases, like a dataset has a height value in INCH unit and a weight in KG unit, so if we plot it in one graph, it will be difficult to observe the pattern. There is another optional work in this stage is data compression. This step is recommended when data have very high dimension. Because when the dimension is higher, the calculation related to statistical analysis will rise dramatically. Also, the result will be difficult to observe if data dimension is higher than 3. However, compression will lose some detail more or less no matter what kinds of method you
are using. So what we need is an approach to keep the useful detail as much as possible. This
time I will turn to the featuring concept in machine learning. In a typical learning problem, the
algorithm will finally come out a decision map (policy) associate to each specific condition. When
the condition is to detailed, it will be impossible to get the policy in a reasonable time. Instead
of using condition, we will use features to characterize some valuable properties of the data which
related to the problem we are solving. For instance, if we have the data which is the length of the
edges of rectangle, we can know the area and perimeter of the rectangle. If the problem is more
relate to the area, we can use area to characterize the rectangle instead of the length of edges. Note
that featuring in its original definition is not only used to compressing data, in some case it can
extend data. But in my cases, it will mainly use for compress data.

The next step is exploratory analysis. This step is to get a brief idea of data set, so we can
decide what will be best approach to analysis the data. The techniques may using here is sampling
and clustering. Firstly, in order to know the distribution, range or other macro property, we can get
some sample from the whole data set which saves a lot of time. The sample amount is usually much
smaller than the data set. After we get the outline of data, some time we need to separate them in
order to apply more specific operation. The approach of separation can be manual or natural, which
is called grouping and clustering. Both of them are used for downstream proposes. The difference
between them is grouping is manually divide data into designated group and clustering is that data
aggregate by their similarity. In most cases, these two techniques are always mixed together. Clus-
ter analysis or clustering is to group the individual by the similarity of their property. The goal
of clustering is to minimize the difference in certain cluster and maximize the difference between
clusters. Cluster analysis itself is not one specific algorithm, but the general task to be solved. Com-
monly used clustering algorithms are: KNN (K-Near-Neighbor), K-means, hierarchical clustering,
Expectation-Maximization clustering, DBSCAN [20].
There is no objectively "correct" clustering algorithm. In most of cases, the most appropriate clustering algorithm for a particular problem often needs to be chosen experimentally, unless there is a mathematical reason to prefer one cluster model over another. Moreover, some clustering algorithm will generate non-exclusive answer, which made it more difficult to evaluate [21].

In my cases, the Apache Hadoop platform will mainly do the job of clustering, because this part has the heaviest calculation jobs and need access the whole data set. Besides, Hadoop will also join the job of data cleaning, featuring and manipulation. The analysis job can be done be R and Excel if its not too complicated. At least, the result must be shown to us by picture and figures. This step we usually refer to software which can convert data into plot like matlab or R. In my case, I will also use a light weight integrated graphic tool pack named OpenCV. OpenCV is released under a BSD license and it's free for use. By embed the CV code in algorithm, the every step of clustering algorithm can be supervised. That will allow me to research its intermediate steps, which is quite useful in optimization process.

2.3 Data Mining Algorithms

A data mining algorithm is a set of heuristics and calculations that creates a data mining model from data. In short, it converts data into meaningful model or pattern. To create a model, some of algorithm can be apply to provide a overview of data in a short time, looking for specific types of patterns and trends. Other algorithm uses the results of this analysis to define the optimal parameters for suitable mining approach. These parameters are then used across the entire data set to extract actionable patterns and detailed statistics [22]. Data mining algorithm can be concluded in several categories.

The first is classification. Classification is commonly used technique for predicting a specific outcome with distinctive characteristics. This technique usually used in such data set has clear causality, and the difference between each feature is easy to tell and measure [22]. For classification
technique, the four most common use algorithms is Logistic Regression, Naive Bayes, Support Vector Machine, Decision Tree.

The second category is regression. Regression is a technique for predicting a continuous outcome based on a function which takes the all possible influence as parameter [22]. Regression usually used for prediction future trend. The common used algorithms for regression is multiple regression and Support Vector Machine.

The third category is clustering. Clustering is helpful in data exploring for finding natural groupings. Items of a cluster are more like each other than they are like items of a different cluster. Common examples include finding new customer segments, and biological sciences discovery [22]. The frequently used algorithms are K-Near Neighbour, K-Means, Orthogonal Partitioning Clustering and Exception Maximization.

The final one is conformity. This technique is used in data aggregation and compression. Though the linear combination and calculation to make data set fitter for any learning method like clustering. In this paper, the electric data firstly passed conformity to catch the key feature related to the problem and reduce it’s dimensions to speed up the calculation in K-means.

A typical data analysis project always make full use of different analysis tools. In my thesis, I will focus on clustering algorithm to see how it works on a power usage data set. Clustering is a natural division of data into different sets of similarity. In clustering, only part of features of data are used in similarity measurement. Cluster is a very strong data modeling technique providing concise summaries of the data. Thus, clustering relates to multi-disciplines and plays a key role in various applications. These applications of clustering are usually dealing with data in the large scale due to many attributes[23]. The purpose of clustering is to determine the intrinsic grouping in a set of unlabeled data. However, it is difficult to tell what is made of a good clustering. As a consequence,
no absolute fair criteria can be independent on the final target of clustering. Users have to take care of this criteria, and the results of clustering would satisfy their needs in this way [24].

2.4 K-means

K-means is one of the most popular unsupervised learning algorithms solving common clustering problem. The procedure is straightforward enough to classify a given data set through a certain number of clusters, says k clusters, fixed a priori. This algorithm was developed by several researchers across multi disciplines, most notable people are like Lloyd (1957, 1982), Forgery (1965), Friedman and Rubin (1967), and McQueen (1967)[25].

The main idea is to divide all item in a data set in to K group to minimize the cohesion in the group and maximize the separation among groups. So at the beginning of the process, K hypothetical centroid will be given to be the initial clustering centroid. To select the item which is belong to the a certain cluster, we used the revised index approach which calculate all the distance from this item to every centroid and pick the cluster which has the smallest distance to the item. Repeat this method until all item has been assign to cluster. This process is called an iteration in K-means. Then, for each cluster, calculate the centroid by the arithmetic average of the item in the cluster. Afterwards, repeat the iteration and calculation of new centroid several times until no item needs to be re-assign to other cluster any more. In this process we may find that the k centroids change their location in each iteration until no more changes are done [26].

The algorithm is composed of the following steps: First select K data points as initial seeds. This selection can be done by random selection and heuristic selection. Heuristic selection is a way that trying to select seeds which close to the final cluster after the algorithm. After that, the seeds are used for the K hypothetical centroid in the algorithm. Then, calculate the distance by a similarity measurement function which is pre-defined by the propose of clustering regarding to the project. After all data point has been assigned to a cluster, recalculate the centroid of the cluster. Then use a
loop to repeat the iteration and controlled by a boolean parameter which indicate if there is any data point has been reassign to another cluster [27]. The process of K-means algorithm can be observed from the example show in figure 2.5. The example is plot in 2-D graphic.

There are three main challenge in K-means clustering: 1. How to choose K, 2. How to measure the similarity, 3. How to choose the initial centroids. Just as discuss above, K-means algorithm do not have designated correct answer, it is hard to judge whether one solution is better than others. Especially the choice of K. For the second problem, we need first to realize that data has multiple types in different format. The three common data types are binary, enormous, numeric. Binary: The value of data is only have either true or false. Enormous: The value of data is only consisted with limited choice . Continuous: The value can be at any position or combination of reality's trustee. There are also existing some measurement accordingly, but no one can handle mixed-type data. More precisely, the method will vary from the project and the problem you have. I’ll explain my method in the paper.
Last challenge is the most easy to evaluate in the aspect of speed. Because the quality of seeds will directly affect the efficiency of the algorithm. As the mechanism explained above, K-means will repeat doing same job until no node need to be reassign. A good set of seeds will make the converge process faster. As you may know, in a typical big-data cases, a round of K-means job will have to go through all node in the data set which may rich GB to PB. So a good set of seeds will save significant amount of time especially in data-rich case. There two commonly use algorithm for seeds selection. One is called K-means++, and the other is called the canopy. K-means has the mechanism as follows: choose the first cluster center uniformly and randomly from data points being clustered; then choose each subsequent cluster center from the remaining data points with probability which is proportional to its square distance from the closest existing cluster center of the point. As for the canopy, the algorithm is even more complicated. This kind of unsupervised pre-clustering algorithm is introduced by Andrew McCallum, Kamal Nigam and Lyle Ungar in 2000 [28]. It is normally applied to the preprocessing step for the K-means algorithm or the Hierarchical clustering algorithm, which helps to speed up clustering operations on large data sets, while other algorithms may be impractical due to the size limitation of data sets.

Another popular optimization approach is Canopy K-means, and it has been adopted in the Apache foundation since 2010. Canopy Clustering is a straight forward, fast and surprisingly accurate method for clustering analysis. All items are represented as a point in a multidimensional vector space. The algorithm uses a fast approximate distance metric and two distance thresholds T1 and T2, which T1 is larger than T2, for calculation. The basic process is to start from a set of points and remove one randomly. Create a Canopy which contains this point and iterate through the remainder of the point set. At each point, if its distance from the former point is smaller than T1, then add the point to the cluster. And if the distance is smaller than T2, then remove the point from the set. This way points that are very close to the original will avoid being frequently reassign in
K-means process. The algorithm loops until the initial set is empty, accumulating a set of Canopies, each containing at least one point. A given point may occur in more than one Canopy [28].

The Canopy algorithm seems to have alluring advantages: no need of pre-set K. But there are two more parameters need to be set in Canopy algorithm: T1 and T2. And it’s even not as intuitive as K setting (Usually K is related some downstream propose). In this paper, Canopy algorithm will not be discussed.
CHAPTER III

CASE STUDIES

3.1 Case Overview

This load prediction project is based on a real data set which is provided by national energy partner who collects the user data by smart meters. The wide adoption of smart meters that measure and communicate residential electricity consumption gives rise to the development of new energy efficiency services [29]. The data used in the project include the hourly power consumption and daily water consumption. In addition to the power and water data, it also combined with weather data provided by NOAA. The amount of data is list as follows:

116 independent users

Smart meter record the usage of the electric every hour

1 year complete electronic meter records (24*365 = 8760)

Each user have a water meter which records water usage once or twice daily.

The time period is from 6/9/2013 to 6/9/2014

The power and water data are collected by smart meter to monitor the daily usage of each individual user. These data are primary used for classifying the user type as well as making prediction of monthly usage. The user type can be define by different aspects. For example, users can be classified into resident and business, or classified into high consumer and low consumer. Also they can be classified into daytime user, night time user or whole day user. In order to make accurate
prediction of power usage, a reasonable user classification is quite important. Although it is not too difficult to calculate the regression but it’s not that easy to find a suitable regression for each individual user. That’s why we need to classify them. Using clustering algorithm, we can group user who has similar consuming habit. So you can expect that they can apply on one regression model without heavy losses of accuracy.

The data received for this case include lots of information. First of all, I need to digest all this data and find out the relationship between them. In order to represent them clearly, I use an E-R diagram which usually used in RDBMS system to illustrate the data. The E-R diagram is list as follows:

3.2 Platform Establishment

In this case, I will use Hadoop as the tool of data aggregation and clustering. The computing model will use MapReduce. So the Hadoop release 1.0.0 and 0.20.2 are good enough for my task. In practice, I use 0.20.2 as my primary software platform and I repeat the work on 1.0.0 version to check compatibility. The hardware used in this project is as follows:
PC: HP Z51 Work Station 4

Hardware configuration:

CPU: E3 - 1225v3 3.2Ghz Quad-Core, Hyper-Threads enable

Memory: 8G DDR3-1600

Storage: Crucial 240G SSD

OS: CentOS 7.0

These four PC are connected by a CISCO 1000M switcher, the network configuration is as follows:

Hosts: (/etc/hosts)

These four PC has established password free SSH connection for carrying out map reduce task.

The detailed software environment settings and Hadoop runtime parameters are list as figure 3.3:

After the Hadoop has been success setup, the system will report the status:
Figure 3.3: DFS Admin status report

Configured Capacity: 532248821760 (495.7 GB)
Present Capacity: 529946578944 (493.55 GB)
DFS Remaining: 529946554368 (493.55 GB)
DFS Used: 24576 (24 KB)
DFS Used%: 0%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0

Datanodes available: 3 (3 total, 0 dead)

Name: 131.238.19.26:50010
Decommission Status : Normal
Configured Capacity: 177416273920 (165.23 GB)
DFS Used: 8192 (8 KB)
Non DFS Used: 614866944 (586.38 MB)
DFS Remaining: 176801398784 (164.66 GB)
DFS Used%: 0%
DFS Remaining%: 99.65%
Last contact: Thu Mar 12 09:58:06 EDT 2015

Name: 131.238.19.24:50010
Decommission Status : Normal
Configured Capacity: 177416273920 (165.23 GB)
DFS Used: 8192 (8 KB)
Non DFS Used: 928415744 (885.41 MB)
DFS Remaining: 176487849984 (164.37 GB)
DFS Used%: 0%
DFS Remaining%: 99.48%
Last contact: Thu Mar 12 09:58:06 EDT 2015

Name: 131.238.19.25:50010
Decommission Status : Normal
Configured Capacity: 177416273920 (165.23 GB)
DFS Used: 8192 (8 KB)
Non DFS Used: 758960128 (723.8 MB)
DFS Remaining: 176657305600 (164.52 GB)
DFS Used%: 0%
DFS Remaining%: 99.57%
Last contact: Thu Mar 12 09:58:06 EDT 2015
3.3 Algorithms

In this section, I will introduce the general program schema for problems solved with Hadoop platform in this case. The scheme includes the format of input and output data file, and major operation in algorithms.

**Data aggregation**

Data input:

| address | size | year | num of rooms | floor | style | name | Elec Meter ID | Wat Meter ID | record |

Mapper:

Key = Elec Meter ID

Value = record

Reducer:

Key = Elec Meter ID

Value += record / number of record

Data output:

| Elec Meter ID | Daily average power usage |

**Data filtering**

Data input:

| address | size | year | num of rooms | floor | style | name | Elec Meter ID | Wat Meter ID | record |

Mapper:

Key = Elec Meter ID
Value = if( record is valid) (record).toString

Reducer: None

Data Join

Data input:

<table>
<thead>
<tr>
<th>Data Src 1</th>
<th>Elec Meter ID</th>
<th>Average Power Daily</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Src 2</td>
<td>Elec Meter ID</td>
<td>Daytime %</td>
</tr>
<tr>
<td>Data Src 3</td>
<td>Elec Meter ID</td>
<td>Monthly S.D.</td>
</tr>
</tbody>
</table>

Mapper1: (Read data src 1)

Key = E. ID

Value = ”s1,” + Average Power Daily

Mapper2: (Read data src 2)

Key = E. ID

Value = ”s2,” + Daytime %

Mapper3: (Read data src 3)

Key = E. ID

Value = ”s3,” + Monthly S.D.

Reducer:

Key = E.ID

ArrayList(String) feature;

if(value.substring(0,1) == ”s1”) feature[0] = value;

if(value.substring(0,1) == ”s2”) feature[1] = value;
if(value.substring(0,1) == "s3") feature[2] = value;


Data output:

<table>
<thead>
<tr>
<th>Elec Meter ID</th>
<th>Daily average</th>
<th>Daytime Per</th>
<th>Monthly S.D</th>
</tr>
</thead>
</table>

**K-means**

Data input:

<table>
<thead>
<tr>
<th>Cluster ID</th>
<th>Daily Pwr</th>
<th>Daily water</th>
<th>Avg Mounthly Pwr S.D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weekend %</td>
<td>Weekday %</td>
<td>Daytime %</td>
<td>Nighttime %</td>
</tr>
<tr>
<td>Q2 Usage</td>
<td>Q2 Usage</td>
<td>Q3 Usage</td>
<td>Q4 usage</td>
</tr>
</tbody>
</table>

Global recorder:

K = Number of clusters

double DifferenceCalc(String Data1, String Data2);

ReAlc = Number of Re-allocate data

Arraylist(String) CenterSet

<table>
<thead>
<tr>
<th>Cluster No.</th>
<th>Center Value</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(Sum(Data in Cluster0) / Num of data in cluster0).toString</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(Sum(Data in Cluster1) / Num of data in cluster1).toString</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>K-1</td>
<td>(Sum(Data in ClusterK-1) / Num of data in clusterK-1).toString</td>
<td>0</td>
</tr>
</tbody>
</table>

Mapper:

Min(DifferenceCalc(Data,CenterSet[0]),DifferenceCalc(Data,CenterSet[1]), ...

...,DifferenceCalc(Data,CenterSet[K-1]);

Key = The index of CenterSet which has minimum difference calculation with the input data item

if( Data.ClusterID != Key) ReAlc++;

Value = Data.toString
Reducer:

Key = Cluster Index number
Value = data.ClusterID = Key
CenterSet[Key].Value += value.todouble;
CenterSet[Key].Number of Data ++

TaskCtrl:
NumOfRounds++
if (ReAIC > 0) Do:
for(0 to K-1):
CenterSet[i].value = CenterSet[i].value ÷ CenterSet[i].Number of data
Start Again.
else: Task end

3.4 Similarity Measurement

Almost all of the clustering algorithm is distance based. Which means, we need to know the similarity between two data points. It is not very difficult for us to calculate the physical of any pair of data point in a N-dimension space. The problem is, in the reality, the data point cannot be as simple as the points on a 2-D square. It may have multiple demission and it may have multiple types (i.e. Binary, categorical/nominal, ordinal). And the range of value vary from types. There are some solution of these problems. For multiple demission, especially high demission data, we usually use dimensionality reduction or sub-space clustering. For the data range problem, we usually use normalization to solve. And to deal with multiple data type, there are corresponding distance algorithm for each of these, but there is no certain method to solve mixed type data. For example, if
we have two objects, a 5-feet-tall boy and a 5-feet-tall girl. They have differences obviously but they may be concerned same if we didn't consider the gender.

In most cases, we don't have that chance to do many experiments and experience is not always that correct. That is, we need to take it into calculation in some forms. There are three forms of data dimension: binary, enormous, numeric.

**Binary**: The value of data is only have either true or false.

**Enormous**: The value of data is only consisted with limited choice.

**Numeric**: The value can be at any position or combination of reality's trust.

And there are corresponding algorithms to measure their distance.

- Euclidean Distance
- Minkowski Distance
- Manhattan Distance
- Chebyshev Distance
- Mahalanobis Distance (Uniform unit)
- Pearson Correlation Coefficient
- Jaccard Coefficient

If we meet the mix of all these types, a function can be used to calculate the combined distance:

\[
d(i, j) = \frac{\sum_{f=1}^{p} (\delta_{ij}(f) - d_{ij}(f))}{\sum_{f=1}^{p} \delta_{ij}(f)}
\]

(3.1)

Nevertheless, the similarity measurement is highly depend on manual setting and downstream propose. Like how to manage the power of each feature and how to select relative feature in a certain question. And some time we need to facing non-scaled variable which is more difficult to measure. Because the general concept of scaled variables is to assign a value manually according
to experience. For instance, if student can be rate as Excellent, good, fair and poor, we always give them 4 - 1 to represent them. But if we meet red, yellow, blue, white, which standard can we use to assign value to them? The similarity measurement still can not find a approach avoiding digitize the conceptual variable, which we need to avoid using in clustering calculation.
CHAPTER IV

EXPERIMENTS

4.1 Result Overview

This project uses K-means algorithm as a tool for distinguishing users, and the K value is not easy to choose. However, we can do some exploratory clustering tests on data sets to get a brief idea of it. This is one of the examples of a serious of tests.

K = 10;
Data vector (Q1 total, Q2 total, Q3 total, Q4 total, annual water usage)
To get better data visualization, I added Q1 to Q4 power usage as annual usage for X axis.

Besides, there are some reasonable combinations of K-value and Data vector I have been put into test. As you can see the choice of K are really depend on what problem you are looking into.

4.1.1 Distinguish family user and home business user

K = 2
Data vector (Average daily power usage, daytime percentage, weekday percentage, weekend percentage)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>1</td>
<td>97</td>
</tr>
</tbody>
</table>
4.1.2 Distinguish climate sensitive user

\( K = 3 \)

Data vector (Q1 total, Q2 total, Q3 total, Q4 total, S.D. of monthly usage)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>1</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
</tr>
</tbody>
</table>

4.1.3 Daily usage distribution

\( K = 2 \)

Data vector (Average daily power usage, daytime percentage)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>46</td>
</tr>
<tr>
<td>1</td>
<td>71</td>
</tr>
</tbody>
</table>
4.2 K-means Optimization

A general problem in big-data analysis which deserve lots of consideration is efficiency. Because of the massive amount of calculation, any tiny improvement in efficiency will lead to significant time saving. Generally speaking, the efficiency of program is determined by three parts: hardware, platform and algorithm. These three parts need to support each other to achieve higher performance. In the thesis I will focus on platform and algorithm parts only. As discussed before, the Hadoop system represent two critical programming concepts: data localization and computing localization. The computing model Map-Reduce which implements these two concepts tremendously reduce the standards of hardware as well as hardware cost.

Moreover, these two concepts may also be the guide to our ways of algorithm optimization, which make it little bit difficult comparing to optimize an algorithm in a single machine. The Map-Reduce model was designed to hide the implement detail of multi node coordination, which made it transparent to programmers. This approach does help programmers concentrate on the realistic problem and reduce their load. However, what behind this model is what we interested in when we consider how to optimize the algorithm.

When we look at the Map-Reduce model we will notice there is a shuffle and sort part between map tasks and reduce tasks. After map task has finished, it would generate .spill file to store the temporary data on local disk. Before this point, the computing only happen on local node. Then, the reduce task will copy their target intermediate data from different node based on HTTP. That means it will be much slower than data I/O on local machine. Although this process is not avoidable in each Map-Reduce process, we sometimes only use map task to deal with some light task like pre-processing before recursively running Map-Reduce.

For K-means algorithm, no matter how it use in single computer or Hadoop compute cluster, the most time consuming step is recurring computing the distance to new center and re-assign data
to new cluster. The number of round is closely depend on the initial seeds and the selection of K. In a real world problem, the K number is usually set manually, and the seeds set generated randomly. If we can find out a way that can pick out a sets set which have similar distribution as the finial clusters centers, it can definitely reduce the rounds in recursion before it gets converge. There are some existing try to solve this problem, like famous K-means++ and Canopy K-means.

K-means++ using a greedy like approach. At first the algorithm will randomly select a data point as seed. Then the algorithm will calculate all the distances between it and other data point and select the one who have the longest distance. After that, calculate the average of seeds set to get the new seeds center. Repeating these process until all K seeds were found. The K++ algorithm implements a simple concept that make seeds as dispersive as possible because K-means algorithm is designed maximize the difference between clusters. So we can image the center of K-means cluster will not be very close to each other. Because each time to select a seed will involve all the data point into calculation, so the time complexity will be O(kN). After many experiments, I
have find out an improved version of this algorithm. In my practice, I found two major problem of this algorithm. Firstly, it’s time costing, especially in data rich scenario. What’s worse, it cannot guarantee it can perform better than random seeds selection. See figure 4.4

Observe from a 2-D data set, I found that using K-means ++ will make the initial seeds too close to the margin as it always selects the longest one. On the contrary, the final clusters centres always distribute evenly in the graph. So I discard this way to select seeds, my algorithm work in this way:

Rearrange data in vector format:

\[
D_x = (f_1, f_2, f_3, f_4, ..., f_n)
\]  

(4.1)

Define similarty measurement \( \text{Dis}(D_1,D_2) \)

\[
\text{Dis} = \sum \left( \frac{D_1f_i - D_2f_i}{k} \right)
\]  

(4.2)

SeedsSet[k];

Randomly select K data point as initial seeds.

Calculate the center \( C_s \) of seeds set.

\[
C_s = (\frac{\sum S_k f_1}{k}, \frac{\sum S_k f_2}{k}, \frac{\sum S_k f_3}{k}, ..., \frac{\sum S_k f_k}{k})
\]  

(4.3)

Calculate the average distance \( D_c \) to the center \( C_s \)

\[
D_c = \frac{\sum \text{Dis}(S_i, C_s)}{k}
\]  

(4.4)

Calculate the distance of one seed to all other seeds, and add them together

\[
\text{SUM}d_i = \sum (\text{Dis}(D_i - D_1) + \text{Dis}(D_i - D_2) + ... + \text{Dis}(D_i - D_k))
\]  

(4.5)
Calculate the standard deviation $D_{sd}$ of all distance sum

$$D_{sd} = \sqrt{\frac{\sum (SUM_{di} - Avg(SUM_{di}))^2}{K}}$$

Select the seed has the minimum distance sum

Randomly choose a data node, if new data node has a larger distance sum and the distance to center is bigger than $D_c$, replace the old one. If not, randomly pick another one until it fit the requirement.

Re-calculate the new center $C_s$, the average distance $D_c$ and $D_{sd}$. If the new $D_{sd}$ is in the range of +/- 10% old range, seeds optimize end. Else repeat doing these above. Also there is a simple condition to determine whether the process is finished. Pick a constant integer $A$. In the process of searching new seeds, if failed more than $A$ time, then abort. If there is continuously $K$ times abortion, then process terminated.

Because my algorithm only pick limited number of data point, there is no need for Hadoop system running Map-Reduce task. Only map task is needed to get the random data point. So there
Figure 4.4: K-means converge speed with growth of data size

Figure 4.5: Seed generation algorithm efficiency
will be no shuffle and sort process. My algorithm is also efficient in time complexity, it takes $O(\kappa^2)$, which is near constant because $K$ is usually very small.

To test the seed selection algorithm, I use a variable amount of data and a fixed amount of $K$ to simulate different situation. In order to eliminates the noise and other interruption, I implement it on a single PC. The test condition is listed below:

CPU: Inter Core I7-3720QM

Memory: 16GB DDR3-1600 Non-ECC

Storage: Samsung 840EVO SSD

In order to simulate the data distribution in real cases, I use different distribution function with different parameter to generate a even distributed data set which is like figure 4.3. To test the performance of the algorithm, the density of data set is change by inserting different amount of data. The figure 4.3 is recorded by OpenCV in a 10000 data case. In the figure, the black dots is a data point, the Green dots is the final clusters center with random seeding.
In the figure 4.4 we can observe that using the seed from K-Means++ always keeps the lowest needs of recursion. The random seeding and K-means++ seeding performance is similar before data size under 10000, and after that, K-means++ algorithm show it’s advantage in efficiency. In this test, the number of cluster is 20. In big data scenario, we have reason to believe the seeds set generated by improved K-means++ will save a lot of time due to less needs of recurring.

The cost to generate a good set of seeds is also need to be considered. In figure 4.5, we can see the seed generation time in different density data set. The random seed selection is the fastest of course, and the time cost raise dramatically for K-means++ algorithm because of it’s quadratic complexity. My algorithm always keep a reasonable increment and will not make to much load to system overall.

To observe the intermediate process of the K-means algorithm, I record the number of data set which has been reassigned to another cluster in a certain round, see figure 4.6. From the figure we can find the K-means algorithm using optimized seeds always has the minimum number of data points need to be reassign. Also, in 10 separate test, I record the worst case of three seed algorithm and point at the figure 4.6.
REFERENCES


