Analysis of Quality Factors of Online Video Streaming Based on Measurement

Anyi Xu,Xiuqing Zhuang,Guijin Niu

School of Computer Science, Ningde University, Fujian, China

Abstract: Spark is a low-latency, distributed computing system for large data sets. Spark is compatible with Hadoop data sources, but about 100 times faster than MapReducer, and is particularly well suited for machine learning. Spark is still in the embryonic stage, not yet high-speed development, with the Spark1.0.0 version of the release, marking the apache's top open source project Spark as a large data upstart, more and more attention by the IT industry will be widely used. Equipped with Spark platform and application of Spark to study the analysis of online video streaming media quality factors. This paper introduces the background knowledge of the research, and introduces and studies the composition and principle of Spark in detail. According to the needs of the experiment, the overall configuration of the platform is completed, its performance is verified, and its machine learning library is studied. First, we introduce the user requirements and architecture models of the widely accepted distributed file system in the industry. Then, the architecture of RDD is introduced. Finally, the relationship between the time of viewing video and the number of buffers is analyzed by KMeans machine learning algorithm, and the relationship of streaming media related factors is summarized. The platform used in the experiment is the Linux Ubuntu12.04LTS , the application is the Apache Spark platform. All the system preparation, debugging and testing are carried out in this experimental platform.

Key words: Spark; machine learning; KMeans; data mining; video streaming media

Chapter 1 Preface

1. Research background and significance of research

Stream Streaming refers to the continuous time base media that uses streaming technology in the Internet. The Media Streaming service means that the multimedia streaming is supported by the multimedia data stream from the server to the client, and the receiving side plays the technology. Compared with the traditional download after the practice, streaming media technology has a great advantage, such as real-time strong, is conducive to the protection of copyright and so on. With the continuous development of computer network and multimedia technology, streaming media live technology more and more widely, the traditional Flash has long been replaced by video streaming media. In recent years, the rapid development of wireless networks and the rapid increase in bandwidth for streaming media technology in the wireless network applications laid a solid foundation.

Today, data processing requirements are very fast, as a replacement for Hadoop, Spark performance than MapReduce to enhance a lot, making it a hot open source project. Apache Spark is a new efficient distributed computing system, is a common parallel computing framework, originated in the University of California at Berkeley AMPLab [2] cluster computing platform. Databricks, Cloudera have decided to support Spark, Spark is considered a big deal in the big data area and is likely to be the big guy in the big data area.

Spark memory computing framework for a variety of iterative algorithms and interactive data analysis, can improve the real-time data processing and accuracy. MapReduce processing framework is good at complex batch operations, landing filtering, ETL (data extraction, conversion, loading), web index and other applications, MapReduce in low-latency business has been criticized.

As a memory-based in-memory data processing platform, it is compatible with Hadoop data sources but runs much faster than Hadoop MapReduce. Spark iteration calculation of the main idea is RDD, all the calculated data stored in the distributed memory. RDD is a collection of read-only objects distributed in a set of nodes. These collections are flexible, and if the data set is lost, they can be rebuilt. Iteration calculations are usually iterative calculations that are repeated for the same dataset, and the data will greatly improve IO operations in memory. This is also the core of Spark: memory computing. Spark is a memory-based iterative computing framework for applications that require multiple data sets for specific data sets. The more the number of times to repeat the operation, the greater the amount of data required to read, the greater the benefits, the amount of data is small but the calculation of the larger concentration of the occasion, the benefit is relatively small.

This paper focuses on the clustering of machine learning methods using distributed system processing to dig large data, through the spark of streaming media quality factors for measurement and analysis, correlation analysis in different technical conditions and the environment of the quality characteristics of streaming media and factors, derived from different elements such as Media content, terminal type, geographical location, viewing time and so on video streaming media quality and the relationship between the degree and extent.

Chapter 2 Spark Development Platform Overview

2.1 Spark background

2.1.1 Introduction to Spark

Spark is an open source cluster computing environment similar to Hadoop, and is very impressive in performance and iterative computing, and is now Apache's top incubator. Spark, developed by the University of California at the University of California at the University of California, is used to build large, low-latency data analysis applications. Spark enabled the memory distribution data set, in addition to providing interactive queries, it can also optimize the iterative workload. Spark is implemented in the Scala language, which uses Scala as its application framework, and Scala's language features also cast out most of Spark's success. The core part of the project code only 63 Scala files, very short and pithy. Unlike Hadoop, Spark and Scala are tightly integrated, where Scala can easily manipulate

distributed data sets as well as manipulating local collections. Although Spark was created to support iterations on distributed datasets, it was actually a supplement to Hadoop that could run in parallel in the Hadoop file system. This behavior can be supported by a third-party clustered framework named Mesos. Spark can be used to build large, low-latency data analysis applications.

2.1.2 Spark's applicable scenario

Spark is a memory-based iterative computing framework for applications that require multiple data sets for specific data sets. The more the number of times to repeat the operation, the greater the amount of data required to read, the greater the benefits, the amount of data is small but the calculation of the larger concentration of the occasion, the benefit is relatively small.

Due to RDD's characteristics, Spark does not apply applications that are asynchronous fine-grained update states, such as web service storage or incremental web crawlers and indexes. It is not suitable for the application model that is incremental. In general Spark's application is broader and more general.

2.1.3 Use in the industry

Spark project started in 2009, open source in 2010, now use: Berkeley, Princeton, Klout, Foursquare, Conviva, Quantifind, Yahoo !, Alibaba, Cloudera, Databricks, IBM, Intel, Taobao, etc. [3] Use Spark's python clone version of Dpark.

2.1.4 Spark vs. Hadoop

Spark provides a variety of data set operation types, unlike Hadoop only provides Map and reduce two operations. Such as map, filter, flatMap, sample, groupByKey, union, join, cogroup, reduceByKey, mapValues, sort, partionBy and other types of operations, Spark these operations called Transformations. At the same time also provide Count, collect, reduce, lookup, save and other actions.

This wide variety of data set operation types, to the development of the upper application of the user to provide a convenient. The communication model between the processing nodes is no longer the only Data Shuffle model like Hadoop. Users can name, materialize, and control the middle of the results of storage, partition and so on. It can be said that the programming model is more flexible than Hadoop.

Figure 2.1.4

2.2 RDD Framework Overview

2.2.1 Introduction to RDD

Resilient Distributed Dataset (RDD) is the most basic abstraction of Spark, is the abstract use of distributed memory, to achieve the operation of the local collection of the way to operate the abstract implementation of distributed data sets. RDD is Spark the core of things, but also the essence of design. It is understood as a large collection, all the data are loaded into memory, easy to reuse multiple times. First, it is distributed, can be distributed in multiple machines, the calculation. Second, it is flexible, in the calculation process, the machine's memory is not enough, it will and hard disk data exchange, to some extent will reduce performance, but can ensure that the calculation can continue. It represents a set of data that has been partitioned, immutable, and can be manipulated in parallel, and different dataset formats correspond to different RDD implementations. RDD must be serializable. RDD can cache to memory, each time the operation of the RDD data set after the results can be stored in memory, the next operation can be directly from the memory input, eliminating the MapReduce a large number of disk IO operations. This is more common for iterative computing machine learning algorithms, interactive data mining, efficiency is relatively large.

2.2.2 The main way to create RDD

1. Create from the Hadoop file system (or other storage system compatible with Hadoop), such as HDFS.

2. The new RDD is obtained by the existing RDD conversion.

2.2.3 Features of RDD

1. It is an immutable, partitioned collection object on a cluster node.

2. Create (eg, map, filter, join, etc) by way of parallel conversion.

3. Failure to rebuild automatically.

4. You can control the storage level (memory, disk, etc.) to reuse.

5. Must be serializable.

6. Is a static type.

2.2.4 Benefits of RDD

1. RDD can only be generated from persistent storage or through the transformations operation, which can be more efficient than the distributed shared memory (DSM), for the loss of part of the data partition only according to its lineage can be recalculated without the need to do a specific checkpoint.

2. RDD invariance, you can achieve class Hadoop MapReduce speculative execution.

3. RDD data partitioning feature, you can improve the performance of the data through the local, which is the same Hadoop MapReduce.

4. RDD is serializable, in the case of insufficient memory can be automatically degraded to disk storage, the RDD stored on disk, then the performance will be a big drop but not worse than the current MapReduce.

2.2.5 RDD storage and partitioning

1. Users can choose different storage levels to store RDD for reuse.

2. The current RDD is stored in memory by default, but when memory is low, RDD spills to disk.

3. RDD in the need to partition the data distributed in the cluster will be based on each record Key partition (such as Hash partition), in order to ensure that the two data sets in the Join can be efficient.

2.2.6 Fault tolerance

For streaming computing, fault tolerance is critical. First of all, we have to clear the Spark RDD fault tolerance mechanism. Each RDD is an immutable distributed recalculateable data set that records a deterministic operation of the lineage, so as long as the input data is fault tolerant, then any one of the RDD partitions (Partition) error or not available, can be used to re-calculate the original input data through the conversion operation.

2.2.7 Usability

Spark enhances usability by providing rich Scala, Java, Python APIs and interactive shells.

2.2.8 RDD internal representation

Each RDD in an internal implementation of RDD can be represented by five aspects:

1. Partition list (data block list)

2. Calculate the function of each slice (calculate the RDD according to the parent RDD)

3. List of dependencies on parent RDD

4. On the key-value RDD Partitioner

5. A list of predefined addresses for each data slice (such as the address of a data block on HDFS)

2.3 Local mode and Mesos mode

Spark supports both the Local call and the Mesos cluster. The algorithm is developed on the Spark. After the local mode is successfully debugged, the Mesos cluster runs directly. The algorithm does not need to be modified in any way except that the file is saved.

Spark's native mode supports multithreading and has a stand-

alone concurrency capability but not very strong. Local mode can save the results in local or distributed file systems, and Mesos mode must be saved in a distributed or shared file system.

2.4 Spark programming model

RDD is a read-only data partition set, note that the data set.

Only when there is an action on the RDD, all the operations on the RDD and its parent RDD will be committed to the cluster.

From code to dynamic operation, the components involved are shown in Figure 2.5.

Figure 2.5

Chapter 3 Spark platform build

3.1 Install the Ubuntu Linux operating system

Using wubi to install ubuntu, I am using the current LTS version of Ubuntu 12.04 LTS.

3.2 Configuration Install the Java JDK

3.2.1 Download jdk

Jdk 1.7 can be downloaded from the link here: http: //www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

If the CPU is 32-bit, choose to download Linux x86, if the CPU is 64-bit, select Linux x64. In general, if the computer is dual-core, certainly support 64-bit operating system. Or you can open the command terminal to run 'uname-a' command to look at the author's notebook to run this command results are as follows:

Figure 3.2.1

Linux ubuntu 3.11.0-22-generic # 38 ~ precise1-Ubuntu SMP Fri May 16 20:47:57 UTC 2014 x86_64 x86_64 x86_64 GNU / Linux

The later x86_64 indicates that the system is 64-bit. I use the version is jdk-7u60-linux-x64.tar.gz

3.2.2 Install jdk

Open the command terminal, the following command is executed in the terminal.

$ Sudo su -

This command will switch to the root user, that is, the highest authority of the user. Because the subsequent implementation of the jdk installation operation is carried out in the / usr / local directory, with the root user more convenient.

$ Cd / usr / local / lib

Cd is the abbreviation of change directory, switch the current directory.

$ Tar -zxvf /home/kavin/jdk-7u55-linux-x64.tar.gz

Tar is linux under the packaging and decompression command line tool, the specific details can refer to 'man tar'. This command will jdk-7u55-linux-x64.tar.gz decompression package to the current directory. After decompression is complete, the implementation of 'ls', you can see the current directory has a new directory called 'jdk1.7.0_55'

3.2.3 Configure environment variables

$ Gedit / etc / profile

Gedit is similar to Windows in Windoes Notepad editor, the file / etc / profile is linux configuration file. This command opens the configuration file for editing.

Add configuration:

Add the following three lines of code at the end of the / etc / profile file:

Export JAVA_HOME = / usr / local / lib / jdk1.7.0_55

Export CLASSPATH =.: $ JAVA_HOME / jre / lib / rt.jar: $ JAVA_HOME / lib / dt.jar: $ JAVA_HOME / lib / tools.jar

Export PATH = $ PATH: $ JAVA_HOME / bin

Save the file and exit.

Linux system, boot automatically after the implementation of / etc / profile configuration file. The export command sets or displays environment variables. The three lines of code, respectively, set up JAVA_HOME, CLASSPATH, and PATH these three environment variables.

$ Source / etc / profile

If you change the / etc / profile configuration file, it will only take effect in the new terminal, the terminal is now using the terminal will not take effect. If you want it to use the terminal, you need to use the source command to run the configuration file. This command will make the three environment variables of 3.2.3 take effect immediately.

3.2.4 Check whether the installation is successful

Figure 3.2.4

$ Java -version

This command checks whether the jdk installation is successful. Run this command, as long as no error indicates that the installation was successful.

Java environment installation and configuration is complete, this is the basis for our build Hadoop, because Hadoop platform is based on Java for development and operation.

3.3 build Hadoop environment

Hadoop installation is more cumbersome, for the following reasons: First, Hadoop has a lot of versions; Second, the official documents are not detailed, and sometimes update out of touch, Hadoop development too fast; Third, the online spread of the Document, either customized to some needs, or adding unwanted steps, or adding easy-to-understand steps. In fact, the installation is a very important step, only installed, to talk about the next step.

3.3.1 Download and install Hadoop

In the Hadoop home page provides a number of download chain http://hadoop.apache.org

I have this experiment, select the current hadoop stable version: Hadoop1.2.1, so I chose to download hadoop-1.2.1.tar.gz

Promote the stable version of Hadoop

$ Cd ~ / setup / hadoop

In the command terminal, '~' that the current login user's home directory. For example, at boot time, the login user is kavin, then in the command terminal, '~' means the directory '/ home / kavin', if the boot time, login user is john, then '~' means '/ home / John 'directory.

3.3.2 Configuration hadoop

Reference: http: //hadoop.apache.org/docs/stable1/single_node_setup.pdf

In accordance with the pseudo-distributed configuration, that is, with a machine running NameNode, SecondaryNameNode, DataNode, JobTracker, TaskTracker 5 tasks.

The configuration file is in the conf directory of the hadoop home directory.

Modify the contents of the core-site.xml file as follows:

<Configuration>

<Property>

<Name> fs.default.name </ name>

<Value> hdfs: // localhost: 9000 </ value>

</ Property>

</ Configuration>

Modify the contents of the mapred-site.xml file as follows:

<Configuration>

<Property>

<Name> mapred.job.tracker </ name>

<Value> localhost: 9001 </ value>
</ Property>
</ Configuration>
Change the contents of the hdfs-site.xml file as follows:
<Configuration>
<Property>
<Name> dfs.replication </ name>
<Value> 1 </ value>
</ Property>
</ Configuration>
Add the following statement to the hadoop-env.sh file:
Export JAVA_HOME = / usr / local / lib / jdk1.7.0_55
JAVA_HOME is defined as the directory where you installed the Java JDK

3.3.3 Install rsync and ssh

'Sudo apt-get installs ssh rsync'

This command installs ssh and rsync. Ssh is a well-known Secure Shell protocol Secure Shell Protocol. Rsync is a file synchronization command line tool.

3.3.4 Configuration ssh login

$ Ssh-keygen -t dsa -f ~ / .ssh / id_dsa

The implementation of this order to generate ssh public key / private key, the implementation process, there will be some tips to enter characters, you can directly enter the way.

$ Cat ~ / .ssh / id_dsa.pub >> ~ / .ssh / authorized_keys

Ssh for remote login when you need to enter the password, if the public key / private key way, you do not need to enter the password. The above way is to set the public key / private key login.

$ Ssh localhost'

Figure 3.3.4

The first time the implementation of this order, there will be a prompt, enter 'yes' and then enter.

3.3.5 Start hadoop

$ Cd ~ / usr / hadoop / hadoop-1.2.1

$ ./bin/hadoop namenode -format

Format NameNode.

$ ./bin/start-all.sh

Start all nodes, including NameNode, SecondaryNameNode, JobTracker, TaskTracker, DataNode.

$ Jps

Check the process is running, then, you should see six java virtual machine process, namely Jps, NameNode, SecondaryNameNode, DataNode, JobTracker, TaskTracker, if this 6 is right, indicating that the start is success.

Figure 3.3.5

At this point, on the standstill successfully installed hadoop. Spark need to run on hadoop. Installed hadoop, then Zang spark, because spark need to use hadoop function.

3.4 Installing Scala

Spark developed using Scala, before installing Spark, first in the various sections will be installed Scala. While Spark 1.0.0 relies on Scala 2.10.4, install this version of scala.

After downloading Scala2.10.4, open the terminal.

$ Tar -zxf scala-2.10.4.tgz

$ Sudo mv scala-2.10.4 / usr / local

Unzip and move to the / usr / local directory

$ Sudo gedit / etc / profile

Add a line of code in / etc / profile

Export SCALA_HOME = / usr / local / scala-2.10.4

Export PATH = $ PATH: $ SCALA_HOME / bin

Save and exit

$ Source / etc / profile

Update the environment variable

$ Scala -version

Test whether the installation is successful

Figure 3.4

3.5 deploy Spark

3.5.1 Download spark-1.0.0 source

Download the source package spark-1.0.0.tgz from Spark's official address.

Http://spark.apache.org/downloads.html

3.5.2 Decompression compilation

Will be spark-1.0.0.tgz extract compilation:

$ Tar -zxvf spark-1.0.0.tgz

Run sbt to compile:

$ Cd ~ / spark-1.0.0

$ Sbt assembly

Figure 3.5.3

This step will download a lot of libraries, and then compile, compile time will probably be about 1 hour.

3.5.3 Set the SPARK_HOME environment variable

And add SPARK_HOME / bin to PATH

$ Sudo gedit etc / profile

Open the / etc / profile Add the following code

Export SPARK_HOME = $ HOME / spark-1.0.0

Export PATH = $ PATH: $ SPARK_HOME / bin

$ Source / etc / profile

Make environmental variables effective

3.5.4 Verify the spark environment

Open the interactive spark command environment, that is, local mode

Start Spark shell: $ ./bin/spark-shell

Figure 3.5.5.1

According to the official document in the quick-start [4] for some file read operation, and the content of the deal, do mapreduce like things. In sparkshell run the most simple example wordcount, enter the code:

Scala> sc.textFile ('README.md'). Filter (_. Contains ('Spark')). Count

The above code counts how many lines of Spark are included in README.md.

Long = 15

That is, the number of lines with 'Spark' is 15 lines.

And then enter scala> val count = file.flatMap (line => line.split (')). Map (word => (word, 1)). ReduceByKey (_ + _)

Scala> count.collect ()

As shown in Figure 3.5.5.2

Figure 3.5.5.2

In the example of word count, map all the text of a text, and then reduce them by word, and finally sum up the number of words. RDD can be read from disk and then kept in memory, improving performance, you can see this and Hadoop most of the disk-based speed is much faster.

Then use the spark's own run script to run the spark program

$ ./bin/run-example org.apache.spark.examples.SparkPi

Pi is roughly 3.14154

Calculate the Pi value

Figure 3.5.5.3

Chapter 4 Machine Learning Algorithm Model

4.1 Cluster analysis

Cluster analysis is one of the key problems in data mining and machine learning. It is widely used in many fields, including machine learning, data mining, pattern recognition, decision support, image analysis and biological information. It is one of the most important data analysis methods. Clustering is the process of grouping data objects into clusters that consist of similar objects. A cluster generated by a cluster is a collection of data objects that are similar to objects in the same cluster, and are different from objects in other clusters. The clustering algorithm can be divided into a hierarchical approach, a hierarchical approach, a density-based approach, a grid-based approach, and a model-based approach.

We generally summarize the data into an unsupervised learning problem. On this issue, our goal is to combine a part of the entity in a sense of similarity and another part of the entity. Clustering is often used for exploratory analysis, or as a component of a hierarchical supervisory study pipeline network.

4.2 K-means algorithm

K-means [6] algorithm is one of the most widely used clustering algorithm based on partition [7], n objects are divided into k clusters, so that the cluster has a high degree of similarity. The calculation of similarity is based on the average of the objects in a cluster. It is similar to the maximum expected algorithm for dealing with mixed normal distributions because they are trying to find the center of natural clustering in the data.

The algorithm first randomly selects k objects, each of which initially represents the average or center of a cluster. For each remaining object, assign it to the nearest cluster based on its distance from the center of each cluster, and then recalculate the average of each cluster. This process is repeated until the criterion function converges.

Figure 4.2

It assumes that the object attribute comes from the space vector, and the goal is to minimize the sum of the mean square errors within each group. Suppose there are k groups Si, i = 1,2, ..., k. Mi is the center of gravity of all elements xj in group Si, or center point.

4.2.1 Algorithm description

1. Select the number of clusters k.

2. Any k clustering, and then determine the cluster center, or directly generate k centers.

3. Calculate the Euclidean distance between each object and the central object in the sample set according to the mean (central object) of all the objects in each cluster and re-divide the corresponding object according to the minimum distance. That is, for each point to determine its clustering center point to calculate its clustering new center.

4. And then calculate the new center of its clustering, that is, to recalculate the mean of each cluster.

5. Repeat the above steps until the convergence requirements are met. (Usually the center of the point is no longer changed).

4.2.2 Performance analysis of the algorithm

Advantage:

1. The kmeans algorithm is a classical algorithm to solve the clustering problem. The algorithm is simple and fast.

2. For large data sets, the algorithm is relatively scalable and efficient because its complexity is about O (nkt), where n is the number of all objects, k is the number of clusters, and t is the number of iterations. Usually k << n. This algorithm often ends with local optimality.

3. The algorithm tries to find the k divisions that minimize the squared error function. When the cluster is dense, spherical or cluster-like, and the difference between clusters and clusters is obvious, its clustering effect is very good.

Disadvantages:

1. The kmeans method can only be used if the average of the cluster is defined, and does not apply to certain applications, such as data that does not apply to a classification attribute.

2. Requires the user to give in advance the number of clusters to be generated.

3. Sensitive to the initial value, for different initial values, may lead to different clustering results.

4. Not suitable for the discovery of non-convex shape of the cluster, or a large difference in the size of the cluster.

5. For 'noise' and isolated point data sensitive, a small amount of such data can have a significant impact on the average.

Chapter 5 Experimental Processing and Experimental Analysis

5.1 Prepare the dataset to be measured

The data source selected for this experiment is the data provided by the mobile client server of the PPTV. The data to be measured is named qos, containing 10140, consisting of six fields: user uid, ip address, watch video time, buffer count, drag count, and non-drag buffer times.

For example, the first line of data in the text is: 357238047592824 221.181.192.29 236 0 0 0

The first two columns use the command to remove.

$ Awk '{print $ 3, $ 4, $ 5, $ 6}' qos.txt> QOS.txt

Print the third, fourth, fifth, and sixth columns of the text document qos and save them in the newly created QOS text document.

The first line of data preprocessing becomes: 236 0 0 0

Look at the processed dataset QOS.

$ Grep " QOS.txt

Print out the contents of the QOS document to the big screen, you can see Figure 5.1 is selected for a paragraph.

Figure 5.1

5.2 Number of data sets to be measured

$ ./bin/spark-shell

Start the Spark shell

Scala> val countlines_data = sc.textFile ('QOS.txt')

Scala> countlines_data.count ()

Get Long = 10140

That is, the length of the text is 10140 lines

5.3 using KMeans algorithm

5.3.1 Download NumPy

A K-means clustering program using MLlib.

MLlib requires NumPy 1.7+

Calling MLsib with KMeans directly in Python will prompt you to install NumPy 1.7 or later.

First install the pip with the following command, pip is a tool for Python to install and manage the extension library.

Sudo apt-get installs python-pip

Install git:

Sudo apt-get installs git

Install Python-dev, Python's development environment for future compilation of other extensions

Sudo apt-get installs python-dev

You can quickly install this library with the apt-get command:

Fsudo apt-get installs python-scipy

If you need to compile and install through pip, you can use the apt-get command to install all the libraries needed for compilation:

Sudo apt-get build-dep python-numpy

And then install the pip command:

Sudo pip installs numpy

This numpy installed on the success, and thus can use Python MLlib.

5.3.2 The python code for the Kmeans algorithm

```
Import sys
Import numpy as np
From pyspark import SparkContext

Def parseVector (line):
Return np.array ([float (x) for x in line.split (")])

Def closestPoint (p, centers):
BestIndex = 0
Closest = float ('+ inf')
For i in range (len (centers)):
TempDist = np.sum ((p - centers [i]) ** 2)
If tempDist <closest:
Closest = tempDist
BestIndex = i
Return bestIndex

If __name__ == '__main__':
If len (sys.argv)! = 4:
Print >> sys.stderr, 'Usage: kmeans <file> <k> <convergeDist>'
Exit (-1)

Sc = SparkContext (appName = 'PythonKMeans')
Lines = sc.textFile (sys.argv [1])
// read data can be from HDFS can also be hard disk as RDD
Data = lines.map (parseVector) .cache ()
K = int (sys.argv [2])
// k clusters.  The number of cluster centers
ConvergeDist = float (sys.argv [3])
// iteration convergence conditions
// randomly initialize K cluster centers
KPoints = data.takeSample (False, K, 1)
TempDist = 1.0

While tempDist> convergeDist:
Closest = data.map
Lambda p: (closestPoint (p, kPoints), (p, 1)))
// closest (category, (point, 1)), 1 is used to follow the statistics of the number of points in each class; lambda is ALS regularization parameters
PointStats = closest.reduceByKey (
Lambda (x1, y1), (x2, y2): (x1 + x2, y1 + y2))
// Calculate the coordinates of the points by category, and the total number of nodes in the category (category, (point vector sum, number of points))
NewPoints = pointStats.map (
Lambda (x, (y, z)): (x, y / z)). Collect ()
// generate a new clustering center for the Map (category, new clustering center)
TempDist = sum (np.sum ((kPoints [x] - y) ** 2) for (x, y) in newPoints)
// calculate the delta value of the current vector vector
For (x, y) in newPoints:
KPoints [x] = y
// update the cluster center to kPoint
Print 'Final centers:' + str (kPoints)
```

5.3.3 Run in Spark

$ ~ / Spark-1.0.0 $ ./bin/spark-submit ~ / kmeans.py 'QOS.txt' 5 0.1

The first parameter is k, the cluster centroids.

The second argument is convergeDist, that is, the iteration convergence condition.

Figure 5.3.3

Then, when the threshold is 0.1, five points represent the center of the cluster.

Clustering center point 1: (3.70995437e + 03,2.16468254e + 00,2.99404762e + 00,1.30952381e-01)

Clustering center point 2: (2.46609167e + 02,1.31449953e + 00,1.96276894e + 00,3.46117867e-02)

Clustering center point 3: (2.33774165e + 03,1.74343675e + 00,2.19272076e + 00,1.06801909e-01)

Clustering center point 4: (1.14494467e + 03,2.24523364e + 00,3.05345794e + 00,8.56074766e-02)

Clustering center point 5: (6.05363436e + 03,1.40969163e + 00,2.00881057e + 00,1.71806167e-01)

5.4 Clustering effects and analysis

5.4.1 Selection of k values in Kmeans

The simplest way to determine the initial cluster center is to randomly select K points as the initial cluster center point.

So there are two ways to select the initial cluster points:

1) Choose the distance from each other as far as possible K points

2) The data is clustered with the hierarchical clustering algorithm or the Canopy algorithm. After the K clusters are obtained, a point is selected from each cluster, which can be the center point of the cluster or the distance is nearest to cluster center point.

The method of this experiment is to choose from each other as far as possible from the K points. First randomly select a point as the first initial cluster center point, and then select the point furthest from the point as the second initial cluster center point, and then select the distance from the first two points of the nearest point as the largest point. The center of the third initial cluster, and so on until the initial cluster center is selected.

1. Given a suitable cluster index, such as average radius or diameter.

2. The diameter of a cluster is the maximum distance between any two points in a cluster.

3. The radius of a cluster is the maximum of all the points in the cluster.

4. As long as we assume that the number of clusters is equal to or higher than the number of real clusters, the indicator will rise slowly, and once the attempt to get less than the number of real clusters, the index will rise sharply.

Figure 5.4.1 is the effect of clustering effect and cluster index when K is from 2 to 9.

Figure 5.4.1

When the value of K is from 2 to 9, the cluster-like cluster is the weighted average of the average centroid distance of K cluster. The horizontal axis is the number of selected clusters, and the vertical axis is the weighted average of the mean centroid distance of clusters. It is obvious that when K is 5, the descending trend of the cluster is the fastest, so the correct value of K should be 5. Figure 5.4.2 is the

specific data:

Figure 5.4.2

5.4.2 Analysis of the results of this experiment

Clustering center point 1: (3.70995437e + 03,2.16468254e + 00,2.99404762e + 00,1.30952381e-01)

Clustering center point 2: (2.46609167e + 02,1.31449953e + 00,1.96276894e + 00,3.46117867e-02)

Clustering center point 3: (2.33774165e + 03,1.74343675e + 00,2.19272076e + 00,1.06801909e-01)

Clustering center point 4: (1.14494467e + 03,2.24523364e + 00,3.05345794e + 00,8.56074766e-02)

Clustering center point 5: (6.05363436e + 03,1.40969163e + 00,2.00881057e + 00,1.71806167e-01)

According to the five cluster center points can be learned, the user's non-drag buffer times relative to the number of buffers and drag the number of times is relatively small, and users in the video viewing time of more than 1000 seconds, there are 2 thousand seconds, 3 thousand seconds, 6 thousand seconds, which is the length of the video and the user's degree of love related to the video.

According to the clustering center point, especially the observation cluster center points 4 and 5, we can see that when the residence time of the video becomes longer, the corresponding non-drag buffer times are increasing, and the number of drags is decreasing. The more users drag the number, indicating that the video may not be attractive, it may be limited user time, of course, will eventually lead to the video to stay on the time to reduce. The increase in the number of non-drag buffers may be related to the user's network, the server where the video is located, the geographic location, and so on. The increase in the number of non-drag buffers may also result in an increase in user impatience, resulting in a reduction in user dwell time. [8]

From the results of this experiment, the number of buffers does not change with the user's viewing time, drag number, and non-drag buffer times. Drag the number of times more than the number of non-drag buffer, the actual number of times associated with the buffer, so in addition to the user drag the video playback buffer, the user's playback terminal, speed, etc. does not affect the video buffer to a large extent.

5.5 Conclusion

When there is a single video dwell time, and the dwell time is much smaller than the length of the entire video, you can know which part of video that may less attractive through the user's approximate dwell time, resulting in the user did not want to continue to watch the video, this can analyze the video streaming media content to make the appropriate decision-making and adjustment.

You can analyze the individual locations of a single video by viewing the user's different geographic locations by analyzing the number of non-drag buffers, as well as analyzing the geographic location and related features related to the streaming media business.

The same video location of the same geographical location and network under the different end of the cluster center can be used to analyze the different users in the different types of terminal player. Of course, it can also be used to analyze the same video in different coding cases.

That is, as long as there is data, you can analyze the video streaming media quality and media content according to the model to, terminal type, location, viewing time and other factors.

Spark is a very powerful data mining tool that requires a deeper understanding. Analyze the data under the era of large data by using it, and in-depth mining data, data can be used to continue the complex analysis to achieve the results you want to know. Product managers can study the user experience, companies can make adjustments to maximize the benefits.

REFERENCES

[1] http://hadoop.apache.org/
[2] AMP:https://amplab.cs.berkeley.edu/software/
[3] Powered By Spark:
https://cwiki.apache.org/confluence/display/SPARK/Powered+By+Spark
[4] Spark quick-start:http://spark.apache.org/docs/latest/quick-start.html
[5] Cluster analysis:http://en.wikipedia.org/wiki/Cluster_analysis
[6]MacQueen, J. B. .Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1. University of California Press. pp. 281 – 297. Retrieved 2009.
[7] K-means clustering:http://en.wikipedia.org/wiki/K-means_clustering
[8]Florin Dobrian, Asad Awan, Dilip Joseph, Aditya Ganjam, Jibin Zhan, Vyas Sekar, Ion Stoica, Hui Zhang. Understanding the Impact of Video Quality on User Engagement. SIGCOMM 2011 .
[9]http://cwiki.apache.org/confluence/display/SPARK/
[10] Athula Balachandran,Vyas Sekar,Aditya Akella,Srinivasan Seshan,Ion Stoica,Hui Zhang.A Quest for an Internet Video Quality-of-Experience Metric.HotNet 2012.
[11] Florin Dobrian, Asad Awan, Dilip Joseph, Aditya Ganjam, Jibin Zhan, Vyas Sekar, Ion Stoica, Hui Zhang. Understanding the Impact of Video Quality on User Engagement.SIGCOMM 2011 .
[12] Ahahzad Ali, Anket Mathur, Hui Zhang. Measurement of Commercial Peer-to-Peer Live Video Streaming. Workshop in Recent Advances in Peer-to-Peer Streaming.August,2006.
[13] Phillipa Gill, Martin Arlitt, Zongpeng Li, Anirban Mahanti. YouTube Traffic Characterization: A View From the Edge. In Proc. IMC, 2007.