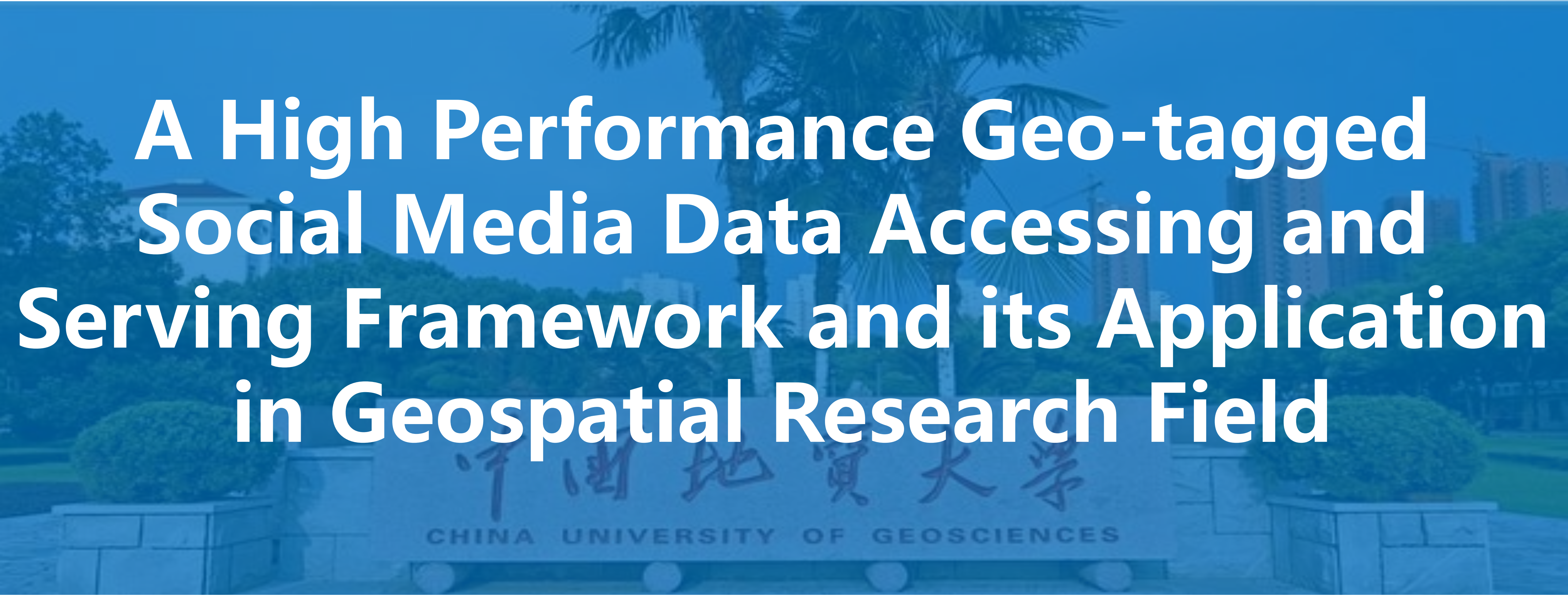
中國地質大學
CHINA UNIVERSITY OF GEOSCIENCES



A High Performance Geo-tagged Social Media Data Accessing and Serving Framework and its Application in Geospatial Research Field

肖 濛

中国地质大学（武汉） 信息工程学院

Introduction

There are more than 2 billion people in the world use all kinds of social media applications. Through the various social media applications, people can create and share all kinds of information by themselves.

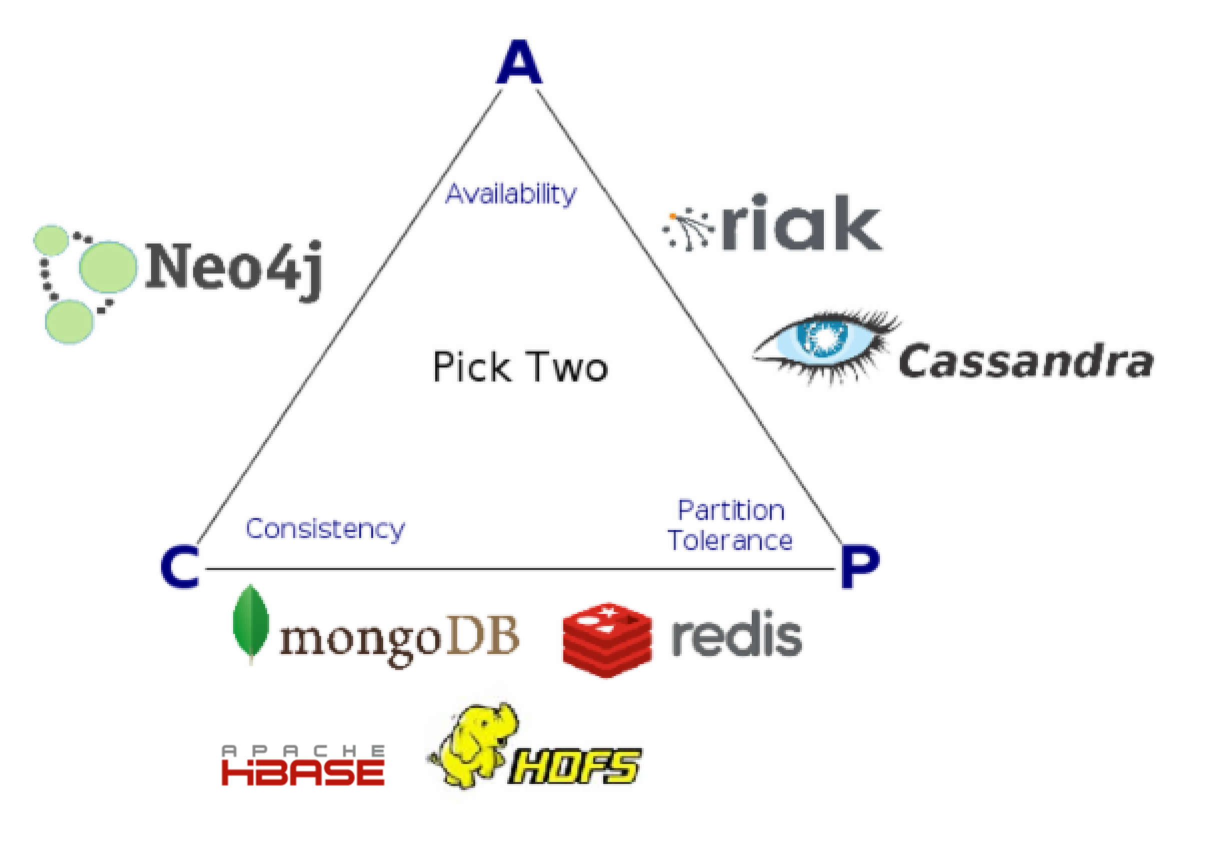
Introduction

Among the heterogeneous social media data, the geo-tagged ones are valuable information, which contain location information which generated by the GPS sensor in the smart phone or the nearest address based on the IP location of a computer.

Why Drop RMDBS?

The traditional GIS industry is generally used in RMDBS like MySQL, but in order to store such a multifarious data like social media Check-in data, RMDBS's bad scalability make it impossible to solve our problem.

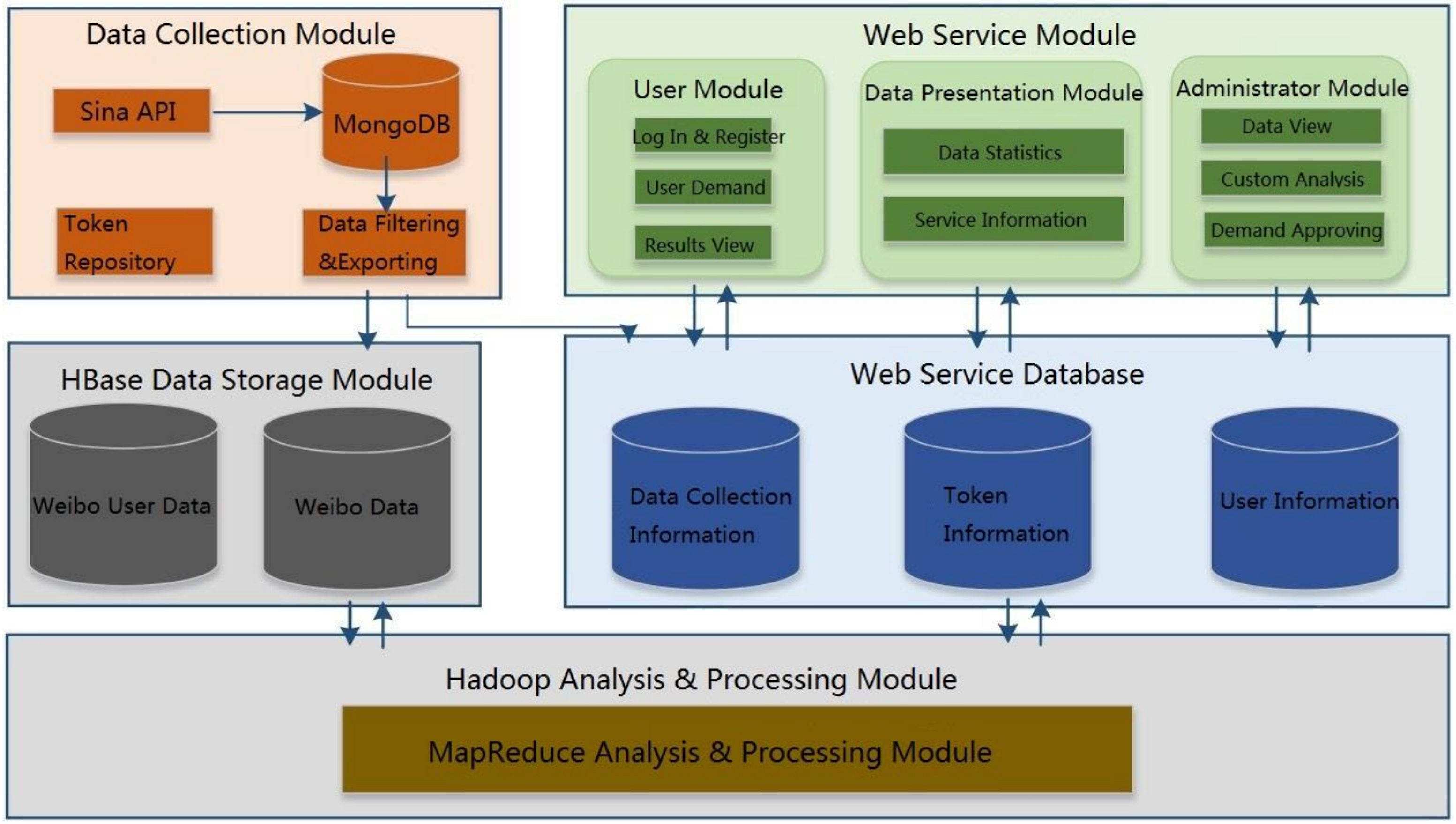
Why HBase?



So When P happens, choose A or C?

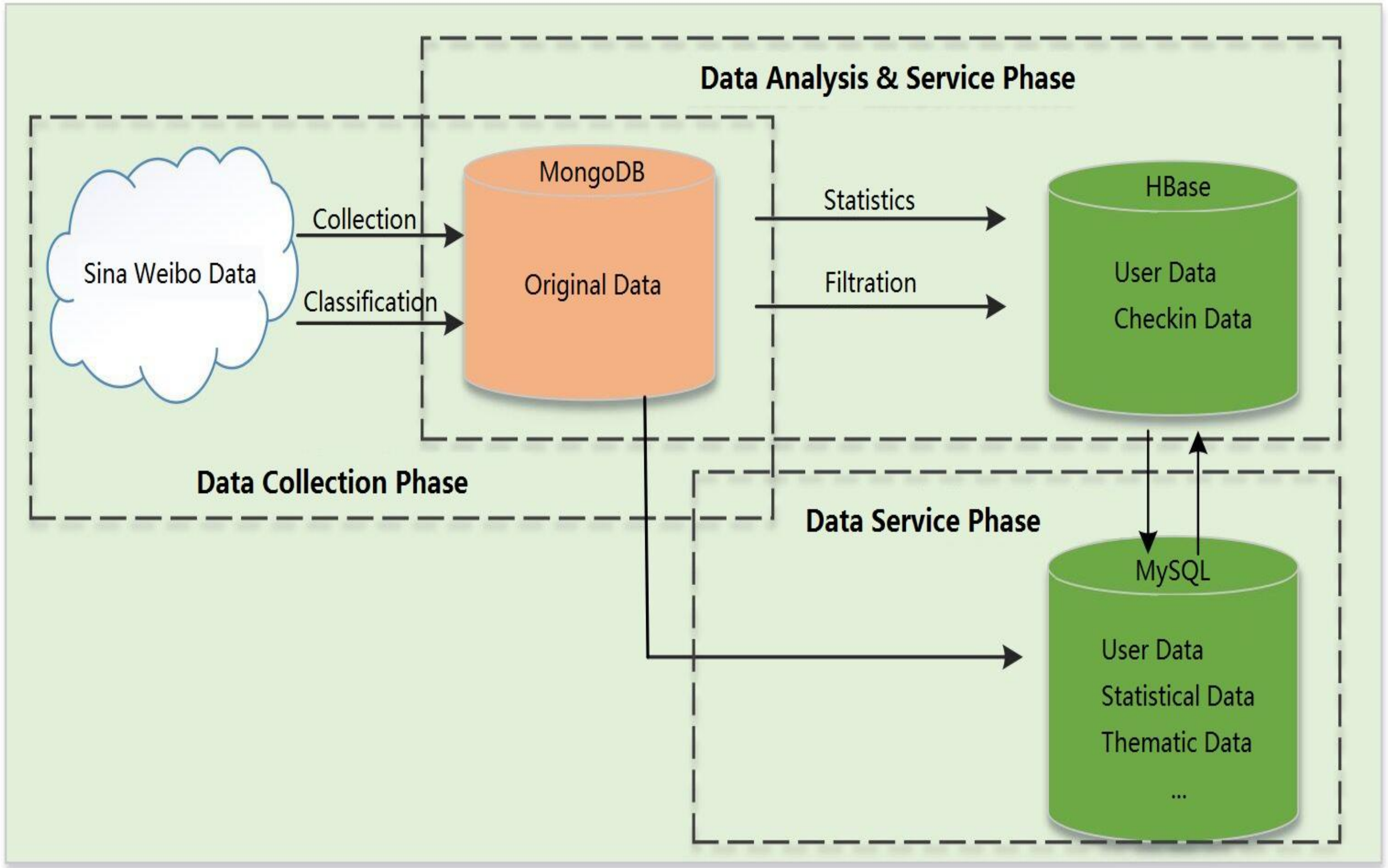
Because of the CAP can only be met two of the three and we are more inclined to a high degree of consistency of the database, which the most active open-source database.

We need to support the MapReduce programming style database to do the data analysis, the only available is MongoDB and HBase. In fact, HBase support for MR is much more better, while compared to the Mongodb base on hard disk, the bottom is based on HDFS. So in this set of systems we use Hbase as the underlying database



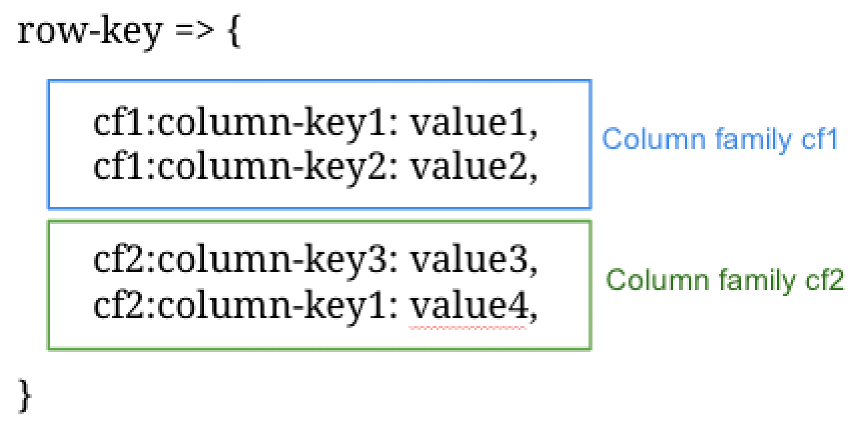
In the previous analysis, we can propose such as the left figure based on a distributed No-SQL database to make social media data get efficient acquisition, storage, analysis and data visualization

Data Grab



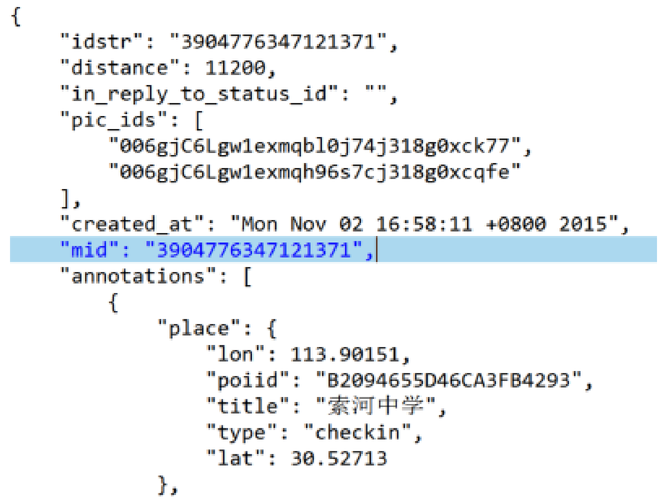
Weibo API is the most efficient way to access to the current data. Based on the official Weibo API interface and MongoDB as a transit database, we propose the left part of the data capture part of the framework

HBase Data Stock

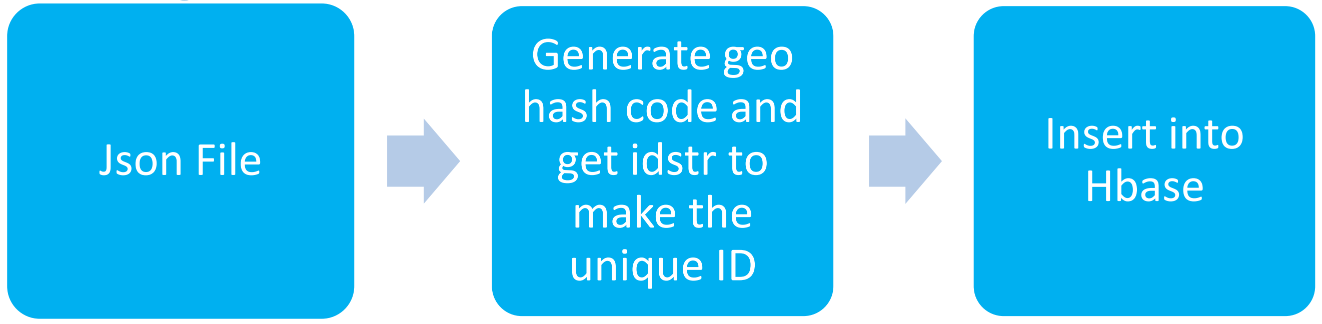


```
row-key => {
    cf1:column-key1: value1,
    cf1:column-key2: value2,        Column family cf1

    cf2:column-key3: value3,
    cf2:column-key1: value4,        Column family cf2
}
```

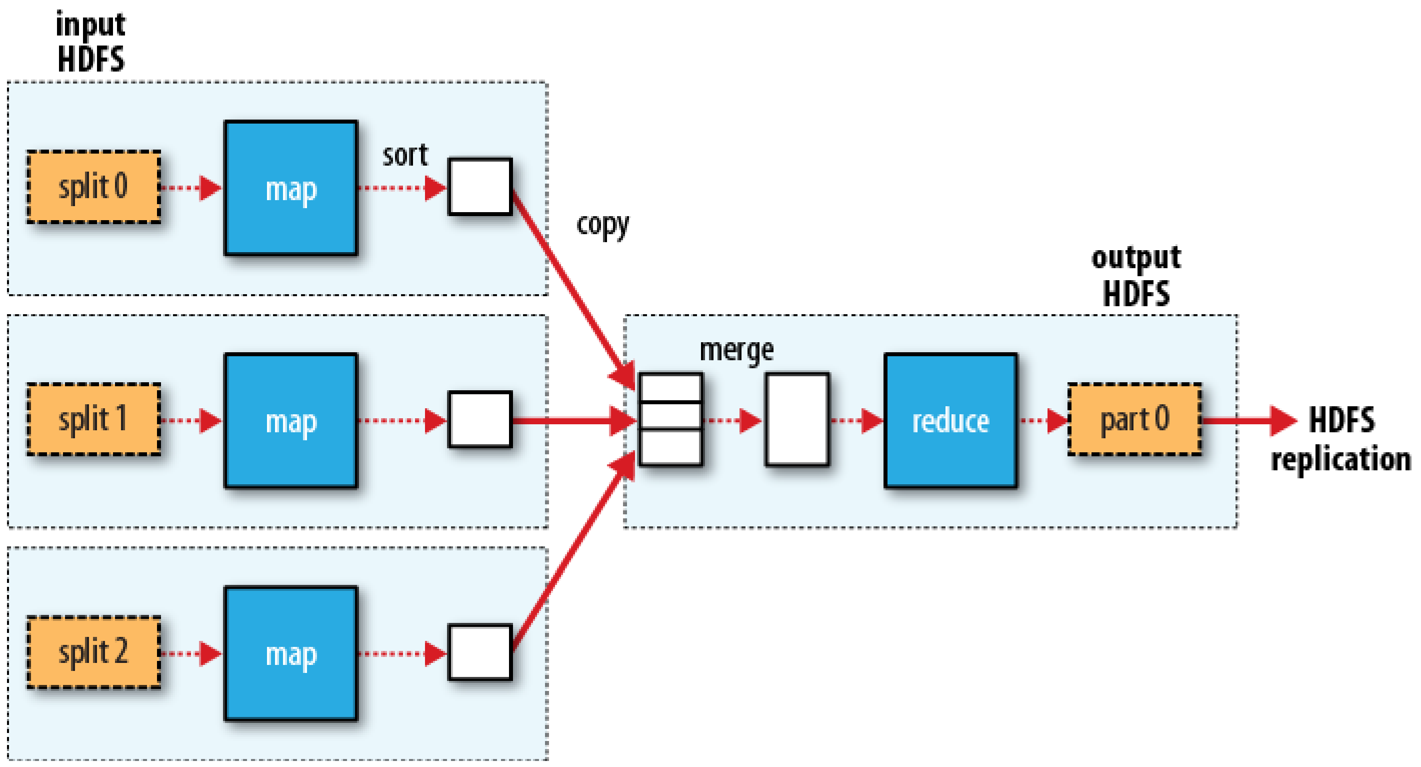


```
{
    "idstr": "3904776347121371",
    "distance": 11200,
    "in_reply_to_status_id": "",
    "pic_ids": [
        "006gjC6Lgw1exmqbl0j74j318g0xck77",
        "006gjC6Lgw1exmqh96s7cj318g0xcqfe"
    ],
    "created_at": "Mon Nov 02 16:58:11 +0800 2015",
    "mid": "3904776347121371",
    "annotations": [
        {
            "place": {
                "lon": 113.90151,
                "poiid": "B2094655D46CA3FB4293",
                "title": "索河中学",
                "type": "checkin",
                "lat": 30.52713
            },
```



As you can see in the top diagram There are three steps to insert the check-in data into Hbase.
Because of HBase builds indexes based on RowKey only, and in our study, the location relationship is the more important relationship between the two Check-in Weibos, we generate a unique ID for each Weibo by its geographic coordinates and idstr. The unique RowKey that represent its location.

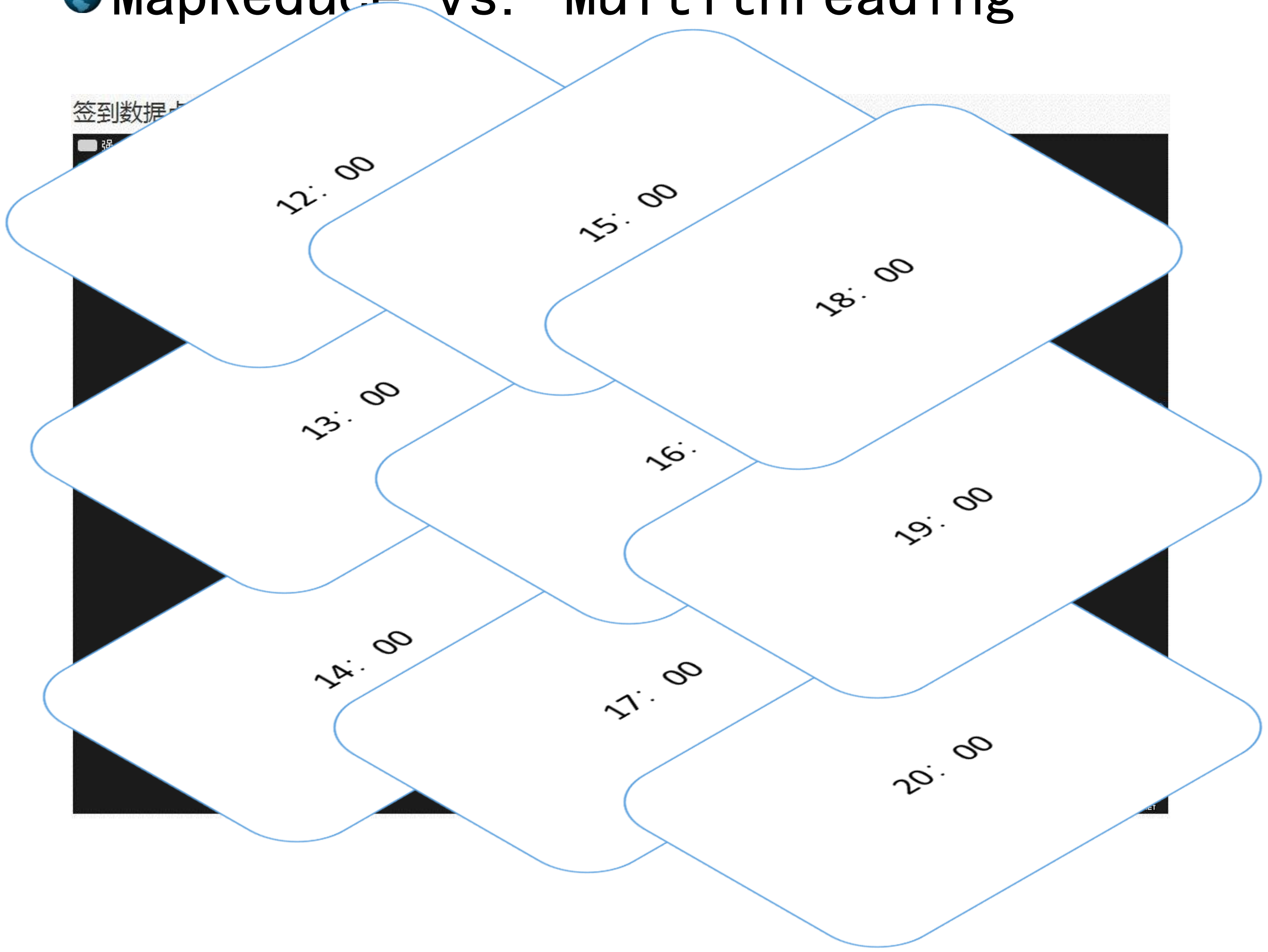
Hadoop & MapReduce



Not only do we periodically insert the Json format data from the grab system into HBase for persistent storage, but also upload some of the hot city data and hot fields to HDFS

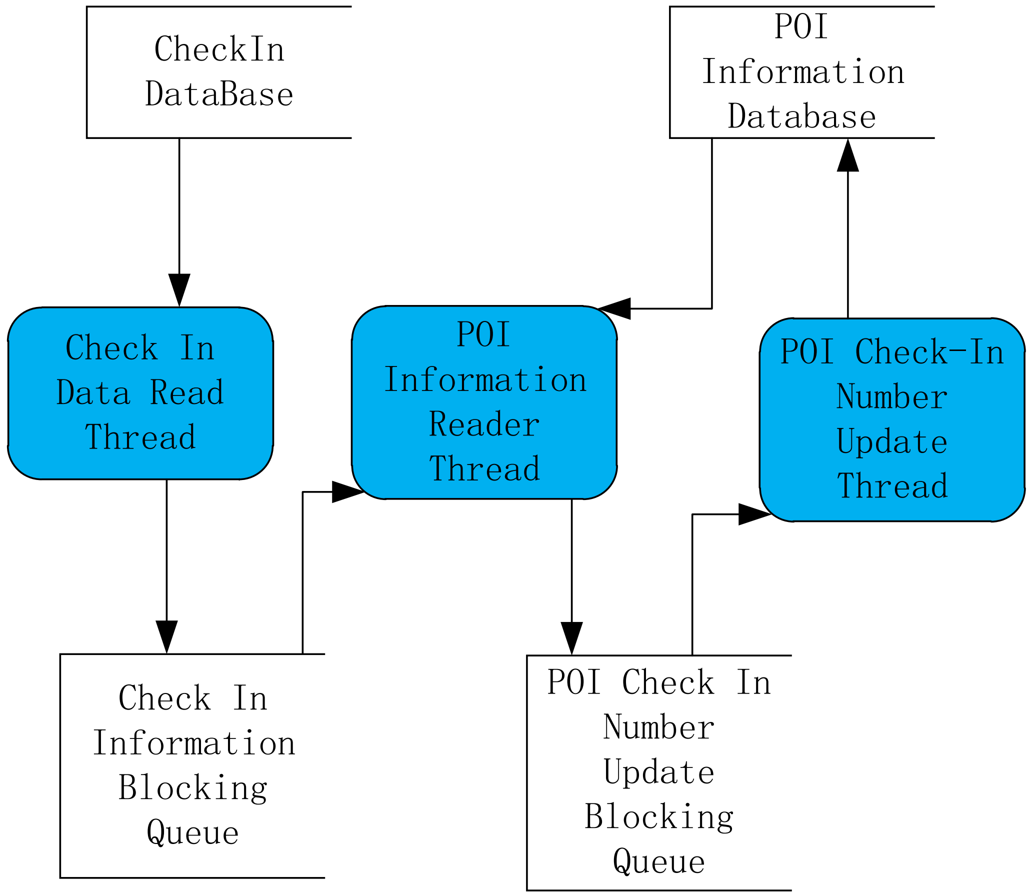
With efficient MapReduce architecture, we only need a thousandth of the java program cost time to complete the complex calculations

MapReduce vs. Multithreading



The left is the 2014 Weibo check-in data visual by a map.
In order to demonstrate the superiority of MapReduce and multithreaded architectures, and to lay the groundwork for subsequent data analysis, we designed a POI placement experiment, which is based on each Check-in Weibo with the correspond POI information. For the Weibo in which the layer of time, so that the POI point’s Check-in number in that time plus one.
This experiment can build a POI hot area model order by times, and in the subsequent data analysis can be used as a trend forecasting model.

MapReduce vs. Multithreading


CheckIn DataBase
POI Information Database
Check In Data Read Thread
POI Information Reader Thread
POI Check-In Number Update Thread
Check In Information Blocking Queue
POI Check In Number Update Blocking Queue

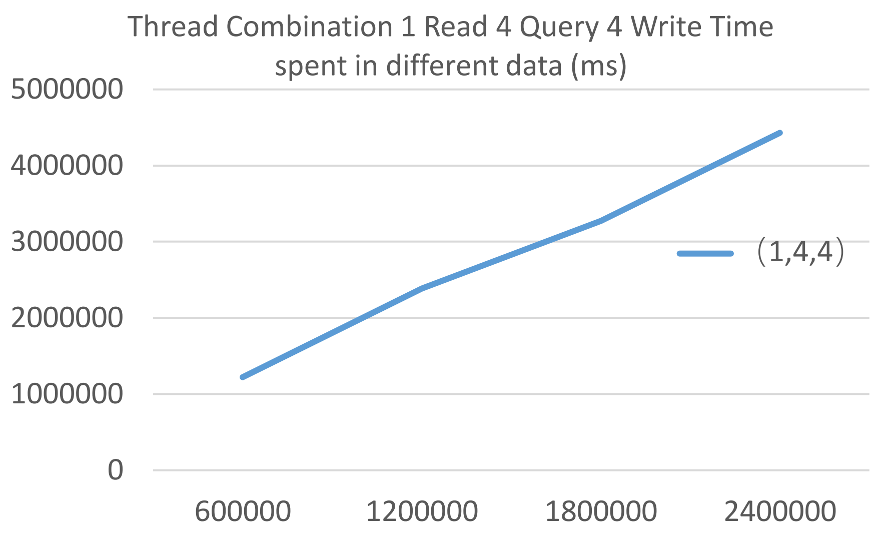
As we can see in the left figure, in order to compare the performance of the MR architecture, we use the Java multi-threaded design pattern.

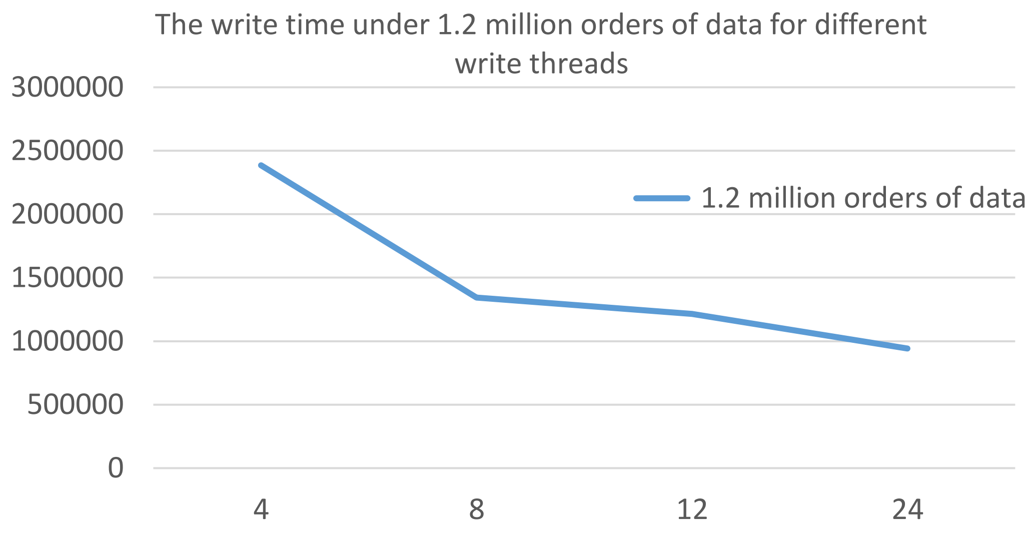
Through a priority queue act as a buffer, and a database read thread as a producer;

M threads through the database to read the Weibo Check-in point to check its POI point information;

N threads to update the database POI point to check the number.

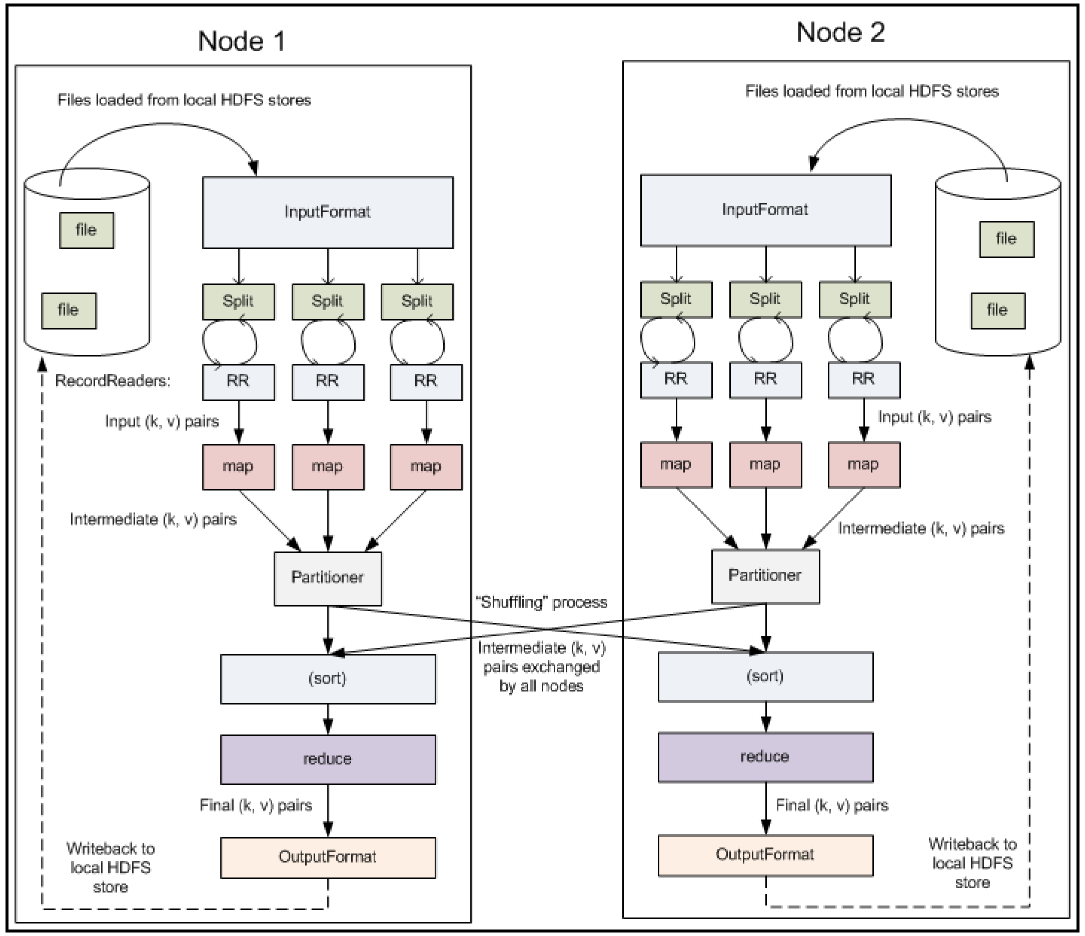
MapReduce vs. Multithreading





Can be seen in the face of millions of levels of data, the use of Java multithreading time spent in more than 20 minutes. With observing the amount of data in the blocking queue we can found that, the main factor restricting multi-threading is POI Check-in Counting part.
When you add more threads to write threads, as you can see in the left figure, the whole program become much more efficient. But they still need minutes to hour level of time to finish this work.

MapReduce vs. Multithreading



For the coding the MapReduce function, the code ideas are as follows

As shown in the figure to the right, at first, we have three years of Beijing Check-in data which is storage in HDFS as Json format. In the Map part, the cluster read the file into LongWritable-Text form, and converted to (POI Inform-1) The form is passed to the Reducer function for counting

In the Reduce part, the KV data from the Map are counted and converted into (POI Inform – times) form then output to the output file.

In the face of Beijing three years of 600 million Check-in data, MapReduce cost only 65 seconds, far better than Java multithreading

HBase vs. MySQL

The right figure is Baidu map when you search "hotel" keyword and generate tag according to the different regions of the POI point number.

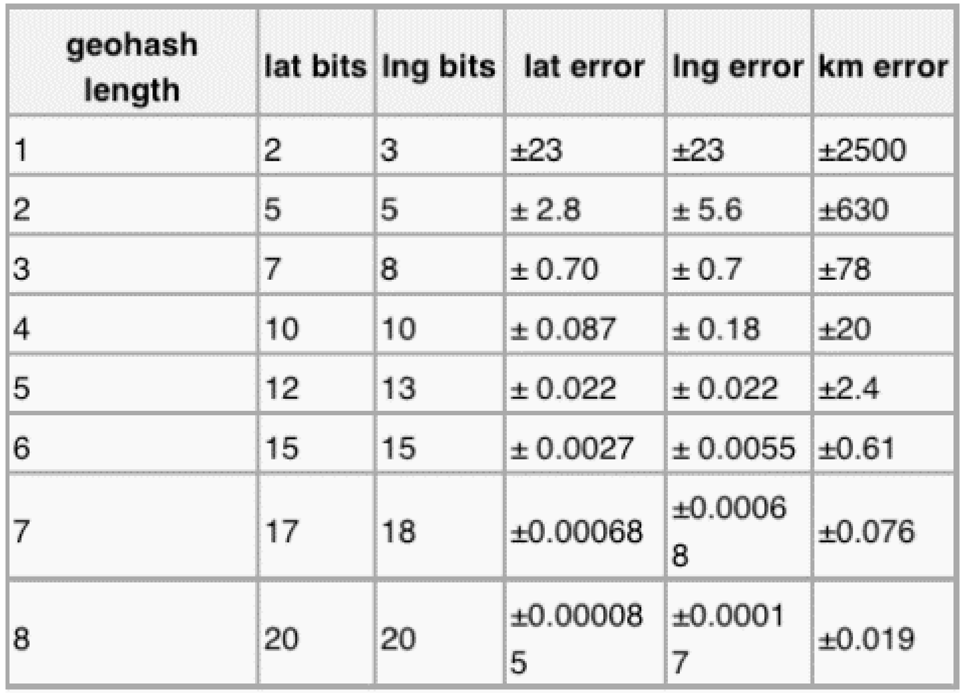
We also want to build the Weibo database to get number of keywords in the region client want, which is one of the core features of our system.

In the previous, we propose that HBase is well suited to our database for scalability and high availability and partition fault tolerance. But can the performance go beyond RMDBS?

HBase vs. MySQL – GeoHash

We use GeoHash in our project.

The GeoHash lib we use can be found at https://github.com/kungfoo/geohash-java



geohash length	lat bits	lng bits	lat error	lng error	km error
1	2	3	±23	±23	±2500
2	5	5	± 2.8	± 5.6	±630
3	7	8	± 0.70	± 0.7	±78
4	10	10	± 0.087	± 0.18	±20
5	12	13	± 0.022	± 0.022	±2.4
6	15	15	± 0.0027	± 0.0055	±0.61
7	17	18	±0.00068	±0.00068	±0.076
8	20	20	±0.000085	±0.00017	±0.019

Geohash gives us a simple way to construct a two-dimensional georeferenced index on a one-dimensional string

HBase vs. MySQL – Environment

HBase Cluster

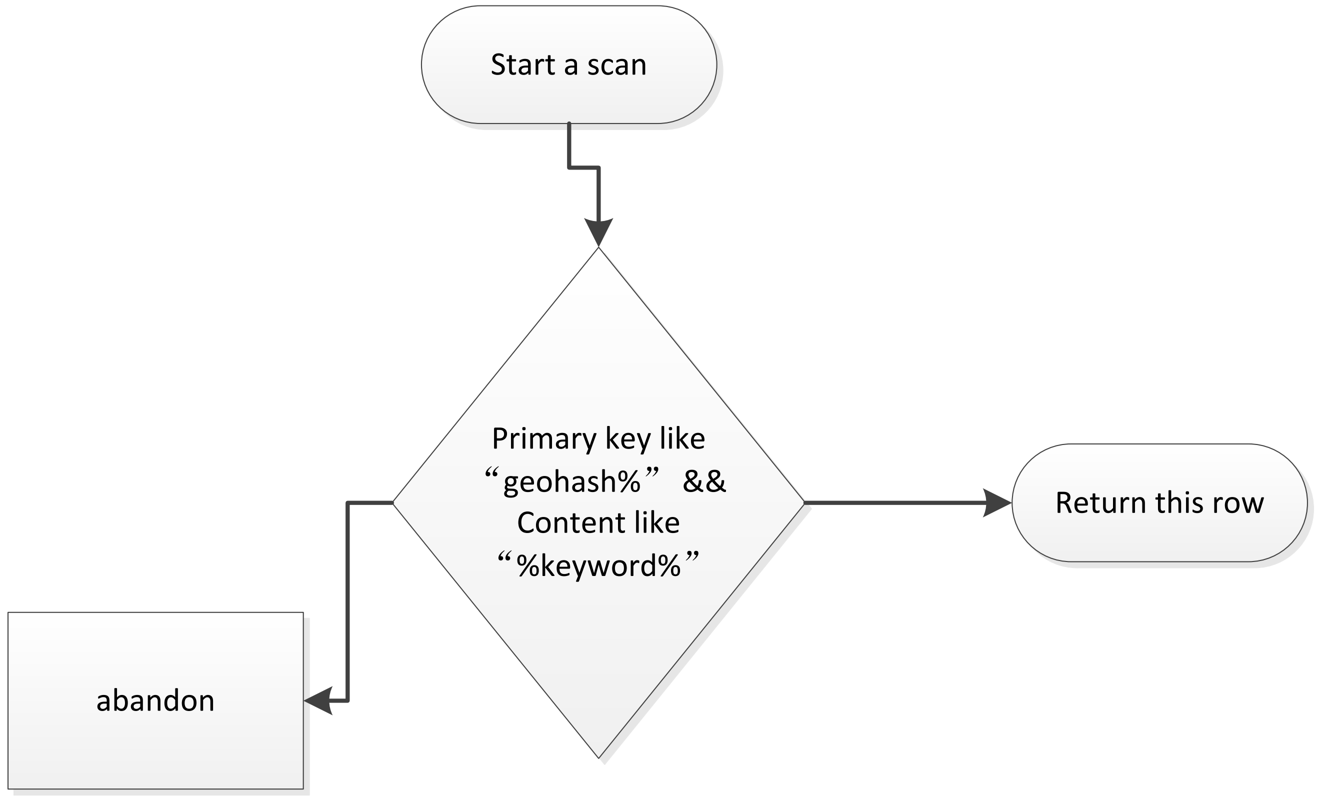
processor： 1.8Ghz processor
RAM： 4Gb
Hard Disk： 500Gb HDD
System environment： CentOS 7.0/JDK 1.8u101/Hadoop 2.6.3/ HBase 1.2.1
Number of cluster machines： 5

MySQL Server

processor： 2.1Ghz processor
RAM： 8GB
Hard Disk： 300G SSD
System environment： Windows Server 2012/MySQL 5.4/ JDK 1.8u101

HBase vs. MySQL – MySQL

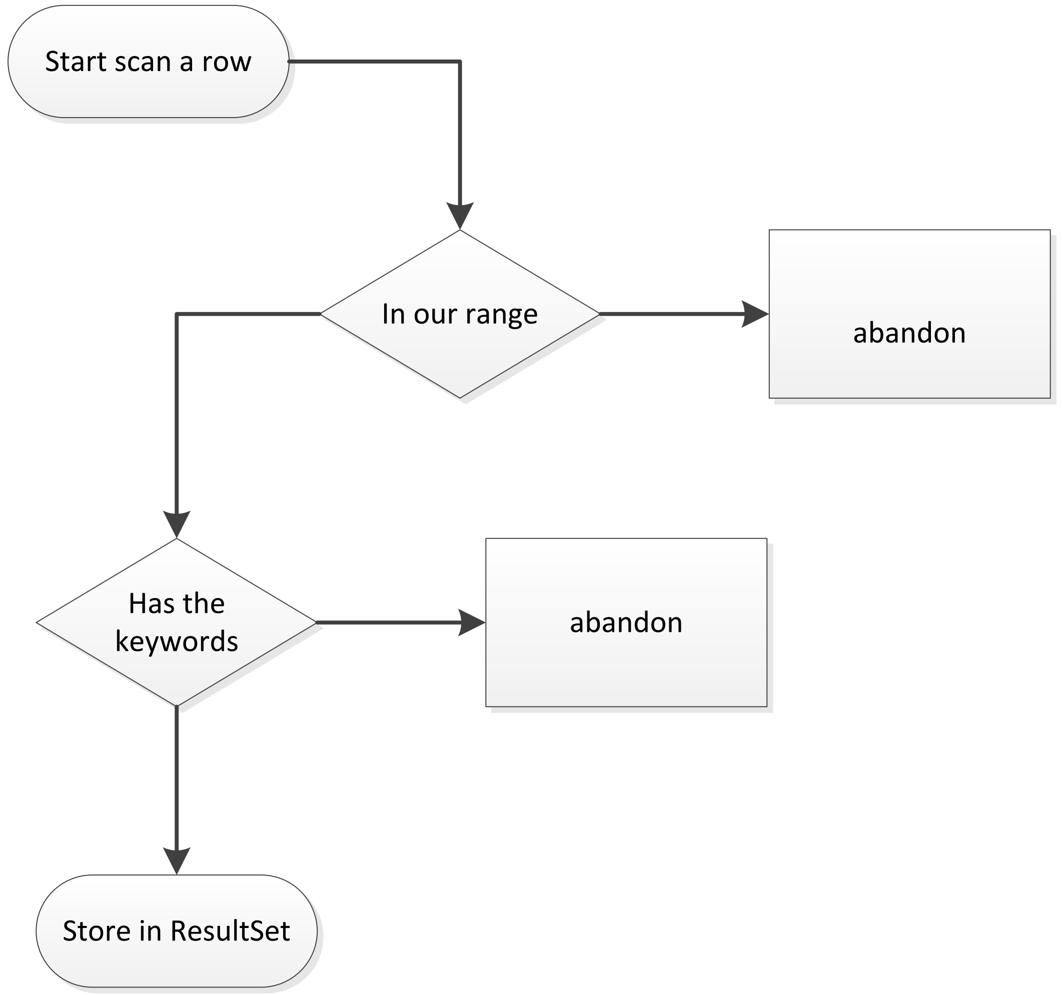
In the MySQL table scan operation, we use regular expression for the content and id to matching for geographical coordinates and keywords. Because of first match id is in the form of "xxx%“. so regular expression can take a shortcut by the index which created on the ID.



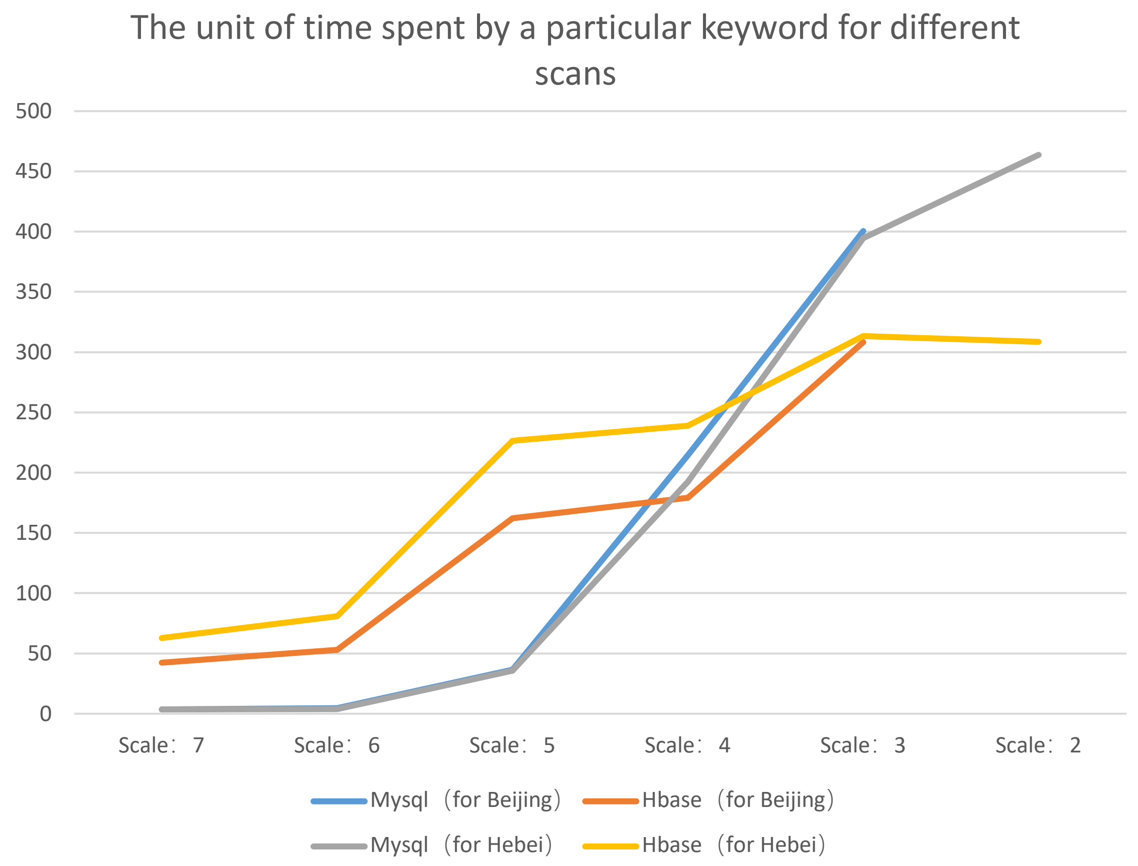
HBase vs. MySQL – HBase

We make Row Key through filtered which has right geohash, and then filtered for the results of this keyword, this process can greatly reduce the time complexity of the keyword filtering operation.

In the HBase search part, by using the Java API that HBase provides, the multiple Filter and Scanner to complete the database scan. We use the RowFilter to find the selected area and SingleColumnValueFilter to complete the confirmation of the existence of the keyword. Also set Scan Cache to optimize scan speed and IO.



HBase vs. MySQL – Performance



You can see from the left chart:

1) When the scanning range is small: MySQL's scanning efficiency is much higher than HBase.

2) But when the amount of data increased. MySQL's scan time increase much faster than HBase. (each size of search range is about four times larger than before). And HBase scan time increase more gentle, and when scale is 3-4 the scanning cost is less than MySQL, and at this time of the scanning range is about 400 square kilometers to 6000 square kilometers.

MapReduce vs. Multithreading – conclusion

In our previous experiments, based on the time of the experiment, we can see that both the efficiency and stability of the system running, MapReduce are more excellent properties, which is largely because the data MapReduce process is store in Hfile which base on a distributed system.

In the conclusion of POI placement analysis of experiments using the MR programming model can easily solve the POI check-in problem in high-speed updates.

HBase vs. MySQL – Performance

As you can see in the previous chart. at the first, MySQL's scanner for small volume data acquisition is very efficient, but by the scanning of large areas, MySQL cost too much time and the system suspended animation Happening sometimes. At the same time in the amount of data to the municipal, provincial HBase scan table in the consumption of the logarithmic level of increase, and MySQL is almost linear increase.

In write part, HBase for multi-threaded insertion provide a non-lock write method, the write speed is much faster than MySQL.

When building a data warehouse for social media data, using HBase as the underlying architecture is suitable.

中國地質大學
CHINA UNIVERSITY OF GEOSCIENCES

Thanks

肖 濛

coco11563@yeah.net
https://github.com/coco11563/BeijingDataProcess

中国地质大学（武汉） 信息工程学院