Applying Hadoop's MapReduce Framework on Clustering the GPS Signals through Cloud Computing

[1]Wichian Premchaiswadi, [2]Walisa Romsaiyud, [3]Sarayut Intarasema
Graduate School of Information Technology
Siam University
Bangkok, Thailand
[1]wichian@siam.edu, [2]walisa.rom@siam.edu, [3]darkman502@hotmail.com
[4]Nucharee Premchaiswadi
Faculty of Information Technology,
Dhurakij Pundit University,
Bangkok, Thailand
[4]nucharee@dpu.ac.th

Abstract— **Year by year, we are considerably witnessing a dramatic increase in the size of data gathered from machines or human interactions. Typically, the data generated by machines is massive, complex and comes from different varieties including sensors collecting climate information, posts being shared in social media sites, videos being posted online, digital pictures, transaction records of online purchases, cell phone GPS signals and so on. Not surprisingly, the amount of data generated by machines is greater than the data generated by human elements. Sensor data (obtained from transportation, logistics, retail, utilities, and telecommunications) has continuously been generated from fleet GPS trans-receivers, RFID tag readers; smart meters, to cell phones. Such data has frequently been used in numerous parallel processing methods so as to optimize operations and drive operational business intelligence (BI) systems scrutinizing immediate business opportunities. Appropriately, MapReduce is a programming model designed for expressing distributed computations on massive datasets and an execution framework for large-scale data processing on clusters of commodity servers. In this paper, we enhanced the Hadoop MapReduce for data-intensive computing on massive datasets of GPS signals. We developed an execution framework for large-scale data processing through the cloud system —in order to reduce the execution time of the cluster systems— as well.**

Keywords- Mobile Location-Based Services; GPS Signals; Cloud Computing; Hadoop Distributed File Systems (HDFS); MapReduce.

I. INTRODUCTION

Today's mobile devices operate independently of each other, using only local computing, sensing, networking, and storage capabilities and functions provided by remote Internet services. It is generally difficult or expensive for a mobile device to share data and computing resources with another. Data is shared through centralized services, requiring expensive uploads and downloads that strain wireless data networks. Collaborative computing is only achieved using ad hoc approaches. Coordinating mobile device data and computing would allow mobile applications to utilize the capabilities of an entire mobile device cloud while avoiding global network bottlenecks. In many cases, processing mobile data in-place and transferring it directly between mobile devices, would be more efficient and less susceptible to network limitations than offloading data and processing to remote servers.

We have extended the Hadoop MapReduce platform derived from Hadoop that supports cloud computing on Android and windows phone mobile devices. Our proposal allows client applications to conveniently utilize data and execute computing jobs on networks of mobile devices and heterogeneous networks of phones and servers. By scaling with the number of devices and tolerating node departure, we allow applications to use distributed resources abstractly, oblivious to the physical nature of the cloud. The design and implementation of the proposal includes experiences in porting Hadoop to the Android platform and the design of mobile specific customizations. The scalability of our proposed method is evaluated experimentally and compared to that of Hadoop. Although the performance of our proposed method is poor for CPU-bound tasks, it is shown to significantly tolerate node-departure offering reasonable performance in data sharing. A distributed multimedia search and sharing application is implemented to qualitatively evaluate the proposed from an application development perspective.

Therefore, the rationale behind the paper is based on the challenge of using mobile hardware through cloud computing. We all know that cloud computing offers various advantages over using traditional hardware such as computational access to multimedia (and sensor data without large network transfers), more efficient access to data stored on other mobile devices, and distributed ownership and maintenance of hardware. Therefore, the idea of opting cloud computing approach inevitably gives rise to many concerns including access-

978-1-4799-0838-7/13/$31.00 ©2013 IEEE

control, incentivisation of users, privacy, and mobile resource conservation. At the same time, the following approach may create many opportunities for interesting new applications and for more resource-efficient versions of existing applications as well.

The remainder of this paper is organized as follows. In Section 2, we review and describe related technology. Section 3 addresses the system architecture. Section 4 describes the design and implementation. And the final section provides a conclusion and describes the future work.

II. LITERATURE REVIEW

At present, there are many terms and techniques concerning the GPS database management based on the Hadoop's MapReduce as the following:

A. Cloud Computing

Cloud Computing is a new paradigm [1-2] that uses dynamically scalable shared resources over the scalable network of nodes. Data centers, web services, and low-end devices can be examples of such nodes. The network of such nodes can be termed as the Cloud and several networks of such nodes can be called the Internet-of-Clouds. The Internet of Clouds involves four major participants which are Cloud providers, brokers, resellers, and end consumers.

We can discuss the entities of Cloud as following [2];

- Cloud Providers: The providers host and manage the underlying infrastructure and offer different services (e.g., SaaS, PaaS, IaaS, and etc.) to the consumers, the service brokers or resellers. Cloud brokers and resellers plays the same role as "Cloud providers".
- Cloud Service Brokers: Service brokers concentrate on the negotiation of the relationships between consumers and providers without owning or managing the whole Cloud infrastructure. Moreover, they add extra services on top of a Cloud provider's infrastructure to make up the user's Cloud environment.
- Cloud Resellers: Resellers can become an important factor of the Cloud market when the Cloud providers will expand their business across continents. Cloud providers may choose local IT consultancy firms or resellers of their existing products to act as "resellers" for their Cloud-based products in a particular region.
- Cloud Consumers: [1] End users belong to the category of Cloud consumers. However, also Cloud service brokers and resellers can belong to this category as soon as they are customers of another Cloud provider, broker or reseller.

B. National Marine Electronics Association (NMEA)

The National Marine Electronics Association (NMEA)[3-4] has developed a specification that defines the interface between various pieces of marine electronic equipment. The standard permits marine electronics to send information to computers and to other marine equipment. Most GPS receivers understand the standard which is called: 0183 version 2. This standard dictates a transfer rate of 4800 b/s. Some receivers also understand older standards. The oldest standard was 0180 followed by 0182 which transferred data at 1200 b/s. An earlier version of 0183 called version 1.5 is also understood by some receivers. Some Garmin units and other brands can be set to 9600 for NMEA output or even higher but this is only recommended if we have determined that 4800 works ok and then we can try to set it faster. Setting it to run as fast as and may improve the responsiveness of the program. Some units also support an NMEA input mode. While not too many programs support this mode it does provide a standardized way to update or add waypoint and route data. Note that there is no handshaking or commands in NMEA mode so just send the data in the correct sentence and the unit will accept the data and add or overwrite the information in memory. If the data is not in the correct format it will simply be ignored. A carriage return/line feed sequence is required. If the waypoint name is the same will overwrite existing data but no warning will be issued. The sentence construction is identical to what the unit downloads, so we can, for example, capture a WPL sentence from one unit and then send the same sentence to another unit. However, we must be cautious whether two units support waypoint names of different lengths or not (since the receiving unit might truncate the name and overwrite a waypoint accidently). Normally, many units support the input of WPL sentences and a few supports RTE.

Accordingly, there are many sentences in the NMEA [18, 22] standard for all kinds of devices that may be used in a Marine environment. Some of the ones that have applicability to GPS receivers are listed below: (all message start with GP.)

GGA - Fix information
GLL - Lat/Lon data
GRS - GPS Range Residuals
GSA - Overall Satellite data
GST - GPS Pseudo range Noise Statistics
GSV - Detailed Satellite data
WPL - Waypoint Location information
ZTG - Zulu (UTC) time and time to go (to destination)
ZDA - Date and Time

The NMEA 2.3 output from the Garmin Legend, Vista, and perhaps some others include the BWC, VTG, and XTE sentences.

The Trimble Scoutmaster outputs: APA, APB, BWC, GGA, GLL, GSA, GSV, RMB, RMC, VTG, WCV, XTE, ZTG.

The Motorola Encore outputs: GGA, GLL, GSV, RMC, VTG, ZDA and a proprietary sentence PMOTG.

While a user is moving, NMEA collects the GPS signals based on the couple of the elements such as latitude, longitude, date, time and speed.

$GPGGA,123519,4807.038,N,01131.034,E,1,08,0.9,545.4, M,46.9,M,,*47
$GPGGA,123530,4807.038,N,01131.045,E,1,08,0.9,545.4, M,46.9,M,,*47
$GPGGA,123545,4807.038,N,01131.050,E,1,08,0.9,545.4, M,46.9,M,,*47

$GPGGA,123552,4807.038,N,01131.067,E,1,08,0.9,545.4,M,46.9,M,,*47

$GPGGA,123601,4807.038,N,01132.000,E,1,08,0.9,545.4,M,46.9,M,,*47

.....

Where:

GGA Global Positioning System Fix Data

123519 Fix taken at 12:35:19 UTC

4807.038,N Latitude 48 deg 07.038' N

01131.000,E Longitude 11 deg 31.000' E

1 Fix quality: 0 = invalid

1 = GPS fix (SPS)

2 = DGPS fix

3 = PPS fix

4 = Real Time Kinematic

5 = Float RTK

6 = estimated (dead reckoning) (2.3 feature)

7 = Manual input mode

8 = Simulation mode

08 Number of satellites being tracked

0.9 Horizontal dilution of position

545.4,M Altitude, Meters, above mean sea level

46.9,M Height of geoid (mean sea level) above WGS84 ellipsoid

(empty field) time in seconds since last DGPS update

(empty field) DGPS station ID number

*47 the checksum data, always begins with *

Figure 1. The Example Decode of selected position sentences from GPS signals on NMEA format [3].

As Fig. 1 illustrates, after decoding the navigation sentences, our system gets the information from current GPS signals through the Cloud and writes them into a text file compatible with the standard format of NMEA.

C. National Marine Electronics Association (NMEA)

MapReduce programs [5-12] are executed in two main phases, called mapping and reducing. Each phase is defined by a data processing function, and these functions are called mapper and reducer, respectively. In the mapping phase, MapReduce takes the input data and feeds each data element to the mapper. In the reducing phase, the reducer processes all the outputs from the mapper and arrives at a final result as show on Fig. 2.

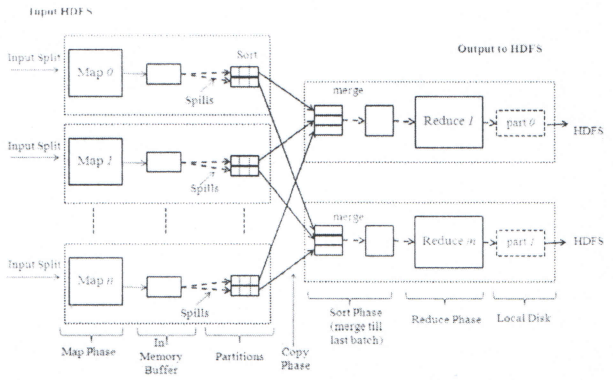

Figure 2. MapReduce Architecture [7-8].

As Fig. 2 illustrates, the MapReduce Framework consists of two mainly folds and subcategories [8-14] as the following: 1) Map Processing: HDFS splits the large input data set into smaller data blocks (64 MB by default [15-16]) controlled by the property dfs.block.size. 2) Spill: When the buffer size reaches a threshold size controlled by io.sort.spill.percent (default 0.80, or 80%), a background thread starts to spill the contents to disk. 3) Partitioning: Before writing to the disk the background thread divides the data into partitions. 4) Sorting: In-memory sort is performed on key (based on compareTo method of key class). 5) Merging: [17] Before the map task is finished, the spill files are merged into a single partitioned and sorted output file. 6) Compression [18-19, 21]: The map output can be compressed before writing to the disk for faster disk writing, lesser disk space, and to reduce the amount of data to transfer to the reducer. 7) Re-duce Operations [22]: The reducer has three phases as follows: Copy, Sort and Reduce.

III. ARCHITECTURAL OVERVIEW

In this section, we present the cloud computing on mobile devices using Hadoop MapReduce. Our method uses the following major steps: (a) The GPS data are stored in the cloud. The Hadoop Map/Reduce system accesses the GPS data through the 'namenode'. (b) The MapReduce framework and the Hadoop Distributed File System (HDFS) are used to process vast amounts of GPS data (multi-terabyte data-sets) in-parallel on large clusters. (c) A Map/Reduce job splits the input GPS data-set into independent chunks which are processed by the map tasks in a completely parallel manner. It sorts the outputs of the maps which are then input to the reduce

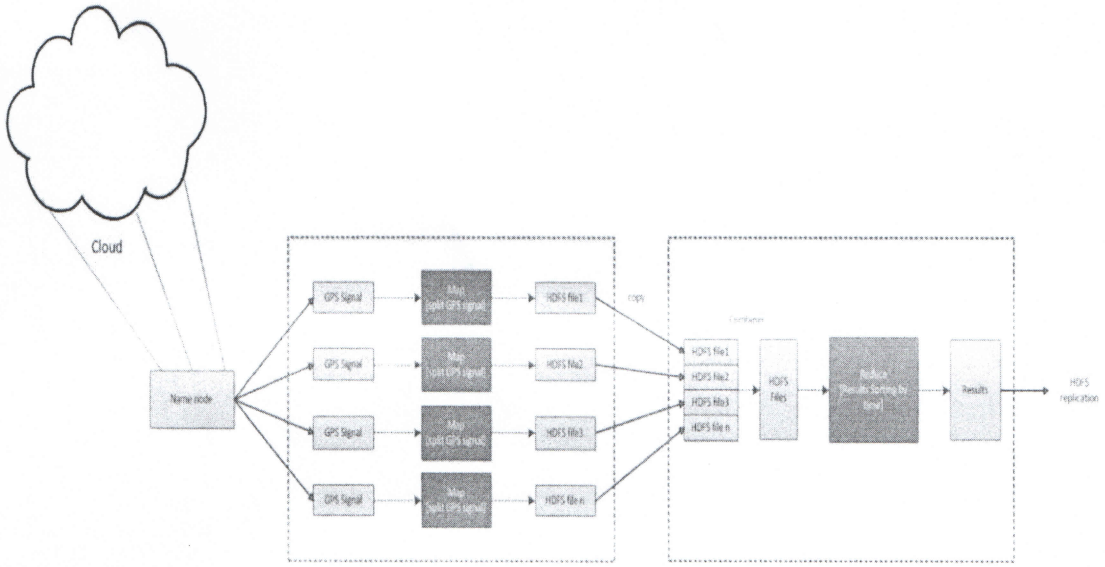

Figure 3. The architectural overview of our proposed application.

tasks. Both the input and the output of the job are stored in a file-system (HDFS files) presented in Fig. 3.

As Fig. 3 illustrates, we found out that in general retrieving data from the cloud takes an hour despite the fact that storing data in a database consumes more than 4GB of memory. Without a doubt, queries take long time to process and return the results. On the other hand, the Map/Reduce technique significantly reduces the overall execution time of queries.

As Fig. 4 illustrates, we have ported the DataNode and TaskTracker processes to work on Android. A Hadoop cluster is configured to run on both Android/Windows Phone devices. Running Hadoop on a cluster of phones is analogous to running Hadoop on a cluster of servers. In both cases, there is one instance of the NameNode and one instance of the JobTracker. These often run on the same machine. The slave machines in the cluster each run DataNode and TaskTracker instances. In this paper, the DataNode and the TaskTracker are run on each phone in separate Android/Windows Phone "service" processes within the same application. Android/Windows Phone applications may consist of multiple processes, some of which run as background services. Since the DataNode and TaskTracker are run as Android/Windows Phone services, they can run in the background of other applications. The performance of these phones is compared to the performance of a cluster of 10 AMD Opteron 1220 machines, each with 4 GB RAM, two Seagate Barracuda 7200.10 320 GB disks, and a Broadcom NetXtreme BCM5721 Gigabit Ethernet controller. Each node runs Debian GNU/Linux 4.0 (etch) with Linux kernel 2.6.18.

Applications							
HDFS and MapReduce Interfaces							
NameNode	Jobtracker		NameNode	Jobtracker		NameNode	Jobtracker
Traditional Server			Android Devices1..n			Windows Phone Devices1..n	



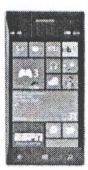

Figure 4. The Hadoop on a Mobile Cluster.

IV. IMPLEMENTATION AND TESTING

The first step towards porting Hadoop to run on Android/Windows Phone was to compile Hadoop's source code into both Android/Windows Phone applications. We planned to create Android/Windows Phone applications that can act as a slave in a Hadoop network (running DataNode and TaskTracker instances). Instantiating the DataNode and TaskTracker was just a matter of including Hadoop's source in the Java build path in Android/Windows Phone project. We started with Hadoop 0.20.0. Hadoop allocates memory buffers that are on the order of 10 to 100 MB. This is too much for Android and Windows Phones application whose heap can grow to a maximum of 16MB. To fix the restriction, the buffer sizes were reduced to about 1 MB. This caused excessive swapping to occur. This swapping was reduced by adjusting the *io.sort.record.percent* parameter. The default values for some timeouts in Hadoop are not long enough for a mobile device network. For instance, the value of the *dfs.socket.timeout* had to be increased to compensate for connection issues. Hadoop uses XML for its configuration files even though the same key-value configuration could be stored in a simpler format, such as a properties file as Pseudocode1.

Pseudocode 1: Distributed Mode on XML file

```
<?xml version="1.0"?>
<!-- core-site.xml -->
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost/</value>
</property>
</configuration>
<!-- hdfs-site.xml -->
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
</configuration>
<!-- mapred-site.xml -->
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:8021</value>
</property>
```

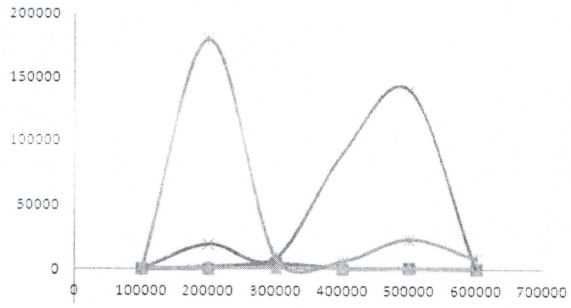

Figure 5. Network receive.

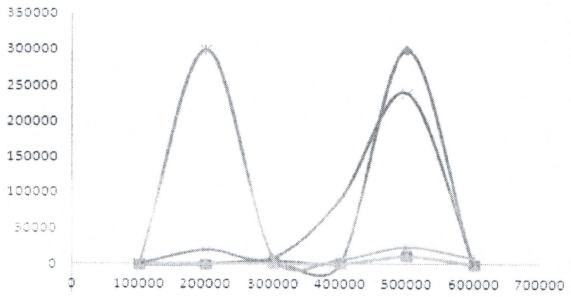

Figure 6. Network send.

Fig. 5-6 illustrates the system resource usage data such as Network based on the time and data file size —to sort benchmark on 3 of 10 phones.

V. CONCLUSIONS

In this paper, we have combined the two concepts of cloud computing and mobile networks with the intention of bringing benefits for mobile users, network operators, as well as cloud providers. Cloud computing has many advantages over traditional computing models. Our proposed approach in this study provided an abstract interface for using data and executing computing jobs on a mobile device cloud. As a result, implementing the distributed multimedia search and sharing application, we could provide a more convenient and sufficiently abstract interface for developing applications that use mobile data.

REFERENCES

[1] I. Paul Zikopoulos and Chris Eaton, "Understanding Big Data", McGraw Hill Professional, Oct 19, 2554.

[2] T. White: Hadoop, "The Definitive Guide (1st edn.)", O'Reilly Media, Inc., United States of America, 2009.

[3] http://hadoop.apache.org/

[4] B. Furht, "Handbook of Cloud Computing", Springer, 2010.

[5] M. Miller, "Cloud Computing: Web-Based Application that change the way you work and collaborate online", Que Publishing, 2010.

[6] W. Premchaiswadi and W. Romsaiyud, "Optimizing and Tuning MapReduce Jobs to Improve the Large-Scale Data Analysis Process", International Journal of Intelligent Systems, Vol. 28, Issue 2, pp. 185-200, 2013.

[7] Hadoop MapReduce Change Log. Release0.22.1 –Unreleased. < http://hadoop.apache.org/mapreduce/docs/r0.22.0/changes.html>, Accepted 02012012.

[8] W. Premchaiswadi and W. Romsaiyud, "Extracting Weblog of Siam University for Learning User Behavior on MapReduce", Int'l Conf of Intelligent and Advanced System (ICIAS), pp. 149-154, 2012.

[9] D. Meng, Jizhong Han, Jianfeng Zhan, Bibo Tu, Xiaofeng Shi and Le Wan, "Transformer: A New Paradigm for Building Data-Parallel Programming Models", Vol. 30, Issue 4, pp. 55-64, 2010.

[10] Y. Hung-chih, A. Dasdan, H. Ruey-Lung and D. Stott Parker, "Map-Reduce-Merge: simplified relational data processing on large clusters", Proc. ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD'07), pp. 1029-1040, 2007.

[11] R. McCreadie, C. Macdonald and I. Ounis, "MapReduce indexing strategies: Studying Scalability and Efficiency", Jour. Information Processing and Management, pp. 1-16, Elsevier, 2010.

[12] Z. Li-na, J. Liu and Y. Zhang, "Data Mixed-extraction Strategy based on the Time Characteristics in CDW", First International Conference on Pervasive Computing, Signal Processing and Applications, pp. 1129-1131, 2010.

[13] P. I. Hofgesang and W. Kowalczyk, "Analyzing Clickstream Data: From Anomaly Detection to Visitor Profiling", ECML/PKDD Discovery Challenge 2005, 2005.

[14] A. Banerjee and J. Ghosh, "Clickstream clustering using Weighted Longest Common Subsequences", Int'l Conf of the Web Mining Workshop at the 1st SIAM Conference on Data Mining, Chicago, 2001.

[15] D. Jiang, A. K.H. Tung and G. Chen, "MAP-JOIN-REDUCE: Toward Scalable and Efficient Data Analysis on Large Clusters", IEEE Trans. Knowledge and Data Engineering, Vol. 23, No. 9, pp. 1299-1311, September 2011.

[16] L. Ma, H. Liao, Y. He, F. Li and Q. Gao, "A Switch Criterion for Hybrid Datasets Merging on Top of Map Reduce", Proc. Eighth IEEE Int'l Conf. Grid and Cooperative Computing (GCC'09), pp. 293-298, 2009.

[17] http://www.nmea.org/

[18] http://msdn.microsoft.com/en-us/library/aa920475.aspx

[19] https://www.42tele.com/Global_Network_Coverage/SMS/Thailand/

[20] Microsoft Corporationm, "Radio Interface Layer (RIL) White paper", June 2004.

[21] http://www.netmite.com/android/mydroid/development/pdk/docs/telephony.html