Research Article

A Tracking System Supporting Large-Scale Users Based on GPS and G-Sensor

Jun Zeng, 1 Minbo Li, 1, 2 and Yuanfeng Cai 1

1 Software School, Fudan University, Shanghai 201203, China
2 Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China

Correspondence should be addressed to Minbo Li; limb@fudan.edu.cn

Received 2 December 2014; Revised 13 March 2015; Accepted 25 March 2015

As the need for tracking objects in our daily life is increasing, how to take advantage of widely used technologies and devices to meet everyone’s need is becoming a burning issue. To achieve this aim, a tracking system supporting large-scale users based on GPS and G-sensors (TSSLSU) is designed to track items by obtaining various pieces of information of the item’s states and analyzing various conditions. In this system, a microprocessor with Global Positioning System (GPS) modules embedded in targets sends data in real-time through General Packet Radio Service (GPRS) and Global System for Mobile Communication (GSM), and an inner G-sensor can fetch real-time state information, such as acceleration and vibration signals. The system designed a message protocol to encapsulate and send these data to servers and also adopted distributed architecture, including backend server cluster, cache queue cluster, database cluster, and web server cluster. This specific architecture provides a better performance when large-scale users visit concurrently. In particular, there are alarm mechanisms to inform the user of any illegal cases through smart phone application and power-saving mechanism for embedded terminal by changing the frequency of sending messages. At any time, user can locate it and query the carrier’s previous path by web page or phone application.

1. Introduction

As the Chinese economy grows rapidly, the number of vehicles, including but not limited to cars and electric bicycles, rises greatly. Sometimes, locating these vehicles became an important function for peripheral product of vehicle [1]. Meanwhile, issues about improper care for the elderly and loss of children have been exposed frequently these years [2]. As a result, people are getting more concerned about the security of the kids, the old men, vehicles, and pets, which creates huge market prospects for tracking technology.

Till now, various tracking systems have been developed, while the result is far from satisfaction. Some GSM-based tracking devices send location information through text message to the phone of the binding user, while such data is not direct enough for users to understand [3]. With the prevailing smartphones, much more can be done to facilitate tracking objects. IDC (Internet Data Center) released its mobile phone forecast recently, predicting that total sales of the smart phones will reach 1.2 billion units before the end of the year, a 23.1 percent increase over 2013 [4].

Based on this premise, we fully take advantage of ubiquitous smart phones and sensor network, to design a sensor-based tracking system, which can support large-scale users and big data storage, with the alarm mechanism. In our system, terminals send location information, which is gathered through GPS module [5], to the server cluster periodically. For the purpose of saving energy, the time gap between two messages can adjust to the condition of carrier automatically, by preset parameters. As to the speed of the carrier, if the speed of the carrier increases to some particular point, the frequency decreases and vice versa. Besides, G-sensor and some necessary parameters are used to monitor the state of the carrier to hit the mark of illegal cases and alarm conditions.

For this system, the terminal may send one message to server cluster every fifteen seconds in normal condition with the aid of GPS sensor; for large-scale users, like 100,000 users concurrently, the data received by server cluster in one single day is big. This system settles this problem as well. To meet the big data challenge, cache queue is adopted to cache...
the data before inserting into database, in which database sharding technology is utilized to distribute the data into various tables according to time. In addition, G-sensor is used to monitor the condition of the carrier, as whether the target has gone out the electronic fence or the target has some unexpected movements. Taking advantage of the GPS and G-sensor, this system can be used to monitor the kids or the elder, as well as being used to locate and track vehicles when stolen. Moreover, web page and smart phone application can be used to locate the carrier and look over its historic path.

Compared to other existing systems, we develop a mechanism to change the frequency of sending message and thus we are able to save energy which is a key factor in sensor reliability, as well as decreasing the network traffic. Besides, innovative application level protocol is designed to ensure the reliability of network transmission.

In this paper, for convenience, vehicle is the experimental carrier, and in the following sections, vehicle is used as an example to discuss the design and implementation of the system.

The rest of this paper is organized as follows. Section 2 presents related literature reviews on current tracking system and technology. Section 3 describes the designation and implementation details of tracking system based GPS and G-sensor. Section 4 illustrates the method for large-scale users’ access, big data process, storage, and query. In Section 5, result is presented to show that how the system is implemented and the performance for high concurrent pressure. Finally, Section 6 accounts for conclusion and future work.

2. Previous Systems

Recently, with the rapid increase in demand of security, various tracking technologies have been developed. Considerable research works focused on locating and tracking. Ganesh et al. [6] developed an antitheft tracking system using only GSM technology, which would be the cheapest source of tracking antitheft system. The terminal in vehicle can send location of the nearest base station to owner’s mobile phone via text message. This approach is cost-effective; however, the locating accuracy is disappointing and the latitude and longitude information is not straight enough for users to understand. Liu et al. [7] designed a tracking system based on Internet of Things, which is controlled by an RFID module. When the vehicle is stolen inside where the vibration sensor mounted will be triggered, and GSM module will send the location information to the owner's mobile phone. Compared to the previous systems, this is a great improvement, but it is still not enough. Lee et al. [8] proposed a simple vehicle tracking system based on smartphone using GPS, GSM, and GPRS technology. The system provides vehicles a microcontroller to monitor their geographic locations and speed. Then the owner can know his vehicle's location in real-time using Google maps and determine the distance and time for the vehicle to arrive at a given destination. But the terminal in this system costs a lot, which only provides some simple functions.

There were also some other similar vehicle systems proposed with antitheft or tracking functions [9], but most of the monitored objects were cars with high moving speed. For ordinary vehicles, however, the cost of such tracking system is unfortunately high. This might prevent customers from purchasing these services and terminals since their high price is one of the main selling disadvantages.

In this paper, a more reliable tracking system supporting large-scale users is proposed. The ubiquitous tracking system provides the most commonly used functions for vehicles. Compared with systems proposed by Lee et al. and Liu et al., the TSSLSU can not only monitor the vehicle’s real-time location and obtain the information of driving route, but also interact with the vehicle by changing the parameters of the embedded terminal and protect it via sending warning messages. Furthermore, the interactive platforms of the system are smartphone and web platform, both of which are easily achievable for vehicle owners.

There is another difference between TSSLSU and other systems. Baidu map is used in this system, since it is the most widely used and more accurate map in China, whereas in most other countries, Google map is more popular. Before displaying path information in map, this coordinate information obtained from GPS satellites has to be treated by a serial of special processes rather than being used directly.

This paper is to present the architecture of tracking system supporting great concurrency based on the sensors of GPS and G-sensor. With regard to the embedded terminals’ access and applications’ access, concurrent access and contagious distribution have been considered and utilized. For frequent data uploading and writing, database cluster and data cache queues are also employed.

3. System Design and Architecture Details

The prime purpose of TSSLSU is to locate vehicle and monitor it by tracking its geographic locations. The tracking terminal embedded in vehicle is able to forward information, including speed, defensive state, and geographical coordinates to the backend server clusters, which then process these data and monitor the vehicle's unusual situations.

The tracking terminal acquires geographic coordinates from GPS satellites along with some other information. It then sends these data via various messages to the backend server cluster, such as locations, states, and speed and some other information. At set intervals, a heartbeat message is also sent to the backend server cluster to keep in touch with servers.

Once the backend server cluster receives a normal message, it writes these data into queues, whose SQL data wait in line to be written into the database of job nodes. After a specific time interval, the database of job nodes synchronizes with the database of backup.

User can log into the web or smartphone application using the terminal’s unique ID and password, monitor the previous and real-time travel paths in Baidu map, and send instructions to terminal. User will receive real-time warning messages of various pieces of information, such as over speed, suspected stolen, off-side, or moved, via the smartphone application. Furthermore, instructions can also be sent to web servers to change parameters of embedded terminal.
The functional architecture of TSSLSU is showed in Figure 1. The details of the design and implementation of TSSLSU are elaborated in the following four parts.

3.1. Embedded Terminal. The embedded terminal is composed of three modules, which is GPRS/GSM module, GPS module, and gravity sensor (G-sensor) module, as shown in Figure 2.

The main chip based on the 32-bit ARM73J-STM RISC processor is MT6260A [10] produced by MediaTek in Taiwan. MT6260A is the microprogrammed control unit of the embedded terminal and integrated single-ended SAW-less RF transceiver for multiband GSM cellular system. Also it supports GPRS modem.

Another chip named MT3336 [11] on the mainboard functions as GPS. This chip contains a CMOS RF down conversion circuitry, combined with digital baseband signal processing. This combination delivers excellent receiving performance with extreme sensitivity and a very low time-to-first-fix.

The third module, BMA222E (F), perceives the changes of acceleration and vibration. BMA222E is a triaxial, low-g acceleration sensor with digital output for consumer applications, allowing measurements of acceleration in three perpendicular axes. This module offers a variable VDDIO voltage range from 1.2 V to 3.6 V and can be programmed to optimize functionality, performance, and power consumption in customer specific applications.

For the final application of the terminal device, a SIM card needs to be inserted into the embedded terminal. GPS antenna receives GPS signals and processes the data in GPS module. The MCU schedules the data and uploads the encapsulated messages to background servers by GSM antenna. The structure is showed in Figure 3.

There are about 300mAh power in the fully charged backup battery. Under normal conditions, the embedded terminal connects to other devices and gets charged by the host device, and if the embedded terminal disconnects with the host device, it gets power directly from the backup battery and changes into sleep mode with low energy consumption. In Figure 4, it shows a completed embedded terminal, whose size is 84 mm as length, 52.6 mm as width, and 15.4 mm as height.
Table 1: Some detailed messages format.

| Action         | Type number | Function                                           | Terminal message format                                                                 | Server message format |
|----------------|-------------|----------------------------------------------------|-----------------------------------------------------------------------------------------|-----------------------|
| Heartbeat      | 2           | Message is sent from terminal to server, to keep the links connected. | [Time, T + num, ISMI, longitude, latitude, other, status]                                | [Time, S + num,]      |
| State          | 3           | Report the location information each period of time. | [Time, S + num, ISMI, longitude, latitude, other, status]                                | [Time, S + num,]      |
| Locate         | 4           | Server sends it to terminal to ask for location information. | [Time, S + num, ISMI, longitude, latitude, other, status]                                | [Time, S + num,]      |
| Parameter set  | 6           | Change the parameter for terminal in long distance. | [Time, T + num, ISMI, longitude, latitude, other, status]                                | [Time, S + num,]      |
| Setup defense  | 8           | Set up defense for terminal to guard against theft. | [Time, S + num, ISMI, longitude, latitude, other, status]                                | [Time, S + num,]      |
| Illegal shake  | 11          | Terminal reports the alarm of illegal shaking after the vehicle is illegally moving. | [Time, T + num, ISMI, longitude, latitude, other, status]                                | [Time, S + num,]      |
| Cross border   | 14          | Terminal reports alert after moving 300 m, when it sets up defenses. | [Time, S + num, ISMI, longitude, latitude, other, status]                                | [Time, S + num,]      |

3.2. Message Protocol. The formats of the most used messages transferred between terminal and servers are shown in (1), (2), and (3):

\[
\begin{align*}
\text{[Time, Type} & + \text{ num, ISMI, longitude, latitude, other, status],} \\
\text{[Time, Type} & + \text{ num, ISMI, mold, mark],} \\
\text{[Time, Type} & + \text{ num, ISMI, mark, status],}
\end{align*}
\]

where “[” and “]” are the start and end of a message, respectively, and “,” is the separator. The format of “Time” is “YYYY-MM-DD HH:MM:SS,” “Type” includes “T” and “S,” where “T” means the message sent by terminals and “S” by servers. “num” shows the different classes of message type and ISMI is the unique serial number of terminals.

To realize the tracking system and reduce the message redundancy, there are over 20 kinds of functionality message types. For different functions, there are some differences for protocol format in detail. The protocol format may contain different parameters according to the defined class. Some parameters may be null, but the “,” has to be retained in a complete and valid message.

In (1), (2), and (3), it shows the comprehensive types of message protocol. For (1), it is the protocol type for collection messages, like heartbeat, state, locate, parameter query, address, and query. For (2), it is the format for fortification messages, such as set parameter, restart terminal, set up defense, and rescind defense. As for (3), it is the message format for warn message, like illegal shake, power off, cross the border, over speed, and low battery. Some parameters may have completely different meanings in each type of format, such as in state message, the other information means real-time speed, while in illegal shake message, the other information means cell-id information. These messages encoded in ASCII are sent with TCP (Transmission Control Protocol) as the transport protocol and GPRS (General Packet Radio Service) network for the data transmission in the network layer. Some detailed information of the format is shown in Table 1.

In TSSLSU, the size of a message encapsulated into a data package should not be larger than 1024 bytes, so that the package can be efficiently transferred by GPRS. In particular, the transfer protocol can guarantee the reliability for every transferred message. For a package transferred from terminal to servers, the time is when the terminal starts to send a message. Then the servers have to reply to a message to inform that the servers have indeed received the message. And the time for the reply message is the same as the previous sending time, so that the terminal could remove the sent message by matching the time. For example, the terminal sends a state message to servers with [2014-12-15 10:00:00, T2, 4600113250310,,,.]. If the servers receive it, they will reply to a message with [2014-12-15 10:00:00, S2,,,.]. After the terminal receives the reply message, the incipient message of type T2 will be removed. However, if the reply message is not received after a long time, such as 1 minute, the terminal will send the same message again. This process, as illustrated in Figure 5, ensures the integrity of data.

For this system, the message protocol is specially designed based on real scenarios. In comparison with previous systems, such as Lee et al. and Liu et al.’s tracking systems, the protocol mechanism can guarantee the arrival for every message in the highest degree. When a terminal sends a message to servers, while the servers stop, the message will be sent several times to try to get to the servers. By contrast, the message may be missed in other tracking systems.
3.3. Path Display. After logging in with username and password in web or smartphone application, the user can query his vehicle’s daily path in Baidu map with HTTP (Hyper Text Transport Protocol) request. The Baidu map API servered by Baidu Map servers can automatically display map and respond to user’s gestures like drag and click.

However, before displaying the final Baidu map, preparation jobs need to be done. At the beginning of the system, GPS geographic coordinates will be received. However, because of China’s law, final map cannot be obtained based on these geographic coordinates. So for a request sent from user, there are several steps to be processed to prepare for the final data of map coordinates.

1. Acquire the accuracy of the absolute coordinates.
2. Get region information based on the absolute coordinates.
3. Query from the offset database.
4. If the offset is in the database, the final coordinate is the summation of latitude, longitude, and the corresponding offset.
5. If the offset is not in the database, obtain the final coordinates after converting coordinates through the interfaces from China’s State Bureau of Surveying and Mapping (CSBSM). Acquire offset information based on the final coordinates and store it into offset database.

Each original coordinate obtained from GPS satellites has to go through all the steps above. With the latter part in step 5, query efficiency will be promoted for next same region coordinates and the offset database will be more complete, as well. In Figure 6, it shows the flow for coordinates conversion.

The user sends a request to web server, which in return acquires a set of processed coordinates based on the user’s and their vehicle’s unique information. Then, in the smartphone or web application, the tracking route and some other information will be displayed in Baidu map.

3.4. Alarm Mechanism. Because some vehicles can be stolen or resold easily, in TSSLSU, special functions, such as illegal shake, cross border, setup defense, are designed to achieve the aim of alarm mechanism and antitheft.

First, the system provides several categories of vehicles’ statuses, such as whether the vehicle is parking or whether the vehicle sets up defense. If the vehicle’s status shows that it is parking and in defense mode, but the G-sensor senses that the vehicle is moving, a warning message will be sent to the owner’s smartphone immediately.

Another monitor function, geo-fence, is to limit the vehicle’s activity to a defined range. In other words, the owner can set a virtual geographic boundary for their vehicle. Once the vehicle crosses this boundary, a warning message will be sent to the owner’s smartphone as well. In this way, owner can interact with terminal and choose border with owner’s actual demand.

Finally, if the vehicles were stolen, the embedded terminal with standby battery would exchange data with servers and application. These data can be used to track and locate the stolen vehicle’s position.

4. Large-Scale Data Storage and Query

The concept of large-scale distributed storage system was taken into consideration when this system was first designed. To break the bottleneck of storage schema in traditional database, cache queue cluster is used before inserting data
into database. Also, to improve the performance of the whole system with large amount of users visiting concurrently, the read and write splitting models have been adopted. Furthermore, two cache layers are added to cache web pages and hot data separately. Sections 4.1, 4.2, 4.3, and 4.4 discuss the details of interaction when processing large-scale accesses, data storage, and user query.

4.1. Terminals Access Servers. Embedded terminals connect to the server cluster, which accepts and processes data of messages by using socket connection. After processing data, the information needs to be stored into the database. Because of the large amount of terminals and high uploading frequency, the number of concurrent accesses is very huge. To solve this problem in TSSLSU, a LVS (Linux Virtual Server) layer has been added between the embedded terminal group and the server cluster. LVS plays the role of the load balancing [12]. It receives numerous requests, which come from terminals, with a virtual IP and maps these requests to idle servers [13]. Servers analyze and process these data. In the end, processed data waits to be written into the database of job nodes that the cache queue is on. After the data has been written into database successfully, the server will send the terminal an acknowledge (ACK) message to inform that the message has been handled.

4.2. Data Process. In TSSLSU, each valid embedded terminal needs to connect to the server cluster and sends messages at least every fifteen seconds. In addition, as the user can change the frequency of sending messages, the number of messages might increase as well. Every day, there would be more than 5760 messages sent by just one terminal. Assume that 100,000 users will visit the system at the same time; then there would be more than 0.575 billion messages per day that need to be successfully written into the database. In this case, if a single, no cache traditional database is used, there would be read and write bottlenecks due to the large-scale data [14]. However, in this system, the cache queues and database server cluster are adopted.

(i) Because of the limitation of the database writing performance, cache queues play a vital part of conserving data that have not been written into database. The queues keep and process these data sequentially with the principle of First in First out (FIFO).

(ii) The database server cluster is divided into one control node and several database groups. Every database group includes one job node and one backup node. The control node keeps the metadata information of job nodes’ data tables and records [15]. The control node regularly schedules job nodes to synchronize their operation log to the corresponding backup nodes to achieve consistency of data in different database nodes. Job nodes send heartbeat including metadata to control node at regular intervals. As separating reading and writing improves performance, the job nodes are only responsible for records’ writing while backup nodes only handles records’ reading [16].

The server processes the data stored in cache queue one by one, and each data in the queue head is calculated according to (4) with the key of terminal serial number (ISMI). After data's hashValue is determined, the queue head data is removed from the cache queue:

\[ \text{hashValue} = (\text{hash (ISMI)} \mod N) , \]

where \( N \) is the number of job nodes. ISMI is terminal's serial number, which is unique.

Figure 7 shows the process how servers visit control node to identify the job node position and then visit job node to write data. The specific steps are shown as follows.

(1) Cache Queue server sends the calculated hashValue to the control node.

(2) The control node looks up the metadata table to identify the corresponding job node according to the hashValue.

(3) The control node returns the location information of the corresponding job node to the Cache Queue server, which sends the query information.

(4) Cache Queue acquires the location of job node and sends request to the job node to insert data.

By matching the calculated hashValue based on (4) in the control node, the corresponding job node that stores the data will be located.

4.3. Database Sharding. With 100,000 users, the database will have 0.575 billion data to insert per day in conservative estimate. Facing such big volume data, the database group gets bigger rapidly over time. Based on this condition, the CRUD (Create, Retrieve, Update, Delete) operations on the database are much more time-consuming. In addition, the resources of each server, such as CPU, disk, memory, and IO, are limited; thus, the server may meet great challenge in daily use when encountering growing data.

To strengthen the scalability and robustness of the system, database sharding technique is adopted in every job node of the database group to facilitate data operation according to the requirements and design of the system. In previous sections, the design of the nodes and data storage method are presented, that is, to distribute the data in different job node which is determined by the ISMI value. Based on the premise that the user only queries the information of his device in some specific day in the path representation function, job nodes are split into multiple tables measured by weeks. The backup nodes are handled similarly as the job nodes. As shown in Figure 8.

Through taking use of database sharding, every shard will maintain the most common used data. In this way, data in every node and table can be operated, managed, and backed up easily. Furthermore, higher read operation speed and concurrency can be realized.

4.4. User Access Web Servers. For 100,000 embedded terminals, there would be over 100,000 smartphone application
users or web users. The owner of a carrier can track it with a smartphone application or web page in PC. Furthermore, commands that control the embedded terminal can be sent by phone application as well. The smartphone application sends a request to the web server, and then the request would be processed by the master control node. Once received, the requests would be distributed into different web servers for load balancing purpose.

Two main functions of the control node are illustrated in Figure 9.

(i) Control node holds the metadata information of web server cluster and this information would be updated periodically.

(ii) Control node is responsible for the distribution of abundant requests from users. For large amount of concurrent queries from users, control node splits concurrent requests by load balancing.

In order to improve performance, especially reducing network traffic, a page caching server is added between load balancing server and web server cluster [17]. When a new request is issued, the page caching server will be checked first to confirm if there exists the result page for the request, instead of requesting to the web server directly. If there is no result page in page caching server, the web server will be visited.

Because the memory of servers is limited, only important pages or frequently visited pages should be stored in the memory. In addition, users may visit plenty of different pages in a short time. Some algorithms like LRU (Least Recently Used) cannot ensure that the pages in the memory are the most popular pages [18]. Here the LIRS (Low Inter-reference Recency Set) algorithm is used to filter cached pages [19]. This algorithm divides pages into two levels: the first level and the secondary level. The first level holds a list for each page. If one page have not been read or written for a while, the reference in the list of the page would be removed. Pages that have been accessed more than twice in a short time will be set into the secondary cache.

The way to access the web application database for web servers is similar to that between servers and embedded terminals. The difference lies in the page caching server for
holding hot data [20]. For the selection of hot data, a counter of the data will be stored to record the number of visited times. Every time user requests the data, the number of the data's counters will add one. Only top $n$ will be considered as hot data.

5. Implementation and Experiment

5.1. Implementation. In TSSLSU, the mainly development platform is JAVA2 Enterprise Edition (J2EE). Database MySQL 5.5 is used for database cluster. The server is Apache JMeter 2.11 combined with Tomcat 7.0 and JDK 1.7.0. Smartphone application is based on the Android 4.0.3. For embedded terminal, compiled language is C programming language.

The backend of the system consists of server cluster which deals with connection requests, cache queue cluster which caches the message to be inserted into the database, and web server cluster which responds to query requests; all of the three parts are implemented in Java. Figure 10 shows the implemented architecture of TSSLSU.

In this paper, a Google smartphone Nexus 4 is used to show the developed phone application and the web application. For either kind of application, the user needs a unique name, which is the phone number or terminal ID and a password to login in. Location and tracking path can be seen in the Baidu map. In the meanwhile, instructions and commands to change parameters of the embedded terminal could be sent by these applications. There are three parts to show how the system is implemented.

The terminal transfers encapsulated message to backend servers, received and analyzed by the servers. Then, necessary information will wait in line to be written into the database of job nodes. In Table 2, the “park” and “antitheft” columns show
the states of the vehicle about whether it is parking, setting up defense, respectively, in which 1 represents yes and 0 stands for no, and the “energy” column shows the power resource, in which 1 presents the terminal is powered by carrier and 0 stands for by backup battery. “gsm” and “gps” show the signal strength of GSM and the number of connected GPS satellites (0∼9). From Table 2, the time interval for each message of a certain terminal is about 10 seconds. If the time interval parameter is changed by the user or the speed of the vehicle changes, the time interval would be changed as well.

The owner can bind his vehicle through serial number with his mobile phone; furthermore, owner can locate the accurate position of his vehicle. Also, previous travel path can be seen as well. Parameters of the terminal can be changed and several kinds of alarm messages can be examined. Figure 11(a) shows the previous travel path and Figure 11(b) shows alarm messages from the smartphone application.

In web application, these functions are similar to the functions in smartphone application. User also needs unique name and password to log in into the website. Then the user can do same operations in phone. Figure 12(a) shows the real-time location for a user’s vehicle and Figure 12(b) shows the trip path in Baidu map.

After finalizing the development and deployment of hardware and software, this system has been used in one city of Shandong province. Through cooperating with China Unicom Corp, the owner only has to pay RMB 10 Yuan per month for service charge.

5.2. Experiment and Discussion. In order to verify the feasibility of the system facing big data challenge, pressure tests for such models including vehicle list, real-time monitoring, tracking playback, and remote command, are carried out. Tests have also been made to ensure the usability and security of the following functions, such as locating, instruction control, and path representation. We checked and evaluated the load capacity of the system in a simulated environment. Based on the statistics of throughput and responsiveness of the system under various concurrent conditions, the maximum concurrent capacity in expected data volume can be determined. To evaluate the reliability of the system, experiments are carried out to test how long the system will work normally under certain load pressure. JMeter is adopted to simulate thousands of users to make requests concurrently.

In the pressure test, all the clients and servers are in the same LAN. The clients are five PC with 4 G memory and dual-core CPU, and the server cluster is running in an IBM 7915R02 server with 8 G memory and quad-core CPU. Each client PC runs apache-JMeter-2.11 to simulate thousands of
| id  | Time       | imsi          | Longitude  | Latitude  | Speed | Park | Anti-theft | gsm | Energy | Backup  | Baidulng | Baidulat |
|-----|------------|---------------|------------|-----------|-------|------|------------|-----|--------|---------|----------|----------|
| 87728 | 2014-06-11 15:31:26 | 460011312504744 | 121.353865 | 31.221718 | 9.1   | 0    | 0          | 1   | 1      | 0       | 4.15 V–4.2 V | 121.36502500 | 31.22568900 |
| 87731 | 2014-06-11 15:31:36 | 460011312504744 | 121.353853 | 31.222052 | 11.3  | 0    | 0          | 1   | 1      | 0       | 4.15 V–4.2 V | 121.36503000 | 31.22602300 |
| 87734 | 2014-06-11 15:31:46 | 460011312504744 | 121.353902 | 31.222882 | 10.2  | 0    | 0          | 1   | 1      | 0       | 4.15 V–4.2 V | 121.36506300 | 31.22625200 |
| 87737 | 2014-06-11 15:31:56 | 460011312504744 | 121.354293 | 31.222345 | 17.5  | 0    | 0          | 0   | 1      | 0       | 4.15 V–4.2 V | 121.36456000 | 31.22630800 |
| 87739 | 2014-06-11 15:32:07 | 460011312504744 | 121.355020 | 31.223228 | 22.2  | 0    | 0          | 0   | 1      | 0       | 4.15 V–4.2 V | 121.36618900 | 31.22627800 |
| 87742 | 2014-06-11 15:32:17 | 460011312504744 | 121.355672 | 31.222250 | 18.8  | 0    | 0          | 1   | 1      | 0       | 4.15 V–4.2 V | 121.36684300 | 31.22695000 |
| 87744 | 2014-06-11 15:32:27 | 460011312504744 | 121.356233 | 31.22213 | 15.6  | 0    | 0          | 1   | 1      | 0       | 4.15 V–4.2 V | 121.36740800 | 31.22647000 |
| 87746 | 2014-06-11 15:32:37 | 460011312504744 | 121.356432 | 31.222008 | 15.1  | 0    | 0          | 2   | 1      | 0       | 4.15 V–4.2 V | 121.36760800 | 31.22594000 |
| 87748 | 2014-06-11 15:32:47 | 460011312504744 | 121.356405 | 31.221558 | 18.2  | 0    | 0          | 2   | 1      | 0       | 4.15 V–4.2 V | 121.36758100 | 31.22549000 |
| 87750 | 2014-06-11 15:32:57 | 460011312504744 | 121.356425 | 31.22182 | 12.5  | 0    | 0          | 2   | 1      | 0       | 4.15 V–4.2 V | 121.36760100 | 31.22514000 |
| 87752 | 2014-06-11 15:33:07 | 460011312504744 | 121.356395 | 31.220968 | 2.6   | 0    | 0          | 2   | 1      | 0       | 4.15 V–4.2 V | 121.36737100 | 31.22489800 |
| 88037 | 2014-06-11 15:42:27 | 460011312504744 | 121.356383 | 31.220912 | 9.9   | 0    | 0          | 8   | 2      | 0       | 4.15 V–4.2 V | 121.36755800 | 31.22484400 |
| 88042 | 2014-06-11 15:42:37 | 460011312504744 | 121.356395 | 31.220533 | 16.4  | 0    | 0          | 8   | 2      | 0       | 4.15 V–4.2 V | 121.36757000 | 31.22446400 |

Table 2: A certain terminal’s message information.
concurrent requests, and the server cluster runs ServerAgent-2.2.1 to respond to the requests.

In this test, each client simulates 2000 valid users to log in to the system and then send 2000 requests to insert condition messages. The GUI in the server shows the real-time processing speed and condition. Meanwhile, JMeter in the client side helps to monitor the performance of the server over time, including the changing usage rate and processing speed of CPU, memory, disk, and network, with the transaction controller for TCP sampler. Through such tests, statistics of the concurrent capacity can be obtained, which can be used to further determine the bottleneck of the system.

Running multiple tests on all five servers, we select one testing result from each sever randomly and show them in Table 3. The test column reflects the specific server. The table records the performance of the servers in the test in which 2000 concurrent requests are processed, including the average responding time. The median column depicts the responding time of 1000 concurrent requests, while the 90%_line column shows the responding time of 90% of full load, which is 1800 concurrent requests. The responding time in the table is measured by millisecond. It can be seen from the table that the amplitude of variation is low, while the average column and max column in row three and row five change greater compared to the other rows which may be due to the slight performance disparity of the client PCs. In general, the experiment data is reasonable.

From Table 3 and Figure 14, a conclusion can be drawn that five clients send total of 10,000 requests concurrently (each of the five sends 2000 requests), and the server is able to respond in time without system crash or service response failure.

**Table 3: Aggregate report data for experiments.**

| Test | Average | Median | 90%_line | Min  | Max  |
|------|---------|--------|----------|------|------|
| t1   | 91251   | 141450 | 144436   | 1475 | 151027|
| t2   | 99606   | 142759 | 145714   | 1211 | 148226|
| t3   | 106856  | 143295 | 146785   | 1253 | 252218|
| t4   | 94627   | 141384 | 144538   | 1576 | 151087|
| t5   | 137972  | 173908 | 200090   | 1396 | 203966|

**Figure 11:** Smartphone application: (a) tracking the path and (b) alarm messages.

**Figure 12:** Web application: (a) locating of the device and (b) tracking the path.
In addition, Figure 13 consists of three parts naming (a), (b), and (c), which depict the CPU, disk I/O, and memory performance, respectively. On the premise that the experiments are undertaken in LAN with 1000 M bandwidth, so the network I/O performance is not taken into consideration and recorded.

It is showed in the graphs that, in the beginning of the experiment, the performance index is great because the servers receive a burst of requests concurrently, and it has to respond to the requests as well as query database and process data in the cache queue. The performance index decreases over time to normal state, though with some fluctuation. As a conclusion, the design and implementation make the system able to support large-scale concurrent users smoothly.

6. Conclusion and Future Works

This system is designed to track the exact location of carrier in real time, such as vehicles, elderly or children, and pets. Moreover, with the help of G-sensor, it can also provide antitheft and monitor functions. A terminal is embedded in the carrier which includes three modules, GPRS/GSM module, GPS module, and G-sensor module. The embedded terminal receives and encapsulates GPS data, combined with other information according to a given format. Then it builds a socket and sends the messages. Considering the potential large scale of users, a distributed architecture is designed to reduce the pressure of every server. In addition, separating reading from writing can further improve the performance of the whole system. Two types of clients are implemented in this system so that users can track their carriers through web application or an application on the smartphone. The path of one embedded terminal, which displays in the Baidu map, can be acquired in both clients. The results have demonstrated the precision and practicability of the system and all the functions of the system are verified in this study.

For the future work, how to transfer message in binary representation and reduce transmission time and bandwidth is the next problem to be solved. At the same time, the large data stored in the system can be mined to find useful information to offer help to other functional departments, such as transportation department and police.
Conflict of Interests
The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments
This paper is based on a project “Electric Bicycle Bodyguard System” which has lasted for one year. The research is supported with partner SIM Technology Group Limited. And the authors would like to thank Weidong Lu.

References
[1] S. A. Hameed, O. Khalifa, M. Ershad, F. Zahudi, B. Sheyaa, and W. Asender, “Car monitoring, alerting and tracking model: enhancement with mobility and database facilities,” in Proceedings of the International Conference on Computer and Communication Engineering (ICCCCE ’10), pp. 1–5, May 2010.
[2] H. Tan, J. Feng, G. Feng, W. Wang, and Y.-J. Zhang, “Traffic volume data outlier recovery via tensor model,” Mathematical Problems in Engineering, vol. 2013, Article ID 164810, 8 pages, 2013.
[3] C. Roadknight, L. Parrott, N. Boyd, and J. W. Marshall, “Real-time data management on a wireless sensor network,” International Journal of Distributed Sensor Networks, vol. 1, no. 2, pp. 215–225, 2005.
[4] http://www.zdnet.com/idc-smartphone-growth-to-continue-reach-1-2-billion-in-2014-7000029947/.
[5] M. Zhao and X. Shi, “A new developed wireless sensor based on GPRS & GPS for dynamic signals of structures,” in Proceedings of the 5th International Workshop on Structural Health Monitoring, pp. 845–851, DEStech, Stanford, Calif, USA, September 2009.
[6] G. S. P. Ganesh, B. Balaji, and T. A. S. Varadhan, “Anti-theft tracking system for automobiles (AutoGSM),” in Proceedings of the IEEE International Conference on Anti-Counterfeiting, Security and Identification (ASID ’11), pp. 17–19, IEEE, Xiamen, China, June 2011.
[7] Z. Liu, A. Zhang, and S. Li, “Vehicle anti-theft tracking system based on Internet of Things,” in Proceedings of the IEEE International Conference on Vehicular Electronics and Safety (ICVES ’13), pp. 48–52, IEEE, Dongguan, China, July 2013.
[8] S. Lee, G. Tewolde, and J. Kwon, “Design and implementation of vehicle tracking system using GPS/GSM/GPRS technology and smartphone application,” in Proceedings of the IEEE World Forum on Internet of Things (WF-IoT ’14), pp. 353–358, IEEE, March 2014.
[9] C. Bo, C. Xin, Z. Zhongyi, Z. Chengwen, and C. Junliang, “Web of things-based remote monitoring system for coal mine safety using wireless sensor network,” International Journal of Distributed Sensor Networks, vol. 2014, Article ID 323127, 14 pages, 2014.
[10] P. Liang, H. Wang, C. H. Peng et al., “Digital transmitter design for mobile devices,” IEEE Communications Magazine, vol. 51, no. 10, pp. 114–123, 2013.
[11] T. Ren, M. Petovello, and C. Basnayake, “Requirements analysis for bit synchronization and decoding in a standalone high-sensitivity GNSS receiver,” in Proceedings of the Ubiquitous Positioning, Indoor Navigation, and Location Based Service (UPINLBS ’12), pp. 1–9, IEEE, Helsinki, Finland, October 2012.
[12] B. Ji, J. Zhou, L. Qian et al., “An implementation of LVS-based highly-available Radius server,” in Proceedings of the IET International Conference on Wireless Mobile and Multimedia Networks (ICWMMN’06), p. 396, November 2006.
[13] P.-J. Yang, Load balancing mechanism for QoS-aware cloud computing using eucalyptus platform [M.S. thesis], Department of Electrical Engineering, 2011.
[14] K. Suto, H. Nishiyama, N. Kato, T. Nakachi, T. Fujii, and A. Takahara, “Toward integrating overlay and physical networks for robust parallel processing architecture,” IEEE Network, vol. 28, no. 4, pp. 40–45, 2014.
[15] M. Kumar, N. Batra, and H. Aggarwal, “Cache based query optimization approach in distributed database,” International Journal of Computer Science Issues, vol. 9, no. 6, pp. 389–395, 2012.
[16] E. Anderson, C. Hoover, X. Li, and J. Tucek, “Efficient tracing and performance analysis for large distributed systems,” in Proceedings of the IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS ’09), pp. 1–10, IEEE, London, UK, September 2009.
[17] A. Dash and B. Demsky, “Integrating caching and prefetching mechanisms in a distributed transactional memory,” IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 8, pp. 1284–1298, 2011.
[18] S. Podlipnig and L. Böszörményi, “A survey of Web cache replacement strategies,” ACM Computing Surveys, vol. 35, no. 4, pp. 374–398, 2003.
[19] Y. Zhiyong, L. Chunlin, L. Yanpei, and A. Malek, “DRMST: a novel dynamic replica management strategy based on TOPSIS in cloud storage,” Journal of Software Engineering, vol. 8, no. 4, pp. 278–286, 2014.
[20] M. Rezvan, K. Pawlikowski, and H. Sirisena, “A distributed cache architecture for routing in hierarchical QoS-capable networks,” in Proceedings of the IEEE International Conference on Communications (ICC ’03), vol. 3, pp. 1749–1753, IEEE, May 2003.