A Telematics Service System Based on the Linux Cluster

Junghoon Lee[1], Gyung-Leen Park[1], Hanil Kim[2], Young-Kyu Yang[3], Pankoo Kim[4], and Sang-Wook Kim[5,⋆]

[1] Dept. of Computer Science and Statistics, Cheju National University,
[2] Dept. of Computer Education, Cheju National University,
[3] Graduate School of Software, Kyungwon University,
[4] Dept. of Computer Engineering, Chosun University,
[5] College of Information and Communications, Hanyang University
{jhlee,glpark,hikim}@cheju.ac.kr, ykyang@kyungwon.ac.kr,
pkkim@chosun.ac.kr, wook@hanyang.ac.kr

Abstract. This paper designs and implements a taxi telematics service system, aiming at providing an efficient framework by means of a Linux cluster to host emerging telematics services that need intensive computing. Combined with global positioning system and radio communication technology, the taxi telematics service system traces the position of taxis, finds a time saving route between start and destination points, dispatches the nearest taxi to the service call point based on the latest traffic information, and finally decides an efficient route for multiple destinations. The performance measurement result demonstrates that the implemented system can process up to 200 map matches for every minute, keeping average response time for other requests below 1.5 seconds.

1 Introduction

[1] Telematics is the blending of computers and wireless telecommunication technologies, with the goal of efficiently conveying information over vast networks to improve a host of business functions or government-related public services[1]. This term later evolved to refer to automation in automobiles, namely, vehicle telematics. Due to the mobility of vehicles, the telematics service is necessarily combined with satellite navigation and geographic information. Additionally, GPS (Global Positioning System) and radio communication technologies have become indispensable in providing various telematics services such as vehicle location tracking, automatic collision notification, and location-driven driver information[2].

One of essential applications of telematics networks is vehicle tracking, which is capable of monitoring the location, movement, status, and behavior of a vehicle. The vehicles may be taxis, rent-a-cars, delivery trucks, and any other

⋆ Corresponding author.

[1] This research was supported by the MIC, Rep. of Korea, under the ITRC support program supervised by the IITA (IITA-2006-C1090-0603-0040).

Y. Shi et al. (Eds.): ICCS 2007, Part IV, LNCS 4490, pp. 660–667, 2007.

automobiles. Particularly, for the call taxi company which runs a lot of taxis, the knowledge on the real-time position of each taxi enables to make a decision on the closest taxi to a service call, the best route, and so on. In addition, with the periodic report on the current speed from each taxi, information on the up-to-date traffic conditions, accidents, and jams can be dynamically monitored and estimated[3]. The collected data can be also exploited to some challenging traffic information technologies such as vehicle relationship management, traffic forecast, and mobility analysis[4].

With such requirements, this paper designs and implements the taxi telematics service system capable of tracing the positions of taxis, locating the nearest taxi to a service call, finding an economic path based on the latest traffic information, and discovering an efficient route for multiple destinations on top of a *Linux cluster*. The Linux cluster is known to be the most cost-effective and scalable way to provide large amounts of hardware and software resources. This parallel processing framework seems prospective since the existing and future telematics service will not only handle large amount of data but also need a significant computing power. The functions mentioned above can be implemented using various fundamental techniques including road network management, map matching heuristics[5], diverse versions of path finding algorithms[6], stable database management, and so on.

Using the taxi telematics system, company managers are able to supervise and guide every driver to travel in the optimal route, which is fuel-efficient. The customers need not wait for a longer period since the system aims to dispatch the nearest available taxi. They can also reach the destination by means of the optimal time saving route. The work of the driver is relieved since all guidance is provided by the automated server. Finally, a newly developed service can be easily integrated into the system.

This paper is organized as follows: After issuing the problem in Section 1, Section 2 provides some background for the target taxi telematics system. Then, Section 3 describes the system architecture and implementation details. After discussing the performance measurement results in Section 4, Section 5 concludes this paper with a brief description on future work.

2 Background

The taxi telematics system has been developed as a milestone project for the *Jeju Telematics City* enterprise in Republic of Korea. As one of the most famous tourist attractions in East Asia, Jeju Island is a popular vacation spot for Koreans and many international visitors. It has a well maintained road network which essentially follows the entire coast (200 km) and crisscrosses between the island's major points. In terms of a road network, there are about 18,000 intersections and 27,000 road segments. This means that the road graph can be built with 18,000 nodes and 27,000 links along with some additional data structures such as POI (Point of Interest), allowing the implementor to store the whole graph in the high-speed main memory, not using low-speed file systems or databases

in disk. Hence, almost every functions can be carried out only within the main memory. With the enterprise mentioned above, most of rent-a-cars have already installed in-vehicle telematics for the purpose of tour guide, navigation, safety service, entertainment, and so on. Moreover, some taxi companies began to employ a call taxi control system which tracks the location of taxis, decides the nearest taxi to reduce the waiting time for a customer, maintains the diverse service records for each taxi[7].

The in-vehicle telematics in taxis contains a GPS receiver as well as air interface, which is CDMA (Code Division Multiple Access) protocol in Korea. Each taxi reports its location and speed every minute with a reasonable communication cost negotiated with a telecommunication company. A remote server is responsible for receiving, managing, and exploiting this information. However, the current system just emphasizes the function of vehicle tracking and taxi selection according to the *Euclidean distance* to the call point. They do not even take advantage of the freshness of traffic data in path finding or taxi selection.

In the mean while, all-day hire of a taxi is one of the most popular tour patterns in Jeju, especially for the first time visitors. The customer can suggest multiple destinations, restaurant preference, and so on. Then, the driver makes a tour plan with this requirement and his own experiences. The tour plan essentially starts from and ends at the customer's hotel as long as the customer does not change his hotel, so it can be considered to be a TSP (Traveling Salesman Problem)[8]. TSP is one of the widely studied NP-hard combinational optimization problems and can be described as follows: To begin with, let $G = (V, E)$ be a graph where V is a set of nodes and E is a set of links, and $C = (c_{ij})$ be the distance or cost matrix associated with E. TSP is to find the cheapest way of visiting all of the nodes and return to the starting point.

Among diverse heuristics to solve TSP, the most famous one is Lin-Kernighan's [9,10]. It is known for its success in efficiently finding near-optimal results, while the core of this scheme consists of link exchange in a tour. A research web site of *Concorde* provides diverse TSP solutions including Lin-Kernighan's in a source level, enabling users to download and integrate it to their own system[9]. Though this software can calculate the route plan efficiently, it needs cost matrix each element of which has the respective cost from node i to j. Each element can be calculated by one-to-many version of the Dijkstra algorithm or A* heuristics[6]. Because the calculating step for each element is independent, each calculation can be performed in parallel using a Linux cluster.

3 System Architecture and Implementation

Our telematics service system consists of the telematics device on each taxi, call control server, telematics server, and Linux cluster as shown in Fig. 1. With this architecture, this service system provides such functions as the dynamic link cost update, taxi assignment, path finding between the two points, and path planning for multiple destinations.

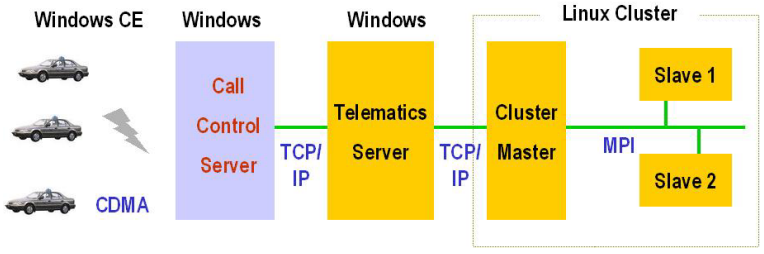


Fig. 1. The architecture of the taxi telematics service system

3.1 Telematics Device and Call Control Server

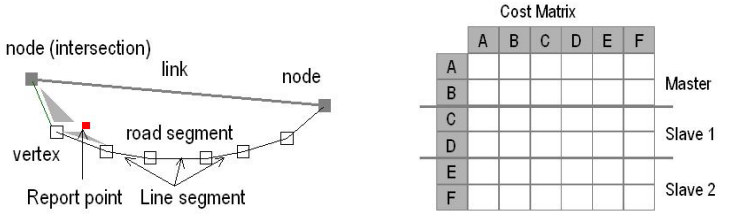
Basically, the telematics device provides taxi drivers with a user interface for the remaining telematics service system. So, the driver can invoke the function he wants via this device. Each telematics device deploys Windows CE 5.0 operating system, and receives information on its location every 1 second from the GPS receiver observing the NMEA specification. The information includes standard time, latitude, longitude, current speed, and moving direction. The telematics device reports these data to the *call control server* residing within the Internet domain. The servers on the wired network can be accessed from each taxi through CDMA air interface, as Windows CE supports TCP/IP communication protocol over the RAS (Remote Access Service) utility.

The call control server tracks and stores the current location of each taxi in the (latitude, longitude) coordinate system as well as to report such data to the *telematics server* for the advanced processing. Currently, up to 200 taxis are tracked simultaneously within our system, and the number of taxis will grow soon. It regenerates a fine-grained request to the telematics server upon receiving a specific function from the human operators or taxi drivers. For example, when a service call arrives from a customer, this server first extracts the list of taxis within 10 km radius from the call point and then sends the list along with the call point itself to the telematics server. After all, this server performs a preliminary functions and plays a role of a gateway to the telematics system.

3.2 Telematics Server

The telematics server provides enhanced functions to the telematics. There are two major data maintained in this server: (1) a digital map as a form of ESRI shape file and (2) a road network as a form of a main memory data structure in a multiple adjacent list graph, respectively. The digital map has full information on each road segment represented as a sequence of lines and vertices. On the other hand, the road network has only nodes and links, which are intersection and two end points of the road segment, respectively. In Fig. 2(a) which illustrates a road segment, a road network in main memory stores just nodes and link, while the digital map stores all nodes, vertices, and line segments. While the digital map is used just for the map matching purpose, the road network is exploited

for sophisticated functions such as taxi dispatching and path finding. Since the road network is the fundamental data to the system, the efficient management of the road network is very crucial to the overall performance of the system. For the town of small or medium size like Jeju, the data is not too large, so it can be stored, processed, searched efficiently only within main memory, avoiding the time-consuming file system calls in a disk.



(a) Digital map and road network. (b) Cost matrix.

Fig. 2. Basic assumptions

The telematics server provides the following functions. First, for the speed report message from the telematics through the call control server, it finds the link that corresponds to the location specified in the message as the (latitude, longitude) pair. As the report can be generated at any place of each road segment along the actual road, the whole digital map file should be searched. For a road segment, the area of each triangle which consists of two end points of each line segment and a report point is calculated, as shown in Fig. 2(a). Then, we can decide the distance from the line segment and the report point and if the distance is less than a given limit, the ID of that link is returned. After finding the corresponding link, the server updates the cost of that link with the reported speed and then stores the record in the system database table in the master node of the Linux cluster. This map matching is endowed with the highest priority as it is periodic and time-sensitive.

Second, for the taxi dispatch request, the server also receives the location of service call and the list of candidate taxis within a radius of 10 km. It decides the nearest taxi to the location of the service call in terms of the network distance by performing the classical Dijkstra's algorithm for multiple destinations. To this end, the telematics server maps all relevant locations represented in the (latitude, longitude) coordinate system to the nearest nodes. This version begins from the location of a service call and proceeds to the current location of candidate taxis by the breadth first search until it reaches any one of them.

Third, for the path finding request with the specification of a source and a destination, it employs a well-known A* algorithm in which the Euclidean distance is exploited for the remaining estimation and network distance for the accumulated cost[6]. In addition, the path finding scheme provides another option which takes into account the current moving direction of a vehicle. After matching the angle between the road segment and the taxi's direction, we can decide the node the taxi will arrive. In the angle matching procedure, the digital map file is used. We also calculate the distance and estimate the time duration from start point to end point.

Finally, receiving a multi-destination planning request, it transfers the request to the Linux cluster after converting the (latitude, longitude) coordinates to the node ID's that are shared with the Linux cluster.

3.3 Linux Cluster

To run a Lin-Kernighan algorithm for the given set of nodes, we need the cost matrix that contains the cost of a path for every node pair in the given set as shown in Fig. 2(b). It takes a significant time to compute the matrix but Linux cluster provides a cost-effective way to enhance the performance of such computation. When the master receives the list of nodes from the telematics server, it partitions the work and distributes the subtasks to the slaves via the MPI (Message Passing Interface) communication primitive[11]. Each computer runs the A* or multi-destination Dijkstra algorithm which calculates the cost from one given node to multiple destinations. This version begins from the start point and proceeds until it reaches all of the destinations. After building the cost matrix, the master runs the Lin-Kernighan algorithm. As it takes less than a second for less than 20 nodes, we don't have to consider the parallel version of a traditional TSP solver.

The cluster master has installed the MySQL DBMS to store the speed report records[12]. As a relational DBMS, MySQL enables to access databases anywhere on the internet. Hence, after installing a MySQL client version on the telematics server, it can not only store the record into the database table, but also make it possible for other analysis server to access the data at any time.

4 Performance Measurement Result

The telematics server runs on the IBM ThinkPad X31 personal computer which has a 1.4 GHz Pentium processor with 512 MB main memory and 100 Mbps Ethernet interface. In the Linux cluster, each of 3 nodes is equipped with a 700 MHz Pentium 3 CPU and 384 MB memory with 100 MBps Ethernet interface. In addition, the NETGEAR 8-port Fast Ethernet switch connects all of cluster nodes to build a private network. Finally, all nodes installed Redhat Linux version 9.0 and LAM-MPI version 7.1.2.

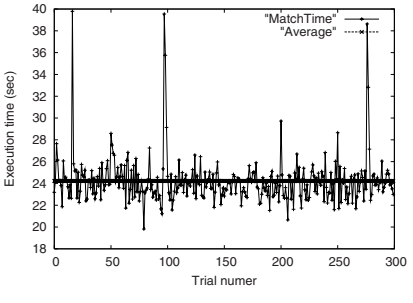


Fig. 3. Performance of map matching

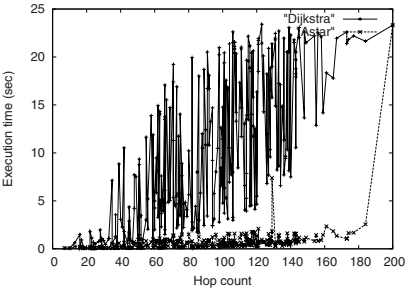


Fig. 4. Performance of path finding

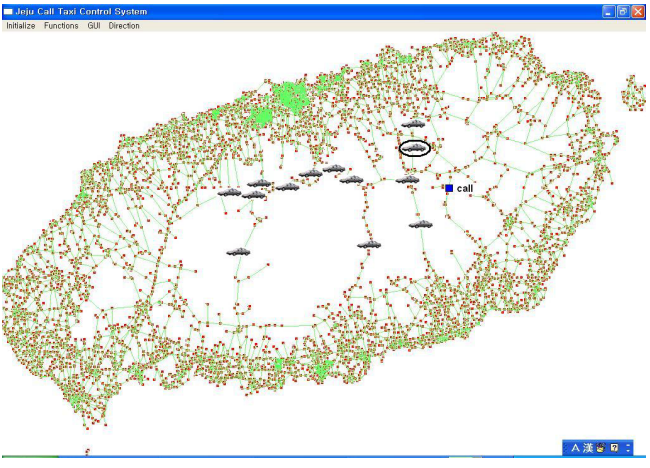


Fig. 5. Finding the nearest taxi to the service call

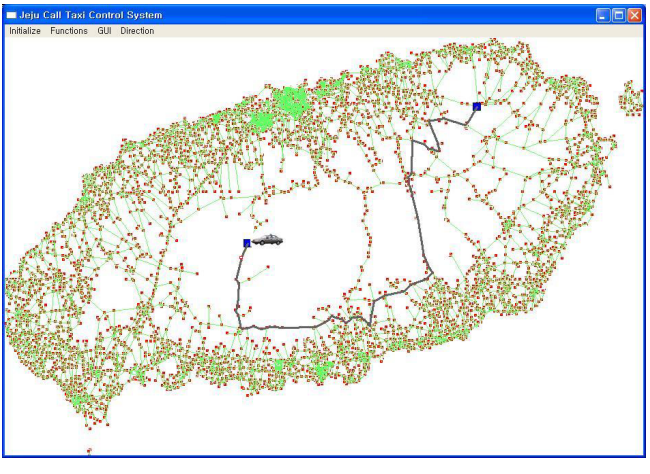


Fig. 6. Multi-destination planning

Fig. 3 plots the execution time of 200 map matches for every 1 minute. The execution time does not exceed 40 seconds (in about 24 seconds on average), giving a sufficient margin to other functions. Fig. 4 plots the execution time of path finding compared with Dijkstra's in the map of the Jeju area. In case there is no feasible route between two points, both schemes show the same execution time, but A* can generally performs 10 $\sim$ 100 times faster than Dijkstra's, providing a reasonable route.

Fig. 5 shows the example of a taxi dispatch result. The closest taxi marked with a circle is found for the candidate to serve the call, as it is the closest in the network distance. The execution time depends on how close the call point is to

such a taxi. Fig. 6 shows the example of a TSP execution result. The execution time also depends on the network distance between the destinations. These GUI displays are implemented in the call control server.

5 Concluding Remarks

In this paper, we designed and implemented a taxi telematics service system for a mid-size city, aiming at providing an efficient framework with a Linux cluster to host a computing-centric telematics service. Taking advantage of the moderate number of nodes and links, we implemented the taxi telematics service system capable of updating the current cost of a link, tracing the position of taxis, finding an efficient path, dispatching the nearest taxi to the service call based on the latest traffic information, and deciding an economic route for multiple destinations. Even though the system is made up with relatively low-performance components, this system can not only meet the system requirements and functional specifications but also be upgraded to achieve a better performance.

As a future work, we are planning to develop a more sophisticated telematics service such as path recommendation, combination of call taxi and tourist information ontology, and a multimedia delivery over the telematics network for advertisement and entertainment.

References

1. http://en.wikipedia.org/wiki/Telematics
2. Kiruthivasan, S., Madan Deepakumar, C., Althaf, S.: Decision Support System For Call Taxi Navigation Using GIS-GPS Integration, MAP India (2006)
3. Lee, S., Lee, B., Yang, Y.: Estimation of Link Speed Using Pattern Classification of GPS Probe Car Data. Proc. International Conference on Computational Science and its Applications (2006) 495-504
4. Jeong, S., Kim, S., Kim, K., Choi, B.: An Effective Method for Approximating the Euclidean Distance in High-Dimensional Space. Proc. International Conference on Database and Expert Systems Applications (2006) 863-872
5. Marchal, F., Hackney, J., Axhausen, K.: Efficient map matching of large global positioning system data sets: Tests on speed-monitoring experiment in Zürich. Journal of the Transportation Research Board (2005) 93-100
6. Goldberg, A., Kaplan, H., Werneck, R.: Reach for A*: Efficient point-to-point shortest path algorithms. MSR-TR-2005-132. Microsoft (2005)
7. Liao, Z.: Real-time taxi dispatching using global positioning systems. Communication of the ACM, **46** (2003) 81-83.
8. Winter, S.: Modeling Costs of Turns in Route Planning. GeoInformatica **6** (2002) 363-380
9. http://www.tsp.gatech.edu/concorde.html
10. Haghani, A., Jung, S.: A dynamic vehicle routing problem with time-dependent travel times. Computer & Operation Research **32** (2005) 2959-2986
11. Pacheco, P.: Parallel Programming with MPI. Morgan Kaufmann Publishers, Inc (1996)
12. Zawodny, J., Balling, D.: High Performance MySQL. O'Reilly Media (2004)