Development of the Integrated Logistics Information System based on Cloud Computing

Minjeong An[1], Minyoung Lee[2], and Hongchul Lee[3]

***Abstract*—**This study presents a prototype of the integrated logistic information system (ILIS) in the Google cloud computing environment. The ILIS is purpose to optimize resources in logistics business area, such as vehicles, human resources and fuel. This system is targeting to the small and medium-sized third-party logistics (3PL) company that are difficult to establish IT infrastructure, by providing remote service in cloud computing environment. In this study, we utilized theories of traveling salesman problem (TSP) and Dynamic Programming (DP) for route optimization. And we have developed the ILIS using Google App Engine and Google Cloud SQL. We intend to contribute to improve the efficiency of logistics through this system.

***Keywords*—**Cloud computing, logistics information system, transportation management system, traveling salesman problem

I. Introduction

LOGISTICS industry has constantly evolved according to increasing of the trade domestic and abroad. So IT technology is essential to logistics industry to reduce cost and to improve the efficiency of logistics. But core components of the logistics system are depending on expensive logistics solutions. In addition, logistics corporations intend to integrate load optimization functions and route optimization functions at a single system. And most of third-party logistics companies are small and medium-sized scale, so they are difficult to establish IT infrastructure and are hard to operate logistics system with respect to the cost. To improve these aspects, we had designed the ILIS based on cloud computing environment at last study. In this study, we have developed the main process and core functions of the ILIS in Google cloud computing environment.

II. Conceptual Design

A. Business Process

The Logistics business is made up of task such as order, transportation plan, delivery plan, load optimization, route optimization, shipping and calculation until complete delivery after receiving order of the customer [1]. "Fig. 1," shows the overall process of logistics business. After receiving customer's delivery order, a logistics company plans the delivery schedule in accordance with the appointment. In the case of long-distance transportation between Central Distribution Center (CDC) and Regional Distribution Center (RDC), logistics managers conduct the load optimization to reduce the number of vehicle because vehicles for transportation is usually large scale. In the case of regional delivery, they conduct the route optimization to minimize the total driving distance or total delivery cost between multiple destinations. If they confirm the planned schedule, driver conducts shipping according to that schedule. During shipping, logistics managers monitor GPS (Global Positioning System) data from vehicles. When drivers complete the delivery to the customer, they perform POD (Proof of Delivery) to the logistics company.

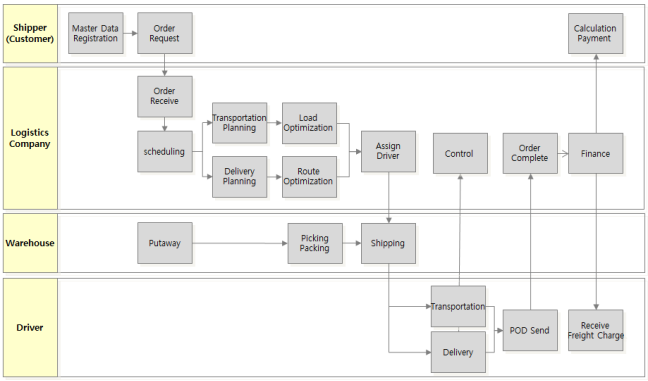


Fig. 1 Overall logistics business process [1]

B. System Configuration

In terms of system, the stakeholders of the ILIS are shippers, logistics companies, warehouse managers and vehicle drivers. For performing this logistics business process seamlessly, we have designed the ILIS by integrating a master data management (MDM), an order management system (OMS), a warehouse management system (WMS), a transportation management system (TMS), a fare management system (FMS) and a vehicle monitoring. The MDM is a base module that manages the critical data for operating the ILIS. The OMS manages customer's order and the WMS controls the movement and storage of materials in warehouse. The TMS plans the transportation and delivery. And the FMS calculates the freight charge with customers or drivers. A vehicle monitoring module is a real time GPS tracking function of the vehicle.

Minjeong An[1] is with the Department of Industrial Management Engineering, Kora University, Seoul, Korea (e-mail: ansebby@korea.ac.kr).

Minyoung Lee[2], was the Department of Industrial Management Engineering, Kora University, Seoul, Korea (e-mail: justmins@korea.ac.kr).

Hongchul Lee[3] is with the Department of Industrial Management Engineering, Kora University, Seoul, Korea (corresponding author's phone: 82-2-3290-3389; e-mail: hclee@korea.ac.kr).

TABLE I
SERVICE MODELS AND DEPLOYMENT MODELS OF CLOUD COMPUTING [1]

Type	Model	Definition
Service Models	SaaS	Software as a Service : SaaS provides application software as a shape of online service. (e.g., SalesForce CRM, Google Apps, DeskAway, Impel CRM, Wipro w-SaaS)
	PaaS	Platform as a Service : PaaS is a broad collection of application infrastructure services (including application platform, integration, business process management and database services) (e.g., Microsoft Azure, Google App Engine, Oracle Cloud Platform, Force.com)
	IaaS	Infrastructure as a Service : IaaS delivers computer infrastructure (including servers, software, data-center space and network equipment) as a service. (e.g., Amazon Elastic Compute Cloud (EC2), Google Compute Engine, Oracle IaaS)
	DBaaS	Database as a Service : DBaaS is a service that is managed by a cloud operator that supports applications. (e.g., Amazon RDS, Microsoft Azure SQL database, Google Cloud SQL, Oracle DBaaS)
Deployment Models	Private cloud	A private cloud is a particular model of cloud computing that involves a distinct and secure cloud based environment in which only the specified client can operate. (e.g., Amazon Virtual Private Cloud(VPC), Microsoft Private Cloud)
	Community cloud	The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns [2]. (e.g., mission, security requirements, policy, and compliance considerations).
	Public cloud	A public cloud is the standard cloud computing model, in which a service provider makes resources, such as applications and storage, available to the general public over the internet. (e.g., Amazon Web Services , IBM SmartCloud, Oracle Cloud, Google App Engine, Windows Azure)
	Hybrid cloud	The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability [2].

Because each module has been developed as an independent software, the logistics company had to integrate each solution for their seamless work. In this system, we have designed an integrating structure of each module with development standard. Specially, we focused on a TMS which is consisted of core scheduling functions with optimization. “Fig. 2,” shows the overview of the ILIS integrating overall process of the logistics business. This describes interaction between main functions of the ILIS. As shown in Fig. 2, the ILIS is essential to the logistics business because real time data processing between stakeholders is important.

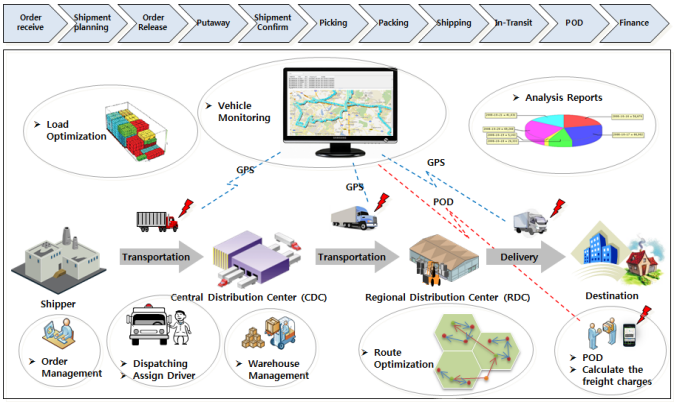


Fig. 2 Overview of the ILIS [1]

III. MATERIAL AND METHODS

A. Cloud Computing

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [2]. Gartner defines cloud computing as "a style of computing where massively scalable IT-enabled capabilities are delivered 'as a service' to external customers using Internet technologies" [3]. Essential Characteristics of cloud computing are on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service [2], [4]. The cloud computing service is classified to public cloud, private cloud, hybrid cloud and community cloud according to deployment target [2]. And it is classified to SaaS, PaaS, IaaS and DBaaS according to service model. Four service models and four deployment models were described in Table I.

The ILIS developed in this study provides a SaaS in public cloud environment. After analyzing technology dependence and provided free services between Google App Engine and Microsoft Windows Azure, we adopted Google App Engine and Google Cloud SQL as development environment. Microsoft Windows Azure is more powerful than Google App Engine in terms of database and function and pricing and supported language [1]. But Google App Engine provides more useful free services [1].

B. Traveling Salesman Problem (TSP)

The traveling salesman problem (TSP) is to find the minimum cost path passing through all the given n cities exactly once [5]–[9]. It is one of the most widely studied combinatorial optimization problems and a well-known NP-hard problem [5], [6].

A large number of exact algorithms have been proposed for the TSP, such as integer linear programming (ILP), branch-and-bound (BB) algorithm and dynamic programming (DP) [6]. These methods could not be practically used due to

their exponential time complexity in the worst case [5].

Therefore, one stream of research has consisted of developing heuristics with a guaranteed worst-case performance [6]. The heuristic algorithms to finding near optimal solution for the TSP are genetic algorithm (GA) [10], ant colony optimization (ACO) [11], simulated annealing (SA) [12], tabu search (TS) [13], neural network (NN) [14]. But they did not guarantee to find the optimal solution.

The multiple traveling salesman problem (MTSP) is a generalization of the TSP, where more than one salesman is allowed to be used in the solution [15]. MTSP extends the problem to a wide variety of vehicle routing problems (VRPs) by incorporating some additional side constraints such as vehicle capacity limits, delivery time windows, as well as time dependent speeds in metropolitan area [15].

In this study, suppose to a single driver and the load already was clustered to the vehicle, we have used dynamic programming to solve TSP for route optimization. The clustering to make shipping group is processed by internal data processing. We describe this process in the process modeling section.

C. Dynamic Programming (DP)

Dynamic programming is a technique for efficiently implementing a recursive algorithm by storing partial results [16]. In this approach we solve small instances first, store the results, and later, whenever we need a result, look it up instead of recomputing it [5]. In dynamic programming algorithms, we construct a solution from the bottom up in an array (or sequence of arrays) [5]. Because the principle of optimality applies in a Shortest Paths problem, we can develop a recursive property that gives an optimal solution to an instance in terms of optimal solutions to subinstances [5].

"Fig. 3," shows a weighted, directed graph. The circles represent vertices, and a line from one circle to another represents an edge.

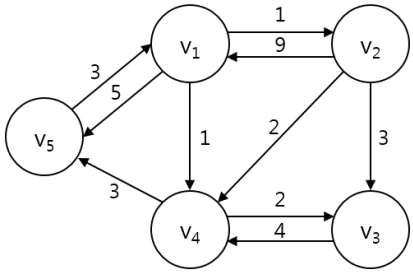


Fig. 3 A weighted, directed graph [5]

An arrow shows the direction. If the edges have values associated with them, the values are called weights [5]. We represent a weighted graph containing n vertices by an array W where

$$W[i][j] = \begin{cases} \text{weight on edge} & \text{if there is an edge from } v_i \text{ to } v_j \\ \infty & \text{if there is no edge from } v_i \text{ to } v_j \\ 0 & \text{if } i = j. \end{cases}$$

The graph in Figure 3 is represented by adjacency matrix like below [5].

$$W = \begin{bmatrix} 0 & 1 & \infty & 1 & 5 \\ 9 & 0 & 3 & 2 & \infty \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \infty & \infty & \infty & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 1 & 3 & 1 & 4 \\ 8 & 0 & 3 & 2 & 5 \\ 10 & 11 & 0 & 4 & 7 \\ 6 & 7 & 2 & 0 & 3 \\ 3 & 4 & 6 & 4 & 0 \end{bmatrix}$$

The matrix D is the distance matrix. It contains the lengths of the shortest paths in the graph [5]. We accomplish this by creating a sequence of $n + 1$ arrays D(k), where $0 \le k \le n$ and where $D^{(k)}[i][j]$ is the length of a shortest path from vi to v_j using only vertices in the set $\{v_1, v_2, \ldots, v_k\}$ as intermediate vertices [5]. We can determine D(k) from D(k-1) as follows [5]:

$$D^{(k)}[i][j] = minimum\left(D^{(k-1)}[i][j], D^{(k-1)}[i][k] + D^{(k-1)}[k][j]\right)$$

The array P is the optimal path matrix [5]. When at least one intermediate vertex exists, P[i][j] is the highest index of an intermediate vertex on the shortest path from vi to v_j [5]. If no intermediate vertex exists, it is zero [5]. The graph in Fig. 3 is represented by the optimal path matrix like below [5].

$$P = \begin{bmatrix} 0 & 0 & 4 & 0 & 4 \\ 5 & 0 & 0 & 0 & 4 \\ 5 & 5 & 0 & 0 & 4 \\ 5 & 5 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 \end{bmatrix}$$

Using the above dynamic programming, we can get the optimal solution, which the minimum distance is 14 and the shortest paths are {1-2-3-4-5-1}.

In the case of a straightforward enumerative of paths, the time complexity is O($n!$) [5]. In the case of dynamic programming, the time complexity can be reduced to $O(n^2 2^n)$ [5]. The memory complexity is $O(n2^n)$ [5], [17]. Although the complexity was reduced, this method is feasible when n is small [5].

In logistics business area, the vertices can be represented to destinations for delivery and the edge is distance between each destination. Because we made shipping group by other process model, the number of vertices are not large number. Moreover our system is going to be run in cloud computing environment, we can get more efficient performance than single local computing environment.

IV. Design Of The ILIS

We have designed the application architecture, process model and data model based on cloud computing technology and TSP theory.

A. System Architecture

We have designed the application architecture of the ILIS in Google cloud computing environment as shown in Fig. 4. Infrastructure layer is a physical layer which is running ILIS. Application layer is the environment which services are developed and deployed and testing. This is provided as a shape of an Eclipse plug-in and Google App Engine SDK. We have designed these service components in a standardized manner using the MVC design pattern.

Master data management component performs management about base data for running ILIS such as customer, product, vehicle, driver, destination, zone, and so on. Order management

component consists of functions that receive, modify and complete order from customer. Transportation management component consists of functions such as delivery scheduling, the route optimization and shipping execution. These components are provided in the shape of SaaS which is generated to Google App Engine project.

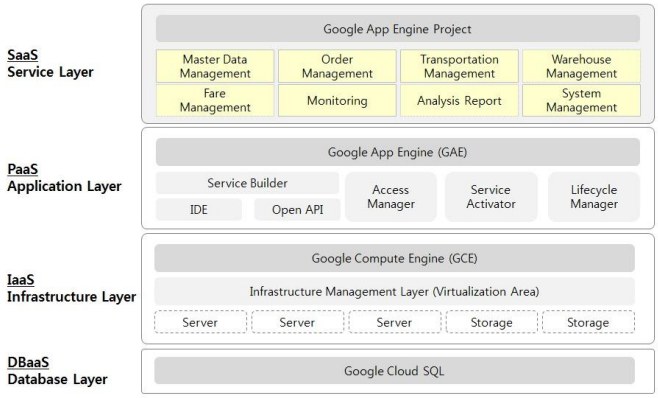


Fig. 4 Application Architecture of the ILIS

We adopted java language and relational database are widely used. "Fig. 5," describes the software architecture of the ILIS according to these technology. After developing and testing in the local development environment, we can deploy to Google App Engine.

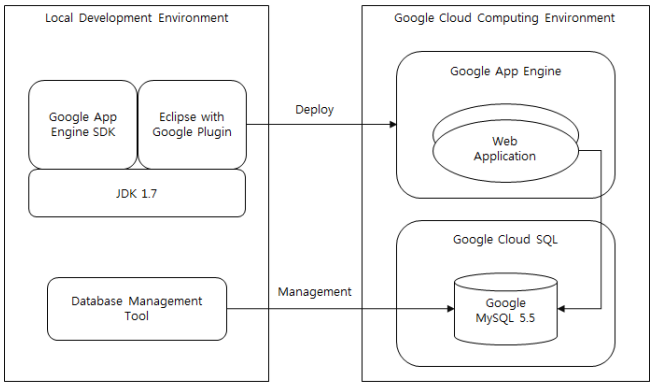


Fig. 5 Software Architecture of the ILIS

This is provide by a shape of the eclipse plug-in. And if we create a Google MySQL instance at the Google Cloud SQL, we can manage cloud database using general database management tool for MySQL.

We have established the development standard based on Model 2 structure, which is Model-View-Controller (MVC) pattern. Because we use Java language, MVC is implemented by Java Bean, JSP and Servlet.

B. Process Model

We have designed the overall process model and focused on clustering before making shortest delivery route. Each task of this process was designed as follows:

1) *Getting Orders:* At first, customer's delivery orders are received at OMS.
2) *Transportation Plan:* Transportation plans are generated at TMS according to product model, destination and appointment time. CDC is selected by product model and RDC is selected by zone of the destination. After checking appointment time, transportation group is generated. The ShippingID is generated by CBM (Cubic Meter) checking of each order based on vehicle capacity. In this step, if an order's capacity is larger than vehicle's capacity, this is split. Next, orders of transportation group are assigned to the available vehicle. In this step, if total capacity of the orders is larger than capacity of the assigned vehicle, the vehicle is added. After clustering like this, the ShippingGroupID is generated. The ShippingGroupID is a primary key of vehicle tracking during shipping.
3) *Confirm Transportation:* The logistics planner confirms transport plan generated in 2).
4) *Delivery Plan:* Delivery plans are generated at TMS according to destination and appointment time. After checking appointment time, delivery group is generated. The ShippingID is generated by same process like 2).
5) *Delivery Route Generation:* For getting shortest path of delivery, TMS generates weight matrix using latitude and longitude of the destination. The weight means distance between each destination. TMS generates the delivery sequence in delivery group by means of dynamic programming described in section III.C using weight matrix.
6) *Confirm Delivery:* The logistics planner confirms delivery plan generated in 5).
7) *Move the Plan Data to Shipping Data:* If scheduled data is confirmed by planner, these are moved to shipping data. Before moving, planner can re-scheduling.
8) *Start Shipping Monitoring:* The moved data can be executed and monitored. If shipping execution is started, GPS data is gathered from vehicle.
9) *POD Update:* After finishing delivery, driver sends POD in real time.
10) *Stop Shipping Monitoring:* When TMS receive POD signal, shipping execution and monitoring are finished at TMS.

"Fig. 6," describes these process model in detail.

C. Data Model

We have designed relational data model of the ILIS by traditional methods shown as Fig. 7. Because Google Cloud SQL is based on MySQL5.5, physical data model is same with general data model for MySQL. This model is made up of order data, master data, shipping plan data, shipping execution data, monitoring data.

V. Development Of The ILIS

We have developed the ILIS according to the development standard established in architecture design step. "Fig. 8," summarizes user interfaces of the ILIS.

This user interface (UI) is operated in Google Compute Engine after deploying by Google App Engine based on Eclipse. "Fig. 8 (a)," shows customer's order list. Transportation plan is displayed and executed in Fig. 8 (b).

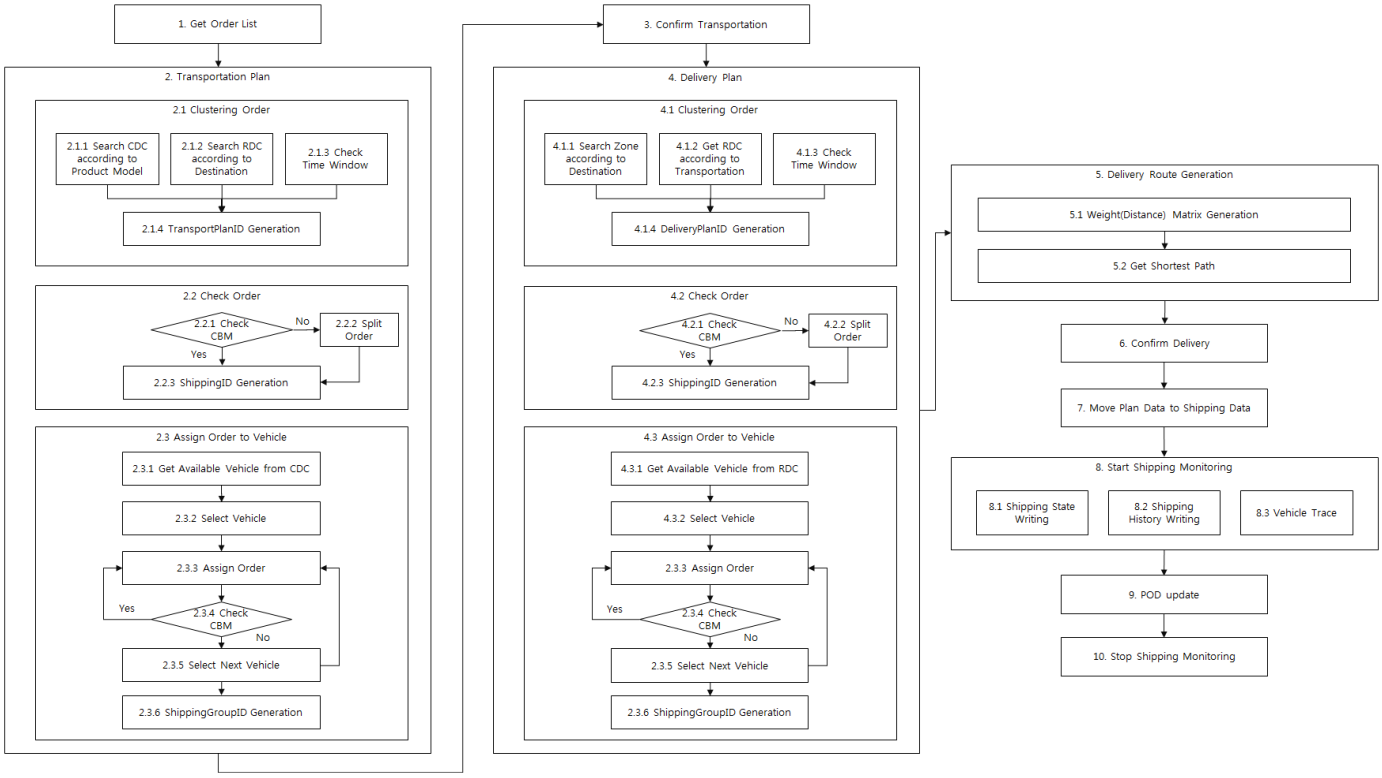


Fig. 6 Process Model of the ILIS

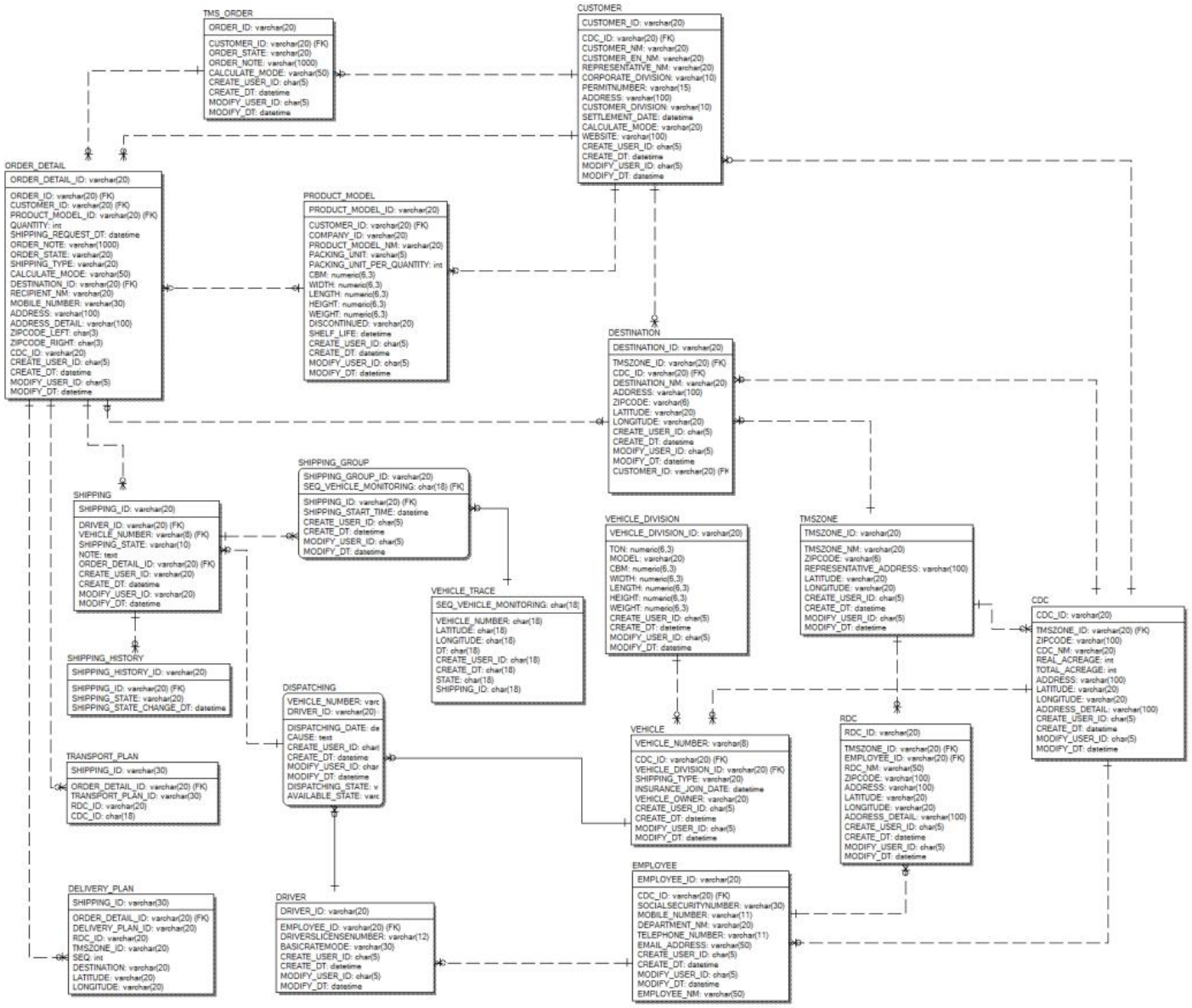
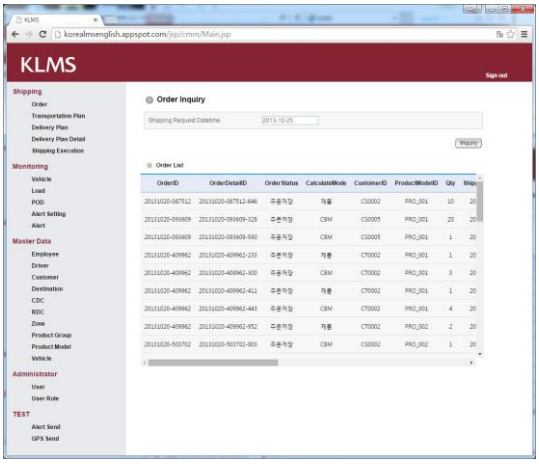


Fig. 7 Physical Data Model of the ILIS

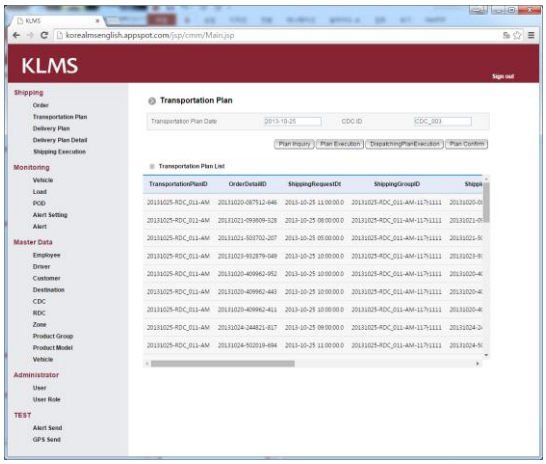
Assignment of the vehicle also can be at this UI. In this UI, a planner can inquiry and confirm shipping and dispatch the vehicle and execute shipping. In delivery plan UI of the Fig. 8 (c), a planner also can manage delivery plan. In delivery plan detail UI of the Fig. 8 (d) and (e), a planner can generate the delivery route. And the optimal route is presented at the map with delivery sequence. After a planner confirms transportation and delivery plan in Fig. 8 (b), (c), a planner can manage shipping status in shipping execution management UI shown as Fig. 8 (f). A driver also can generate POD at this UI in mobile environment, after finishing delivery.

"Fig. 9," shows the monitoring UI of the important object, such as vehicle, POD, load. "Fig. 9 (a)," shows movement route data of vehicles during driving. We have developed this UI using Google Map API and GPS data of the vehicle. In this UI, a planner can compare real movement route with planned route. "Fig. 9 (b)," displays real time POD status and Fig. 9 (c) shows movement history of the load.

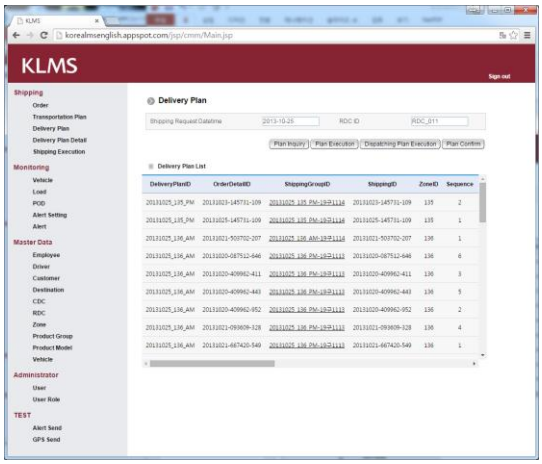
(a) Order Inquiry



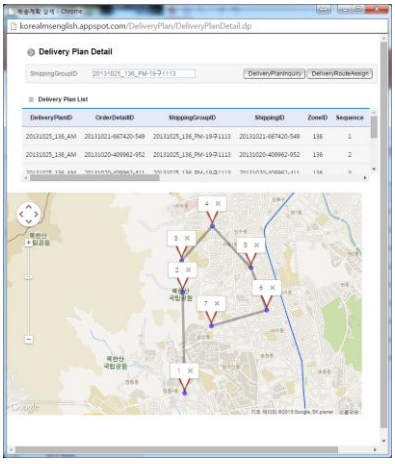
(b) Transportation Plan



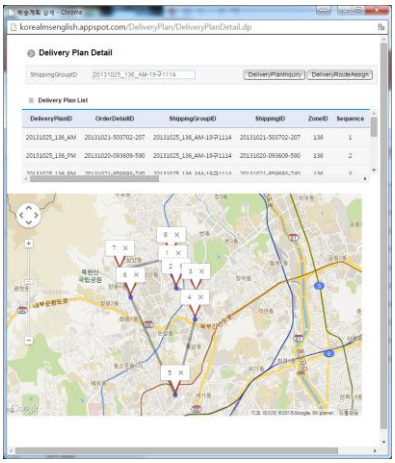
(c) Delivery Plan



(d) Delivery Plan Detail



(e) Delivery Plan Detail



(f) Shipping Execution Management

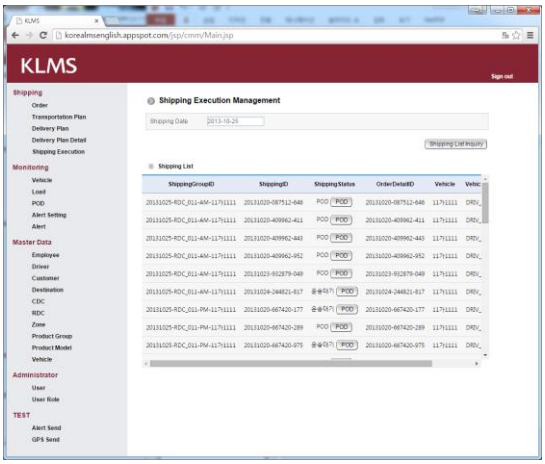


Fig. 8. Major UI in the ILIS

(a) Vehicle Monitoring (b) POD Monitoring (c) Load Monitoring

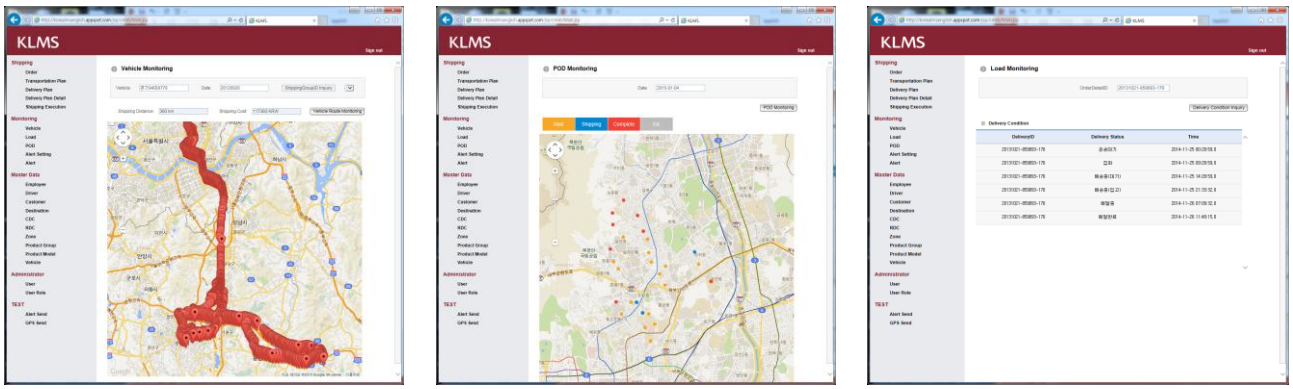


Fig. 9. Monitoring UI in the ILIS

VI. Conclusions And Future Works

In this study, we proposed a prototype of the ILIS in the Google cloud computing environment. The ILIS can contribute to the optimization of logistics operations through offer to the small and medium sized logistic company by an economical cost. The function of the route optimization can reduce the driving distances through suggestion about shortest paths before starting delivery. Because the ILIS is serviced by mobile web, logistics work could be more efficiently performed by real time.

Our future works are consideration about security and data collision because this system is sharing by many logistics company. And a multi-tenancy architecture of database has to be designed to share with other logistics company. In terms of data traffic and size, real time GPS data is small and simple but this is huge with the course of time. Therefore Big Data technology could be more suitable for vehicle monitoring. We have plan to vehicle monitoring using BigQuery which is provided from Google Storage. And we are going to carry out quantitative measurement about performance of routing in cloud computing environment. The systems of solving optimization problem which need to large amount of hardware resources such as CPU and memory could be run more efficiently in cloud computing environment.

Acknowledgment

This work was supported by the BK21 Plus (Big Data in Manufacturing and Logistics Systems, Korea University).

References

[1] M. J. An and H. C. Lee, "Design of the Integrated Logistics Information System based on Cloud Computing," *Lecture Notes on Software Engineering*, vol. 3, no. 1, pp. 31-34, February 2015. http://dx.doi.org/10.7763/LNSE.2015.V3.160

[2] *The NIST Definition of Cloud Computing*, NIST Special Publication, SP800-145-2011

[3] *Cloud Computing: Defining and Describing an Emerging Phenomenon*, Gartner, Inc., Stamford, CT., 2008.

[4] G. Reese, *Cloud Application Architectures*, Sebastopol, CA: O'Reilly, 2009, pp. 1-46.

[5] R. E. Neapolitan and K. Naimipour, *Foundations of Algorithms using C++ Pseudocode*, 3rd ed. New York, U.S.A.: Jones and Bartlett, 2004, ch. 3, pp. 91-133.

[6] L. Gilbert, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 52, no. 2, pp. 570-578, July 1993.

[7] Y. Dumas, J. Desrosiers, E. Gelinas and M. M. Solomon, "An optimal algorithm for the traveling salesman problem with time windows," *Operational Research*, vol. 43, no. 2, pp. 367-371, Apr. 1995. http://dx.doi.org/10.1287/opre.43.2.367

[8] A. Langevin, F. Soumis and J. Desrosiers, "Classification of travelling salesman problem formulations," *Operations Research Letters*, vol. 9, no. 2, pp. 127-132, Mar. 1990. http://dx.doi.org/10.1016/0167-6377(90)90052-7

[9] G. B. Dantzig, D. R. Fulkerson and S. M. Johnson, "On a linear-programming, combinatorial approach to the traveling-salesman problem," *Operations Research*, vol. 7, no. 1, pp. 58-66, Feb. 1959. http://dx.doi.org/10.1287/opre.7.1.58

[10] S. Chatterjee, C. Carrera and L. A. Lynch, "Genetic algorithms and traveling salesman problems," *European journal of operational research*, vol. 93, no. 3, pp. 490-510, Sep. 1996. http://dx.doi.org/10.1016/0377-2217(95)00077-1

[11] M. Dorigo and L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53-66, Apr. 1997. http://dx.doi.org/10.1109/4235.585892

[12] P. P. Yip and Y. H. Pao, "Combinatorial optimization with use of guided evolutionary simulated annealing," *Neural Networks, IEEE Transactions on*, vol. 6, no. 2, pp. 290-295, Mar. 1995. http://dx.doi.org/10.1109/72.363466

[13] J. Knox, "Tabu search performance on the symmetric traveling salesman problem," *Computers & Operations Research*, vol. 21, no. 8, pp. 867-876, Oct. 1994. http://dx.doi.org/10.1016/0305-0548(94)90016-7

[14] C. K. Looi, "Neural network methods in combinatorial optimization. Computers & Operations Research," *Computers & Operations Research*, vol. 19, no. 3, pp. 191-208, May 1992. http://dx.doi.org/10.1016/0305-0548(92)90044-6

[15] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," Omega, vol. 34, no. 3, pp. 209-219, June 2006. http://dx.doi.org/10.1016/j.omega.2004.10.004

[16] S. S. Skiena, Dynamic Programming, 2nd ed. London, U.K.: Springer-Verlag, 2008, ch. 8, pp. 273-315.

[17] R. Bellman, "Dynamic programming treatment of the travelling salesman problem," Journal of the ACM, vol. 9, no. 1, pp. 61-63, Jan. 1962. http://dx.doi.org/10.1145/321105.321111

Minjeong An was born in Seoul, Korea. She obtained the Master Degree in School of Industrial Management Engineering at Korea University, Korea in 2008. She is currently a Ph.D. Candidate in School of Industrial Management Engineering at Korea University, Korea. Her research interests are logistics information system and cloud computing architecture.

Minyoung Lee was born in Seoul, Korea. He is currently a Master & Ph.D. student in the School of Industrial Engineering at Korea University, Korea since 2014. His research interests are logistics information system and cloud computing architecture.

Hongchul Lee was born in Seoul, Korea. He obtained the Ph.D. in School of Industrial Engineering at Texas A&M University, USA in 1993. He is currently working as a professor in the School of Industrial Management Engineering at Korea University, Korea since 1996. His research interests are production information system and logistics information system and Supply Chain Management (SCM).