Fastest Route Search for Large Scale Traffic-Data Graph on Giraph Using Bidirectional Dijkstra Algorithm with Potential Value Forecasting Awareness

코시아완 요하네스, 권준호†, 홍봉희
부산대학교 컴퓨터공학과
{khosiawan,jhkwon,bhhong}@pusan.ac.kr

Yohanes Khosiawan, Joonho Kwon†, and Bonghee Hong
Department of Computer Engineering, Pusan National University
†Institute of Logistics Information Technology, Pusan National University

Abstract

The traffic management issue on megapolitan regions keeps arising and demanding a more effective solution. Building more roads or highways is not a solution anymore; public transportation is becoming a later option since the increasing welfare of the citizens makes their prestige in having and using their own personal vehicles greater. With this condition, the traffic density on the roads is increasing dramatically, giving the equal dramatic increase to the amount of data needed to be processed. In this paper, we propose to use modified Bidirectional Dijkstra Algorithm for performing the fastest route search on Giraph framework, working seamlessly with HBase on Hadoop system.

1. Introduction

South Korea and Japan, as the leading countries in smart transportation system, are pouring much money into their transportation research and implementation. South Korea allocates their funds for its transportation sector up to $230 million annually until 2020 [1]. In accordance with this great passion, in Seoul, there is even a specific program called “Smart Seoul 2015” [2]. One of this program’s goals is to provide traffic information in real time such as the vehicle density in particular roads.

Not just in Seoul, but this vision has or is going to be followed by other areas such as Ulsan, Busan, and other areas throughout the country; which may lead to nationally integrated transportation system in the future. Furthermore, in Korea, there are more than 30 million smart phone and 4 million tablet PC users now [2]. To handle that enormous amount of requests and complex traffic data computation in real time, we need an effective large scale data processing system. By utilizing the traffic data, we can provide a service that helps a driver to find the fastest route to the destination place. This traffic data is represented as graph, containing vertices (places or intersections) and edges (roads), and processed using graph search algorithm. Nowadays, so many parties use Hadoop to process big data, but it is not efficient to process graph data. In this paper, we are going to use Giraph [3] (which is a framework, like MapReduce framework, specialized for processing graph data) with HBase [4] on Hadoop [5]. In Giraph, we will use Bidirectional Search [7] (Bidirectional Dijkstra [6]) algorithm with some modification to search the fastest route from the traffic information graph. The proposed modification is on the weight calculation; the weight property contains *length* and *density* attributes. The *length* attribute’s value represents the length of the edge (road). While the *density* attribute will consist of two values: *current* (integer) and *potential* (*integer* return-value function). The size of the array is equals to the number of

incoming edge(s) of the source node. The *current* value refers to the current density level on the particular road, but the *potential* value is calculated based on the incoming road(s). Some related works will be explained in Chapter 2, then the detail about the fastest route search algorithm will be described in Chapter 3, and we will wrap the whole things up into a conclusion at the last chapter.

2. Related Work

In [7], the author describes about Bidirectional Search, also called Bidirectional Dijkstra, which executes Dijkstra algorithm forward from the source node and backward from the destination node. Based on this well-known algorithm, the author will modify the weight calculation so it suits the functional requirement for the proposed system.

The modified algorithm will be implemented using Giraph framework. Giraph is similar to the common MapReduce framework that we used to know, but one of the big differences is: in Giraph, it is not *map* and *reduce*, it is just about *map*; the illustration figure is shown in Figure 1.

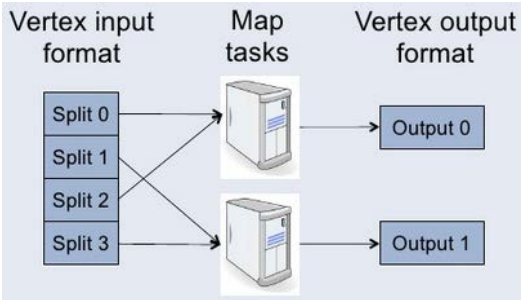

Figure 1: Giraph is a single Map-only job in Hadoop

Giraph would read the data from HBase. HBase is a column-oriented database system built for Hadoop. HBase will receive and store the data to HFile, and these HFiles are typically stored in HDFS; which provide scalable, persistent, and replicated storage layer for HBase [9].

For the field implementation of the sensor networks, we assume system [10] is already well deployed and running. We can get the data including the number of cars, speed, and length of the vehicles.

3. Fastest Route Search Algorithm

Traffic data is changing rapidly every second; it makes the computation of fastest route not a trivial thing. If we keep updating our graph computation every second, it would consume a huge amount of resources. But if we update the graph computation rarely, we will not get the actual and accurate result of the fastest route.

In this paper, the author is thinking about adding forecasting factor into the computation so that we can decrease the frequency of graph computation update, but also could provide a reliable fastest route search result.

The properties involved here are *length* and *density*. We get the length value pretty straightforward from the road's length. But for the *density* property, we need to do some calculation. We must keep the calculation not too complex but effective since we need to think about its affect to the system performance. The *current* density could be obtained by comparing the length of the total vehicles (*vlength*) on a row with the length of the road – regardless the number of rows in the road; since in [10], the data collection is done per row already. And we assume that each road has $1 \leq c \leq 2$ *current* density value (depends on 1-way or 2-ways). The potential density value is obtained from the calculation of the incoming road(s) towards the particular road. The pseudocode of the *potential* density algorithm is as follows:

1. ***D*** = destination node
2. ***p*** = potential incoming *vlength*
3. ***o*** = # of outgoing edges of a particular node

```
4.  BEGIN
5.  FOR each e_i in edges
6.     p[0]=0; p[1]=0;
7.     E= one of two ends of e_i;
8.     EE= the other end of e_i;
9.     FOR each r in rows
10.       IF r.D==E THEN
11.          p[0]+=r.vlength
12.       ELSE p[1]+=r.vlength;
13.       ENDIF
14.    ENDFOR
15.    p[0]/=E.o;    p[1]/=EE.o;
16.    IF p[0]!=0
17.       FOR each e in E.edges
18.         FOR each r in e.rows
19.            r.potential[i] ← p[0]/e.length
20.         ENDFOR
21.       ENDFOR
22. ENDIF
23.    IF p[1]!=0
24.       FOR each e in EE.edges
25.         FOR each r in e.rows
26.            r.potential[i] ← p[1]/e.length
27.         ENDFOR
28.       ENDFOR
29.    ENDIF
30. ENDFOR
31. END
```

4. Conclusion

As the big data issue arises, many people choose Hadoop as their solution. In fact, Hadoop cannot efficiently handle all types of input data. For handling graph data, there is a Hadoop-based framework called Giraph. Using the well known powerful bidirectional search, armed with the *potential* density forecasting awareness, the author expect an accurate and reliable fastest route search result during the implementation.

Acknowledgement

This research is supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2012-(H0301-12-4014)) supervised by the NIPA (National IT Industry Promotion Agency).

5. Reference

[1] CNN. South Korea and Japan Streets ahead in smart transport. <http://articles.cnn.com/2010-04-08/tech/urban.smart.transport_1_south-korea-seoul-bus-carpool-lane> accessed on October 12, 2012 - 17:08 KST.

[2] Smart Seoul 2015. <http://english.seoul.go.kr/library/common/download.php?fileDir=/community/&fileName=SMART_SEOUL_2015_4.pdf> accessed on October 12, 2012 - 10:11 KST.

[3] Apache Giraph. <http://giraph.apache.org/> last accessed on October 13, 2012 - 14:17 KST.

[4] Apache HBase. <http://hbase.apache.org/> last accessed on October 13, 2012 - 14:20 KST.

[5] Apache Hadoop. <http://hadoop.apache.org/> last accessed on October 13, 2012 -14:23 KST.

[6] A. V. Goldberg, C. Harrelson, H. Kaplan, R. F. Werneck. Efficient Point-to-Point Shortest Path Algorithms. <http://www.cs.princeton.edu/courses/archive/spr06/cos423/Handouts/EPP%20shortest%20path%20algorithms.pdf> last accessed on October 13, 2012 - 20:00 KST.

[7] D. De Champeaux. Bidirectional Heuristic Search Again, Journal of the ACM, Volume 30 Issue 1, Jan. 1983, pp. 22-32.

[8] A. Ching. Processing edges on Apache Giraph. < http://www.slideshare.net/Hadoop_Summit/processing-edges-on-apache-giraph>, p11, last accessed on October 13, 2012 - 20:14 KST.

[9] L. George. HBase: The Definitive Guide. O'Reilly, 2011.

[10] S. Coleri, S. Y. Cheung, P. Varaiya. Sensor Networks for Monitoring Traffic. Allerton Conference on Communication, Control and Computing, 2004.