MOIR/MT: Monitoring Large-Scale Road Network Traffic in Real-Time

Kuien Liu
Institute of Software, Chinese Academy of Sciences, China
keliu@itechs.iscas.ac.cn

Ke Deng
The University of Queensland, Brisbane, Australia
dengke@itee.uq.edu.au

Zhiming Ding
Institute of Software, Chinese Academy of Sciences, China
zhiming@ercist.iscas.ac.cn

Mingshu Li
Institute of Software, Chinese Academy of Sciences, China
mingshu@iscas.ac.cn

Xiaofang Zhou
The University of Queensland, Brisbane, Australia
zxf@itee.uq.edu.au

ABSTRACT

Floating Car Data (FCD) provides an economic complement to infrastructure-based traffic monitoring systems. Based on our previous MOIR platform [5], we use FCD as the data source for large-scale real-time traffic monitoring. This new function brings a challenge of efficiently handling of streaming data from a very large number of moving objects. Server overload problems can occur when a system fails to process data and queries in real-tme, which can lead to critical issues such as unbounded delay accumulation, lost monitoring accuracy or lack of spontaneity. These problems can be addressed by adopting suitable load dropping decisions. In this work, we demonstrate several load shedding techniques, focusing on decision-making based on data attributes. With the end results being quantified and visualized using real data for a large city, this proof-of-concept system provides a convincing way of validating our ideas.

1. INTRODUCTION

Rapidly increasing adoption of in-vehicle positioning and communication technologies (e.g. GPS and GSM) enables efficient detection and transmission of real-time traffic information, such as traveling speed, direction and the current position. Such vehicles can be used as traffic speed sensors that continuously generate information. This provides us a new source of data, known as *Floating Car Data* (FCD)[1], to overcome the reliance on costly infrastructure-based measurement devices, such as road sensors, cameras, radars and loop detectors. With the continual collection of traffic updating data from moving vehicles, the current traffic status for the entire road network can be derived, instead of just major roads where roadside monitoring hardware are installed. However, FCD based monitoring applications are challenged by voluminous data and queries they have to deal with, and also by the dynamic change in data rates.

In traffic monitoring applications, one can imagine that a continuous query is issued to monitor the traffic situation of each road segment in a monitoring area, which can be the entire city. When FCD are reported, each FCD records need to be mapped to the road network and the query results for the affect roads need to be updated. When a large number of FCD records flow in which exceed the processing capability of the system, the processing delay will increase and the value of data to real-time monitoring will decrease over time. Since timeliness is a critical measure of Quality of Service (QoS) for real time applications, a fraction of FCD data have to be dropped, possibly at the cost of reduced monitoring accuracy, which is defined as how close the query results are comparing to the real situation on the ground. So the research question is how to decide how much and which FCD records should be dropped in order to balance monitoring accuracy and timeliness.

Data stream load shedding is a problem that has been investigated recently ([1, 8]) due to the practical importance in several application domains including stock market monitoring. However, the existing methods fail to effectively handle our problem of traffic monitoring for road networks. In [8], for a given set of queries which form a query network, some queries can be strategically dropped based on their processing cost and the load shedding objective. The queries with the highest costs are dropped gradually until the load shedding objective is achieved. This method cannot handle our problem effectively even though our problem aims to achieve the same load shedding objective. The reason is that our optimization focus is to achieve the highest possible accuracy while their focus is to maximize the number of queries processed. Dropping the high cost queries cannot help to achieve high accuracy. Moreover, the queries in our problem have very similar processing costs.

[1]Some works are based on the assumption of using Advanced FCD (AFCD) in which data can be processed and matched to the underlying road network. AFCD requires considerable extra on-board capability for storage and processing, incurring higher in-vehicle hardware cost and data schema related issues for universal applications. The spirit of our work, however, is applicable to AFCD.

In [1], a method based on a feedback loop to control the load shedding is proposed. It assumes that the data arrives at similar rates. The load shedding objective in this case is decided by the processing power and incoming streaming data. Once the load shedding objective is known, the load shedding is distributed to queries. If a query has a lower accuracy based on previous data, the sampling rate of this query will be higher for the coming streaming data. Otherwise, it should be lower. The problem is that it works only in the situation when the rate of incoming data is steady. In our application, the query accuracy is not necessarily related to the rate of shedding. For example, a large number of FCD in road A report the same speed while a small number of FCD in road B report very different speeds. It is much more important for our system to process more data in B than in A. That is, the speed change is not related to the number of incoming data processed, so it is impossible to estimate the accuracy using their method. Speed changes play a more important role here in making any decision for load shedding; thus pre-processing of incoming data is needed to detect speed changes. Furthermore, their accuracy optimization is based on the statistics from the previous time frame. If new data have different characteristics, such statistics may not be suitable for new data. However, the adoption of a pre-processing step can help to generate a summary of newly arrived data. This up-to-date summary provides better information for system performance optimization.

Inspired by the above observations, we propose to use speed changes over a road as an importance parameter for load shedding decision-making. An update of traffic situation in a road is only triggered when the speed of this road changes. Two main goals of our demo are: 1) to develop a FCD-based Data Stream Management System (DSMS) for traffic information processing and traffic situation visualization capable of supporting large-scale road networks; and 2) to implement our new load shedding method to handle large scale FCD stream and compare with other load shedding methods. An overly complicated shedding object selection process means more time spent on selecting data items in order to remove them. Therefore, our goal in this work is to design a light-weight load shedding framework that can make complex load shedding decision efficiently.

2. OVERVIEW OF MOIR/MT

This section gives an overview of the technical aspects of our demonstration. Firstly, we briefly review the MOIR platform [5]. Then, we summarize the architecture of MOIR/MT and introduce the load shedding mechanism focusing on feature labeling and load adapting.

2.1 MOIR Platform

MOIR[5] is an on-going project conducted in the Joint Lab on Data and Software Engineering between Institute of Software, Chinese Academy of Sciences (ISCAS) and The University of Queensland. It is designed as an integrated platform for implementing and experimenting new kinds of applications and algorithms for large scale moving objects, especially to support spatiotemporal data indexing and query[3, 4], trajectory predication[6], convoy discovery[7], and road network constrained applications[2]. MOIR runs in the context of detailed digital road maps with 38,000 road segments and 55,000 road intersection points, and real spatiotemporal data of over ten thousand taxis in Beijing.

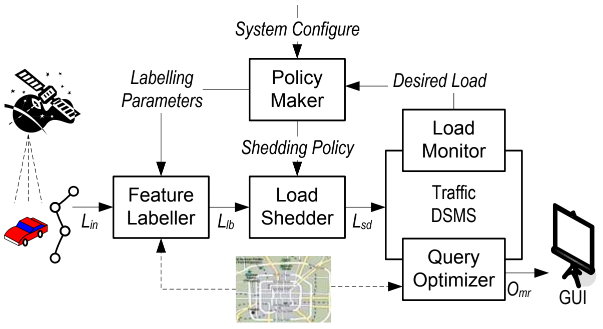


Figure 1: Data flow of MOIR/MT

MOIR facilitates two essential conveniences which motivate us to choose it as our demo infrastructure. First, its flexible architecture allows us to extend it with new functions easily. Second, it also uses large-scale practical floating car data as the data source. Taxis' long running trajectories and extensive coverage make them a proper source of traffic data. It does not involve any privacy issues.

2.2 MOIR/MT Architecture

Figure 1 illustrates the data flow of MOIR/MT framework, in which the entire monitoring task can be divided into two parts: traffic DSMS and incoming data stream preprocessing. These two parts together deal with load fluctuation gracefully and provide approximate query results to users when overloading occurs. If the traffic monitoring system is underloaded, the FCD stream L_{in} will be fully inputted into *Traffic DSMS*, then queries can be answered accurately. On the other hand, if overloading is detected, *Policy Maker* will generate load labelling parameters and shedding policies, and dispatch them to *Feature Labeler* and *Load Shedder* respectively. Then *Load Shedder* will be activated and starts to discard L_{lb} to a desired load level L_{sd}. The load shedding decision-making is based on the features of FCD tuples which are labeled on in *Feature Labeler*. In the *Traffic DSMS* part of our system, *Load Monitor* and *Query Optimizer* will contribute to our monitoring task. *Load Monitor* keeps watching system load continuously and provide information for *Policy Maker* to make load shedding decision. *Query Optimizer* is used to optimize query plan in DSMS. As we concentrate on the first part, we will only discuss the load shedding process in details here.

2.3 Feature Labeling

When the system is heavily loaded, it is preferable to shed some less important load rather than to shed without discrimination. In many applications, users are more interested in the changes of monitoring objects. By connecting the importance of data and changes of speeds in our application, we propose to use speed changes as our importance coefficient. We assign a speed value to every road. This value is averaged from all cars' speed reported in the near term on a road. Since each FCD tuple has a speed value, the difference between this value and the road's speed is defined as the speed change ΔS. For example, a road's last speed is $70km/h$, and all the new tuples on this road have speeds around $10km/h$, so all ΔS of these tuples are around $60km/h$. It is highly possible that there are some abnormal situations which we have to pay attention to. On the contrary, if a road's speed is $90km/h$ and all tuples have speeds around this speed, update does not need to be triggered and we can ignore this road for the current update. A greater change will be more important. The *Feature Labeler* will locate a FCD tuple onto road, then compute ΔS to append it to this FCD tuple.

When there are multiple ΔS on the same road, the largest one will be added into every FCD tuple on that road. The reason is that there are always multiple tuples on a road in a time period and their ΔS may be different. The result is that a road will have several importance values. Some tuples with smaller ΔS will be dropped, so the speed calculation will be performed over remaining value with higher ΔS. This result can be very different from the one over all tuples. In order to avoid multiple importance values appearing on the same road, we use the highest ΔS to label all the tuple on a road. In this way those important tuples can be detected.

2.4 Load Adapting

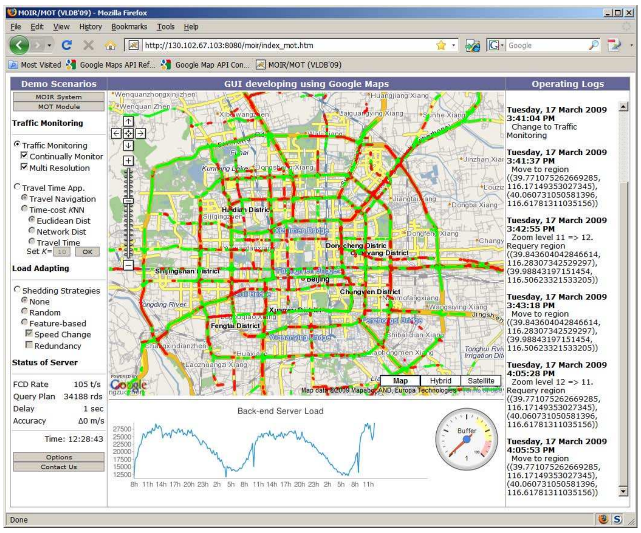
FCD is generated by vehicles, and sent to the traffic monitoring center. Each tuple will trigger an update command to refresh the traffic status of corresponding roads, leading to processing of queries on the road network. However, not all queries need to be triggered. In fact, the data stream arrives in a bursty way depending on many factors, such as stability of wireless communication, sampling cycles configured by hardware vendors, and travel patterns of citizens etc. If the arriving rate exceeds the capacity of the system, performance degrade is unavoidable. Within our system, overloading will be handled gracefully, such that the applications built upon traffic monitoring module will gain approximate answers rather than being suspended. *Load Shedder* will refer to information(namely location, speed change and redundancy) added by *Feature Labeler* to utilize shedding policies.

The amount of load that should be shed is decided by *Policy Maker*. *Load Monitor* estimates the cost for all the queries in DSMS. Suppose the total cost for queries is L and the system capacity is C, then the excessive load is $C_{exc} = L - C$. To overcome the overload situation, data tuples in some queries must be dropped so the load can be remained within the system capacity. Our strategy is to drop the least important tuples so that the cost of remaining queries is smaller than or equal to C.

As we have the information about the cost and importance for all the queries, the queries can be dropped from the least important ones upwards until the excessive load disappears. Suppose there are k queries $q_1, q_2, ..., q_j, ..., q_k$ and the cost for each tuple is c_j, all the data tuples corresponding to them will be dropped. The data rate on these queries is r_j. We have the equation:

$$C_{exc} \leq \sum_{j=1}^{k} r_j \times c_j$$

According to the last dropped query, we can obtain the ΔS of it and we call this value threshold δS. Any data in a query with ΔS lower than the threshold will be dropped. In other words, the load can be controlled by adjusting the threshold. Configuration from system manager can be implemented in this way. Alternatively, it can be done through allocating compulsory mark to corresponding data, so those queries triggered by these data can be exempted from load shedding.



Traffic Monitoring

(●) Traffic Monitoring
[✓] Continually Monitor
[✓] Multi Resolution

( ) Travel Time App.
(●) Travel Navigation
( ) Time-cost *K*NN
(●) Euclidean Dist
( ) Network Dist
( ) Travel Time
Set *K*= 10 OK

Load Adapting

( ) Shedding Strategies
(●) None
( ) Random
( ) Feature-based
[✓] Speed Change
[ ] Redundancy

Status of Server

FCD Rate	105 t/s
Query Plan	34188 rds
Delay	1 sec
Accuracy	Δ0 m/s

Time: 12:28:43

Options
Contact Us

```
Tuesday, 17 March 2009
3:41:04 PM
 Change to Traffic
Monitoring

Tuesday, 17 March 2009
3:41:37 PM
 Move to region
((39.771075262669285,
116.17149353027345),
(40.060731050581396,
116.61781311035156))

Tuesday, 17 March 2009
3:42:55 PM
 Zoom level 11 => 12.
Requery region
((39.843604042846614,
116.28307342529297),
(39.98843197151454,
116.50623321533205))

Tuesday, 17 March 2009
3:43:18 PM
 Move to region
((39.843604042846614,
116.28307342529297),
(39.98843197151454,
116.50623321533205))

Tuesday, 17 March 2009
4:05:28 PM
 Zoom level 12 => 11.
Requery region
((39.771075262669285,
116.17149353027345),
(40.060731050581396,
116.61781311035156))

Tuesday, 17 March 2009
4:05:53 PM
 Move to region
((39.771075262669285,
116.17149353027345),
(40.060731050581396,
116.61781311035156))
```

Figure 2: The interface of MOIR/MT

3. DEMONSTRATION

MOIR/MT is implemented as an extension module to MOIR [5], whose kernel data engine is extended with traffic information processing and load shedding capabilities implemented in C++. We build a data feeder to inject the real data set from taxis in the city of Beijing into the system, and the data rates can be adjusted according to our requirement. In our demo, we will firstly implement the basic idea of using FCD in traffic monitoring by animated traffic visualization. And then, a more interesting and challenging function is brought forward to deal with fluctuating load with different load shedding methods. Finally, we present two traffic based applications to show the feasibility of importing our techniques into practical scenarios.

3.1 Basic FCD based Traffic Monitoring

The traffic situation on road network will be continually updated when FCD tuples are injected into system. With front-end user graphic interface (GUI), users can look up traffic information in the region of interest (e.g., by dragging and dropping region marks to select a region). When the traffic status changes, the color of the corresponding road lines will change continually. As there are limited memory to save history traffic data in DSMS system, and on some roads there are no FCD be reported in certain time period, the speed of those roads can be unknown for the current time. By referring to traffic information of surrounding roads to fill up the speed of unknown roads, users can gain an approximated but smooth display of the whole road network.

In many cases different resolutions of traffic querying are required. For example, a traffic manager may zoom in to show traffic jam spots or zoom out to glance at the overall city traffic using the front-end map view; at the same time a car driver may query the traffic status in the area around. A layered indexing method is introduced to support multi-resolution query. It organizes all the roads into a multi-layer structures according to road's attributes like width, length and etc. Each layer is assigned to a resolution level. When a query is submitted to the back-end, its resolution level will be obtained and only roads in relevant layer will be monitored while other roads will be ignored; thus the consumption of system resources is reduced.

To fasten the visualization on Web-based front-end, several advanced methods are adopted in our implementation, such as geographical data encoding and drawing technologies to reduce transmission costs, graphic object caching and reusing principle to reduce visualization response time.

3.2 Load Shedding Strategies Comparison

The amount of input load and the occupancy rate of buffer are displayed on the system to give a clear indication about when the load shedding is triggered and how efficient it is. Indicators such as data delay, loss rate, response time, communication cost and server load are provided to give a quantitative view.

Our system is able to show the performance difference of applying various load shedding methods. One example is shown in the following figure 3, where the map on the upside uses the strategy of random dropping and the one below uses our dropping policy. Black spots show incorrect query results with errors greater than $20km/h$ when 10% data are shed. Superiority of our method can be seen clearly.

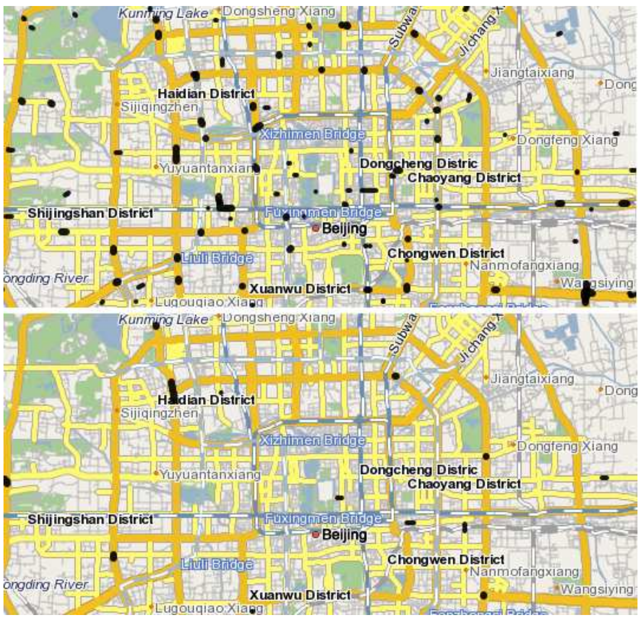


Figure 3: Error distribution in different load shedding strategies (upper subfigure: random drop; lower subfigure: feature-based drop)

By observing our results, we find that FCD have a high level of redundancy, which means parts of data provide overlapping traffic information such as similar speed values and locations. If we drop some of redundant data, it will not bring in unbounded errors but can still save time in computation. The impact of data redundancy can be observed using our system.

3.3 Examples of Traffic-related Applications

MOIR/MT will show two advanced applications based on real-time traffic situation. One is (continual) kNN query ($k = 20$ in left of figure 4). Neighbors' distance can be measured in three different ways, i.e. Euclidean distance, road network distance and travel time distance. The other application is the shortest time routing. Given two positions on the map, route costing shortest time (i.e., dash line in figure 4) between these two positions will be acquired, as well as the shortest path using network distance (i.e., dot line in figure 4). Referring to the example of figure 4, these two pathes are quite different. Although we use a simple method, which is totally based on the current road situation, to estimate the travel time, other more advanced methods can still be used in our system.

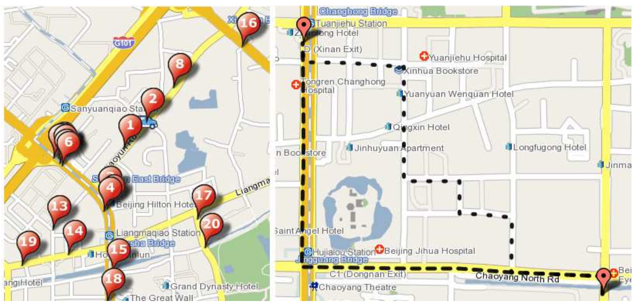


Figure 4: Examples of kNN and shortest path query

4. CONCLUSIONS

In this demo, we have developed a load adaptive framework to process and monitor large scale traffic data stream in real time. Experiences with our prototype indicate that our proposal of utilizing semantic relations between traffic data and road network to improve query accuracy when server overloading exists is a light-weight yet powerful technique. In the future, we will consider enlarging the load monitoring scope, and enriching the semantic concepts of data labeling from processing costs to query similarity.

5. REFERENCES

[1] B. Babcock, M. Datar, and R. Motwani. Load shedding for aggregation queries over data streams. In *ICDE'04*.

[2] Z. Chen, H. T. Shen, X. Zhou, and J. X. Yu. Monitoring path nearest neighbor in road networks. In *SIGMOD'09*.

[3] K. Deng, X. Zhou, and H. T. Shen. Multi-source skyline query processing in road networks. In *ICDE'07*.

[4] K. Deng, X. Zhou, H. T. Shen, S. Sadiq, and X. Li. Instance optimal query processing in spatial networks. *The VLDB Journal*, 18(3):675–693, June 2009.

[5] Z. Ding, L. Guo, K. Liu, H. Wu, and X. Zhou. Moir: A prototype for managing moving objects in road networks. In *MDM'08*.

[6] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou. A hybrid prediction model for moving objects. In *ICDE'08*.

[7] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of convoys in trajectory databases. In *VLDB'08*.

[8] N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In *VLDB'03*.