Travel Time Estimation Using NiagaraST and latte

Kristin Tufte[1,2], Jin Li[1], David Maier[1], Vassilis Papadimos[1], Robert L. Bertini[2], James Rucker[1]

[1]Department of Computer Science
Maseeh College of Engineering & Computer Science
Portland State University
Portland, Oregon

[2]Department of Civil and Environmental Engineering
Maseeh College of Engineering & Computer Science
Portland State University
Portland Oregon

{tufte, li, vpapad, maier, jgrucker}@cs.pdx.edu, bertini@pdx.edu

ABSTRACT

To address increasing traffic congestion and its associated consequences, traffic managers are turning to intelligent transportation management. The latte project is extending data stream technology to handle queries that combine live streams with large data archives, motivated by needs in the Intelligent Transportation Systems (ITS) domain. In particular, we focus on queries that combine live data streams with large data archives. We demonstrate such stream-archive queries via the travel-time estimation problem. The demonstration uses the new latte system which has been developed using the NiagaraST stream processing system and the PORTAL transportation data archive.

Categories and Subject Descriptors
H.2.4. Query Processing

General Terms: Performance, Design

Keywords: Data Stream Management Systems, Hybrid Queries, Stream-Archive Queries

1. INTRODUCTION

Traffic congestion, and the associated delay and economic costs it causes, are a source of significant concern. In the United States over the past twenty years, vehicle miles traveled for passenger cars grew 44%, but miles of interstate highway increased less than 8%! In response, transportation departments are moving towards intelligent transportation management. Much of the data available for use in intelligent transportation management is in the form of data streams, such as inductive loop detector data, Automatic Vehicle Location (AVL) systems on buses, and live traffic signal data. Further, many metropolitan areas, including Portland, Oregon, are storing these data streams, creating large transportation data archives. While the goal of the latte project is to apply data stream management technology to Intelligent Transportation System (ITS) applications, we are developing general technology for efficient execution of queries combining live data streams and archived data – *stream-archive queries* as we call them. The basis for latte is the NiagaraST stream-processing system.

Computing real-time transportation metrics is an important piece of intelligent transportation management. In particular, the Federal Highway Administration (FHWA) has put a high priority on municipalities providing travel time information. The latte project has access to the live stream of freeway data that PORTAL receives from the Oregon Department of Transportation (ODOT) as well as the PORTAL transportation data archive, which has been archiving ODOT freeway detector data as well as other transportation-related data for over two years [1]. In our demonstration, we show how stream-archive queries can be used to improve travel-time estimation for the Portland region. The stream-archive queries run over a re-played stream of ODOT freeway data and data form the PORTAL archive. The demonstration will show the execution of stream-archive queries, will display travel-time estimates and the expected variability in those estimates, and will visually depict how different methods of accessing the data archived affect data access patterns in the archive and the accuracy of answers.

2. STREAM-ARCHIVE QUERIES

Data-stream researchers have developed technology for efficiently processing queries over data streams – for executing window queries, for handling bursts and lulls in the inputs and so on; however, limited attention has been given to combining data from large data archives with live streams. (We note that accessing data from an archive is a different problem from conveying data to an archive.) We begin by discussing the travel-time-estimation problem and then discuss issues involved in effectively accessing a large data archive.

2.1 Travel-Time Estimation

Accurate travel-time estimates have shown value for drivers in cities around the United States. Travel time estimates are provided by transportation departments to drivers via Variable Message Signs on highways as well as via the Internet and telephone services. In addition to current estimated travel times, it is also important to provide travel-time reliability. For example, if one wishes to be sure to arrive at a destination on time, knowing that travel time is less than 15 minutes with a 90% probability is perhaps more important than knowing that the expected travel time is 10 minutes. Current travel-time estimates are best done using a combination of live and archived data; further historical data is required for estimating travel-time reliability.

2.2 Corresponding Periods

To estimate travel times and their reliability, we wish to compare and combine live data with archived data from "similar" time periods in the past. While this demonstration focuses on travel-time estimation, we believe that the concept of comparing current data to "similar" historical data is applicable to many applications. As a simple example, we might estimate travel times by using a combination of today's data and data from the previous ten Tuesdays (assuming today is a Tuesday). In this case, we define similar as "the past ten Tuesdays"; however, many other definitions of similarity may be applicable: same time of year, same weather conditions, same traffic conditions. We define a *corresponding period* to be a past period that is similar (by some definition) to the current time period. An important aspect of similarity is that it can depend on current data values in the live stream. That is, we will not know statically in advance what we want to retrieve out of the archive – rather, we must examine the current data to determine which pieces of the archive are relevant. We consider below some likely notions of similarity that impose challenges for retrieving archive data for corresponding periods.

Structural Similarity: A structural definition of similarity is based on some aspect that correlates to the archival data organization, in our case, likely by time or road segment. Thus similarity definitions based on some offset in time – same day of week, same month last year – will result in corresponding periods that are contiguous spans at predetermined intervals in the archive.

Metric-Based Similarity: For near-term predictions, we may want to find periods that are similar under metrics suggested by traffic-flow theory. For example, we might judge two time periods to be similar based on the demand—measured in Vehicle Miles Traveled (VMT) in a road segment—or on the time since the onset of congestion (characterized by the combination of vehicle density and speed).

Similarity Based on External Conditions and Events: Some notions of similarity may not depend on directly on the content of data in the period of interest, but rather on associated conditions or events. For example, we might want periods similar in terms of weather conditions, level of daylight or incident occurrence.

2.3 Archive Access

The definitions of corresponding periods above need different types of archive access. Structural similarity requires fairly regular and predictable (though dynamic) access to a set of data. Metric-based similarity and similarity based on external conditions have less regular access patterns. If we expect to support many stream-archive queries at once, the archive access must be efficient.

An obvious way to execute stream-archive queries is to issue a query to the archive for every tuple or group of tuples that arrives on the stream. Such a strategy may be successful for estimating a single travel time for a single route, but is not expected to scale. In the demonstration, we will visually show the effect of different definitions of similarity on archive access.

Porthole Scans: We have developed the "porthole scan" approach to support archive window scans of corresponding periods, initially based on structural similarity.

We explain with an example—consider a query that compares windowed vehicle counts over live traffic data to average window counts of the same time and day over the past four weeks to determine how much better or worse traffic is than usual. Say today is Friday. To execute this query, we open one "porthole" (scan point in the archive) for each of the most recent four Fridays. Each porthole produces a sequence of windowed counts that are averaged and then joined with windowed counts from the live stream. Note that the "stream" of average window counts from the archive must be synchronized with the stream of windowed counts from the live stream.

Using our execution strategy, a stream-archive query is divided into a porthole scan plus a stream query that combines the results with the live data stream. Among the issues we are addressing: deriving appropriate porthole scans from a stream-archive query, dividing query processing between porthole sub-queries and the stream query, architecting and optimizing porthole queries to minimize resource usage while producing data at a specific rate, and synchronizing porthole queries with the live streams.

Persistent Panes: It is infeasible in general to pre-compute and store aggregates over corresponding periods to support archive window scans, if different queries are using a different definition or number of similar periods. However, we can pre-compute and store sub-aggregates ("persistent panes") over the archive that can support several different window archive scans. While this approach reduces computational costs, it does add complexity. For example, we must deduce if and how an aggregate can be constructed from persistent panes.

3. SYSTEM DESCRIPTION

The latte system combines the NiagaraST stream processing system and the PORTAL data archive.

NiagaraST: NiagaraST is a stream query engine that extends the Niagara Internet query system developed at the University of Wisconsin[3]. NiagaraST inherits Niagara's system architecture and query execution model, extending it to support stream processing by introducing punctuations and windowed operators. Punctuations [4] unblock blocking operators and limit the state that stateful operators must maintain. Windowed operators in NiagaraST use our WID window semantics [2]. NiagaraST does not require that stream order be enforced or maintained, but rather leverages punctuation in query operators to track stream progress. We term this approach *out-of-order processing* (OOP). We have shown that OOP outperforms the standard in-order approach in terms of execution time, memory usage and latency [2].

PORTAL: PORTAL (Portland ORegon Transportation Archive Listing) [1] is the official archive of transportation data for the Portland metropolitan region. PORTAL receives a live stream of data from ODOT, with speed, count and occupancy data for freeways around Portland, which it has archived since July 2004. PORTAL also archives weather and traffic-incident data and stores its data in a PostgreSQL database. PORTAL provides a web site that provides traffic metrics for use by transportation planners, managers and researchers.

latte: The latte system combines NiagaraST with the PORTAL archive. We modified NiagaraST so that queries over the PORTAL database can be used as input to a NiagaraST query. Further, NiagaraST connects to the live ODOT data stream, so that it can process queries that combine the live stream and the data archive.

4. DEMO DESCRIPTION

We demonstrate stream-archive query processing in latte using the travel-time-estimation problem. We show dynamic travel-time estimation for a freeway network, including estimates of travel time variability. We show the queries that are used to access the archive and visually demonstrate archive access patterns.

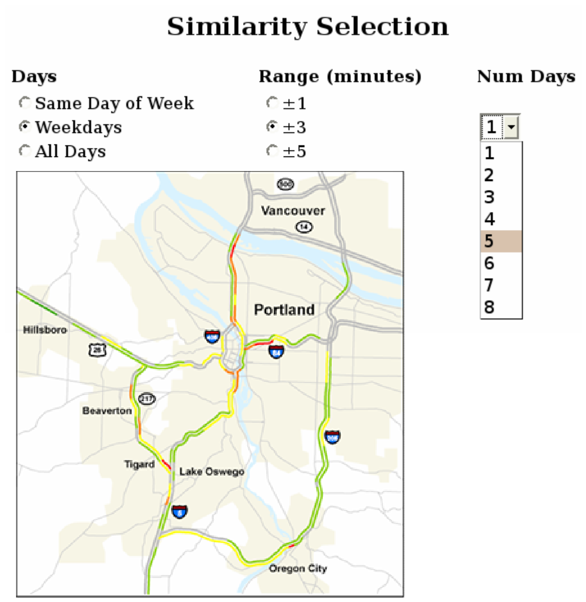

Figure 1 Display of Travel-Time Estimates

Our demonstration uses three types of data: a re-played data stream, the PORTAL database and probe vehicle data. Demonstration times in Beijing will be low traffic times in Portland, so we will re-play a live stream that PORTAL has received from ODOT. Probe vehicle data are records of actual cars' positions as they travel a roadway. By incorporating such data into our demo, we can compare our estimates with the actual travel of the probe vehicle, indicating the accuracy of our predictions.

Our demo has two primary components: display of travel-time estimation and visualization of data access patterns in the archive.

4.1 Display of Time Estimates

Figure 1 displays a similarity selector and a comparison of travel time estimates on the freeway network in the Portland metropolitan region. The selector allows the user to choose between a variety of similarity measures. The map displays a comparison between the current travel times and the travel times estimated using the selected "similar" historical data. In Figure 1, current travel times are being compared with travel times on the five most recent weekdays.

4.2 Visualization of Data Access on the Server

Our demonstration will present a visualization of archive access patterns, to show the effects of different similarity definitions, multiple queries, and different access strategies.

Archived loop-detector data can be visualized using a 2-D grid, with the axes being time and detector location (which correlates with the current physical layout of the data). As data is accessed in the database, squares in the grid are highlighted. Figure 2 shows an example of such a visualization. Each square represents data from a particular sensor over a 5-minute time period. Colored squares indicate recent data access, with darker shades being the most recently accessed. In Figure 1, we show estimates for travel times for multiple highways; this is indicated in Figure 2 by the two different sections of sensors being accessed. Further, sensor access is not necessarily consecutive as sensor numbers may not be consecutive along a route. Finally sensors may be numbered backwards.

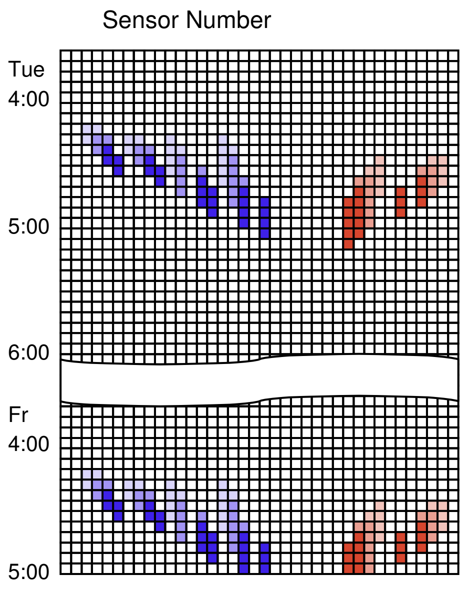

Figure 2 Visualization of Archive Data Access

5. ACKNOWLEDGMENTS

The authors of this paper gratefully acknowledge ODOT for providing us with access to their data and Dennis Mitchell and Jack Marchant, in particular, for their invaluable assistance. This work is supported by NSF Awards (grant number IIS 00-86002 and IIS 06-12311) and NSF CAREER Award (grant number 0236567).

6. REFERENCES

[1] Bertini, R.L., Matthews, S., *et al.* "ITS Archived Data User Service in Portland, Oregon: Now and Into the Future." 8th International IEEE Conference on Intelligent Transportation Systems, Vienna, Austria, September 13-16, 2005.

[2] Li, J., Maier, D., Tufte, K., Papadimos, V. Out-of-Order Processing in the NiagaraST Stream System. http://www.cs.pdx.edu/~jinli/oop.pdf

[3] Naughton, J., DeWitt, D., Maier, D. *et al.* The Niagara Internet Query System. *IEEE Data Engineering Bulletin* 24(2):27-33 (2001)

[4] Tucker, P., Maier, D., *et al.* Exploiting Punctuation Semantics in Continuous Data Streams. *Transactions on Knowledge and Data Engineering*, 15, 3 (May 2003).