A Framework for ITS Data Management in a Smart City Scenario*

Luca Carafoli[1], Federica Mandreoli[1,3], Riccardo Martoglia[1,3] and Wilma Penzo[2,3]
[1]*FIM, University of Modena and Reggio Emilia, Modena, Italy*
[2]*DISI, University of Bologna, Bologna, Italy*
[3] *IEIIT-BO/CNR, Bologna, Italy*
[1]*<name.surname>@unimore.it,* [2]*wilma.penzo@unibo.it*

Keywords: Intelligent Transportation System:Smart City:Relational Databases:Stream Data:Data Reduction Techniques

Abstract: In this paper we introduce a technological framework to efficiently support data management in a modern Intelligent Transportation System (ITS). The proposed technology enables the efficient storage of a variety of recent/historical/static data and guarantees its effective querying by supporting continuous as well as one-time queries for the delivering of real-time traffic services. The framework also offers a scalable solution for coping with the acquisition of huge volumes of data by employing data reduction techniques in Vehicle-to-Infrastructure transmissions. Experimental evaluation on the Linear Road ITS benchmark and along various simulated scenarios demonstrates that the proposed framework efficiently supports smart city data needs.

1 Introduction

The congestion in urban areas is, and will become more and more, a crucial problem of modern society and many investments are going to be made in the direction of the so-called smart cities, whose objectives include the improvement of citizens' life-style by pushing innovation in urban mobility (Streetline Inc., 2011). Smart services, specifically designed to improve people's quality of life by making their lives easier and more efficient, can exploit the recent development of sensing technology to collect a multitude of data from various sources, e.g. vehicles, toll gates, traffic lights, parking meters, weather stations, and others. Nevertheless, they are also expected to access historical and static data, e.g. traffic statistics, street cleaning programme, upcoming exhibitions. For example, by sensing and detecting parking activity in real time, together with information about possible road works or street cleaning programme, a driver would be encouraged to take an alternative means (e.g. bus, metro, bike sharing) knowing that no parking is available, or that it is not allowed, at the desired destination. Similarly, a mobility manager could reinforce dynamically the transportation services on the basis of traffic conditions and of the expectation of large inflows due, for instance, to sporting events. In this perspective, in order to make smart transportation services sustainable, a modern Intelligent Transportation System (ITS) should:

- collect, process and manage both real-time/historical data coming from heterogeneous sources, and static data, in a transparent way;
- provide timely information to be used by advanced services that pursue the city's needs;
- be scalable in order to manage huge amounts of data in the rush hours.

To this end, according to (Cotton, 2011), data management is a primary challenge to be faced. A careful/rough implementation of it makes an ITS effective or not.

In order to cope with large volumes of continuously streaming data in use to common software applications, Data Stream Management Systems (DSMSs) have been introduced (Abadi et. al., 2003; Arasu et al., 2006). These systems natively support continuous queries (CQs) over (continuous unbounded) streams of data according to windows where only the most recent data is retained (Terry et al., 1992). Once data goes out of the windows it is deleted from the system. This model does not completely fit the needs of a modern ITS in a smart city context, where, besides CQs, also one-time queries (OTQs) on a variety of recent/historical/static data have to be supported, e.g., to provide for statistical data and traffic forecasts.

*This work is partially funded by the Italian Council Industria 2015 PEGASUS Project.

In order to fulfill these needs some DSMSs have moved towards integrating DBMS functionalities within their own architectures (e.g. (Abadi et. al., 2003; Arasu et al., 2006)). These approaches have the main drawback of requiring to redesign from scratch a core of well-established DBMS techniques that can not be reused as such in a DSMS architecture. Further, in these solutions (Arasu et al., 2006), according to the DSMS philosophy, historical data necessarily has to be converted into a stream before being queried, thus producing an unbearable overhead for the system.

Other works (Botan et al., 2009) propose two-layered solutions with the DSMS relying on the functionalities provided by a DBMS. The combination of these systems, each designed for specific and opposite goals, does not solve the DBMS inefficiencies in storing/retrieving data at the rate a DSMS requires.

In this paper we introduce a framework that offers data management facilities to make ITS services sustainable in a smart city, along two main directions:

Data Storage and Querying, by providing ITSs with a technological support that enables the efficient storage of a variety of streaming data and its effective querying for the delivering of real-time traffic services. This is achieved through specific extensions to the Transactional Storage Manager (TSM) of a relational DBMS, where a new type of table, called *streaming table*, is introduced;

Data Flow Acquisition, by offering a scalable solution for coping with the gathering of huge volumes of data. Complementary to approaches that aim at system upgrades, and with the goal of fostering data management sustainability, our work aims at selecting information at the sources, by discarding redundant and less relevant data since the acquisition phase (Carafoli et al., 2012).

The paper is organized as follows. Section 2 gives an intuition of the information needs to implement smart traffic services. Section 3 sketches the architecture of the proposed framework. In Section 4 details are given about the Extended DBMS while the experienced data reduction techniques are introduced in Section 5. Section 6 shows results of experimental evaluation and conclusions are drawn in Section 7.

2 A Use Case

The use case we adopt in this paper concerns managing parking availability in a "smart" way, so to contribute to reduce traffic congestion and pollution in cities. It is implemented as an enhanced version of Dynamic Parking Price (DPP) service (Streetline Inc., 2011) that leverages real-time information gathering and analysis. When a driver looks for parking, the service suggests him/her the most convenient available one and promptly computes its hourly cost and the best route to reach it. The most convenient parking is not always the cheapest one. For instance, it could be the easiest to reach according to traffic conditions or the nearest if it is snowing. To rank the parking alternatives, the service exploits information coming from three kinds of sources, parking meters, vehicles and weather stations, together with past driver behavior and base parking prices.

It is worth noting that the DPP service needs to query and process real time data coming from the sources, together with historical, i.e. past driver behavior, and static data, i.e. base parking prices.

Whenever a driver submits a parking request, DPP must retrieve the vehicle's position through location-based services. This is done by means of a traditional SQL-based *one-time query* (OTQ) on the position reports acquired from the vehicles.

Then, it has to calculate the route to the best parking according to the traffic conditions. To obtain them, DPP relies on traffic monitoring services, which work on aggregate information. For instance, it relies on a *continuous query* (CQ) that, every minute, returns the average traveling speeds of each segment.

To compute parking prices, DPP relies on base prices that are increased or decreased by means of multipliers that depend on source status conditions. For instance, in case of a sunny day the weather multiplier would be 1, while in a snowy day it could be lower. In fact, in case of a heavy snowfall, managers could lower the parking prices to encourage people to leave their vehicles and move with alternative means, such as metro or street cars. So DPP needs to join the base prices, that is static information, with the real-time information coming from different sources.

3 Architecture

The data management framework we propose relies on a typical ITS scenario where distributed data sources collect raw data from the real world and send them in the form of data streams to a Control Centre (CC) through GPRS. As to data sources, we assume that cars are equipped with On-Board Units (OBUs), small mobile computing devices that acquire data through GPS and accelerometer units and perform real-time GPRS communications. The CC stores and processes this data to provide end users with various services enhancing mobility.

As shown in Figure 1, the framework acts both at data source level and at CC level. At data source level, in order to reduce communication costs and CC workload, the framework implements data reduc-

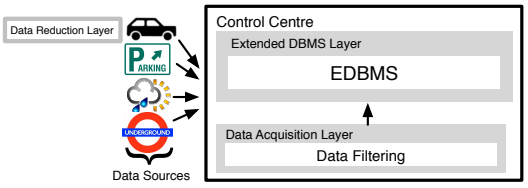

Figure 1: The framework architecture

tion techniques on the OBUs. Thanks to these techniques, OBUs aim to send useful information only, while avoiding redundancy.

The remaining layers are localized in the CC. The Data Acquisition Layer has the goal to convert data coming from heterogeneous sources to common formats. For example, as to geographic coordinates, one possible format is the Universal Transverse Mercator (UTM-NAD83) (NOAA, 2008). In this case, all data coming in the system in different formats, e.g. Degrees, minutes and seconds - WGS84 (NIMA, 2000), needs to be converted.

Finally, the Extended DBMS layer augments a traditional DBMS with the capability of retaining data streams beyond their real-time processing, by managing and storing them transparently as standard data. This is an essential requirement for a modern ITS where many advanced services join real-time data with past trends to derive not only the current situation but also to forecast incoming events.

4 The EDBMS

The EDBMS is in charge of storing data streams coming from sources and to promptly make them available to be queried together with other kinds of data, such as historical data, e.g. traffic statistics, and/or static data, e.g. parking locations.

To this end, besides standard tables, it also supports a new kind of table, called *streaming table*, that stores real-time data. A streaming table, differently from traditional tables, undergoes continuous writes and, besides OTQs, it also supports CQs. The design choice of extending a traditional DBMS towards data stream management benefits of the exploitation of a considerable amount of long-established query processing and data management techniques.

In order to manage streaming table creation, we introduce the SQL-like statement `CREATE STREAMING TABLE`. The main idea is that each data source is linked to a streaming table that stores its data streams. For instance, in our DPP example, a streaming table `VEHICLES` is created to store the position reports received from vehicles:

```
CREATE STREAMING TABLE
VEHICLES(TIME DATE, VID INT, SPD INT,
XWAY INT, LANE INT, DIR INT, SEG INT,
POSITION POINT) RANGE INTERVAL '2'
WEEK.
```

where `RANGE` defines the time period data stays in the database. It is worth noting that the `VEHICLES` table stores geo-temporal data. To this end, the `POSITION` attribute is a `POINT`, which is made up of the coordinates of the vehicles in the map, and the temporal attribute `TIME` is managed with specific methods.

To insert data in streaming tables, we introduce some changes to the standard SQL `INSERT` statement:

```
INSERT INTO VEHICLES VALUES FROM
''VEHICLES.STR''
```

where the `FROM` argument is a stream for *continuous* data insertion. According to the insert semantics, an insert request is required for each data insertion. However, this is incompatible with data streams because it drastically slows down the insertion rate. In our approach, the `INSERT` statement is submitted only once and, then, the TSM exploits ad-hoc mechanisms to speed up continuous stream data storage.

Further, as for traditional tables, indices can be defined to speed up data access. For instance, the SQL command below creates a new index on the attribute `VID` of the `VEHICLES` streaming table:

```
CREATE INDEX IDX_VEHICLES ON
VEHICLES(VID)
```

With reference to the DPP service scenario, Table 1 shows three different kinds of queries that can be submitted on three tables: `VEHICLES`, `WEATHER` and `PARKING`. `VEHICLES` is a streaming table as described previously. `WEATHER` is a streaming table that receives a weather report update from a weather bureau every 10 minutes. The report contains various information, such as weather conditions, temperature and other meteorological measures. `PARKING` is a traditional table that stores general information about parkings, such as the area they are located in (the same field is also stored in the `WEATHER` table), the base price, the number of parking spaces and so on.

It is worth noting that the minimal extensions SQL should undergo to support streaming tables and CQs make the EDBMS a powerful means to design and manage hybrid data-intensive applications.

5 Data Reduction Techniques

In order to support ITS services, the CC receives a large amount of updates, coming in real time from the vehicles' OBUs in the form of continuous streams of GPS-derived time-stamped data. This causes an increase of the communications between the vehicles and the CC and, in general, of the load in all the framework layers.

Q1:	**Q2:**	**Q3:**
SELECT segment, AVG(speed) FROM VEHICLES [WINDOW INTERVAL '1' MINUTE] GROUP BY segment SAMPLE INTERVAL '1' MINUTE	SELECT SUM(w.snow), p.id FROM WEATHER w [WINDOW INTERVAL '30' MINUTE], PARKING p WHERE w.area=p.area AND w.condition= "snow" GROUP BY p.id SAMPLE INTERVAL '10' MINUTE	SELECT position FROM VEHICLES WHERE time=*NOW* AND vid=*VID_X*

Table 1: Q1 is a CQ that calculates the average speed of each road segment. According to the `SAMPLE INTERVAL` statement, it runs every minute and the `WINDOW INTERVAL` statement sets the query's domain to the data arrived in the last minute. Q2 shows the possibility of querying data coming from both streaming and standard tables, joining them as needed. The goal is to have an intuition about snowfall intensity. Q3 is an OTQ on a streaming table returning the current position of a vehicle.

Differently from approaches that aim at system upgrades by employing massive technologies, e.g. parallel processing databases, data grids, etc., we adopt a complementary perspective and foster data management sustainability by employing data reduction techniques to select information at the sources, by discarding redundant and less relevant data since the acquisition phase. The main objective is to reduce the number of data communications, bringing the important benefits of cutting communications costs together with server side update costs. Data reduction techniques are implemented in OBUs and they can be divided in two main categories:

Independent techniques: the OBU autonomously decides when to issue updates to the server. The techniques we consider in this context are:

Simple Sampling: an OBU transmits a data report at a fixed time rate T. This policy has also been considered in other ITS approaches (Xu et al., 2002) and for purposes different from data reduction. Using this policy, the movement of the vehicle is represented as a constant time "jumping vector" that is independent from the actual movements and the road network;

Space Sampling: this policy sends a position report to the CC when the vehicle covers a fixed distance D. Therefore, differently from simple sampling, an OBU transmits only if it is actually traveling. This policy represents the vehicle movements as constant distance "jumping vectors". In order to improve accuracy, a variant of this policy ("stop" version) allows vechicles to send additional position reports when they stop or restart their motion;

Map-Based Sampling: The basic idea behind this approach is that the vehicle moves on map segments. Therefore, in its simplest version, it sends a position report whenever the end of a map segment is reached. Similarly to space sampling, also map-based sampling allows vehicles to inform the CC about stops and restarts ("stop" version). Furthermore, instead of relying on map segmentation data, it can decide to transmit whenever a turn in the motion trajectory is detected ("turn" version).

Information-Need techniques: the OBU decides when to transmit on the basis of specific information made available by the CC. The techniques we consider in this category are:

Deterministic Information Need: The main idea of the deterministic information need policy is to prevent small differences in velocity from being transmitted to the CC (Kerner et al., 2005). To this end, a velocity threshold V is used and the OBU sends its position report when $|s-\overline{s}(seg)_{t,\tau}| \geq V$, where s is the vehicle speed and $\overline{s}(seg)_{t,\tau}$ is the average speed calculated by CC in that segment;

Flow Information Need: The flow information need policy performs data reduction by using randomization (Ayala et al., 2010). That means that every OBU has an equal chance of transmitting to the CC, regardless of its speed. We will test also an alternative version of this technique where OBUs verify if they could send their report whenever a turn is detected ("turn" version), as for Map-Based Sampling.

6 Experimental Evaluation

In this experimental evaluation, we will show the results we obtained from the specific tests that are mainly focused on the overall performance and scalability of the EDBMS layer and the effectiveness of the data reduction techniques. Without loss of generality, we will consider vehicles (i.e., their position and speed sensors) as the primary source of incoming information. In particular, we will report on two kinds of tests which stressed the framework capabilities in reacting to simulated smart city-like workloads:

- two 3-hours simulations with very intense ITS-based input rates, based on the Linear Road stream data management benchmark (Arasu et al., 2004) which is a reference in this field, for testing the performance and robustness of our EDBMS layer in real-time storing, indexing and querying incoming data (Section 6.1);
- four simulations reproducing actual traffic condi-

tions of as many cities in different parts of the world (Bologna, Roma, Camden and Beijing) to put to the test the data reduction techniques in such real cases (Section 6.2).

6.1 Data Storage and Query Processing

In the Linear Road stream data management benchmark, a multitude of cars move on multiple lanes of a virtual highway and pay dynamically calculated tolls. In particular, the input data (position reports, toll updates and so on) is generated at varying levels of complexity (i.e., number of simulated expressways), where the data for testing each expressway reaches a rate of 100000 reports per minute. A system is said to achieve a so-called *L-rating* if it meets a 5 second response time constraint for all toll queries while supporting an input level encompassing L expressways of data (e.g., rating L1 for 1 expressway). Note that we will stress the unique capabilities of our EDBMS layer in executing all the complex real-time requests while, at the same time, we will go beyond the standard benchmark and also require the system to maintain the full stream history of position reports and to make such history always queryable. We will indicate these extended one- and two-expressway scenarios as L1+ and L2+, respectively.

Besides standard table support from PostgreSQL 9.0, all the additional data structures and management code are implemented in Java 1.6, exploiting Oracle BDB 11gR2. All the experiments are executed on an Intel 2.66Ghz Win7 workstation, equipped with 4GB RAM and a 500GB 7200rpm SATA disk.

It is important to note that, in all Linear Road scenarios, the more time goes by, the more data enter the system. Figure 2 shows, for each simulation minute in scenario L2+, the average response time computed up to that minute (line); the growing input workload is depicted on the background. As we can see, the average response time increases very slowly even in the final part of the simulation, when more than 200000 tuples per minute have to be processed.

To contextualize these results, on our hardware (or a closely comparable one), none of literature state-of-the-art systems achieves a level of performance exceeding L2 (Abadi et. al., 2003; Botan et al., 2009), the same supported by our system but in the much more demanding extended L+ scenarios.

6.2 Data Reduction

In order to throughly evaluate the effectiveness of the different data reduction techniques of our framework, in this second part of our experiments we will discuss the simulations performed on the following scenarios:

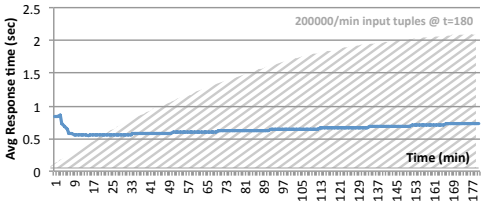

Figure 2: Average response time detail over the 3-hours simulation time, at level L2+

Bologna (BO, for short): a portion of the city center, i.e. a typical european urban scenario, involving medium-width and narrow streets, where vehicle stops are quite short and infrequent and the vehicle average speed is low;

Roma (RO, for short): another european portion involving non-central junctions and crossroads in Rome (Italy), with fast and multi-way segments, vehicle stops infrequent, and high traffic density;

Toll Plaza (TP, for short): a portion of multi-way road junctions in Camden (New Jersey, USA), including a toll-payment station, with medium traffic density and nearly one vehicle out of two stops;

Beijing (BJ, for short): a portion of Beijing traffic network, with very intense, congested and heterogenous (i.e. not only cars but also a significant number of motorbikes and bicycles) traffic conditions, where vehicle stops are frequent and long.

All the scenarios simulate the incoming data exchanges needed to effectively support a typical traffic monitoring service, i.e. for maintaining the average travelling speed of each segment in a sufficiently precise and updated way. The simulations are ten minutes long and have been created in the PTV Vision VISSIM[2] software taking into account real data on the detailed topography, traffic volumes, vehicle speeds and flows representing typical traffic conditions in such environments.

In order to evaluate the actual effectiveness of each technique, we will consider the communication cost (messages sent per second) together with the average error produced by the CC in computing the average speeds. The error is computed as a mean of the ”distances” between the expected values and the actual values (as available from the complete simulation data). We plotted the achieved results in Figure 3. All techniques were tested with a large variety of values for their distinctive parameters, producing a series of points in the graph, and in all their respective variants (including the “stop” ones, sending stop and restart information, and the “turn” ones, sending the needed data each time a turn is identified in the vehicle trajectory). Note that, since space sampling achieved worse results than simple sampling for

[2] http://www.ptv-vision.com

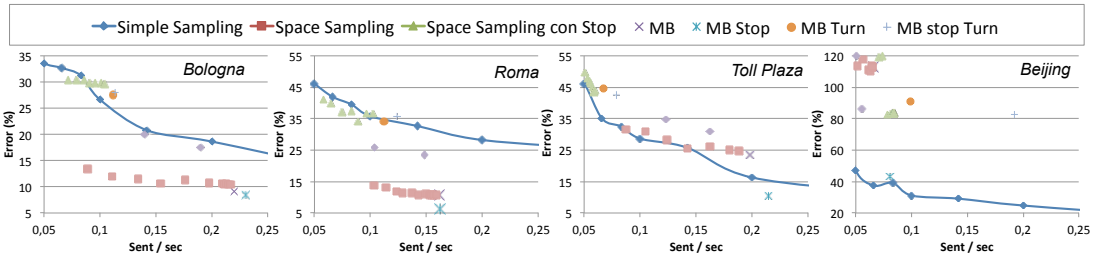

Figure 3: Performance detail graphs for data reduction techniques in the four scenarios

all scenarios, for a better readability space sampling results are not reported in the graphs. Starting our analysis from the BO and RO scenario, we can see that the flow-based technique (without turn option) is able to produce the most satisfying performances, significantly decreasing the simple sampling error level. Indeed, the regularity/predictability of the traffic and the infrequent stops performed by the vehicles are well suited to such kind of techniques, while in TP the higher number of stops makes their effectiveness less evident. Finally, in BJ, which is a very complex scenario, with very irregular traffic flows (also due to the conspicuous presence of different vehicle types), the error levels of most advanced techniques are quite high: in this case, simple sampling appears the simplest and most effective option.

7 Conclusions

In this paper we presented a technological framework to efficiently support data management in modern Intelligent Transportation Systems (ITSs). Data is collected from various sources and stored in an Extended DBMS (EDBMS) that guarantees efficient storage and access to a variety of recent/historical/static data. To the authors' knowledge, no proposal exists that merges natively DBMS and DSMS paradigms.

The EDBMS's transactional storage manager exploits a new type of table, called streaming table, to store live as well as past streamed data. Streaming tables are subjected to continuous writes and support both CQs and OTQs. The EDBMS provides query capabilities that span streaming tables as well as traditional relational tables in a transparent way. As to scalability issues related to the management of huge amounts of incoming data, we experienced the employment of V2I data reduction techniques in the data acquisition phase. A preliminary experimental evaluation on the standard Linear Road ITS benchmark and along various simulated traffic scenarios has shown the storage/query efficiency offered by the streaming table implementation and the effectiveness of the employed data reduction techniques.

REFERENCES

Abadi et. al., D. (2003). Aurora: a new model and architecture for data stream management. *VLDBJ*, 12(2):120–139.

Arasu, A., Babu, S., and Widom, J. (2006). The CQL continuous query language: semantic foundations and query execution". *VLDB J.*, 15(2):121–142.

Arasu, A., Cherniack, M., Galvez, E., Maier, D., Maskey, A., Ryvkina, E., Stonebraker, M., and Tibbetts, R. (2004). Linear Road: A Stream Data Management Benchmark. In *Proc. of VLDB*, pages 480–491.

Ayala, D., Lin, J., Wolfson, O., Rishe, N., and Tanizaki, M. (2010). Communication Reduction for Floating Car Data-based Traffic Information Systems. In *Proc. of GEOPROCESSING*, pages 44–51.

Botan, I., Alonso, G., Fischer, P., Kossmann, D., and Tatbul, N. (2009). Flexible and scalable storage management for data-intensive stream processing. In *Proc. of EDBT*, pages 934–945.

Carafoli, L., Mandreoli, F., Martoglia, R., and Penzo, W. (2012). Evaluation of data reduction techniques for vehicle to infrastructure communication saving purposes. In *Proc. of IDEAS*, pages 61–70.

Cotton, B. (2011). Moving citizens in the smarter city - using a framework approach to plan intelligent transportation systems strategies and implement solutions. http://www.frost.com.

Kerner, B., Demir, C., Herrtwich, R., Klenov, S., Rehborn, H., Aleksic, M., and Haug, A. (2005). Traffic state detection with floating car data in road networks. In *Proc. of ITSC*, pages 44 – 49.

NIMA (2000). Department of defense world geodetic system 1984, its definition and relationships with local geodetic systems. Technical report, TR8350.2.

NOAA (2008). Nad 83 (nsrs2007) national readjustment final report. Technical report, NOS NGS 60.

Streetline Inc. (2011). Becoming a smart city. http://www.streetline.com.

Terry, D., Goldberg, D., Nichols, D., and Oki, B. (1992). Continuous queries over append-only databases. In *Proc. of SIGMOD*, pages 321–330.

Xu, Q., Hedrick, K., Sengupta, R., and VanderWerf, J. (2002). Effects of vehicle-vehicle/roadside-vehicle communication on adaptive cruise controlled highway systems. In *Proc. of VTC*, volume 2, pages 1249–1253.