Data Stream Processing Infrastructure for Intelligent Transport Systems

Eric Bouillet, Mark Feblowitz, Zhen Liu, Anand Ranganathan, Anton Riabov, Schuman Shao, Don Schlosnagle, Fan Ye
IBM T. J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532, USA
Email: {ericbou, mfeb, zhenl, arangana, riabov, smshao, daschlos, fanye}@us.ibm.com

Abstract— **Intelligence Transportation Systems are critical to improve the efficiency of modern transportation. A system that is flexible and powerful enough to handle diverse demands from a large user base, is still elusive. Studies have shown that developing and integrating the various components constitute a significant portion of the capital cost and complexity of such systems.**

In this paper, we present a stream processing infrastructure we call System S. System S enables the deployment of large scale applications. It supports a mechanism for sharing multi-party data sources, software components, and even intermediate results allowing a reduction in the cost of software integration, and ownership.

We experiment the stream processing infrastructure with a Fleet Management Center, and demonstrate how the infrastructure can be used to address unique issues in traffic management.

I. Introduction

The rapid growth of transportation demands and the slow increase in transportation capacity have created severe traffic congestions and incurred great economic costs. Intelligent transportation systems that can utilize various kinds of data sources such as vehicle sensors, traffic cameras hold the potential to alleviate the dire situation. Building a powerful and flexible traffic management system, however, is a tremendous challenge.

Traffic data come from many heterogeneous sources as streams, in different forms, content and quality. They can be noisy and bursty, with limited availability guarantees. Both spatial and temporal analysis is needed to correlated data from sources in large geographic areas or time periods. The demands of analytical results, come from many user groups, such as drivers, highway patrol, department of transportation. They pose not only large numbers of simultaneous queries, but also queries of significantly different nature.

It is not surprising that building and maintaining an intelligent transportation system incurs significant cost and complexity. Studies [3] have shown that developing and integrating software components that access and analyze data streams from heterogeneous sources consumes almost half of the total capital cost in a Fleet Management Center. Various software components, each of which accessing a different type of data source, or analyze data of different content, quality, may come from different vendors, not necessarily developed for interoperability. Composing them together to satisfy diverse user requirements, is even more difficult.

We conclude from this study that software development and integration play a major part of Intelligent Transport Systems (ITS). It is thus important that we make these software artifacts as *reusable* and *modular* as possible. We must be able to compose them effectively in different applications that answer different kinds of queries. Such a system requires standard and reusable mechanisms for interfacing the data with the outside world. In order to facilitate growth and incorporation of new technologies, the system must enable the creation and deployment of new applications without disrupting existing ones. Furthermore, new applications should be able to reuse intermediate derived streams produced by existing applications in order to minimize duplicate or redundant processing of data. The system must also provide a consistent security and privacy enforcement mechanism for the use of derived data, in order to insure that confidential information is not leaked or compromised across the applications. The system must allow location-sensitive applications to adapt to the current locations of the entities they are tracking or assisting. In addition, the system must adapt to the to the addition of new data sources, to the quality of the data produced by the sources, and to changes in system resources consumed by different applications.

In this paper, we propose System S, an infrastructure that supports the large scale deployment of stream processing applications on the fly. These applications ingest data from many heterogeneous data sources. Applications in System S consist of many interconnected modular, reusable software components (called *Processing Elements*), each of which takes data of certain content and format and perform various analysis. Together they produce the highly summarized end results needed by users. The system incorporates components for managing data sources and system resources, and sharing derived securely streams between applications.

We make several contributions in this paper. We propose a data stream processing architecture as an execution context for ITS applications. This greatly reduces system integration costs for transportation systems. We incorporate mechanisms to manage different data sources, and to adapt applications to location-sensitive requirements. Finally, we build a Fleet Management applications in the streaming processing system to demonstrate the flexibility and power of our approach.

The rest of the paper is organized as follows. We give an overview of the system in Section II. Sections III IV explain how application adaptation and data source management are accomplished. We illustrate a fleet management application in V and conclude in Section VI.

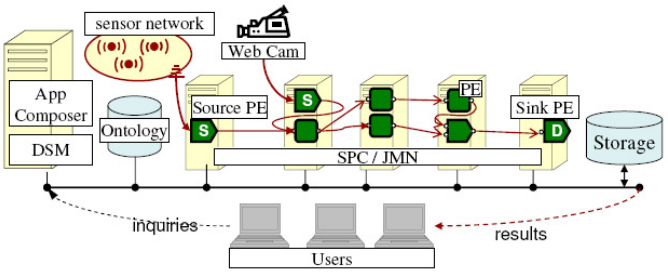

Fig. 1. System S

II. System Overview and Examples

We have implemented our approach within a stream processing system referred to as System S. A sample configuration of System S with a deployed application is shown on Figure 1. The system is centered around a *Stream Processing Core* (SPC) [1], a scalable distributed middleware for stream processing. The SPC jointly with other system components deploys application workflows to multiple processors, and manages PEs and streams.

The SPC consists of a large cluster of dedicated processors and an execution context upon which stream processing applications are deployed. Each stream processing application consists of a processing graph, which is a directed acyclic graph of stream processing components, namely data sources and processing elements (PEs).

Data sources deliver streaming *primal data* into the system and PEs perform various types of operations on streaming data - filtration, annotation, transformation, stream join, etc. PEs are individually deployable and reusable software components and are interconnected by multi-access (single-writer, multiple-reader) streams. PEs and data stream instances in SPC are statically assigned individual security and privacy labels, inherited in part from the applications' effective user IDs, and the PE classes. SPC connects PEs to the data streams only if their respective security and privacy labels allow it. If necessary the labels can be adjusted using "downgrader" PEs that sanitize sensitive information (e.g. remove identifying information, etc). Applications can tap onto past or present (real-time) data. Past data is stored by the system and can be processed like a stream.

End-users interact with System S by specifying stream queries that describe the data of interest to them. We refer to System S queries using the name *inquiries*, to distinguish them from traditional SQL queries, because of the application to mostly streaming and unstructured data, because inquiries can span long time frames, and because they are satisfied by a diverse collection of stream-based processing algorithms. Inquiries are pursued by assembling and deploying processing graphs of interconnected PEs that can produce the data of interest. These processing graphs may be assembled manually or automatically. In this paper, we demonstrate System S's ability to support Intelligent Transport System oriented applications using processing graphs that have been assembled manually by an application developer or administrator and that can be parameterized for different use-cases. In [2] we describe an ontology-based application composer for System S that automatically composes the processing graph for an inquiry using detailed descriptions of data sources and PEs in ontologies along with AI planning algorithms. We shall not go into the details of the automatic composition here.

PEs can be co-deployed on the same processing node (hardware) and can also be replicated across multiple nodes. Data can be processed serially, from one PE to another (linear graphs), but parallelizable tasks can be performed more efficiently by streaming data to concurrently operating distributed PEs. These aspects of the PEs' lifecycle management and resource allocation, are carried out by Job Manager (JMN), an integral system component of System S. JMN receives requests to deploy PEs with their flowgraph specifications in the form of *Job Description Language* (JDL). JMN interprets the JDL, computes the adequate resource allocation with respect to resource availability, and QoS constraints and priorities specified in the JDL, and deploys the PEs in SPC. In addition JMN provides the API to monitor and terminate deployed PEs.

Another component of System S is the *Data Source Manager* (DSM) which manages and activates data sources based on requests and feedbacks from applications and users. DSM provides a virtualization of the data source from the application perspective, allowing application developer (or automatic composer) to focus on instrumenting the data analytics independently of how the data is brought into the system.

Each data source delivers primal data into System S by way of a *SourcePE*. The SourcePE (denoted by "S" in Figure 1), a special type of PE that also runs in the SPC, serves the sole function of bringing data from an external source into the system. SourcePEs encapsulate the drivers, and other source dependent configuration setup to access a the broad variety of external sources. The data is packaged into internally streamable and processable *Stream Data Objects (SDOs)*. SDOs are lightweight data containers that carry data from SourcePEs to PEs, from PEs to other PEs, and eventually to SinkPEs (the dual of SourcePEs, denoted by "D" in Figure 1) that function to deliver SDO data as inquiry results for use outside of the SPC (e.g., to an end user, to a data store, etc.). Source PEs and Sink PEs allow the mechanisms for interfacing with the outside world to be reusable. The same Source and Sink PEs can be reused with different parameters for bringing in data from similar data sources and for pushing out results in similar manners.

For purposes of illustration, consider as a simplified example an inquiry requesting traffic congestion reports for a particular roadway intersection. Such an inquiry may draw audio data from a sound sensor data source and apply an Audio Pattern Analysis PE, which examines the data for alignment to known audio patterns to determine the level of congestion at the intersection (isolating the lower thread in Figure 2). In order to improve the accuracy of such an assessment, the application may use a Video source, extracting images from the video stream and examining them for alignment to visual patterns of congestion at an intersection (the upper thread). The end result would then be achieved by joining feeds from the two analytic chains.

This simple workflow can be extended further, by including additional PEs and data sources. In fact, it describes a small part of an application domain we call Fleet Management

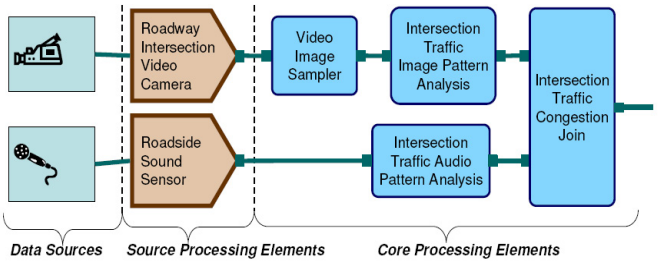

Fig. 2. A Stream Processing Graph Example

Center service (FMC), and we will be using this domain to illustrate our approach throughout the paper. Applications in FMC provide vehicle routing services based on the analysis of real-time data obtained from sensors. In FMC, routes for a delivery truck are set at the beginning of the shift, informed by any known traffic conditions at the time of departure. The standing inquiry established for each vehicle watches for changes in traffic conditions that warrant route replanning to provide an updated route. The routes are based on a collection of destinations and the current location of the vehicle (determined by a GPS transmitter in the vehicle). Accompanying the standing, vehicle-specific routing inquiries are inquiries that examine streaming data from multiple sensors and other sources and update the roadway and traffic conditions, with special focus on corridors covering known vehicle destinations.

An important feature of our infrastructure is the ability to reuse intermediate streams across different applications. Once an application (or processing graph) is built and deployed in response to a users query, the set of intermediate streams produced in this processing graph are added to the set of data sources that are managed by the DSM component. These intermediate streams are called derived streams. Subsequent applications that are deployed can make use of these derived streams. The main advantage of this feature is to allow sharing of common processing across applications and avoiding unnecessary duplicate computation.

III. Application Adaptation

In our system, a processing graph that is constructed and deployed in response to a user's inquiry is static through its lifetime. It is not easy to reconfigure the processing graph dynamically, since that would involve deploying additional PEs, stopping existing PEs, and changing the set of connections between different PEs and sources. Performing such operations dynamically would take some time, and would mean that the application will be down for that period of time. Besides, our model of sharing streams and components across different applications makes it difficult to do such dynamic reconfiguration.

However, most fleet-management applications do need some amount of adaptation at runtime. These applications often provide location-sensitive services to moving vehicles and would need to tap onto data from different sources depending on the current location of the vehicle.

In order to deal with these kinds of applications, we use a two-step approach, where application-independent intermediate information is produced in the first step, and actual results that satisfy a user's inquiry are produced in the second step. The key aspect of this process is to identify relevant intermediate information that can be obtained in an application-independent manner. This identification of useful intermediate information is done by a human who is designing the system.

In the FMC domain, for instance, traffic information about different locations may be useful to different kinds of applications (required by different users) and is hence identified as relevant information for a wide class of applications. Hence, inquiries that ask for traffic information at various locations are deployed, and the results are stored in an intermediate Road Conditions Database. In the second step, processing graphs corresponding to users' inquiries are constructed and deployed; these processing graphs use the intermediate useful information stored in the database. For example, inquiries that ask for the best route to a destination can use the current and past traffic information in the Road Condition Database. The two-step approach allows processing graphs of different applications to remain static. Applications get relevant location-dependent useful information from the database, which in turn stores the current and past values of the information extracted from the sources.

At the same time, not all the information produced by all the sensors are required by some application. In order to minimize the volume of data extracted from different sensors, and the processing performed on this data, our system employs a feedback mechanism to determine which sources are required by any application at any point of time. Applications provide their current requirements for the intermediate useful information they need, and the DSM (Data Source Manager) component decides which sets of sources should be activated or deactivated at any point of time, using a combination of description-logic and location-based reasoning. A source must be active at some point of time if there exists at least one application that requires derived information from that source at that point of time. Otherwise, the source may be deactivated, by deactivating the source PE, which causes data from the source to stop flowing into the system.

IV. Data Source Management

The Data Source Manager (DSM) component provides a repository with services for managing external data sources that are available to the system, as well as derived streams produced by applications within the system. The information stored in the repository consists of the system descriptions of the data sources, such as their access methods, and other metadata such as the semantics of the data produced by the source and its quality and rate. Applications express their requirements for a data source in the form of a semantic query. DSM executes the query on its repository to determines the list of potential data sources suitable for the application. Data sources are connected using Source PEs that are configured accordingly. After deployment the various applications can refine the desired data source semantic descriptions, e.g. limit their interest to sensors located in certain areas, and DSM use this information to activate or deactivate the data sources accordingly. The main tasks of the DSM include managing

information about all sources, deploying source PEs to connect external sources, pro-actively discovering new sources, monitoring the availability and quality of existing sources, and activating or deactivating sources depending on current application needs. DSM simplifies the task of integrating of a large variety of heterogeneous data sources in the system, which typically account for a non-negligible portion of the capital cost of operating such a system.

V. Application to Fleet Management Services

In this section we illustrate the capabilities of the system to support an Intelligent Transport System application. We use a Fleet Management Center application we have implemented for System S, and demonstrate how an existing application can be further upgraded to include other services.

Figure 3 depicts a number of processing graphs for the FMC inquiries. The upper section depicts the route update inquiries, and the lower section depicts the location condition update inquiries. In the route update section are the various PEs - analytic modules that receive the streaming data and perform functions such as generating the K best potential travel corridors, deciding on routes based on vehicle size and, where available, traffic conditions, and fusion of potentially conflicting condition reports. The two main results of these inquiries are route updates for the vehicles and updates to the list of currently active locations, which guide the focus of the condition-assessment inquiries (in the lower part of the graph).

The lower section contains an inquiry per known data source (weather sensor networks in this example), drawing data from the source, processing the data to determine conditions, and updating a Location Conditions store, the sole recipient of condition inquiries' result data. This data is retained for some limited duration and triggers rerouting in the upper inquiries. Other data sources might include non-sensor data sources such as local radio weather report sources or traffic incident report sources. But these are secondary sources, drawn from other service providers, possibly providing stale reports. For rapidly changing conditions, it would be better to draw data from directly accessible sensors or sensor networks, using such devices as traffic surveillance cameras, in-road vehicle sensors, wireless sensors capable of sensing temperature, barometric pressure and humidity, infrared remote temperature detectors (to detect road surface temperatures), and even roadside microphones (to analyze traffic noises). In more permanent installations where power consumption is not a factor, each of these can produce high volumes of streaming data, which can be processed by many stream processing applications deployed in the system, to dynamically process the signals into usable results. There are a few non-sensor sources, providing information not limited to a single geographic area; these may deliver weather updates that span a wide range. There are many more sensors, though, since the scope of the data is limited by the sensor's range: a traffic camera at street intersection can only deliver data within the camera's (possibly fixed) range. Many sensors covering anticipated travel corridors must be be deployed, and activated by DSM when respective travel corridors are active, with their data processed by separate stream-interconnected PEs, in response to inquiries.

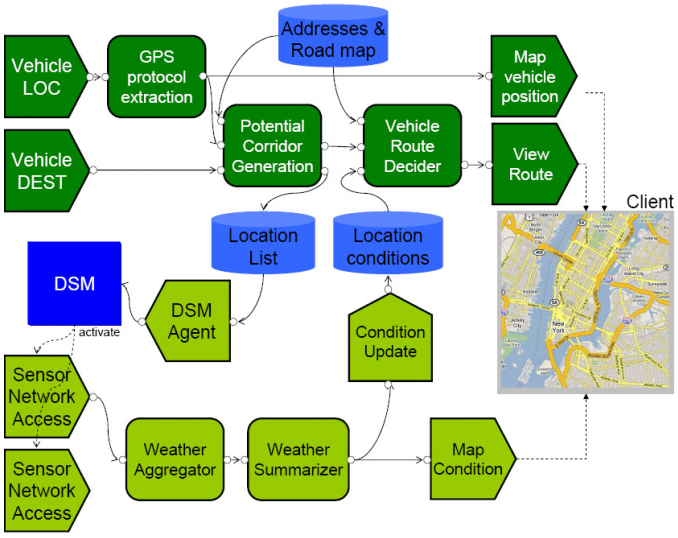

Fig. 3. Fleet Management Center Application

A. Fleet Management Center Architecture Overview

The FMC application depicted in Figure 3 consists of the following PEs:

Vehicle LOC: are source PEs that receive raw GPS data of current vehicle locations.

GPS Protocol Extraction: translates GPS data from Vehicle LOC, into a form that is suitable to other PEs.

Vehicle DEST: are source PEs that retrieve current vehicle destinations from a database or user console.

Potential Corridor Generation: joins vehicles LOC with respective DEST coordinates and generates corridors that might be traversed by vehicles. The corridors are stored locally in the location list database for use by other PEs.

Vehicle Route Decider: computes shortest routes from a vehicle's current location to all its assigned destinations.

DSM Agent: translates the location list into semantic queries for the DSM connection management which in turns activate the appropriate source PEs to satisfy the requests. There is one specialized DSM agent per modality, e.g. weather, traffic report.

Sensor Network Gateway PE: is a source PE specialized to query sensor gateways. DSM connection management dynamically reallocate source PEs in response to requests from the DSM agent.

Weather Aggregator and Weather Summarizer: extracts weather data, and analyze its content to estimate the impact on traffic delays.

Condition update: persist weather related location condition to a Road Condition database.

The application also includes sink PEs (map vehicle position, view route, and map condition) that present the results of the user inquiry to a user interface.

Note that in practice we would further decompose the application and distribute the load into smaller PE elements. For example we can partition the area covered by the vehicle fleet into sub-regions, each handled by a separate instance of the application presented above. A regular expression based filter formulated in the JDL can be applied on the PE input ports so that the respective instances receive only SDOs that are

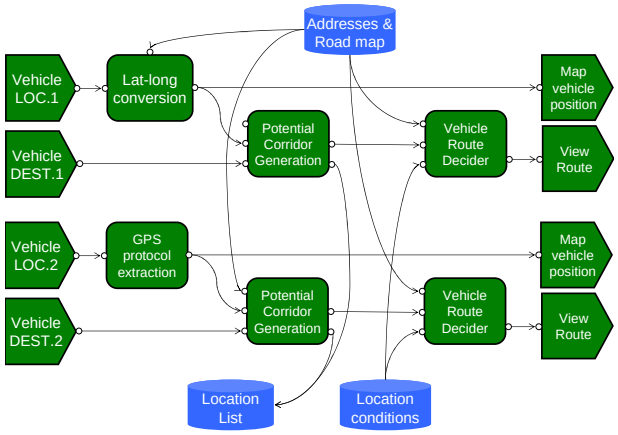

Fig. 4. Incremental deployment of new route computations applications to serve different fleets

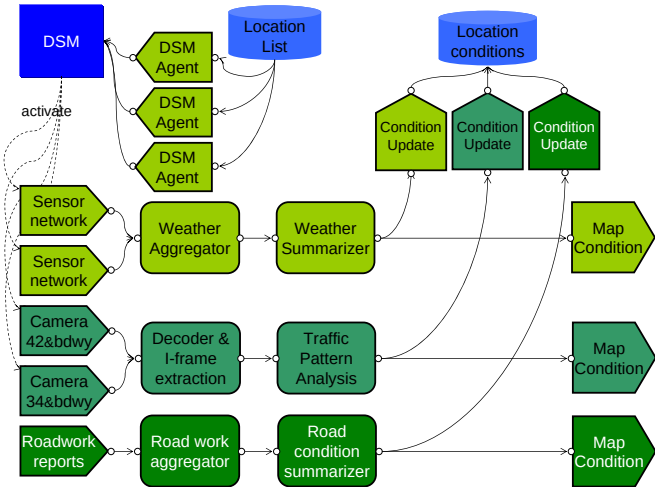

Fig. 5. Incremental deployment of new modalities to improve the route-condition analysis

relevant to their assigned sub-region. Also the Vehicle Route Decider is normally a two-staged process with street-level (intra-city) routing, and highway (inter-city) level routing. The two stages are more efficiently implemented in separate PEs, with dedicated PEs for each city. System S encourages and facilitates such decomposition practices.

B. Incremental Application Deployment

The location list and the location condition databases provide a separation between the PEs that update the database, and PEs that read their data. This separation allows the PE in the top part of Figure 3, which update the corridors, to be deployed independently of the PEs in the lower part, which are responsible for updating the location conditions. As shown in Figure 4, several route selection PEs can be deployed in parallel to handle fleets of vehicles that have different routing requirements. Similarly, diverse applications can be deployed to update the location conditions with new modalities that corroborate, or improve the accuracy of the location condition as shown in Figure 5.

C. Data Stream Reuse

Derived data streams from existing applications can also be registered in DSM and reused to support other applications. Reuse is allowed within the limits of the respective application security and privacy constraints, which are enforced by SPC. For instance if the vehicles managed by the FMC application are shuttle vans, we can extend the application to dispatch vans to pick new customers along their assigned route as illustrated in Figure 6. In this example, the application uses a PE that takes customer pickup and drop-off street addresses in input, a PE that converts the addresses in lat-long coordinates, and a PE that matches the coordinates to the routes (derived stream of the original FMC application), currently assigned to the vans. The PE selects the van that is the best match for this customer and modifies its route to include the customer.

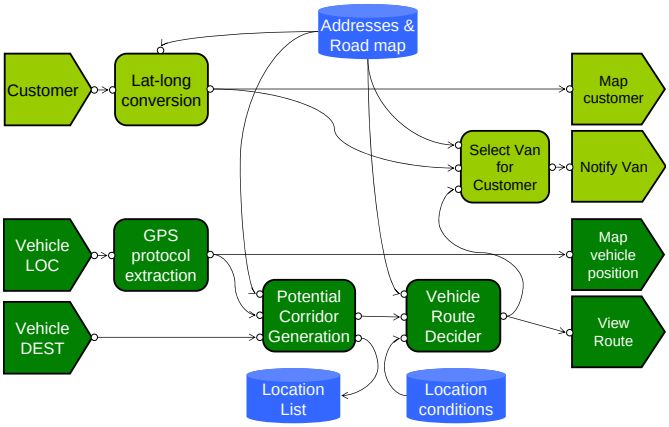

Fig. 6. Deployment of new applications reusing streams from an existing application

VI. Conclusion

In this paper we describe System S, a stream processing platform, and demonstrate how it can be leveraged for the deployment of large-scale Intelligent Transport Service applications. System S provides a reliable infrastructure with efficient resource management and secure data transmission. System S addresses several of the challenges faced in the implementation of ITS applications. In particular it promotes the paradigm for reusable and modular software components that can be composed in different applications to answer different kinds of queries; it provides reusable components for interfacing the data with the outside world; and it provides the mechanisms for managing concurrent applications, and share derived streams across applications. We illustrates System S using a Fleet Management Center application, and show how it is used to compose and extend such an application.

References

[1] N. Jain, L. Amini, H. Andrade, R. King, Y. Park, P. Selo, and C. Venkatramani. Design, implementation, and evaluation of the linear road benchmark on the stream processing core. In *SIGMOD'06*, June 2006.

[2] A. Riabov and Z. Liu. Planning for stream processing systems. In *AAAI'05*, July 2005.

[3] US Department of Transportation. Intelligent transport services benefits costs and lessons learned databases. *http://www.itscosts.its.dot.gov/its/benecost.nsf*, 2005.