

Framework for bringing data streams to the grid

Beth Plale

Computer Science Department, Indiana University, Bloomington, IN, USA

E-mail: plale@cs.indiana.edu

Abstract. Data streams are a prevalent and growing source of timely data, particularly in the scientific domain. Just as it is common today to read starting conditions such as initial weather conditions, for a scientific simulation from a file, it should be equally as easy to draw starting conditions on-demand from live data streams. But efforts to date to bring streaming data to the grid have lacked generality. In this article we introduce a new model for bringing existing data streams systems onto the grid. The model is predicated on the ability to identify stream systems that meet the criteria of being a “data resource”. We establish the criteria in this article, and define a grid service architecture for a data streams resource that leverages standardization efforts in the Global Grid Forum. We discuss key research issues in realizing the data streams model. We are currently developing a prototype of this architecture using our dQUOB system.

Keywords: Grid computing, data stream systems, publish-subscribe, continuous queries, OGSA-DAI, data management, dQUOB

1. Introduction

Data streams are a prevalent and growing source of timely data [16]. Stream applications are broad: sensor networks monitor traffic flow on US Interstates; NEXRAD Doppler radars continuously generate streamed data for weather forecasting and prediction. Network traffic, financial tickers, and web servers continuously generate data of interest. The literature describes numerous systems in existence for handling streaming data. But whereas these existing applications are often designed explicitly to serve the data streams, in the future we expect data streams to be viewed as yet another input source to be consulted at will and on demand. Just as it is common today to read starting conditions for a scientific environmental simulation from a file, it should be equally as easy to draw those starting conditions on demand from live data streams.

Scientific on-demand applications are distributed and have significant computational needs or data access needs. The grid service distributed computing model is an attractive alternative for architecting these applications because it promotes encapsulation, modularity, and extensibility by decomposing an application

into a set of independent services that use a widely accepted and standardized network communication protocol. The grid service model also supports scalability by its provisioning for secure access to resources that cross administrative domains. Earlier models of distributed computing failed to address the security, accounting, and social aspects of intra-domain computing. Prototype Grids, those in existence today that span multiple departments on a university campus, multiple government labs such as Grid3 [11], or multiple sites on a company intranet, demonstrate the benefits of harnessing disparate and widely disbursed computational resources. But while significant gains have been made in middleware support for applications that move terabyte files across vast distances, existing efforts to date to integrate data streams generated from instruments and sensors to the grid are only now beginning to emerge [8,6].

A naive solution to the problem of bringing data streams to the Grid is to treat each data stream source as a separate grid service. When a client needs to access stream data, it discovers the service by means of a standard discovery mechanism, then contacts the service asynchronously and passes it a request. We ar-

gue that while this model of one-service-per-stream is reasonable when the generator of the data stream is a large scale instrument, the model cannot be extended across the full range of data stream generation devices, from the hadron collider down to the tiny mote [27]. To illustrate, some sensor networks communicate by means of an ad hoc routing protocol implemented on top of the MAC layer [28]. These protocols are too low level to support SOAP [26], the agreed upon protocol for grid service to grid service communication. Additionally, tiny motes work in a power conservation mode so their radios are generally turned off except when transmitting. This mode of operation conflicts with the grid service policy that every grid service must respond to commands to manage its lifetime. Additionally, a grid service must be reachable by another grid service, should have its existence registered in a repository, and so on.

We propose a new model for bringing data stream systems onto the Grid. Intuitively, the model treats data stream systems as a single unit and gives that system a presence on the grid. Access to all such stream systems is through a single, common access protocol. The model addresses the recent explosion in stream handling systems, most noticeably in sensor networks. A model for bringing stream systems to the grid must be general enough to accommodate as broad a range of existing systems as possible. As an example, the sensor network that monitors a wetland can be brought to the grid while still retaining its MAC layer protocol for efficient communication. Its point of presence on the grid enables other grid services to contact it and other stream systems by means of a common access interface.

Not all data stream systems meet the criteria of our model. As we discuss in Section 2, a data stream system must satisfy the property that a distributed global snapshot can be determined by examining only the data in the data stream. Access to the stream system through the grid service is based on the model pioneered by OGSA-DAI [4]. OGSA-DAI defined a general database query access protocol to a grid service that serves a database management system. That is, through XML “perform” documents, a user issues database queries and updates to a grid service. The grid service interprets the message and interacts with the database management system to carry out the request. In a similar manner, we define access to a stream service as declarative database query and update requests.

The model that we propose of flexible access to streaming data on the grid is based on the following three assumptions:

- *Aggregation of data streams* – as streams become more prevalent and stream generation devices become smaller, richer interrogation over the streams is required. The value to a client of a data stream system increases dramatically when streams can be aggregated and global behavior can be interrogated.
- *Stream access through database operations* — database query languages are an intuitive way to think about stream access. The recent burgeoning interest in the database research community on data streams reinforces this view.
- *Grid service-based access to data streams* – grid service access to data streams should be organized around providing access to the data in the streams, not to the hardware that generates the data. It is the data that is the valued resource.

In order to fully understand the breadth of data stream systems, we survey the literature and categorize the systems into three broad categories: data manipulation systems, stream routing systems, and detection systems. Of these categories, two satisfy the property that a global snapshot can be taken by examining stream data alone. System categorizations are discussed in Section 3.

We then define a general architecture capable of handling any streaming system that meets the requirement of a *data resource*. The architecture leverages the Global Grid Forum specification for grid service access to database management systems. There are fundamental differences between a database management system and a stream system, however, that make the definition of an architecture and API a challenge. We discuss the architecture and challenges in Section 4.

The remainder of the paper is organized as follows: in Section 2 we define the model of a data stream resource on the grid. In Section 3 we categorize streaming systems along orthogonal axes in order to expose the essential defining characteristics. The architecture of the system and key research issues in realizing the data streams model is covered in Section 4. The paper concludes with a discussion of related work and conclusions, Sections 5 and 6 respectively. Our current effort is to realize this architecture using our dQUOB system [22].

2. Stream resource model

The term “streams” can mean many things. Results stream from a database a row at a time; a sequence

of requests streams to a web server; a stock ticker, a click stream, and keystrokes from a keyboard are all streams. We distinguish a *data stream* as an indefinite sequence of time-sequenced events (also called “messages” or “documents”.) Events are marked with a timestamp indicating the time at which the event is generated, and often include a logical time indicating the time at which the application specific event occurred. Data streams differ from message passing systems in that data streams loosely couple distributed components with asynchronous time sequenced data whereas message passing systems support parallel or tightly coupled systems where communication is often synchronous. Data streams differ from mouse or keyboard events because the latter tightly couple an I/O device to a process.

The model we propose for bringing data stream systems onto the Grid is broad so as to encompass a wide range of existing data stream systems for sensor networks and instruments. It treats the data streams of a stream system as a single unified data resource, like a database. The data resource is brought to the grid and exposed as a single resource. In our model a set of data streams is viewed as a single data resource that must be managed, and must support user access. Access to the resource is through a grid or web service that accepts queries and updates in a declarative query language. Our model accommodates ad hoc wireless networks, a large-scale instrument, or a stream-oriented database. This broad model is more scalable than the more common alternative whereby every data stream generator is web service accessible. As sensors get smaller, they tend to become more specialized in the kind of information they generate. The smaller the sensor, the more important becomes not the local knowledge obtained from a single sensor, but the global knowledge derived from combining information from multiple sensors. How meaningful is it, we posit, to learn that a light reading taken from a single sensor over a 3 cm area is on?¹

The relationship between a stream system and grid service is illustrated in Fig. 1. This figure contains three data resources, the one to the left is a network of tiny mote-like sensors communicating by an ad hoc routing protocol over a MAC layer. The one to the right is a regional network of weather sensors generating binary encoded (i.e., netCDF) data published on TCP connections. At the bottom are sources that stream

data into a database management system. All three data stream systems are brought onto the grid by a grid data service point-of-presence, labeled “GS”. The unique aspect of the model is that all the three data stream systems are accessible through a common query protocol supported by the “GS” grid service. The user, shown at the top of the diagram can query the data in all three systems by issuing declarative queries in a continuous query language. The model does not restrict the query language employed in the implementation of the individual stream systems.

The three data resources shown in Fig. 1 are defined by the data flowing in the streams, and not by the mechanics of the container that holds or generates the data. By defining the resource in terms of the data streams, and not the devices that generate the streams, the model effectively encapsulates implementation so that stream system developers can change the organization of their system, providing transparency to the users. A developer could move from a single stream system as a set of motes streaming directly to a grid or web service, to the organization at the bottom of the figure where all stream data is stored to a single database management systems, and the user would see no difference.

Distributed global snapshot of stream system. Our model hinges on the notion of a data resource in that stream systems that meet the criteria of a data resource fit within our model. Thus we carefully define a data resource and the criteria for inclusion. A *data resource* is a collection of information that is coherent and meaningful. *Coherence* is the level of integration of diverse elements or relationships. *Meaningful* means that the integrated diverse elements have an assigned function in our language and thought system. A relational database has coherence and is meaningful in that the tables in the database are related to one another and the tables when viewed as a collection, provide information that a person may be interested in. As such, data resources are amenable to rich interrogation, analysis, and manipulation.

Stream applications are organized around the production and consumption of data. In some stream systems, the data generated by producers contains the producer’s state or behavior. In traditional distributed applications, a distributed global snapshot consists of a snapshot of the processes in a distributed application plus all messages in progress between the processes. Because stream systems already export their state, the distributed global snapshot of a data stream application can be determined by examining the data streams alone. That is, *we can safely draw conclusions about*

¹As we show, our model does not preclude obtaining this information.

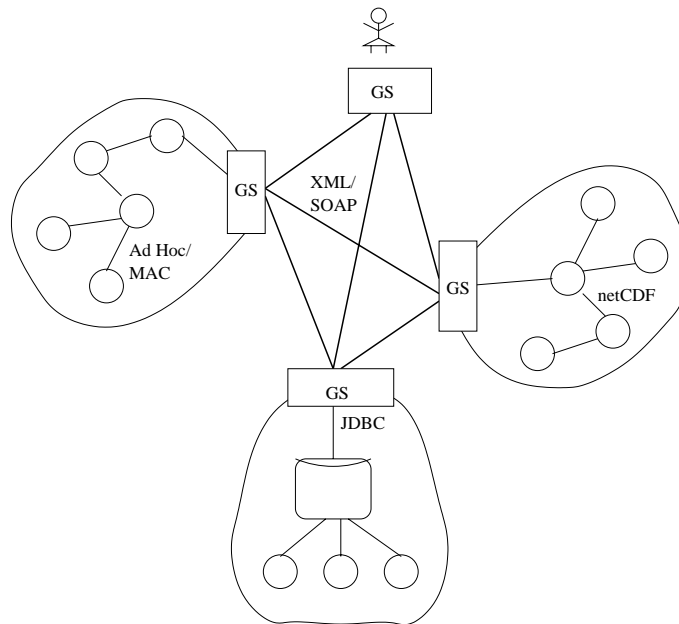


Fig. 1. Three data stream systems available on the Grid. The systems share a common query access interface while preserving the native efficiency of the various protocols underlying the data streams.

behavior of the distributed application simply by examining the streams. This defining characteristic makes stream systems quite different from distributed systems in general in that embodied in their data streams is a global snapshot. Because of this property, certain stream systems can be considered a data resource, just as a database is considered a data resource. When a distributed global snapshot can be obtained from viewing the streams only, the stream system can be considered a data resource. Because the streams have coherence and meaning, rich interrogation and analysis can be performed.

3. Categorizing data streams systems

Data stream systems are generally middleware systems or middleware over database management systems, that operate on data streams. These systems fall into three general categories: stream routing systems, data manipulation systems, and stream detection systems. The categories can be distinguished on the orthogonal characteristics of timeliness demands on the response and on the quantity of stream data that must be examined to process an arriving event (see Fig. 2.) Timeliness, along the X-axis, is the latency between event generation and delivery of a result at one or more consumers. If a decision must be made in on the or-

der of milliseconds, then the timeliness demands are high; if source to destination delivery and processing can take minutes, hours, or days, then timeliness demands are low. Stream routing and detection systems have high timeliness requirements. The second characteristic is size and number of stream chunks analyzed. Systems that make decisions based on localized, single-stream, single-event chunks are low along the Y-axis. Those that must examine larger amounts of data either spatially (i.e. across streams) or temporally (i.e. within streams) in order to make a decision are high along the Y-axis. Though stream systems do not always partition neatly into these categories, in particular there is considerable overlap in data manipulation and detection systems, the general distinction is valid and forms the basis for the conditions under which a set of data streams can be considered a data resource. We refer to the act of operating on stream data as *decision-making*. Determining to where an event should be routed, whether or not to discard an event, or apply to it a transformation, whether or not a query is satisfied by the recent arrival of an event, are all forms of decision-making over data streams. We discuss each system in more detail.

Stream routing systems. Stream routing systems disseminate events (or documents or information about events) to interested recipients. These systems are known by many names: publish/subscribe

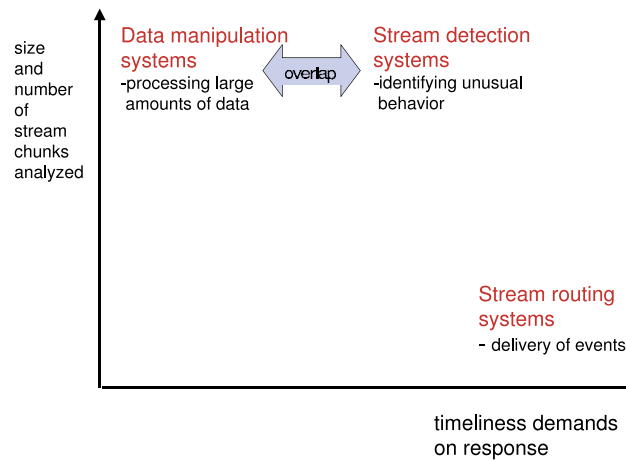


Fig. 2. Types of data stream systems.

systems such as ECho [10], NaradaBroker [14], and Bayeux [29]; selective data dissemination system such as XFilter [2], document filtering system such as Xyleme [20], message-oriented middleware (MOM) [17]. Stream routing systems are distinguished by the locality of the information needed to make the decision. Decisions are made almost exclusively upon the arriving event. Though some history may be maintained for instance to optimize performance, decision-making is largely localized to the immediate event at hand. The high delivery rates expected of such systems ensure that the decisions are kept simple.

A simple stream routing system is illustrated in Fig. 3. A remote broker is shown receiving and distributing stock quote events. Users register their interest in particular stocks through submission of a query to the broker. The query might be, for instance, a Boolean expression, path expression, or Xpath query. Event arrival triggers query evaluation. This is quite unlike a database where query evaluation is triggered by the arrival of the query. An arriving event is matched against the long-standing queries, and is then routed to the consumers indicated by the set of matching queries. The queries are long lived and are executed repeatedly. The expectation of these systems is that millions of queries can be active at any time. Key to achieving timeliness is the efficient matching of arriving events against a large number of long standing queries.

Data manipulation systems. Data manipulation systems are general stream processing systems that transform, filter, and aggregate data streams. Processing often results in the generation of new streams. There are looser timeliness requirements on the results of these systems than for stream routing or stream detection

systems. For example, a large-scale instrument or set of instruments that generates large data sets can make the data sets available to the science community on the scale of hours later, and after having undergone extensive transformation processing. The types of decisions in data manipulation systems can be framed as requests for data and for manipulation of the data, thus the language used to express the requests must be flexible enough to express these complex requests [5,19,22, 21]. Data manipulation systems can be based on the assumption of periodic streams, that is, the assumption of periodicity for all streams in the system. Sensor network systems display this characteristic.

Data flow programming problems are another form of data manipulation system wherein data flows that originate at one or more data generators, are consumed at one or more consumers, and undergo filtering and transformation along the way. This functionality is provided in systems such as dQUOB [22] and DataCutter [7]. In work done at Cornell on ad hoc wireless networking, intermediate nodes aggregate information flowing from sensors to source [28].

Detection systems. Detection systems detect anomalous or changed behavior in remote distributed entities. In these systems asynchronous streams are the norm, that is, no assumptions about periodicity can be made. Stream detection systems are used to monitor performance, such as in R-GMA [12], Autopilot [23], Gigascope [9], changes in HTML pages, such as in Conquer [18], or safety critical systems, such as dQUOB [24]. Though overlap exists between detection systems and data manipulation systems, the focus of the system drives the kind of support provided to users. A detection system that provides timely detec-

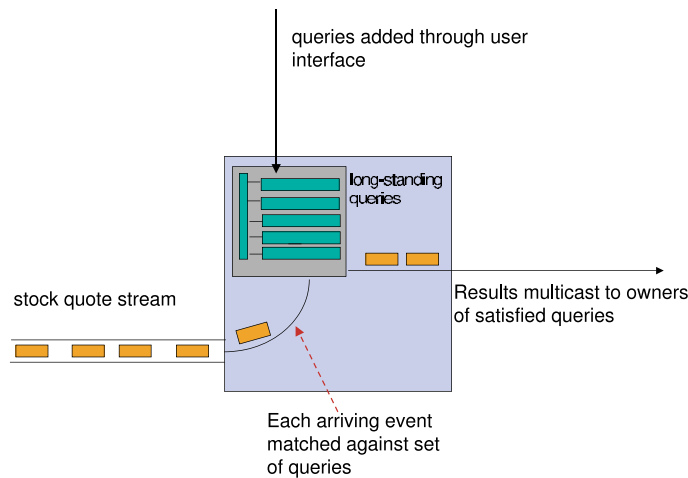


Fig. 3. Stream routing system example.

tion of anomalous behavior might put an emphasis on temporal operators.

4. Architecture of stream resource

Data-driven applications require access to data resident in files, databases, and streams. Access to data in streams should be as intuitive and uniform as access to these other mediums. We believe that this goal is best achieved within the grid services model of distributed computing by viewing the data streams generated by data manipulation and detection systems as a “data resource” that is accessible through a grid service interface by means of a database query language. By modeling a data stream system as a data resource, we are able to provide rich query access to the global snapshot that is inherent in these stream collections. This leads to a definition of a virtual stream store as the architectural basis of the stream resource.

We define the **virtual stream store** as a *collection of distributed, domain-related data streams and set of computational resources capable of executing queries on the streams. The virtual stream store is accessed through a grid service. The grid service provides query access to the data streams for clients. Event stream providers are external to the virtual store. The streams they generate are external to the store until explicitly published to the store. Data streams produced as a result of query execution (i.e., views) are part of the virtual stream store in which they are generated.*

An example data stream store, illustrated in Fig. 4, consists of nine data streams and associated computational resources. The computational resources, Ci,

are contained within the virtual stream store but can be physically widely disbursed. Computational resources are located in physical proximity to data streams. The definition does not prohibit a computational resource from also being a provider. The providers, which include the radar in the lower left, and six sensors, two per sensor node for nodes C1, C2, and C4, are excluded from the virtual stream store. This is an important feature of the model. Through the feature, the model can accommodate a database implementation of a data streams system, that is, where data streams are resident in a database and are serviced by long running queries managed through extensions to the database management system [5,15]. Exclusion of providers benefits the provider by allowing it to retain control over which data streams are visible to a virtual stream store and when the streams become visible.

A client accesses the virtual stream store by submitting continuously executing, long running queries to the Grid Data Service. Access to results is through an instance of a Rowset Data Service that is created on a per-client basis to serve streamed responses to the user. In the following section we probe more deeply into database access interface for a data stream store and define open research issues.

4.1. Grid service interface

A challenge in this work is to define a uniform access interface to the stream system. Our starting point is the work done by the University of Edinburgh team with OGSA-DAI [4]. Access in a grid service is defined in terms of ports and methods. A *port* is a logical grouping of methods. Ports are named units to which

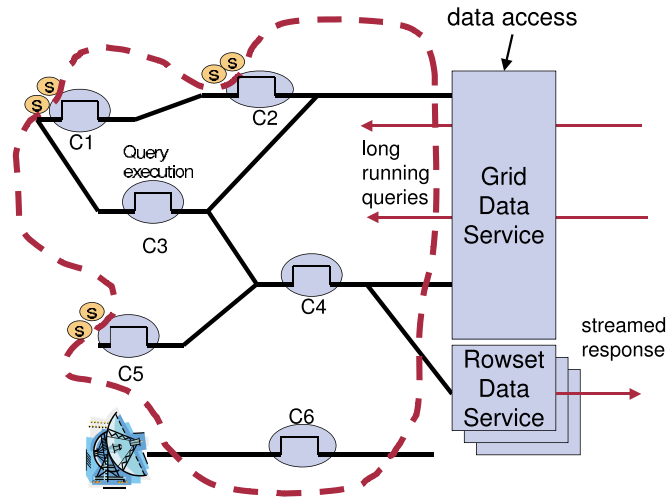


Fig. 4. Virtual stream store within thick dotted lines accessed through a grid service. The data providers are external to the store.

a client connects. The functionality of a grid or web service is defined by the ports it exports. In Fig. 5 we list the ports and methods of the grid service interface for a streaming system. The methods are named in terms of the model of the virtual stream store, but the functionality is general enough to extend to the stream systems we considered.

As can be seen in Fig. 5, the access interface is organized into four interfaces, with each interface corresponding to a port. The “SQL logical interface” is the interface through which a user submits long running queries. The results of a long running query is a streamed response that is retrieved either as a stream or by a time-based request. The “Stream publish interface” allows a provider to register a stream with a virtual stream store (as well as change its attributes or later remove it.) The “Admin logical interface” allows an administrator to create a new virtual stream store and define its properties.

4.2. Research challenges

We motivate the discussion of research challenges with two example stream systems. In an example taken from [28], sensor nodes are limited computational devices equipped with several kinds of sensors (e.g., temperature, light, PI). Nodes are connected to neighbors in an ad hoc wireless network and use a multi-hop routing protocol to communicate with nodes that are spatially distant. Special gateway nodes connect to components outside the sensor network through long-range communication such as cables or satellite links; all communication with users goes through the gateway node.

Queries are long running and periodic; streams are assumed to be synchronous. The query sensor network aggregates readings of the distributed sensors at each timestep. In Fig. 4(a) four sensor nodes are depicted, A, B, C, and Gate. Gate is the gateway node that communicates with the Grid Data Service to return a stream of aggregated values, one per timestep, for each sensor in the system. As can be seen, the periodic output from the sensor network is an aggregation of values for sensors x , y , and z at each timestep. The user in this example might be a portal interface that graphically displays results in real time.

The second example is visualization flow processing as provided in dQUOB [22] where compute nodes are located on the path between the parallel model (WRF1, WRF2) and its visualization; see Fig. 6. Each node on the path performs filtering, transformation, or aggregating. Transformation might convert the data from spectral domain to grid domain by application of an FFT. Another might join data streams from the model with a stream from the user and filter according to the user preference in a particular region of 3D space. Figure 6(b) depicts two model components, WRF1 and WRF2, that push data through a transformation node (T) and filter node (F). At node T the streams WRF1.x1 and WRF2.x2 are joined and the function T applied to the result. The resulting stream of events, Y, are streamed to node F where the function F is applied, resulting in stream Z. The user in this example could be a visualization tool enabled for application steering.

Integrating data streams from instruments and sensors into grid computing raises a number of research

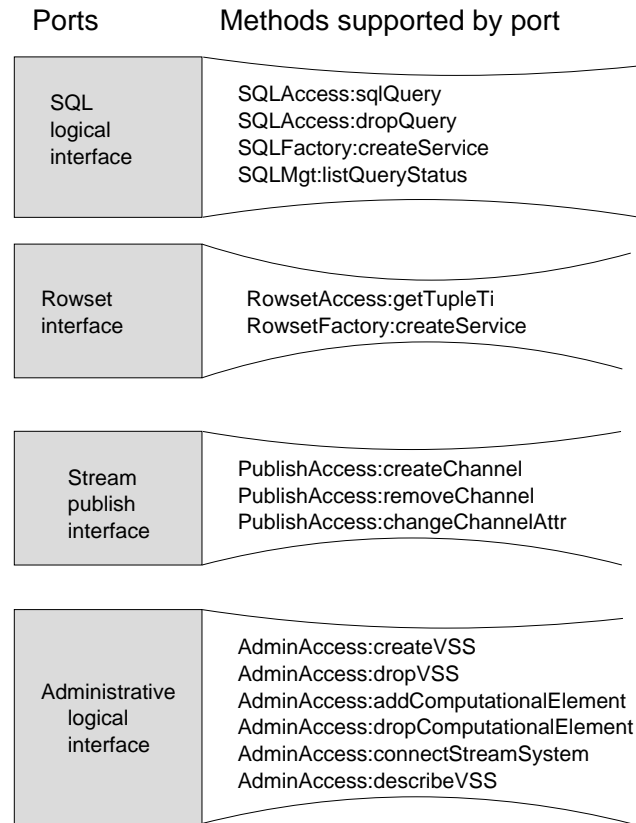


Fig. 5. Stream service interface definition.

issues in query distribution, data distribution, query management, and query instantiation.

Query distribution. The virtual stream store will in most cases be a distributed resource that provides location transparency. That is, the user writes a single query as if all streams (tables) were centrally located. But since the streams are distributed, some form of query distribution must be provided. As this kind of functionality is nearly universal in all stream systems we examined, it could be provided as a pluggable component in the Grid Data Service. The OGSA-DAI [4] reference implementation, for instance, is structured to handle pluggable modules in the Grid Data Service.

Data distribution. Data distribution models differ across the systems we examined. For instance, in the sensor network example of Fig. 6(a)), records of the same sensor type but from different nodes have the same schema, and collectively form a distributed table. Where sensor nodes A, B, and C export the stream of their sensor y . The distributed table Y is the union of all $A.y$, $B.y$, and $C.y$. The operation performed on that distributed table is the aggregation of events based on timestamp. This is seen by the result streamed from C,

namely $\text{Agg}(A.y, B.y, C.y)$. Not shown is that C acts as an intermediate node also for streams $A.x$ and $B.z$ but does not operate on these. In the sensor network, data is distributed.

In other examples, such as the flow model of dQUOB [22], the data is federated in that the data from each node is treated as a separate table. While neither approach is superior, the virtual stream store and its grid service interface should be flexible enough to handle different distribution schemes.

Query distribution in databases often includes assembling results from distributed sources. This functionality is less important in a stream system because aggregation of results is often done within the system itself. In the flow-programming example, the query is broken into subqueries and placed in the virtual stream store in relation to other subqueries based on proximity to the sources. The results are returned from the stream system in their completed form.

Management of long running queries. The queries themselves reside for an extended period in the virtual stream store so lifetime is an issue. Query lifetime is often specified by means of extensions to the query lan-

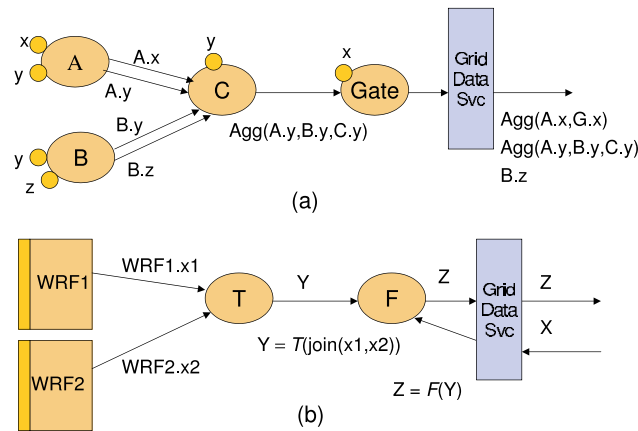


Fig. 6. Data distribution in sensor network and data flow network.

guage. If this support is not provided within the stream store, it would need to be provided by the grid service interface. The result of a long running query is a series of tuples generated over time that must be delivered by means of a stream delivery mechanism. These asynchronous requests are accommodated by the GGF DAIS grid service [3] by means of a special handling agent called a Rowset Data Service.

Query instantiation. Query instantiation is the realization of a query in the virtual stream store. The user specifies a query as say, an ASCII SQL statement, but the query must then be transformed into an executable entity in the virtual stream store. This instantiation is typically handled in a stream system-specific way thus support in the grid service must be modular and pluggable. To illustrate, suppose a set of hosts are available for use. In the Gigascope system [9], queries are pre-compiled into the executables that run on each of the nodes. In the case of dQUOB, the user submits a query that is compiled into a Tcl script that is then sent to the remote host. At the host are a runtime system and a dQUOB library. The runtime system receives the script and invokes a Tcl interpreter. Through the process of interpretation the script calls the dQUOB library which instantiates the query as C++ objects of operators (e.g., select, project) connected as a DAG. Thus the query is transported as a compact script, but runs efficiently as compiled code at the node. Further, the dQUOB query language allows definition of a user-defined procedures to be executed as part of the query. These code chunks are dynamically linked into the query execution environment at runtime.

Updates. An update is the act of publishing to a data stream in a virtual stream store. For reasons of performance and flexibility, data publication from devices,

sensors, and instruments must be independent of the grid services interface. That is, the streams and query nodes within the virtual stream store should not bound by the requirement to understand web service interface descriptions (i.e., WSDL) and use the SOAP transport protocol. Sensor nodes on an ad hoc wireless network, for instance, do not have sufficient resources or protocol support to support the communication overhead inherent in the grid service model.

5. Related work

Numerous stream systems have been cited in Section 3. The Aurora [1] is a continuous query system. The authors developed a fully functional prototype and runtime system designed to run on a single server and using TCP sockets as its underlying communication protocol. Aurora is an example of a system that our model and architecture should support.

Related efforts in grid services for stream data are smaller in number than stream system efforts. The Antarctic monitoring project [6] proposes a grid services architecture for interacting with an environment sensing device located in Antarctica. The grid services architecture provides access to and control of the sensor device, and accepts the data stream from the remote device via an iridium satellite phone network. The work differs from ours in that the service supports the combined functionalities of device management and access a stream of data.

The OGSA-DAI project [4] has developed a set of services for connecting databases to the grid based on emerging standards in the Global Grid Forum. Through OGSA-DAI interfaces, disparate, heterogeneous data

resources may be accessed and controlled as though they were a single logical resource. The grid service interface is described by WSDL. OGSA-DAI work is complementary to our work and in fact will serve as a key component in our implementation of the streaming architecture proposed in this paper.

Globus [13] has implemented a prototype reference implementation of the OGSI infrastructure as described in Open Grid Service Infrastructure. This Java-based toolkit provides a Grid Service hosting container that can run within a standard Web Services hosting container. The toolkit provides WSDL documents describing standard Grid service portTypes, Java implementations of these portTypes, sample services and support infrastructure to manage logging, security, service deployment and service lifetime management for example.

The GATES [8] project is a relatively new project in data stream systems. The authors propose a data flow model to stream processing, and focus their efforts on coding the computational elements in the flow in such a way that they are grid services that respond to commands to adapt their behavior to messages received about changes in their environment.

Narayanan et al. [25] discuss a services oriented software infrastructure that provides database support for accessing, manipulating, and moving large scale scientific data sets. The service model differs from our work in that the target data resource is a database of large scale data sets. R-GMA [12] is a stream detection system for performance monitoring of the grid middleware. It could be cast into the architecture described in this paper, however we envision the data stream store as having value to application users, not grid middleware services. Additionally, R-GMA supports a more limited access language than is provided by our system.

6. Conclusion

Data streams are an increasingly prevalent source of timely data. As streams become more prevalent and the devices that generate them grow increasingly smaller, demand for richer interrogation of the contents of the streams will be required. In this article we have defined a model for bringing stream systems to the grid. The model is based on the acknowledgment that viable stream systems already exist, so the model should be general enough to provide a means by which these existing systems can be brought to the grid. The model applies to stream systems that can be considered a “data

resource”. The criteria for inclusion as a data resource is established. The article then lays out an architecture for the model, and identifies the underlying research challenges. We are currently realizing this architecture by means of our dQUOB system [22].

Acknowledgements

The author would like to thank the reviewers for their insightful comments, Ying Liu, of Indiana University for her work on the interface specification, and the UK OGSA-DAI team for productive discussions.

References

- [1] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul and S. Zdonik, Aurora: A new model and architecture for data stream management. *In Very Large Database (VLDB) Journal* **12**(2) (August 2003), 120–139.
- [2] M. Altmel and M.J. Franklin, *Efficient filtering of XML documents for selective dissemination of information*, in Proceedings of 26th VLDB Conference, 2000.
- [3] M. Antonioletti, M. Atkinson, S. Malaika, S. Laws, N.I Paton, D. Pearson and G. Riccardi, *Grid data service specification*, in Global Grid Forum GWD-R, September 2003.
- [4] M. Antonioletti, N.C. Hong, A. Hume, M. Jackson, A. Krause, J. Nowell, C. Palansuriya, T. Sugden and M. Westhead, *Experiences of designing and implementing grid database services in the OGSA-DAI project*, in Global Grid Forum Workshop on Designing and Building Grid Services, September 2003.
- [5] S. Babu and J. Widom, *Continuous queries over data streams*, in: International Conference on Management of Data (SIGMOD). ACM Press, 2001.
- [6] S. Benford et al., *e-Science from the antarctic to the GRID*, in Proceedings of UK e-Science All Hands Meeting, September 2003.
- [7] M. Beynon, R. Ferreira, T. Kurc, A. Sussman and J. Saltz, *Dat-acutter: Middleware for filtering very large scientific datasets on archival storage systems*, in Eighth Goddard Conference on Mass Storage Systems and Technologies/17th IEEE Symposium on Mass Storage Systems, College Park, Maryland, March 2000.
- [8] L. Chen, K. Reddy and G. Agrawal, *GATES: A grid-based middleware for distributed processing of data streams*, in Proceedings Thirteenth IEEE International Symposium on High Performance Distributed Computing (HPDC), Honolulu, Hawaii, IEEE Computer Society, June 2004..
- [9] C. Cranoe, T. Johnson, V. Shkapenyuk and O. Spatscheck, *Gigascope: a stream database for network applications*, in International Conference on Management of Data (SIGMOD), ACM Press, 2003.
- [10] G. Eisenhauer, *The ECho event delivery system*, Technical Report GIT-CC-99-08, College of Computing, Georgia Institute of Technology, 1999. <http://www.cc.gatech.edu/tech.reports>.
- [11] R. Garder et al., *Grid2003 production grid: Principles and practice*, in Proceedings Thirteenth IEEE International Symposium on High Performance Distributed Computing (HPDC), Honolulu, Hawaii, IEEE Computer Society, June 2004. .

- [12] S. Fisher, *Relational model for information and monitoring*, in Global Grid Forum, GWD-Perf-7-1, 2001.
- [13] I. Foster and C. Kesselman, Globus: A metacomputing infrastructure toolkit, *International Journal of Supercomputer Applications* **11**(2) (1997), 115–128.
- [14] G. Fox and S. Pallickara, An event service to support grid computational environments, *Journal of Concurrency and Computation: Practice and Experience. Special Issue on Grid Computing Environments* (2002).
- [15] D. Gawlick and S. Mishra, *Information sharing with the Oracle database*, <http://otn.oracle.com>.
- [16] L. Golab and M.T. Oszu, Issues in data stream management, *SIGMOD Record* **32**(2) (June 2003), 5–14.
- [17] A.K. Gupta and D. Suciu, *Stream processing of Xpath queries with predicates*, in International Conference on Management of Data (SIGMOD). ACM Press, 2003.
- [18] L. Liu, C. Pu and W. Tang, *Continual queries for internet scale event-driven information delivery*, IEEE Transactions on Knowledge and Data Engineering, Special issue on Web Technologies, January 1999.
- [19] S. Madden and M.J. Franklin, *Fjording the stream: An architecture for queries over streaming sensor data*, in International Conference on Data Engineering (ICDE), 2002.
- [20] B. Nguyen, S. Abiteboul, G. Cobena and M. Preda, *Monitoring XML data on the web*, in International Conference on Management of Data (SIGMOD). ACM Press, 2001.
- [21] C. Nippl, R. Rantza and B. Mitschang, *Streamjoin: A generic database approach to support the class of stream-oriented applications*, in International Database Engineering and Applications Symposium IDEAS, 2000.
- [22] B. Plale and K. Schwan, Dynamic querying of streaming data with the dQUOB system, *IEEE Transactions in Parallel and Distributed Systems* **14**(4) (April 2003), 422–432.
- [23] R. Ribler, J. Vetter, H. Simitci and D. Reed, *Autopilot: Adaptive control of distributed applications*, in IEEE International High Performance Distributed Computing (HPDC), August 1999.
- [24] B. (Plale) Schroeder, S. Aggarwal and K. Schwan, *Software approach to hazard detection using on-line analysis of safety constraints*, in Proceedings 16th Symposium on Reliable and Distributed Systems SRDS97, IEEE Computer Society, October 1997, pp. 80–87.
- [25] N. Sivaramakris, T. Kurc, U. Catalyurek and J. Saltz, Database support for data-driven scientific applications in the grid, *Parallel Processing Letters* **13**(2) (2003).
- [26] W3C. Simple object access protocol (SOAP) 1.1. www.w3c.org/2000/xp/Group, 2000.
- [27] H. Wang, D. Estrin and L. Girod, *Preprocessing in a tiered sensor network for habitat monitoring*, in EURASIP JASP special issue of sensor networks, 2002.
- [28] Y. Yao and J. Gehrke, *Query processing for sensor networks*, in First Biennial Conference on Innovative Data Systems Research, Asilomar, CA, January 2003.
- [29] S. Zhuang, B. Zhao, A. Joseph, R. Katz and J. Kubiawicz, *Bayeux: An architecture for scalable and fault-tolerant wide area data dissemination*, in Proceedings Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001), June 2001.