

Using Global Snapshots to Access Data Streams on the Grid

Beth Plale

Indiana University, Bloomington IN 47405, USA
plale@cs.indiana.edu
<http://cs.indiana.edu/~plale>

Abstract. Data streams are a prevalent and growing source of timely data. As streams become more prevalent, richer interrogation of the contents of the streams is required. Value of the content increases dramatically when streams are aggregated and distributed global behavior can be interrogated. In this paper, we demonstrate that access to multiple data streams should be viewed as one of deriving meaning from a distributed global snapshot. We define an architecture for a data streams resource based on the Data Access and Integration [2] model proposed in the Global Grid Forum. We demonstrate that access to streams by means of database queries can be intuitive. Finally, we discuss key research issues in realizing the data streams model.

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 [12]. 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 that handle streaming data. But whereas 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 upon demand. Just as it is common today to read starting conditions for an environmental simulation from a file, it should be equally as easy to draw those starting conditions on demand from live data streams.

These on-demand applications will be distributed and will have either significant computational needs or significant data access needs. As such, the Grid is an attractive computing framework because it promotes modularity through a service-oriented computing model, and provides scalability by virtue of its ability to amass resources that cross administrative domains. Early grids, those in existence today that span multiple departments on a university campus or sites on a company intranet, demonstrate the benefits attained from harnessing disparate and widely disbursed computational resources. Existing efforts to date to

integrate stream systems into the grid have been ad hoc in nature [5]. Needed is a general architecture under which existing stream systems can be brought onto the grid. The model needs to be closely aligned with the specifications under development in the Global Grid Forum (GGF) because these specifications are defining the future direction of data access in grid computing.

Flexible access to real-time streaming data on the Grid is based on three requirements:

- *Aggregation of data streams*: as streams become more prevalent, richer interrogation over the streams is required. The value of the 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 coherent meaning of a set of distributed entities such as sensors, not physical hardware.

The contribution of this paper is severalfold. We demonstrate that access to multiple data streams can be viewed as deriving meaning from a distributed global snapshot. We define an architecture for a data streams resource based on the Data Access and Integration [2] model proposed in the Global Grid Forum. We demonstrate that access to streams by means of database queries can be intuitive. Finally, we discuss key research issues in realizing the data streams model. Our current effort is to realize this architecture using our dQUOB system [18].

The term “streams” is very broadly defined. Section 2 addresses this ambiguity by categorizing streaming systems along orthogonal axes in order to expose the essential defining characteristics. In Section 4 we define the virtual stream store as a collection of domain related streams. In Section 5 we identify key research issues. The paper concludes with a discussion of related work and conclusions, Sections 6 and 7 respectively.

2 Data Stream Systems

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. We refer to the act of interrogating a stream as *decision-making*. 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.

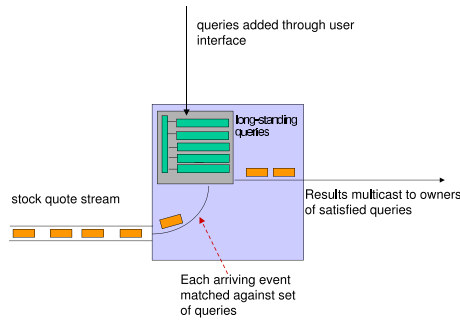


Fig. 1. Stream routing system example

Data stream systems are middleware systems that operate on data streams. These systems fall into three general categories: stream routing systems, data manipulation systems, and stream detection systems. Each is discussed below.

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 systems such as ECho [8], NaradaBroker [10], and Bayeux [23]; selective data dissemination system such as XFilter [1], document filtering system such as Xyleme [16], message-oriented middleware (MOM) [13]. Stream routing systems are distinguished by the locality of the information needed to make the decision. Decisions are made based 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 Figure 1. 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 query arrival. An arriving event is matched against the long-standing queries, and then is 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 on 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 transformative 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 [4, 15, 18, 17]. 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 [18] and DataCutter [6]. In work done at Cornell on ad hoc wireless networking, intermediate nodes aggregate information flowing from sensors to source [22].

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 [9], Autopilot [19], Gigascope [7], changes in HTML pages, such as in Conquer [14], or safety critical systems, such as dQUOB [20]. 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 detection of anomalous behavior might put an emphasis on temporal operators.

3 Distributed Global Snapshot of Stream System

We assert that data manipulation and detection systems taken together form a class of stream systems that meet the criteria of a data resource. A *data resource* is a collection of information that satisfies the properties of coherence and meaning. A relational database has coherence and meaning in that the tables in the database are related to one another and the relationships have meaning. As such, the database is amenable to rich interrogation, analysis, and manipulation.

Stream applications are organized around the production and consumption of data and as such, they export their state or behavior. Unlike a distributed application wherein a distributed global snapshot consists of a snapshot of the processes in a distributed application plus all messages in progress between processes, the distributed global snapshot of a data stream application can be determined from examining the data streams alone. That is, *we can safely draw conclusions about behavior of the distributed application simply by examining the streams*. This defining characteristic makes stream systems quite different from stream routing systems and from distributed systems in general in that embodied in their data streams is a global snapshot. This condition is sufficient for these systems to be considered a data resource in the same way as a database is considered a data resource. That is, the global snapshot over a set of streams satisfies the requirements of coherence and meaning.

4 Architecture for 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 framework 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 provide rich query access to the global snapshot that is inherent in these stream collections. This leads to a definition of a virtual data resource for streams management.

We define the “virtual stream store” as a *collection of distributed, domain-related data streams that satisfy the properties of meaning and coherence. Supporting the stream store is a set of computational resources located in physical proximity to data streams on which query processing can be carried out. The virtual stream store is accessed by means of a grid service that provides query access to the data streams. Event stream providers are external to the virtual store. The streams they generate are external to the store unless explicitly published to it. Data streams produced as a product of data stream processing (i.e., views) are automatically part of the virtual stream store.*

An example data stream store, illustrated in Figure 2, consists of nine data streams and associated computational resources. The computational resources, C_i , 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 C_1 , C_2 , and C_4 , 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 [4, 11]. 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. In the following section we probe more deeply into the suitability of a database access interface for a data stream store and define open research issues.

5 Details

We have demonstrated that the virtual stream store architecture supports an important class of data streaming problems on the grid. From the numerous systems cited in Section 2 based on database concepts, it should be clear that database queries are a viable way to access stream data. For example, a practical solution to distributed monitoring is for a nuclear reactor monitoring system to feed its streaming data into a time-based database and perform post mortem

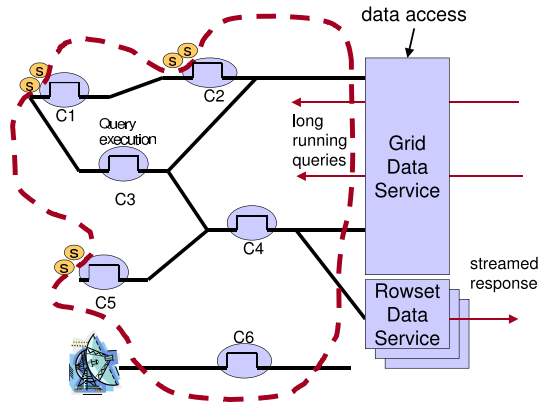


Fig. 2. Virtual stream store within thick dotted lines accessed through a grid service. The data providers are external to the store

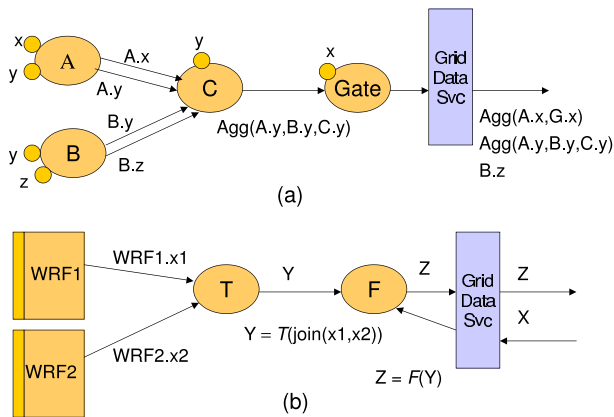


Fig. 3. Data distribution in a sensor network

processing of the data by means of issuing queries. The difference between this approach and an on-line approach should be one of implementation, not access interface. In this section we identify open research issues, but first set the stage by giving two examples from existing data stream systems.

In an example taken from [22], sensor nodes are limited computational devices equipped with several kinds of sensors (e.g., temperature, light, PIR). 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 Figure 3(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 [18] where compute nodes are located on the path between the parallel model (WRF1, WRF2) and its visualization; see Figure 3. 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 3(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.

5.1 Research Issues

Integrating data streams from instruments and sensors into grid computing raises a number of research 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 [3] 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 Figure 3(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 [18], 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 language. 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 [2] 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 [7], 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 be 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.

6 Related Work

Numerous stream systems have been cited in Section 2. Related efforts in grid services for stream data are smaller in number. The OGSA-DAI project [3] has developed a set of services for connecting databases to the grid based on the GGF Grid Data Services specification. The 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. In the Antarctic monitoring project [5], the authors propose 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.

Narayanan *et al* [21] 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 [9] 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.

7 Conclusion

Data streams are a prevalent and growing source of timely data. As streams become more prevalent, richer interrogation of the contents of the streams are required. We argue in this paper that the value of the streamed content can be dramatically increased when a collection of streams is viewed as interrogating a distributed global snapshot. We define an architecture for a virtual stream store as embodying the global snapshot, and provide access to the store through a grid services Data Access and Integration [2] model. Our current effort is focused on defining access semantics to the virtual stream store, and on providing access to the results for clients who demand highly asynchronous streams and extremely timely results.

References

1. Mehmet Altmel and Michael J. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *Proceedings of 26th VLDB Conference*, 2000.
2. Mario Antonioletti, Malcolm Atkinson, Susan Malaika, Simon Laws, Normal Paton, Dave Pearson, and Greg Riccardi. Grid data service specification. In *Global Grid Forum GWD-R*, September 2003.

3. Mario Antonioletti, Neil Chue Hong, Ally Hume, Mike Jackson, Amy Krause, Jeremy Nowell, charaka Palansuriya, Tom Sugden, and Martin 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.
4. Shivnath Babu and Jennifer Widom. Continuous queries over data streams. In *International Conference on Management of Data (SIGMOD)*, 2001.
5. Steven Benford and et al. e-Science from the antarctic to the GRID. In *Proceedings of UK e-Science All Hands Meeting*, September 2003.
6. M. Beynon, R. Ferreira, T. Kurc, A. Sussman, and J. Saltz. Datacutter: 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.
7. Chuck Cranoe, Theodore Johnson, Vladislav Shkapenyuk, and Oliver Spatscheck. Gigascope: a stream database for network applications. In *International Conference on Management of Data (SIGMOD)*, 2003.
8. Greg 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
9. Steve Fisher. Relational model for information and monitoring. In *Global Grid Forum, GWD-Perf-7-1*, 2001.
10. Geoffrey Fox and Shrideep Pallickara. An event service to support grid computational environments. *Journal of Concurrency and Computation: Practice and Experience. Special Issue on Grid Computing Environments.*, 2002.
11. Dieter Gawlick and Shailendra Mishra. Information sharing with the Oracle database. 2003.
12. Lukasz Golab and M. Tamer Ozsu. Issues in data stream management. *SIGMOD Record*, 32(2):5–14, June 2003.
13. Ashish Kumar Gupta and Dan Suciu. Stream processing of Xpath queries with predicates. In *International Conference on Management of Data (SIGMOD)*, 2003.
14. Ling Liu, Calton Pu, and Wei Tang. Continual queries for internet scale event-driven information delivery. *IEEE Transactions on Knowledge and Data Engineering, Special issue on Web Technologies*, January 1999.
15. Sam Madden and Michael J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *International Conference on Data Engineering ICDE*, 2002.
16. Benjamin Nguyen, Serge Abiteboul, Gregory Cobena, and Mihai Preda. Monitoring XML data on the web. In *International Conference on Management of Data (SIGMOD)*, 2001.
17. Clara Nippl, Ralf Rantzau, and Bernhard Mitschang. Streamjon: A generic database approach to support the class of stream-oriented applications. In *International Database Engineering and Applications Symposium IDEAS*, 2000.
18. Beth Plale and Karsten Schwan. Dynamic querying of streaming data with the dQUOB system. *IEEE Transactions in Parallel and Distributed Systems*, 14(4):422 – 432, April 2003.
19. Randy Ribler, Jeffrey Vetter, Huseyin Simitci, and Daniel Reed. Autopilot: Adaptive control of distributed applications. In *IEEE International High Performance Distributed Computing (HPDC)*, August 1999.

20. Beth (Plale) Schroeder, Sudhir Aggarwal, and Karsten Schwan. Software approach to hazard detection using on-line analysis of safety constraints. In *Proceedings 16th Symposium on Reliable and Distributed Systems SRDS97*, pages 80–87. IEEE Computer Society, October 1997.
21. Narayanan Sivaramakris, Tahsin Kurc, Umit Catalyurek, and Joel Saltz. Database support for data-driven scientific applications in the grid. *Parallel Processing Letters*, 13(2), 2003.
22. Yong Yao and Johannes Gehrke. Query processing in sensor networks. In *First Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, January 2003.
23. 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.