Tracking Stream Provenance in Complex Event Processing Systems for Workflow-Driven Computing *

Nithya N. Vijayakumar and Beth Plale
Department of Computer Science
Indiana University
{nvijayak, plale}@cs.indiana.edu

ABSTRACT

Workflow-driven, dynamically adaptive e-Science is a form of scientific investigation often using a Service-Oriented Architecture (SOA) paradigm, designed to use large-scale computational resources on-the-fly to execute workflows consisting of parallel models, analysis, and visualization tasks. In the Linked Environments for Atmospheric Discovery (LEAD) project, with which our team is involved, our research has centered around event processing and mining of observational and model generated weather data such that users can dynamically trigger regional weather forecasts on-demand in response to developing weather.

In this paper we describe stream provenance in complex event processing (CEP) systems. Specifically, we give an information model and architecture for stream provenance capture and collection, and evaluate the provenance service for perturbation and scalability.

Keywords

Event Driven Architectures, Provenance, Mining, Meteorology, Load Forecasting

1. INTRODUCTION

Computational science research has fully embraced the sensor and instrument revolution by, among other things, incorporating streaming events from these devices into their real-time analysis models (such as fluid dynamics or finite element analysis). This is applicable to a large number of areas including modeling the power grid, wild fires, building structures under stress, traffic flows, and hurricane and tornado forecasting. Dynamic data-driven application systems (DDDAS) [5], a term describing these kinds of systems, is gaining some currency in the research community. DDDAS systems are often in the form of workflow-driven middleware built according to the principles of service-oriented architectures (SOA). The middleware provides the dynamic capabilities needed by computational science applications. The component functionality is provided by web services that are described by WSDL and communicate through SOAP and XML based messages. Services are generally either persistent or created on-the-fly, and are discoverable through a service registry. The service abstraction can be extended to include not only the middleware but the application specific models, mining and analysis, and visualization tasks as well.

In the Linked Environments for Atmospheric Discovery (LEAD) [10] project with which our team is involved, we have constructed a distributed SOA that provides meteorology researchers and students secure access to complex weather forecasting models, assimilation and visualization codes, and data through a web portal. Complex event processing (CEP) plays a key role in LEAD because it enables real-time response to the weather. This support means scientists can, for example, set up a 6-hour weather forecast over a region of say a 700 sq. mile bounding box, and submit a workflow that will run sometime in the future, where that point in time is determined by when an SQL-based complex event processing (CEP) query detects severe storm conditions developing in the region that the scientist has specified.

Event processing provides the critical link between the user-driven workflows and the external environment (the weather), making the former responsive to the latter in highly customized ways. Weather triggered workflows, which our research has enabled, are a scientific research outcome of LEAD, not having been demonstrated in a repeatable way before. In LEAD, we use an SQL-based event processing system called Calder [24, 12] to mine the real-time weather data and trigger forecast workflows [26]. Figure 1, depicts the relationship of Calder stream mining service to other LEAD components, with communication carried out through an event notification bus.

shows the services in LEAD and their interconnection through a common event notification bus.

In this paper we describe weather-focused triggering in the LEAD SOA, but the main focus is on showing how provenance collection can be applied to a CEP system to extend its usefulness. Data provenance is information that helps determine the derivation history of a data product, starting from its original sources [20]. This definition is a useful starting point for defining provenance in the context of a CEP system. Our model of an SQL-based CEP system is one of queries responding to input streams event-by-event within a

*This project is funded by DOE DE-FG02-04ER25600, NSF EIA-0202048 and NSF ATM-0331480

VLDB '07, September 23-28, 2007, Vienna, Austria.

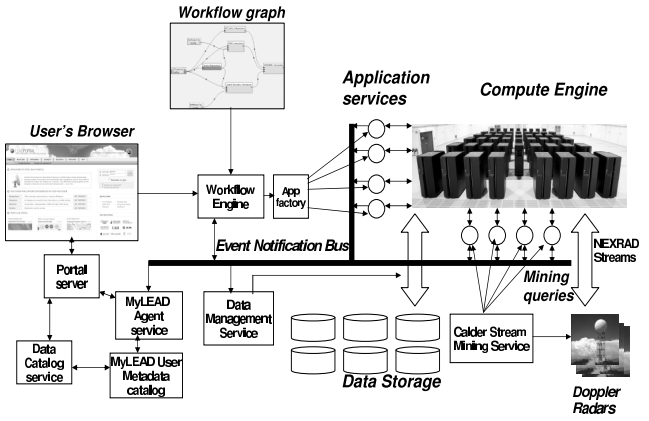

Figure 1: Relationship of Calder stream mining service and other services in LEAD SOA.

stream and concurrent across streams. The query generates a time-bounded output stream. Provenance of this system is in essence the historical information about the activity of, or applied to, the entities in the system. When the entities are queries and streams, one might be interested in understanding which data streams are mined more frequently than others. If one also captures queries and events, more interesting trends can emerge over time. One can start to see group behavior, which we discuss further below. *Data stream provenance then is information that helps determine the derivation history of a data product where the data product is the derived time-bounded stream.*

In stream mining of the weather the events that are of interest are the anomalies. A significant increase in event production observed over a short period of time, for instance, could mean a weather phenomena is attracting interest. While we are fully cognizant of the impact a large-scale weather phenomena such as hurricane Katrina that devastated the Gulf Coast in 2005 might have on the SOA, this is the first system to provide on-demand weather forecasting to a broad audience, so we do not yet have a good picture of overall system utilization. It may be that small scale phenomena will, on the whole, create the majority of the workload for the system. We need to study these behaviors, and a provenance system can help.

Stream provenance captured about streams and queries during use can additionally establish such things as correlations between significant events (e.g., storm occurrences). With this information, a client tool could anticipate resource needs by examining the provenance data and discover trends in weather forecast model output, to determine when the next wave of users will arrive, and where their resources might need to be allocated.

As stated earlier, the purpose of this paper is a framework for provenance collection in a CEP system. We describe an information model and collection framework that extends the model described in [25]. We further describe a performance study on the provenance collection system done to understand its behavior under conditions requiring scalability.

The prototype provenance service is implemented as part of the Calder distributed events processing system [24, 12]. The Calder system enables execution of SQL-based continuous queries on data streams. It uses a query planner service that optimizes and distributes queries to the computational nodes [11]; sophisticated algorithms to join streams with asynchronous arrival rates [16, 17]; and a Kalman filter based approximation technique that estimates the input values when there are gaps in stream data [23].

The remainder of the paper is organized as follows. Section 2 discusses related work in provenance collection. A brief overview of the LEAD SOA is given in Section 3. The information model is given Section 4 and collection framework in 5. Section 6 describes the perturbation overhead and scalability of the provenance service. Conclusions and future work are discussed in Section 7.

2. RELATED WORK

The most common use of provenance has been for establishing the credibility of data and for reproducibility. Provenance management in different applications has been widely explored in the three recent workshops [28, 27, 3]. In scientific, engineering and business workflows, typically data is repeatedly copied, corrected, and transformed as it passes through numerous databases or services. Understanding where data has come from and how it arrived in a database or filestore is of crucial importance to the trust a user will put in that data, yet this information is seldom properly captured [3]. The scientific and grid communities track and integrate provenance information of workflows in order to support reproducibility [6].

Several provenance tracking solutions exist for e-Science [20]. The Virtual Data Grid [7] supports an integrated treatment of not only data but also the computational procedures used to manipulate data and the computations that apply those procedures to data. The Karma Provenance Service [19] is used as a focal web service to collect and query over data and workflow provenance. PASOA [21, 8] investigates the concept of provenance and its use for reasoning about the quality and accuracy of data and services in the context of e-Science.

As far as we know, our provenance approach is the first to target stream provenance. The dynamic adaptation scenario of mining NEXRAD II data for storms and mesoscale cyclones was discussed briefly in [26]. This paper on the other hand focuses on the collection framework and extensions to the information model that enables forecasting of load on the system.

3. MOTIVATION

The LEAD Portal and its service are available for use 24x7. Many of the middleware services run as persistent services on the LEAD grid, a networked collection of clusters spanning University of North Carolina Chapel Hill, National Center for Atmospheric Research, Indiana University, Oklahoma University, National Center for Supercomputing Applications and University of Alabama Huntsville. The models, which are highly parallel, are deployed on TeraGrid resources as needed. Streaming data is available at each site through the Unidata Internet Data Dissemination System [4]. The sites are connected by means of Internet2. This is depicted in Figure 2.

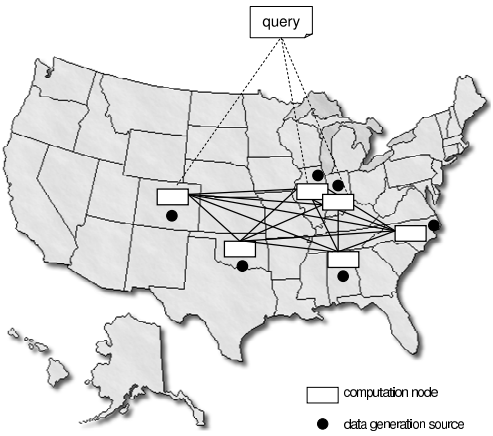

Figure 2: Event processing queries can be deployed to any of the sites in the LEAD grid (rectangles) where there is a high bandwidth Internet 2 connection between sites. Base data streams are delivered to each site through the Unidata Internet Data Dissemination system.

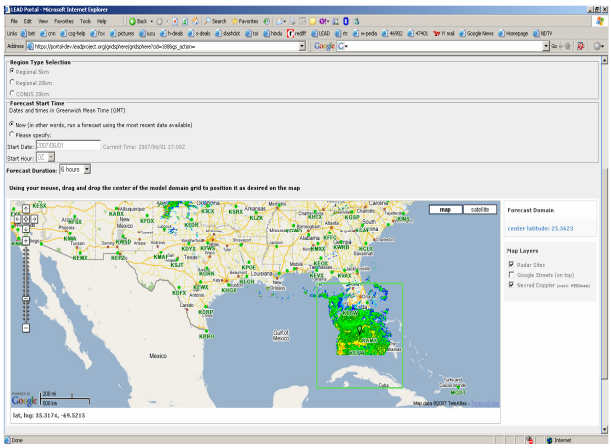

Figure 3: Workflow configuration in LEAD portal.

User interaction with the system is through the LEAD portal [10], where a user can log in and either create a new workflow or use an existing template. One of the input screens, shown in Figure 3, allows the user to configure parts of the workflow by specifying among other things, the geospatial region over which the forecast and assimilation is to run, the resolution of the model (grid spacing), and the forecast duration (that is, how far into the future the forecast will predict. A 6 hour forecast has a far higher probability of being right at the outer bound than would an 84 hour forecast.) The geospatial range selected in this figure is the same geospatial range used to determine the area over which the CEP event detection will take place. This is why the locations of the WSR-88D Doppler radars (NEXRAD) are overlayed on the map. In the figure, the geospatial range is a box layed over parts of Florida and Gulf of Mexico.

Event detection is carried out over Level II data [14] produced by the NEXRAD radars. Filtering, aggregation, self-joins are supported by the Calder events processing system. The data mining algorithms are part of the ADaM data mining toolkit developed at University of Alabama Huntsville [18].

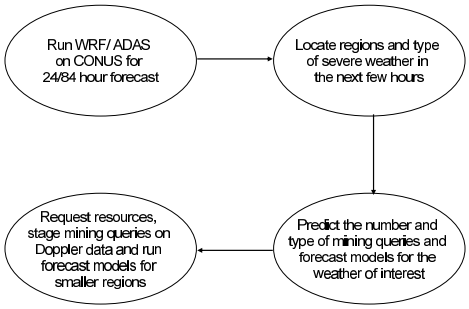

Figure 4: Motivating scenario: Prediction of resource usage in dynamically adaptive computing.

The main atmospheric model components that the LEAD SOA supports are the Weather Research Forecast model (WRF) [29] model and components of the Advanced Regional Prediction System (ARPS) [2]. The WRF model is a mesoscale numerical weather prediction system used both in operational forecasting and as a research vehicle for atmospheric research needs. WRF is highly configurable, taking in some 1000 parameters. ARPS is a complete system that includes a real-time data analysis and assimilation system, the forward prediction model and a post-analysis package. ADAS [1] assimilation draws in real-time data from multiple data sources, including Doppler radar values to initialize a model for short-term forecasts.

The provenance collection framework we describe in this paper can be used to interact with these models in useful ways as follows (see Figure 4). The results of a WRF run over the entire Continental US that produces a 24 hour forecast and an 84 hour forecast, will give a meteorologist or a software agent acting on behalf of a user a good picture of what is to come. We can use this information to anticipate events processing usage, and reduce startup time by staging the data mining algorithms to the computational node in advance. The provenance service's historical view of all the queries submitted in the past can be mined for this and other promising solutions.

4. STREAM PROVENANCE INFORMATION MODEL

The information model for the provenance service discussed in this section, which is an extended version of the model presented in [25] consists of streams and queries as its primary entities. Streams are one of two types: base streams or derived streams. Base streams originate outside the events processing system, such as at an instrument, sensor network, or a service. Derived streams are produced by event processing that takes place on base streams or other derived streams. The user interface to the events processing system is a declarative SQL query language in-

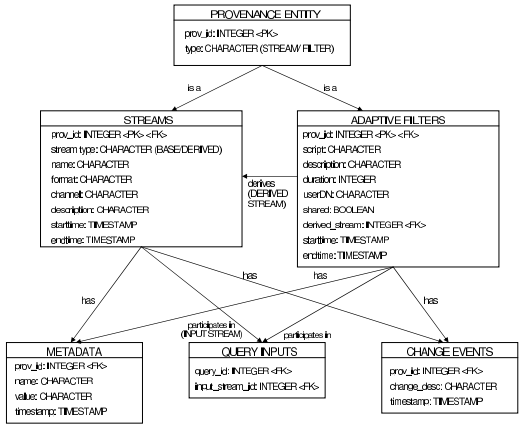

Figure 5: Provenance information model.

terface. Specifically, the Calder system supports a subset of SQL extended with special operators that invoke user-defined functions during query execution. The queries are time-bounded, that is they have an associated lifetime after which they are removed from the system.

The information model shown in Figure 5 consists of six major entities that are related to one another. Streams contain change-events, participate in query-inputs, and have metadata. Adaptive filters (the queries) generate change-events, accept query-inputs, and also have metadata. The relations between the entities are sufficient for us to construct a graph through time of the derivation history of one anomaly event.

The provenance information for the entities is collected in parts: static provenance - information gathered during registration of streams and queries; and dynamic provenance - information gathered during the processing. Metadata is part of the static provenance while a change to the stream rate is an example for dynamic provenance. In the next section, we discuss the implementation of the provenance service and the instrumentation of the components in the Calder system to enable provenance collection.

5. ARCHITECTURE

Calder [24, 12] is a distributed events processing system with a centralized service that accepts query requests, optimizes and deploys the queries across distributed query execution engines where they are executed for the duration of their lifetime. Calder monitors load at the query processing engines and can spawn new query execution engines on-demand to distribute load. The query planner distributes incoming queries to the available query execution engines based on query optimizations, reuse rules, and predicted query and network costs [11]. The provenance service of Calder implements the provenance models described in the previous section and supports a service oriented interface for querying the provenance information. Its architecture is outlined in Figure 6.

During execution, the query processing engines and queries themselves will send updates to the provenance service. When an event of interest occurs, the provenance information propagates as shown in Figure 6. In this figure, we have distinguished the different services of Calder and their interactions. The query planner is responsible for deploying the filtering query. The resource catalog service, is a service that maintains the current state of the system by maintaining a database view over the provenance data. It interacts with the query planner and the provenance service thus keeping track of the current resources in the system. It also provides a GUI for monitoring the current queries, streams and computational resources in the system.

Users register the input streams and filter queries by invoking the provenance service. Registration of derived stream is made by the system when a new query is submitted. Once a stream/filter query is registered, users can append its provenance dataset with additional information like annotations and metadata. The static provenance data is stored in the provenance repository. The provenance service is updated by the query planner and query execution engines when an event of interest occurs. Events of interest include:

- a new query is started
- the execution plan of a query changes due to a set of streams going down or a processing node failure
- an existing query expires
- stream rate changes
- missing events in a stream
- stream/query approximations and accuracy changes

5.1 Instrumentation

Any system that is gathering information about the same or different system, as it executes, has to deal with the cost in terms of performance loss of the instrumentation. Instrumentation, often called 'sensors' and 'actuators' are added to the observed system. When an instrumentation point is reached, an event is generated and sent to a global observer, which in our case is the provenance service.

For the provenance collection, we extend the provenance service of Calder with a monitoring thread for dynamic collection of provenance information. The provenance information model is updated based on these events. The flexible service interface of Calder enables a scalable framework. We adopt a publish-subscribe paradigm for communication between Calder components and the provenance service. This enables a lower overhead on the sender end. By adopting a reliable messaging service at the lower level we can reduce the overhead on the sending components to keep track of updates that have been registered with the provenance service and those that fail. We used WS-Messenger [9], a web services based message broker that can decouple event producers and event consumers and achieve scalable, reliable and efficient message delivery [9].

The query planner updates the provenance service when a new query is submitted, when an existing query expires, or when the distribution plan changes. To enable this, the query planner, implemented as a JAVA webservice, is instrumented with a monitoring thread that uses a WS-Messenger client jar file that sends updates to the provenance service.

At the query execution engine, monitoring is conducted on a per stream basis by a newly introduced provenance generation operator. Stream provenance tracking captures, among other information, the mode changes that streams undergo, and accuracy information about the stream. The

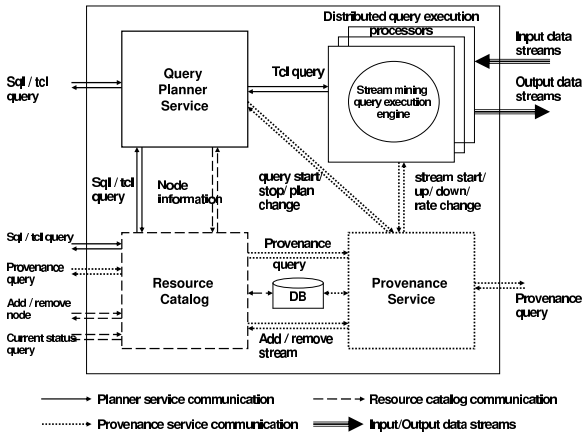

Figure 6: Architecture of stream provenance collection.

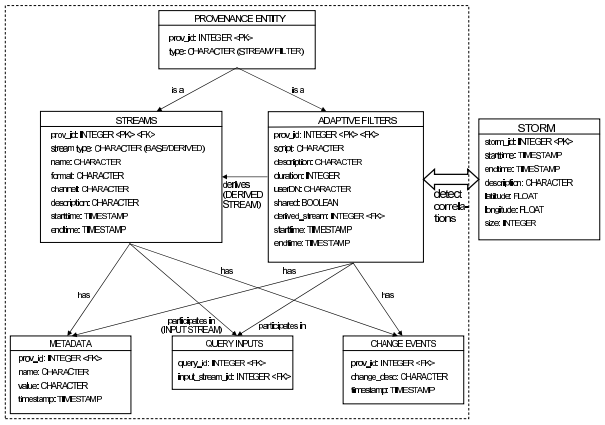

Figure 7: Provenance information model relationship to Storm entity.

former are detected as rate changes in the streams. In meteorology, this might be a rate change that comes about when a radar shifts from operating in "clear" mode to operating in "storm" mode. In the latter mode, the radar does more scans of the sky in an effort to collect more information. Instrumentation in stream provenance tracking can be done very cleverly when the complex event processing uses a declarative query language such as SQL.

Taking the approach of using a query operator to instrument code is a unique contribution to the literature of instrumentation in its own right. The declarative language nature of SQL opens a door for inserting sensors without users being aware. Declarative queries must undergo optimization and instantiation into a procedural representation anyway, so inserting an additional operator or two into the parse tree can be done without changing the semantic meaning of the query or the results. It is quite natural to insert a provenance collection instrumentation into the query plan as additional "hidden" operators. When the operator is reached, it generates an event that is sent to the provenance service. This operator is responsible for sampling input streams, detecting rate changes and missing events and updating the provenance service correspondingly. The query execution engine in Calder is implemented in C++ and sends XML WS-Messenger messages using a socket connection to the WS-Messenger broker. The provenance service subscribes to the WS-Messenger broker to receive the corresponding update messages.

An important criterion for provenance collection in a distributed stream processing system is that the query execution processors have synchronized clocks. This is important because the provenance service receives update information from the query planner, data providers and query execution engines of ordered events based on timestamp. Network Time Protocol (NTP) [15] is a protocol for synchronizing the clocks of computer systems over packet-switched, variable-latency data networks. By ensuring all the nodes in the Calder mesh have NTP installed on them, we keep the nodes more or less synchronized within a few milliseconds. The TeraGrid [22] offers a variety of third-party applications and community codes that are installed on the high-performance systems at the TeraGrid organizations. Currently NTP is used locally in many TeraGrid sites, but not part of the software stack. By enforcing NTP as a required software, we can synchronize the host nodes at all the TeraGrid sites.

5.2 Logging and Prediction

The provenance service uses a relational database with XML front end to store the provenance history of streams and filter queries. When a query is under execution, due to its real-time nature there may be changes in query plans and approximations. These updates are sent to the provenance service as XML change events associated with a timestamp. At the provenance service, the XML events are shredded into a MySQL [13] relational database.

The Calder components and the provenance service communicate using WS-Messenger [9] publish-subscribe system. WS-Messenger uses a notification broker for communication between services. The notification broker provides an abstraction layer between an event source and an event consumer so that they can communicate without knowing the location of each other. The event consumer can be offline when the event source publishes an event. The event source is relieved from the burden of handling subscription registrations and delivering events to all the event consumers. WS-Messenger uses a database to catalog the subscriptions. If the WS-Messenger server accidentally crashes, previous subscriptions can be retrieved from the database to restore the WS-Messenger server to the status before the crash. The database is also designed to temporarily save undeliverable messages to support reliable message delivery. In our experiments we found that the WS-Messenger tries at least twice to deliver a message before saving it as being undelivered.

To enable the prediction of mining query load under different stormy conditions, we append the provenance model to include storm information as shown in Figure 7. This extension is specific to the motivating scenario discussed in Section 3. To enable such a scenario we need to establish correlations between the storms and the queries submitted on them. In our provenance implementation, we do this using geographical co-ordinates and timestamps. The size and location of a storm can be described by its center latitude and longitude (eye of the storm) and its size (radius). Data mining queries are submitted on the Doppler radars

that correspond to the region of interest. Identifying the radars that fall within a storm region and mapping it with the queries submitted at the relevant time, enables us to predict the mining query load on the system. By looking at the type of mining algorithms used in these queries, we establish associations between the storms and the mining queries.

Resources for hosting the queries are obtained by placing requests to the TeraGrid computational nodes [22].

6. EXPERIMENTAL EVALUATION

We performed perturbation overhead and performance scalability experiments on the provenance service of Calder. The perturbation overhead can be defined as the overhead imposed on the normal functioning of the system due to provenance collection. Our first set of experiments measure the perturbation overhead imposed on the query planner service and the query execution engines due to the collection of provenance information. The second set of experiments measure the scalability of the provenance service w.r.t to handling updates and query invocations.

For all experiments, the experimental setup is as follows: The provenance service was hosted on a dual processor 2.8 GHz Dell Precision Workstation with 2 GB memory and running RHEL 4. The query planner service and the computational mesh consisting of four query execution engines, were executed each, on a Dual AMD 2.0 GHz Opteron with 4 GB RAM executing RHEL 4. The user applications for the experiments were executed on a single processor Dell OptiPlex 3.2 GHz machine with 1 GB running RHEL 4. All the host machines were connected with 1 GB Ethernet LAN.

6.1 Perturbation Overhead

Provenance collection incurs a performance loss in the system due to instrumentation. This is typically referred to as the *perturbation* of the collection system. Ideally, we would like the perturbation overhead to be negligible. We measure the perturbation overhead by measuring the time taken to send provenance updates as compared to the overall execution time of the component.

Query Planner Service

The Calder query planner generates a query distribution plan and deploys it at query execution engines. The query planner updates the provenance service in three instances: when a new query is submitted; when an existing query expires, or when the distribution plan changes. The latter may occur when there are changes to available computational resources and query optimizations. As the update sent from the query planner to the provenance service is similar in all three cases, the perturbation overhead will be similar as well. Our first experiment measures the overhead introduced in query submission time by provenance tracking.

The experiment was set up with four query execution engines each running on a single node of a computational cluster. The query planner is configured to distribute queries across nodes in a round robin fashion. For the purpose of this experiment, we configured the query planner to employ no partial or full query reuse, so each new request results in one new query deployed at a query execution engine. A client application submits 100 queries sequentially to the Calder system. We measure the round trip query submission time and the time it takes for the query planner to register the query with the provenance service. As shown in Table 1, the provenance updates impose an overhead of $\sim$ 10 ms on total query submission time, roughly a 10% overhead added to total query submission time, which is reasonable. Overall query submission time may vary slightly based on how the query planner service is configured for optimizations and reuse, but the provenance overhead is constant.

Table 1: Provenance perturbation overhead on query submission time.

Measure	Mean (ms)	STD (ms)
Query submission time	95.218	18.288
Provenance update overhead	10.873	7.429

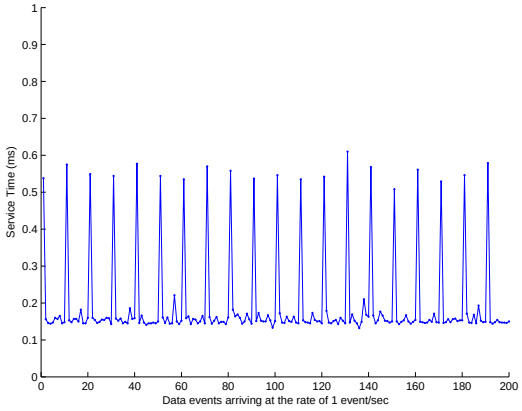

Figure 8: Provenance perturbation overhead on query service time where mode change (provenance collectable) events arrived at a fixed interval.

Event Generation Overhead

In Calder, we instrument the queries for provenance collection. The query execution engine incurs some overhead in throughput because of the instrumented queries. The provenance collection operator generates events when a significant change in the rate of the stream is detected or when the accuracy of the stream data changes.

In this experiment we measure the overhead incurred on the query execution engine as a result of the provenance monitoring. We set up the query execution engine to have a single query running. The query accepts a single input stream that is streaming events at a rate of 1 event/second. The provenance collection operator is configured so that it generates a provenance event every 10th event. This fixed cycle is somewhat artificial, but does not compromise the overall measurements, and it provides an easy to read graph of the stability of provenance collection base overhead.

The cost metric of this experiment is *service time*. This is the elapsed time from instant an event enters the query execution system until it is processed and sent out as part of the derived stream. In order to keep the results simple, the query we use is a select all query executing on a single stream.

The service time under normal processing (without provenance updates) is 0.15 ms. As shown in Figure 8, at every 10th event, the service time spikes to 0.6 ms, an increase

of $\sim$ 0.45 ms. The increase in service time is attributed to the cost of connecting to the WS-Messenger broker [9] and sending the XML update message. We observe that collection imposes a 3x overhead on service time, but note that provenance updates are generated when something changes in the system like a stream going down, coming back up or a significant rate change. The service time is dependent on the query and typically mining queries have service time in the range of 600 ms [26], while the overhead for sending a provenance update remains constant.

6.2 Scalability Analysis

The performance of a system can be measured by analyzing its ability of scale w.r.t to users. We subject the provenance service to simultaneous updates from multiple sources and query invocations from multiple users and measure the response time.

Handling updates

This experiment measures the ability of the provenance service to handle multiple updates at the same time. We measured the processing time of the provenance service - the time taken from the instant the user sends a notification to the instant the provenance service completes the corresponding update. This update processing time gives a measure of how up-to-date the system is. We bombarded the provenance service at different update rates by simulating many clients sending provenance updates simultaneously. The size of each update XML message is 340 bytes. We measured the incoming rate at the provenance service and the overall time taken for handling each update.

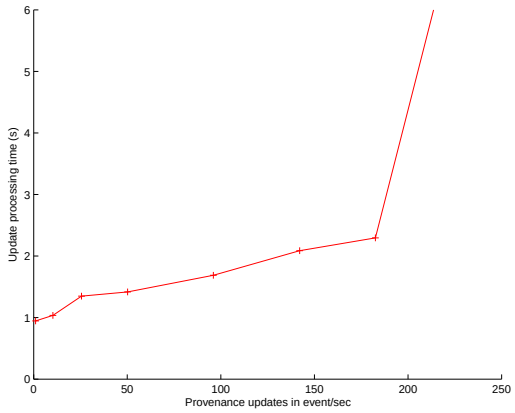

Figure 9: Update handling time of the provenance service under increasing load.

Figure 9 plots the incoming update rate on the X-axis and the average update processing time on the Y-axis. The update processing time is nearly 1 second per update, for it includes the time for marshalling and unmarshalling XML messages, communication latency and the time taken by the provenance service to update the database. From Figure 9, we can see that the provenance service can handle updates at the rate of 200 updates/sec before saturating. Once saturation is reached, the provenance service can no-longer provide an up-to-date view of the system.

The updates collected by the provenance service denote changes to the system. From our experience, we believe that 200 updates/sec is sufficient for our application because collection is for major state changes, so frequency of provenance event generation per generation source is low.

Query Invocations

The fourth experiment measures the round-trip request response time experienced by an user invoking the query interface of the provenance service in the presence of multiple other users. The request response time is the total time as seen by the user for one invocation of the provenance service. We simulated multiple users invoking the provenance service simultaneously. Each user was made to sequentially submit multiple provenance queries. At any point of time, the provenance service had a specific number of users invoking it. We measured the request response time with different number of users for different result sizes. The experimental setup is the same as described in the beginning of this section.

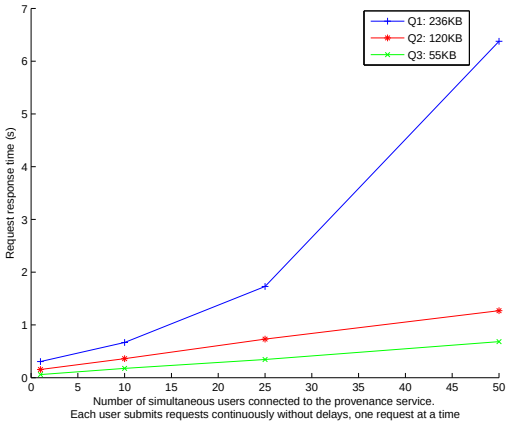

Figure 10: Provenance query invocation from perspective of user.

Figure 10 shows the number of users in the X-axis and the request response time in the Y-axis. The round trip request response time includes the communication latency and the time taken by the provenance service for processing the request. We measured the response time across multiple users for three different result sizes as shown in the Figure 10. A larger result size can be associated with increase in processing time of the query at the provenance service as well as increased latency for marshalling, transmitting and unmarshalling the results through WS-Messenger. From Figure 10, it can be seen that the response time increases linearly with the result sizes and the number of users.

7. CONCLUSION

We describe an information model and collection framework for provenance collection in a complex event processing system and its implementation in the Calder events processing system which is used for mining real-time weather data in LEAD. We discuss the scenario of forecasting of mining queries in LEAD by extending the provenance model to include storm information. Our approach uses location coordinates and timestamps to establish correlation between

storms and the mining queries. We further discuss the results of our performance study on the provenance collection system and show that our provenance service has a low overhead and scales well with increasing number of updates and queries.

We are interested in further investigating provenance in the context of load forecasting and estimation of resource usage in large-scale scientific applications using complex event processing solutions. Our ongoing research extends the current mining work in three areas: extend the queries to mine streams over time, e.g., if a storm detected at 12:05 triggers a workflow, when the storm is detected again at 12:15 another workflow need not be triggered; mine streams of model generated forecast data instead of observational data thus enabling detection of severe weather conditions to occur further into the future; and support extensions to the events processing system by including format conversion and mining algorithms as web services.

8. REFERENCES

[1] ADAS. http://www.caps.ou.edu/~kbrews/adasfull.html.

[2] ARPS. http://www.caps.ou.edu/ARPS/index_flash.html.

[3] International provenance and annotation workshop, May 3-5, 2006. http://www.ipaw.info/ipaw06/.

[4] M. S. Baltuch. Unidata's internet data distribution (IDD) system: Two years of data delivery. In *Proceedings of the Thirteenth International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology*, Anaheim California, February 1997.

[5] DDDAS: Dynamic Data-Driven Application Systems. http://www.dddas.org/.

[6] E. Deelman and Y. Gil, editors. *Workshop on the challenges of scientific workflows*. May 2006. Technical Report, Information Sciences Institute, University of Southern California.

[7] I. Foster, J. Vockler, M. Wilde, and Y. Zhao. The virtual data grid: A new model and architecture for data-intensive collaboration. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2003.

[8] P. Groth, M. Luck, and L. Moreau. A protocol for recording provenance in service-oriented grids. In *International Conference on Principles of Distributed Systems*, 2004.

[9] Y. Huang, A. Slominski, C. Herath, and D. Gannon. WS-Messenger: A web services based messaging system for service-oriented grid computing. In *6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, 2006.

[10] LEAD: Linked environments for atmospheric discovery. http://www.leadproject.org.

[11] Y. Liu and B. Plale. Multi-model based optimization for stream query processing. In *KSI Eighteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'06)*, July 2006.

[12] Y. Liu, N. N. Vijayakumar, and B. Plale. Stream processing in data-driven computational science. In *The 7th IEEE/ACM International Conference on Grid Computing*, Barcelona, Spain, 2006.

[13] Mysql. http://www.mysql.com/.

[14] NEXRAD Level II Data. http://www.roc.noaa.gov/NWS_Level_2/.

[15] Network Time Protocol. http://www.ntp.org/.

[16] B. Plale. Leveraging run time knowledge about event rates to improve memory utilization in wide area data stream filtering. In *Proceedings Eleventh IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pages 171 – 176, Edinburg, Scotland, August 2002. IEEE Computer Society.

[17] B. Plale and N. Vijayakumar. Evaluation of rate-based adaptivity in joining asynchronous data streams. In *Proceedings of International Parallel and Distributed Processing Symposium*, Denver, USA, 2005. IEEE Computer Society.

[18] J. Rushing, R. Ramachandran, U. Nair, S. Graves, R. Welch, and A. Lin. ADaM: A data mining toolkit for scientists and engineers. In *Computers and Geosciences*, volume 31, pages 607–618, 2005.

[19] Y. Simmhan, B. Plale, and D. Gannon. A framework for collecting provenance in data-centric scientific workflows. In *International Conference on Web Services*, 2006.

[20] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *ACM SIGMOD Record*, 34(3):31–36, 2005.

[21] M. Szomszor and L.Moreau. Recording and reasoning over data provenance in web and grid services. In *Intl. conference on ontologies, databases and applications of semantics, Lecture notes in computer science*, volume 2888, pages 603–620, 2003.

[22] TeraGrid. http://www.teragrid.org.

[23] N. N. Vijakumar and B. Plale. Prediction of missing events in sensor data streams using kalman filters. In *To appear in ACM Workshop on Knowledge Discovery from Sensor Data*, 2007.

[24] N. Vijayakumar, Y. Liu, and B. Plale. Calder query grid service: Insights and experimental evaluation. In *Intl Symposium on Cluster Computing and the Grid (CCGrid)*, 2006.

[25] N. Vijayakumar and B. Plale. Towards low overhead provenance tracking in near real-time stream filtering. In *Internationl Provenance and Annotation Workshop, LNCS 4145*, 2006.

[26] N. N. Vijayakumar, B. Plale, R. Ramachandran, and X. Li. Dynamic filtering and mining triggers in mesoscale meteorology forecasting. In *International Geoscience and Remote Sensing Symposium (IGARSS)*, 2006.

[27] Workshop on data provenance and annotation, December 1-3, 2003. http://www.nesc.ac.uk/esi/events/304/.

[28] Workshop on data derivation and provenance, October 17-18, 2002. http://www-fp.mcs.anl.gov/~foster/provenance/.

[29] The Weather Research and Forecasting (WRF) Model. http://www.wrf-model.org/index.php.