Provenance-Aware Sensor Data Storage

Jonathan Ledlie, Chaki Ng, David A. Holland,
Kiran-Kumar Muniswamy-Reddy, Uri Braun, Margo Seltzer
Division of Engineering and Applied Science
Harvard University, Cambridge, MA
pass@eecs.harvard.edu

***Abstract*— Sensor network data has both historical and real-time value. Making historical sensor data useful, in particular, requires storage, naming, and indexing. Sensor data presents new challenges in these areas. Such data is location-specific but also distributed; it is collected in a particular physical location and may be most useful there, but it has additional value when combined with other sensor data collections in a larger distributed system. Thus, arranging location-sensitive peer-to-peer storage is one challenge. Sensor data sets do not have obvious names, so naming them in a globally useful fashion is another challenge. The last challenge arises from the need to index these sensor data sets to make them searchable. The key to sensor data identity is *provenance*, the full history or lineage of the data. We show how provenance addresses the naming and indexing issues and then present a research agenda for constructing distributed, indexed repositories of sensor data.**

I. Introduction

Readings and events emerging from a sensor network may be consumed immediately or stored for later analysis. In many cases it is useful to combine data from distinct sensor networks, and often sensor data is still useful for historical analysis long after it is collected.

For example, while traffic data from London's Congestion Zone is useful immediately to ticket non-paying drivers, it is also useful in other ways: it could be aggregated over time to estimate the effects of changing Zone size, or it could be combined geographically with data from other cities to gather a broader picture of traffic. Even deeper insight might be gained by merging historical traffic data with historical weather data.

Other existing sensor applications that exhibit some or all of these properties include volcano monitoring [31], city-wide structural monitoring [22], biological field research [8], supply chain management [17], and military sensing [26].

In this environment it becomes necessary to be able to name and search for sensor data sets, whether in real-time or in archival storage. Traditional approaches to indexing massive quantities of distributed storage (*e.g.,* content indexes) are not terribly useful when that content is primarily a stream of sensor readings. Clearly, any data set must also have a description, and the data itself may have annotations; for example, one might mark when individual sensors were replaced with newer models having slightly different properties, or when software on the sensor devices was upgraded. Such descriptions and annotations must also be searchable.

These requirements have implications for the organization of storage systems for sensor data. This paper discusses the research challenges related to naming and indexing sensor data, including a discussion of constructing such an index atop a distributed system. We address three questions:

- What are the right attributes to index? We reason in Section II that sensor data must be indexed by its *provenance* metadata: the history of how and when it came to be.
- How will people use historical sensor data? That is, what are the applications for which we are designing? We show in Section III that they will use data both near their source and at a variety of sites elsewhere in the network. In particular, because sensor data is *locale specific*, it may be inappropriate to store it in arbitrary (or randomly chosen) places; instead it should be stored near the network or its primary users.
- What storage architecture will make indexing provenance metadata and serving locale specific clients feasible? In Section IV, we propose a set of evaluation criteria and examine how different models for distributed storage and indexing fare under our requirements.

We present our research agenda in Section V.

II. How to Index

Before we can index anything, we must choose the granularity at which to index it. We could conceivably index every sensor reading, or *tuple*, individually. However, this appears infeasible, due to the sheer number of readings, and also not necessarily useful, as individual sensor readings in isolation have little meaning. A better solution is to index *tuple sets*, collections of readings grouped by some property, typically time. For example, a tuple set might contain all the readings of a particular type over the span of one hour or one minute. To make retrieval practical, each such tuple set must have a unique name.

A. Provenance as Name

Tuple set names could be conventional, self-describing filenames, like `volcano_vesuvius_10_11_04`. However, unstructured strings of this kind incur several problems:

- They require a complicated naming convention, to allow for all the possible things that could be named.
- Such naming conventions are equivalent to a hierarchical index. Section IV-B discusses the resulting problems.
- Strict adherence to the convention is necessary because no enforcement or cross-checking is possible.
- To be fully descriptive, these names may become arbitrarily long. Without special support, handling these names becomes difficult, both for humans and for machines.

- Because the structure of the naming convention is not exposed, these names will be hard to index automatically. They also cause problems for user-friendly query engines.
- Additional important information about the data may not be readily expressible in the filename even under the best possible naming conventions. In our earlier example of sensor replacement, one might want to know when particular sensors were upgraded; this information cannot reasonably be encoded in a conventional filename.
- Again because the structure is not exposed, it is difficult to recognize the relationships among data sets. One tuple set might be the results of passing another through some postprocessing filter, such as image sharpening, but there is no practical way for an indexing engine to “know” this.

The fundamental problem is that the name is trying to encode a collection of attributes. In many areas, such identifying information is called *provenance* [2], [6], [7]. For example, in high-energy physics, provenance metadata tracks the complete history of a research result. The provenance for the data in a publication describes all the various analysis and collection steps, all the way back to the raw data collected in a particle accelerator. In the archival community, provenance metadata describes the history of a document, the people who assumed responsibility for it, and, in the digital world, any format transformations applied to it [29].

The provenance of a collection of data is not just a useful description. It is the single, unique identifier for that data set. In a very real sense, this makes the provenance the *name* of the data set. For this reason, provenance should be a first class property. Instead of encoding the name as a string, we represent it fully as a collection of name-value pairs. Of course, traditional names remain useful as well.

The specific details of the provenance are likely to be application-specific or at least community-specific. Different communities will likely develop their own standards for provenance metadata. For example, the VOTable format is a domain-specific DTD that is augmented with provenance [23].

B. Indexing Provenance

Because the complete provenance of any particular tuple set is likely to be large, most queries will probably not be a simple matter of looking up a name and retrieving the data; instead users will search for data sets based on subsets of the attributes and values found in provenance metadata. Different users will tend to query by different attributes depending on their goals; for example, given traffic sensor data framed as car sightings, a commuter investigating alternate routes will likely search by sensor location, but someone assessing the city-wide impact of new one-way street assignments will likely search by time.

If these car sightings are amalgamated from different sensor networks of different types (cameras, magnetometers, etc.) where the raw data is postprocessed in different ways, someone investigating anomalies in the data reporting might query based on origin: looking up the magnetometer readings that generated some suspect sighting data, or finding tuple sets handled by a particular postprocessing program. Such queries are often recursive, as there may have been several steps involved with multiple intermediate data sets, each with its own provenance.

Thus, the indexing structures in sensor data storage systems must provide for efficient lookups in many dimensions, as well as efficient recursive or transitive queries. Simple relational or XML-based name-to-value schemes are not sufficient and will not work well unless augmented with other structures.

III. What to Query

In this section, we consider the types of queries that a sensor data repository should support. We begin by discussing document versioning, as this is a familiar framework for working with provenance metadata. We then look at the requirements of research communities in the sciences, where provenance issues are increasingly important. Finally, using these examples as motivation, we turn to queries on sensor data provenance.

A. Document Versioning Systems

Document versioning systems are provenance management systems. When multiple programmers are working on the same program, they will be editing concurrently and (largely) independently. Systems like CVS [11] allow programmers to coordinate; however, they also track changes over time and record who did what. Typical queries on such systems include:

- Show me the file as it is now, or as it was yesterday.
- Show me all changes to this file since last week.
- Show me when each line in this file was inserted.
- Find the person who removed this error code.
- Get me all files tagged “Release 1.1”.

These queries are all reasonably well supported by CVS and similar systems. However, most document versioning systems are file-oriented. Queries that span files in complex ways, or involve data that has been copied from one place to another must generally be performed manually.

B. Experimental Data in the Sciences

Research communities demand good provenance support for their experimental data. This is necessary to support reproduction and validation of research results when large data repositories are available. For example, the Sloan Digital Sky Survey [28] collects data on millions of astronomical objects. This data comes from observatories across the world and is synthesized onto a universal ‘map’ to allow queries by position. Other communities, including biology [13], chemistry [10], and physics [16], have deployed similar databases.

Furthermore, many data sets are derived from others as analysis steps are performed. The provenance of a derived data set is the provenance of the original data plus the provenance of the tools used to do the derivation.

Provenance is particularly important for derived data; if a problem is found with the original data or with an analysis tool, all downstream data is tainted and must be locatable. Other typical queries on research data provenance include:

- Find all the raw data from which this data set was derived.
- Show me what I need to reproduce this result.
- Find an experiment that answers this question.
- I am up for tenure. Show everyone who has used my work.
- Find experiments similar to mine.

The queries in this domain tend to be more complex than those in the document versioning system. Nearly all the queries have some component of transitive closure, a construct not well supported by conventional query systems.

C. Sensor Queries

Sensor applications require all the same capabilities we saw in the two previous examples, and pose new demands of their own. Consider a sensor-enabled ambulance team [30]. EMTs arriving at an accident or mass casualty event place sensors (*e.g.*, pulse oximeters, EKGs) on the patients. These sensors monitor vital signs in real time. The resulting data is streamed to the ambulance, to dispatchers who route patients to medical facilities, and ultimately also to the correct hospital emergency room. Initially, this data is identified by patient, date/time, location, etc. As it moves through the system, it gets processed and filtered, and is thus enriched with additional provenance.

All of this data and metadata represents critical information, not only about each patient, but also about the emergency care infrastructure itself. These two aspects involve queries of considerably different natures. Queries about an individual patient might include:

- Show me everything we've done for this patient.
- Show me the heart rate from moment of arrival until now.
- Show me the results of oxygen treatment.
- Feed this patient's data into our automatic diagnostic tool and suggest the appropriate hospital or trauma center.

Examples of queries about the system might include:

- Give heart rate profiles for everyone handled by EMT X.
- Find me all patients with signs of arrhythmia.

D. Characteristics

From the above examples, we derive the following set of requirements for provenance-based sensor data storage:

Storage should be near the sensors. Sensor data may be valuable for arbitrarily long periods of time. (Weather data collected by hand goes back over a hundred years and can be expected to remain valuable indefinitely.) Furthermore, sensor networks can generate a huge volume of data: a regional traffic sensing network that records every passing car could easily generate terabytes of data per day. Transmitting all this data long distances over the network is unnecessarily expensive; also, the data is often most valuable near the source. Boston traffic data belongs in Boston, not in Singapore or even Seattle.

Data has multiple consumers. Real-time sensor data is probably of most value to its immediate collector, but many parties may have use for the archives. We cannot anticipate all applications, so both access and packaging must be flexible.

Recursive queries are common. All the usage scenarios make heavy use of transitive closure queries. These may go both backwards, to find ultimate origins, and also forwards, to find derived data that may be many generations downstream.

Distributed queries are common. Because raw sensor data should be stored near the sensors, aggregating over multiple sensor networks is inherently a distributed operation. Furthermore, aggregate data sets derived from such queries will probably be stored where they are created, so the transitive closure queries tracing the history of data will be distributed as well.

Query needs are heterogeneous. Though application domains share various characteristics, there is no reason that specific applications from different domains need commonality at the query language or data organization level. Nonetheless, it seems useful to share common infrastructure (network and storage resources) and also to be able to perform queries across domains. The traffic and weather communities might not agree beforehand on how to store and represent their data sets, but they may later want to query across them. This argues for the ability to federate data and processing.

IV. Design Space

In this section, we first present criteria for evaluating provenance index and query architectures. We then discuss several potential models in these terms.

Scalability. The system might need to scale in many ways: some possibilities include the number and size of tuple sets, rate of tuple set production, number of indexes, depth of ancestry, number of hosts, and distance across which the system is distributed.

Reliability. Data, local or non-local, will become inaccessible if provenance metadata becomes corrupted or is lost due to a system crash. The system must recover provenance metadata to a state consistent with its data after a system failure.

Query Result Quality. In information retrieval terms, there are two aspects to this: *precision*, the fraction of returned results that are relevant to the query, and *recall*, the fraction of relevant results that are actually returned.

Usability. The content and structure of provenance information depends on the application domain. The system must support storing domain-specific provenance, and must allow queries in whatever form is most appropriate for each domain.

Speed. Provenance metadata is accessed more frequently than its data. The system must perform at a reasonable speed, even on complex queries such as the transitive closures discussed previously.

Resource Consumption. The system must not impose excessive overhead, particularly on the network. Index metadata must be widely accessible, and will be both updated and accessed often. If distributed, updates may use a lot of network bandwidth; if centralized, query traffic may instead.

These criteria are not independent. Different models (and implementations) offer different tradeoffs among them. For example, a system with good precision and recall will generally be relatively slow and expensive. Similarly, increasing reliability by distributing index information will incur additional bandwidth requirements for index updates.

We now turn to the architectural models. Due to space constraints, we examine only the most critical criteria affecting each model.

A. Centralized Models

In a centralized system, provenance metadata is sent to some central data warehouse, where it is examined and indexed; query processing is then done within the warehouse. (As discussed in Section III, the warehouse would not store actual sensor data.)

This offers speed, simplicity, and ease of use. The conventional wisdom is that centralized indexing cannot scale. However, the success of Google [18] and Napster [15], among others, suggests otherwise. Centralized setups are also as likely as any to be able to handle recursive queries and provide effective backups.

The Trio project is a centralized database system that manages not only data, but also the provenance and accuracy of the data [32]. Buneman *et al.* offer a a centralized XML database that handles user annotations and allows tracking the path of a single datum through various transformations [5].

For sensor data, however, a central index has three shortcomings. First, query processing on real-time sensor streams is already becoming distributed [1], [24], [27]. And second, even though Google has indexed eight billion web pages [18], it may not be scale to the volume of updates associated with sensor data or data aggregated over many sensor networks. Finally, even though both data and its provenance are read-only once collected, when the index is only loosely coupled to the actual data there is a risk of inconsistencies creeping in: the linkage back from the index to the data might break or end up pointing to the wrong thing.

Nonetheless, despite these issues, the success of centralized indexing in practice makes it a standard against which to compare any other mechanism.

B. Distributed but Stable Models

If one assumes that the hosts involved are stable – permanent participants with reasonable reliablity and availability – there are four conventional architectures that require no centralized installation.

The first of these is the distributed database. Distributed databases inherently provide unified schemas [21], a useful property. However, they have limited ability to process recursive queries (*e.g.*, transitive closure), and optimizing continuous, distributed queries is still an open problem.

A second model, the federated database, uses multiple autonomous database systems, each with its own specific interface, transactions, concurrency, and schema [4]. A federated system does provide the illusion of a unified schema, but the fact that the components are truly disjoint systems may lead to slow access.

Both of these models provide strong consistency: full transaction semantics. However, this may be overkill for sensor data, given that the provenance index will be effectively append-only.

A third model, choosing availability over consistency, relies on soft-state and a mostly stable network. Three variations come from the scientific community. The Replica Location Service, or RLS, provides a unified lookup service for replicas of large data sets [9]. Its metadata lookup service is distributed, reducing update and query load, and it relies on periodic updates to keep its soft-state from becoming stale. Another example, the Storage Resource Broker, or SRB, is an instantiation of a simple federated database, storing metadata as name-value pairs and dividing itself into zones for scalability [3]. Third, the PASOA project, which examines trust relationships, includes a protocol for managing provenance in a client-server environment [19].

These examples from the Grid do provide worldwide access to large data sets. For RLS, much of this scaling, however, comes from an assumption that the exact name of each data set is known. Meanwhile, SRB's metadata model denies transitive closure, which is essential for handling provenance. Still, these models do support locale-specific query processing: data is stored at the producers and replicated at consumers; it is shipped to neither a central nor an arbitrary location.

The fourth model is the filename (or URL) model: organize the material into a hierarchical namespace and then use the hierarchy to partition the data across a distributed network of servers. While this approach is very practical for many applications, for sensor data it is inappropriate. Hierarchical naming systems are fundamentally limited by the need to choose a significance ordering for the attributes. This is a bad fit for any problem where no natural ordering exists, as is typically the case for the attributes that make up provenance metadata. For example, astronomical data will likely be tagged with both spatial coordinates and observation wavelength, and neither is a more general attribute than the other. Choosing either one as most significant will make querying on the other difficult.

C. Distributed and Unstable Models

It may not be feasible to rely on stable participants; if so, a different class of architecture is needed. The most widely-used mechanism in this class is the distributed hash table, or DHT [12], [14]. However, DHTs do not appear to be a suitable solution.

First, storing data objects by hashing a key inherently assumes that the location of these objects is unimportant. This is not the case for sensor data. The DHT-based database Pier [20] proved slow because of poor data placement. Second, periodic updates of distinct queriable attributes to DHTs scale to only tens of thousands of updaters [25]. This is inadequate. Third, getting even this much update performance would require that all participants have good network connectivity and plenty of processor power; this is more expensive than maintaining the stable servers required by other architectures. Finally, support for efficient recursive queries is so far nonexistent.

V. Research Agenda

We have shown that provenance-aware storage is useful for sensor network applications. With this storage come interesting research challenges.

A *Provenance-Aware Storage System*, or PASS, has four fundamental properties distinguishing it from other storage:

- Provenance is treated as a first class object.
- Provenance can be queried.
- Nonidentical data items do not have identical provenance.
- Provenance is not lost if ancestor objects are removed.

The first goal is to construct a purely local PASS. As provenance metadata is large and contains cross-references among files, just storing and indexing offers challenges; in particular, one needs efficient support for transitive closure queries.

Once this is done, the second goal is to allow merging collections of local PASS installations into single globally searchable data archives. This requires distributed naming and indexing schemes, and support for distributed queries. Designing efficient distributed transitive closure techniques is likely to keep researchers busy for years to come.

Other challenges abound. Our model does not inherently involve replication, as data is locale-specific, but replication is desirable for reliability and for query performance. Supporting replication cheaply is an interesting problem.

Security is essential as well, as much of the data collected in sensor networks (*e.g.,* medical data) is private. Much of this data is valuable even when aggregated to preserve privacy. What degree of aggregation is necessary? How does one represent the provenance of such aggregates? How do regulatory moves like HIPAA affect the situation? And how do we provide strong guarantees that privacy policies will be enforced?

Relatedly, sometimes one wants to abstract provenance away. For example, one probably wants to know what compiler compiled the program that did a particular analysis step; compilers are subject to optimizer bugs that can invalidate results. But for most purposes, it is far more useful for this information to be reported as "gcc 3.3.3" rather than as a detailed record of gcc's own provenance and change history. How does one identify these abstractions and take advantage of them?

VI. Conclusion

Sensor data alone, decoupled from its origins, will only be useful for the most prosaic applications. Instead, it must be tightly bound to searchable information about the sensors that produced it and any programs that processed it. Building suitable indexes on this *provenance* is a challenging problem.

Given reasonable assumptions about (1) the physical locations of where data comes from and goes to, and (2) the arrival rate of new tuples, no existing storage/query model offers a satisfying fit. A new architecture must be developed.

Constructing a distributed provenance-aware storage system requires solving both of these problems.

References

[1] D. Abadi, D. Carney, U. Cetintemel, et al. Aurora: A New Model and Architecture for Data Stream Management. *VLDB*, Aug. 2003.

[2] B. Barkstrom. Data Product Configuration Management and Versioning in Large-Scale Production of Satellite Scientific Data Production. In *Software Configuration Management*, 2003.

[3] C. Baru, R. Moore, A. Rajasekar, and M. Wan. SDSC Storage Resource Broker. In *CASCON*, Toronto, Canada, Nov–Dec 1998.

[4] Y. Breitbart, H. Garcia-Molina, and A. Silberschatz. Overview of multidatabase transaction management. *VLDB J.*, 1992.

[5] P. Buneman, S. Khanna, K. Tajima, and W. C. Tan. Archiving scientific data. In *SIGMOD Conference*, Madison, WI, 2002.

[6] P. Buneman, S. Khanna, and W. Tan. Why and Where: A Characterization of Data Provenance. In *ICDT*, 2001.

[7] R. Cavanaugh, G. Graham, and M. Wilde. Satisifying the Tax Collector: Using Data Provenance as a way to audit data analyses in High Energy Physics. In *Workshop on Data Derivation and Provenance*, Oct. 2002.

[8] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, Costa Rica, April 2001.

[9] A. L. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, and R. Schwartzkopf. Performance and Scalability of a Replica Location Service. In *HPDC-13*, Honolulu, HI, June 2004.

[10] Collaboratory for Multi-scale Chemical Science. `http://cmcs.org`.

[11] Concurrent Versions System. `http://www.cvshome.org`.

[12] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *SOSP*, October 2001.

[13] DBCAT. `http://www.infobiogen.fr/services/dbcat`.

[14] P. Druschel and A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *SOSP*, October 2001.

[15] S. Flanning. Napster. `http://www.napster.com`, 2000.

[16] I. Foster, J. Voeckler, M. Wilde, and Y. Zhao. The Virtual Data Grid: A New Model and Architecture for Data-Intensive Collaboration. In *CIDR*, Asilomar, CA, Jan. 2003.

[17] M. Gaynor, M. Welsh, S. Moulton, A. Rowan, E. LaCombe, and J. Wynne. Integrating wireless sensor networks with the grid. *IEEE Internet Computing*, July/August 2004.

[18] Google. `http://www.google.com`.

[19] P. Groth, L. Moreau, and M. Luck. Formalising a protocol for recording provenance in grids. In *Proceedings of the UK OST e-Science Third All Hands Meeting*, Nottingham, UK, Sept. 2004.

[20] R. Huebsch, J. Hellerstein, N. Lanham, B. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *VLDB'03*, Berlin, September 2003.

[21] D. Kossmann. The state of the art in distributed query processing. *ACM Computing Surveys*, 32(4), 2000.

[22] V. Kottapalli, A. Kiremidjian, J. Lynch, E. Carryer, T. Kenny, K. Law, and Y. Lei. Two-Tiered Wireless Sensor Network Architecture for Structural Health Monitoring. In *SPIE'03*, San Diego, CA, May 2003.

[23] B. Mann. Some Provenance and Data Derivation Issues in Astronomy. In *Workshop on Data Derivation and Provenance*, Oct. 2002.

[24] R. Motwani et al. Query Processing, Resource Management, and Approximation in a Data Stream Management System. In *CIDR*, Asilomar, CA, January 2003.

[25] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Distributed resource discovery on PlanetLab with SWORD. In *WORLDS*, San Francisco, CA, Dec. 2004.

[26] E. Page. The SECURES Gunshot Detection and Localization System, and Its Demonstration in the City of Dallas. In *5th Battlefield Acoustics Symposium*, Ft. Meade, MD, September 1997.

[27] P. Pietzuch, J. Shneidman, M. Welsh, M. Roussopoulos, and M. Seltzer. Path Optimization in Stream-Based Overlay Networks. Technical Report TR-26-04, Harvard University, Sept. 2004.

[28] Sloan Digital Sky Survey. `http://www.sdss.org`.

[29] R. Sproull and J. Eisenberg. Building an Electronic Records Archive at NARA: Recommendations for Initial Development, 2003. `http://www7.nationalacademies.org/cstb/pub_nara1.html`.

[30] M. Welsh, D. Myung, M. Gaynor, and S. Moulton. Resuscitation Monitoring With a Wireless Sensor Network. In *Circulation: Journal of the American Heart Association*, October 2003.

[31] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring Volcanic Eruptions with a Wireless Sensor Network. In *Second European Workshop on Wireless Sensor Networks*, Istanbul, Turkey, January 2005.

[32] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In *CIDR*, Asilomar, CA, January 2005.