The Case for Fine-Grained Stream Provenance

Boris Glavic [#], Kyumars Sheykh Esmaili [*], Peter M. Fischer [*], Nesime Tatbul [*]

[#] *Database Group, University of Toronto, Canada*
glavic@cs.toronto.edu

[*] *Systems Group, ETH Zurich, Switzerland*
{kyumarss, peter.fischer, tatbul}@inf.ethz.ch

Abstract: The current state of the art for provenance in data stream management systems (DSMS) is to provide provenance at a high level of abstraction (such as, from which sensors in a sensor network an aggregated value is derived from). This limitation was imposed by high-throughput requirements and an anticipated lack of application demand for more detailed provenance information. In this work, we first demonstrate by means of well-chosen use cases that this is a misconception, i.e., coarse-grained provenance is in fact insufficient for many application domains. We then analyze the requirements and challenges involved in integrating support for fine-grained provenance into a streaming system and outline a scalable solution for supporting tuple-level provenance in DSMS.

1 Introduction

Tracking provenance, exploring which input data led to a given query result, has proven to be an important functionality in many domains such as scientific data management, workflow systems [D$^+$07] and relational database systems [C$^+$09]. Previous techniques have been traditionally classified according to their granularity: *Coarse-grained* provenance tracks dependencies between input and output data at a very abstract level (e.g., streams), whereas *fine-grained* provenance does so for individual data items in the input's data collections (e.g., tuples or attribute values).

Surprisingly, in the area of data stream management systems (DSMS), there has been little work beyond *coarse-grained* provenance (e.g., tracking the sensor sources from which a data item originates [V$^+$07, L$^+$10]). Recently, Huq et al [H$^+$10] have proposed to achieve *fine-grained* stream provenance by augmenting *coarse-grained* provenance with timestamp-based data versioning, focusing specifically on query result reproducibility at reduced provenance metadata storage cost.

In this position paper we show that in fact many diverse stream processing applications require *fine-grained* provenance, we broadly identify the main challenges involved in providing such support, and outline an approach to address these challenges in a scalable way.

Use case	Provenance generation	Provenance retrieval		
	Relevant events & queries	Lifetime	Retrieval	Response times
Ad-hoc human inspection	all events (events of interest not known beforehand)	time-bound (minutes to hours)	iterative drilldown & point queries	(milli)seconds
Stream query debugging	selected queries & events	debugging session	lookup & replay & interactive drilldown	(milli)seconds
Indicator-based assurance	selected queries, all events	retention period for indicators	Point & Analytic queries	offline
Event warehousing	selected queries, all events	application-dependent	Analytic queries	offline

Table 1: Provenance Use Cases and Requirements

2 Motivation and Use Cases

Event stream processing has recently been gaining attraction in applications that not only need to deal with large amounts of observed data, but also require the ability to trace an output data item generated by the DSMS back to the input data that contributed to its existence. Table 1 summarizes the result of a survey that we performed to identify data stream applications which require *fine-grained* provenance [F+10]. Due to space limitations, we will only discuss the first use case in more detail and for the rest give brief summaries.

Ad-hoc human inspection: In monitoring and control of manufacturing systems, sensors are attached to machines and to key points along a supply chain as well as on the support infrastructure. Sensor readings are processed by a DSMS in order to detect critical situations such as machine overheating or low inventory. These detected events are then used for automatic corrections and also to notify the human supervisors who need to assess the relevance of these events. To do so, the human operators need to understand from which inputs these events where derived (i.e., the individual temperature readings). This requires *fine-grained* provenance for events. Because of the interactive nature of human inspection, the original events and their provenance become relevant only for short periods of time, but should be provided efficiently to enable interactive drilldown.

If the DSMS outputs a machine overheating alarm event, the user would want to understand which sensors measured high temperature values.

Stream query debugging & diagnosis: The high complexity of streaming queries requires support for diagnosing system behaviour, up to the scope of events or even attributes. Provenance helps in exploring the computational steps and the data that led to an observed result and in understanding how errors have propagated. The scope of inspection can be limited to particular queries or events of interest.

Indicator-based assurance: Monitoring and control systems often adhere to strict accountability requirements and need to provide proof for correct operations, which are expressed as indicators. Provenance helps to establish the validity of these indicators by

providing the input events and computations they are based on.

Event warehousing: Event warehousing is used to collect raw and derived event streams for mining and analysis. Provenance exposes how events became part of the warehouse. Full provenance needs to be captured to allow complex analysis over such data.

The use cases summarized above clearly show that *fine-grained* stream provenance is a critical requirement in several important application domains, each of which pose a diverse set of requirements in terms of the generation, storage, and retrieval of provenance.

3 Challenges and Opportunities

In this section, we provide a discussion of the challenges that have to be addressed to be able to extend a DSMS with efficient support for *fine-grained* provenance.

Infinity, Performance, and Aggregation: The key challenges for *fine-grained* provenance for data streams stem from the combination of infinite data, high performance requirements and the prevalence of aggregating operators.

Since data streams can potentially be infinite, we do not necessarily have a full view on all items of a stream (e.g., some items may have not appeared yet). Moreover, it may be impractical or even impossible to preserve all seen items for later processing. These problems are often further aggravated by the strict performance requirements set up by streaming applications, including high data rates and low latency. In addition, typical streaming workloads combine multiple events into one, e.g., by aggregating or inferring higher-level events and thereby reducing the load on downstream operators. *Fine-grained* provenance needs to represent information about all contributing data items from the input, thus negating many of these savings.

The high performance requirements of DSMS call for efficient provenance generation that only instantiates complete provenance information when it is actually requested. Yet computing provenance information *lazily*, i.e., only when it is requested, is not a good strategy for DSMS, because access to the input data is required to compute the provenance of an output data item and naive storage of the complete input is unfeasible. *Eager* generation of provenance for all outputs does not solve this problem, because the size of *fine-grained* provenance allows only for short-time storage of this information. Limited access to the input data also restricts the direct applicability of standard optimizations for propagation based approaches for provenance generation. A propagation-based approach computes provenance by propagating annotations through the operators of a query that represent partial provenance information. In relational systems these approaches can be optimized by propagating, e.g., only the identifiers of input data items in the provenance of an output. If necessary, the complete input data items represented by these identifiers can be reconstructed from the input data, which is not necessarily available for DSMS. *Coarse-grained* source provenance tracking mostly ignores these challenges, since, given the small size of information, such as the sensor or stream it is derived from, it is reasonable to annotate a data item with *coarse-grained* provenance.

Non-determinism: Some mechanisms applied by DSMSs to cope with issues like high input rates (e.g., load shedding or approximations), unpredictable behavior of input sources (e.g., delays or disorder), and certain operator definitions (e.g., windowing on system time), may lead to non-deterministic behaviour. The non-deterministic nature of some DSMSs severely restricts the use of some of the standard techniques developed for database provenance for these systems. For instance, query rewrite techniques usually require reproducibility of query results to deal with operations like aggregation.

Order: In contrast to the set or bag model of relational databases, data streams are typically modeled as ordered sequences, requiring a provenance model that incorporates order. This ordering, however, can be exploited to compress provenance information (see Sec. 4).

In summary, most of the challenges for developing a provenance-enabled DSMS stem from the transient nature of streaming data, its performance requirements, and the non-determinism introduced to deal with high input rates and infinity. A major challenge is to find a solution that balances the amount of data needed for provenance representation, efficiency of provenance generation, and performance of provenance retrieval.

4 Solution Outline

We now present an approach to integrate provenance support in a DSMS, that addresses the challenges outlined in Section 3. We strive for an eager propagation approach that generates compressed provenance with low overhead. The provenance of an output is modeled as a set of tuple identifiers, while keeping the original data items at the sources or inside the query plan as long as they are needed for provenance retrieval. These identifier sets will then be used to retrieve and reconstruct the complete input data items in the provenance.

Propagation: Instrumenting the physical operators of a DSMS to propagate identifier sets can be done very efficiently as long as the number of input items from which an output is derived from is small. However, as discussed in Section 3, typical DSMS query processing involves operations that compute a relatively small number of outputs from a large number of inputs, i.e., we can expect the typical provenance of such queries to be too large to be processed in this way. Fortunately, the relationship between an output data item and its provenance follows specific patterns based on the operators used in the query. These patterns can be used to apply very effective and specialized compression techniques to reduce the load on the system. For example, a prevalent windowing method applied by DSMS is sliding windows of fixed size. Given that streams are ordered, the content of such a window can be represented as a single interval. If the slide is relatively small, the provenance of two subsequent data items will overlap to a large extent. Thus, it may be beneficial to store only the differences between the provenance of subsequent outputs.

Reconstruction for Retrieval: If provenance information is requested by a user, the input data items in the provenance have to be reconstructed from the set of identifiers used to represent the provenance. Obviously, this step needs access to the input data. As discussed before, storing all the inputs for an unlimited amount of time is not feasible. Thus, we en-

vision an approach that stores only these parts of the input that belong to the provenance of an output and purges inputs and provenance data using backpropagation of the computed provenance and e.g., timeouts, driven by the application needs outlined in Section 2.

Covering Interval Compression: A more radical approach to reduce the overhead caused by provenance generation would be to only propagate covering intervals during regular query execution. Instead of computing the complete provenance of an output, we only compute a (minimal) continuous subsequence (represented as a covering interval) from the input that is guaranteed to contain all data items from the provenance. This approach has the advantage of reducing the provenance generation overhead during the execution of streaming query to a constant factor. However, this comes at the price of additional computational cost during retrieval and additional overhed in order to provide the information on which input data to keep. At retrieval time we have to compute the actual provenance by feeding the input from the covering interval into a copy of the original query network. Clearly, this approach is not applicable to queries with non-deterministic behaviour. Creating a copy of the network does provide room for additional architecture options, in particular in distributed settings: The copy does not need to be run on the same nodes as the DSMS, but can be instantiated in a warehouse or on-demand in a cloud-style setting.

5 Conclusions

In this work we motivated the need for fine-grained provenance in DSMS, discussed the challenges involved with supporting this kind of functionality, and outlined a solution. The conflicting requirements of precision and performance require a comprehensive approach that carefully trades off expressiveness with a moderate overhead during processing and retrieval, providing ample opportunities for research.

References

[C+09] James Cheney et al. Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases*, 1(4), 2009.

[D+07] Susan Davidson et al. Provenance in Scientific Workflow Systems. *IEEE Data Engineering Bulletin*, 32(4), 2007.

[F+10] Peter M. Fischer et al. Stream Provenance Use Cases. Technical report, ETH Zurich, 2010.

[H+10] Mohammad R. Huq et al. Facilitating Fine Grained Data Provenance using Temporal Data Model. In *DMSN*, 2010.

[L+10] Hyo-Sang Lim et al. Provenance-based Trustworthiness Assessment in Sensor Networks. In *DMSN*, 2010.

[V+07] N. N. Vijayakumar et al. Tracking Stream Provenance in Complex Event Processing Systems for Workflow-Driven Computing. In *EDA-PS Workshop*, 2007.