6-2008

Advances and Challenges for Scalable Provenance in Stream Processing Systems

Archan MISRA
Singapore Management University, archanm@smu.edu.sg

Marion BLOUNT
IBM TJ Watson Research Center

Anastasios KEMENTSIETSIDIS
IBM TJ Watson Research Center

Daby SOW
IBM TJ Watson Research Center

Min WANG
IBM TJ Watson Research Center

Citation

MISRA, Archan; BLOUNT, Marion; KEMENTSIETSIDIS, Anastasios; SOW, Daby; and WANG, Min. Advances and Challenges for Scalable Provenance in Stream Processing Systems. (2008). *Provenance and Annotation of Data and Processes: Second International Provenance and Annotation Workshop, IPAW 2008, Salt Lake City, UT, June 17-18, 2008: Revised Selected Papers*. 5272, 253-265. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/678

Advances and Challenges for Scalable Provenance in Stream Processing Systems*

Archan Misra, Marion Blount, Anastasios Kementsietsidis, Daby Sow, and Min Wang

IBM T.J. Watson Research Center
Hawthorne, NY, USA
{archan,mlblount,akement,sowdaby,min}@us.ibm.com

Abstract. While data provenance is a well-studied topic in both database and workflow systems, its support within stream processing systems presents a new set of challenges. Part of the challenge is the high stream event rate and the low processing latency requirements imposed by many streaming applications. For example, emerging streaming applications in healthcare or finance call for data provenance, as illustrated in the Century stream processing infrastructure that we are building for supporting online healthcare analytics. At anytime, given an output data element (e.g., a medical alert) generated by Century, the system must be able to retrieve the input and intermediate data elements that led to its generation. In this paper, we describe the requirements behind our initial implementation of Century's provenance subsystem. We then analyze its strengths and limitations and propose a new provenance architecture to address some of these limitations. The paper also includes a discussion on the open challenges in this area.

1 Introduction

To enable an emerging class of *cyber-physical* computing applications, several stream computing platforms and middleware have been developed (e.g., Aurora [1], SPC [2]) to provide scalable, high throughput processing of sensor-generated data streams. In such systems, the arriving data are essentially *ephemeral*; to support low-latency processing of the data streams, stream operators perform only one pass over the arriving data, which are then typically discarded. In turn, this typically limits the forms of provenance in these systems to *process provenance*, i.e., determining which stream operators contributed to the generation of a particular data item.

Remote health monitoring represents an extremely important application domain for stream computing. To enable automated near-real time analysis of high volumes of medical sensor streams, we have been building, over the past year, an infrastructure, called Century [3], that permits the scalable deployment of online medical analytics. Stream analysis in the medical domain requires the Century infrastructure to support both process and *data* provenance, to support capabilities such as "offline dependency analysis" or "historical data replay". From a technical standpoint, data provenance imposes a

* This work was supported by the IT R&D program of MIC/IITA under the project/grant/funding number 2006-S-602-01 (Development of Stream-based Distributed Interoperable Health care Infrastructure Supporting Provenance and QoE).

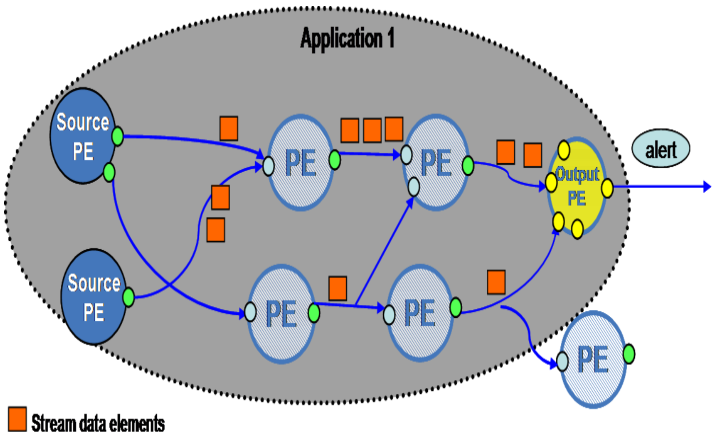

Fig. 1. Illustrating Data Provenance in a Stream Analysis Infrastructure. An application is represented as a directed acyclic graph (DAG) of nodes, with each node representing an operator or a Processing Element (PE). Provenance reconstruction involves determining the set of causative data elements belonging to streams that lie *upstream* of a particular data element.

novel challenge in data streaming systems, that of *stream persistence* [10,12]. At any point in time, given an output (e.g., a medical alert) generated by a stream processing graph, the Century Provenance system must not only recreate the processing graph that created the output, but also provide *all* the elements of the intermediate data streams that generated it. A data provenance solution for streams faces a couple of challenges:

– It must preserve the *high processing throughput* of the infrastructure, implying that the provenance solution cannot introduce significant additional processing overhead for every individual stream data element.
– It must *not impose a prohibitive storage load*, both in terms of the *volume of data*, as well as the *insertion rate* of data items requiring storage.

To support data provenance for such high throughput environments, we have previously introduced a model-based provenance solution, called Time-Value-Centric (TVC) provenance [17], which uses an explicit specification of the dependency relationship between input and output streams at every node (hereafter called a *Processing Element* or *PE*) in the processing graph. The notion of data provenance, involving the identification of multiple stream elements at upstream PEs belonging to a processing graph, is graphically illustrated in Figure 1.

This paper first describes some challenges with data provenance, based on our initial experiences with a TVC-based provenance solution for Century. The TVC approach does result in lower processing overhead, compared to the conventional annotation approach (which would require every element of every data stream to carry along a much longer set of stream elements as metadata). However, our experience reveals two new challenges for a pure model-based approach:

– Fundamentally, it has to contend with the increased *storage insert rate* that results from the need to persist the individual elements of *every* data stream occurring within a stream processing graph.
– The provenance model must reconcile and track potential discrepancies between the granularity at which stream data are produced and consumed by PEs within a processing graph. This discrepancy surfaces in extensible stream computing systems, where the data is not strongly typed, where the set of operators (PEs) is not closed and where different PEs choose to consume data at different granularities.

While both of these features need to be addressed, the issue of *much larger stream storage rates* is fundamentally more challenging and requires a change in the basic provenance model. The need to store both external and intermediate streams will impose an infeasibly high workload on commercial database systems. Accordingly, we shall propose a new hybrid provenance architecture, called *Composite Modeling with Intermediate Replay (CMIR)* that solves the problem of stream persistence by defining TVC-style dependency relationships only over a set of PEs (rather than at each individual PE) and by using *data replay* to recreate the data elements of streams internal to the PE set. We shall also discuss a set of open challenges and issues, with a goal of soliciting new approaches from the provenance community for tackling these challenges.

The rest of this paper is organized as follows. Section 2 provides an overview of the basic TVC primitives and their use in a representative analytic application, and then introduces two observed challenges. Section 3 introduces the suggested CMIR model for data provenance in stream computing platforms, and then describes the related technical challenges. Section 4 describes our current solution for resolving the granularity mismatch between output and input data elements. Section 5 then surveys prior relevant work and the paper concludes in Section 6 with a summary of the main points.

2 The TVC Model for Century and Resulting Limitations

The TVC model [17] specifies a set of primitives that are used to define a causative relationship between the data elements generated at the output port of a PE and the data elements arriving at its input ports. The TVC model differs from conventional annotation-based approaches for data provenance, which would need to embed a potentially large set of input data identifiers as metadata in every output data element (owing to the *statefulness* of stream operators, which implies that each output may be influenced by a large number of input data samples). TVC exploits the observation that the input-output dependencies for most PEs can be specified in terms of some invariants– while each output data element may have a variable set of causative input elements, this set may be indirectly determined through the application of these invariant primitives.

The TVC model supports the following primitives for dependency specification:

– *Time:* This primitive captures dependencies where an output data element is generated based on a past time window of past input data elements. For example, the notation $O_{m1}(t) \leftarrow I_{n1}(t-10, t-2)$ indicates that an output element generated at a time $t = 80$ on output port $m1$ depends only on those input elements that were timestamped with values in the interval $(70, 78)$, on input port $n1$.

- *Value:* The 'value' primitive defines a dependency in terms of predicates over the attributes of the input data elements. For example, a value primitive like $O_{m2}(t) : \{alertLevel = 1\} \leftarrow I_{n2}(t) : \{(systolic > 130)\&\&(diastolic > 100)\}$ indicates that an output element with 'alertLevel=1' depends only on *all* past input samples that satisfy the corresponding predicates over the (systolic, diastolic) attributes.
- *Sequence:* The 'sequence' primitive expresses dependencies in terms of the sequence number of arriving elements. For example, a sequence primitive $O_{m3}(t) \leftarrow I_{n3}(i-30, i)$ indicates that an output element depends on the most recent 30 samples of input data.

A TVC dependency relationship may be composed by arbitrary conjunctions and disjunctions of these basic primitives. Moreover, for significantly enhanced expressiveness, the specifications allow each TVC term to specify a combination of (time, sequence, value) triples. Each element of such a triple has a unique 'order' term, which defines an evaluation order for these primitives, with the output sub-stream of a lower order primitive acting as the input stream for a higher order primitive. As an example, the dependency relation $O_{45}(t) \leftarrow I_{97}\{(t-1d, t, order = 2), (systolic > 130, order = 1)\}$ implies that the causative set for an output element of port 45 may be reconstructed by first obtaining the sub-stream of input elements on port 97 that have '$systolic > 130$' and then picking all the elements of this sub-stream that have been received in the last day. Figure 2 shows the specification of TVC primitives in a sample processing graph in Century.

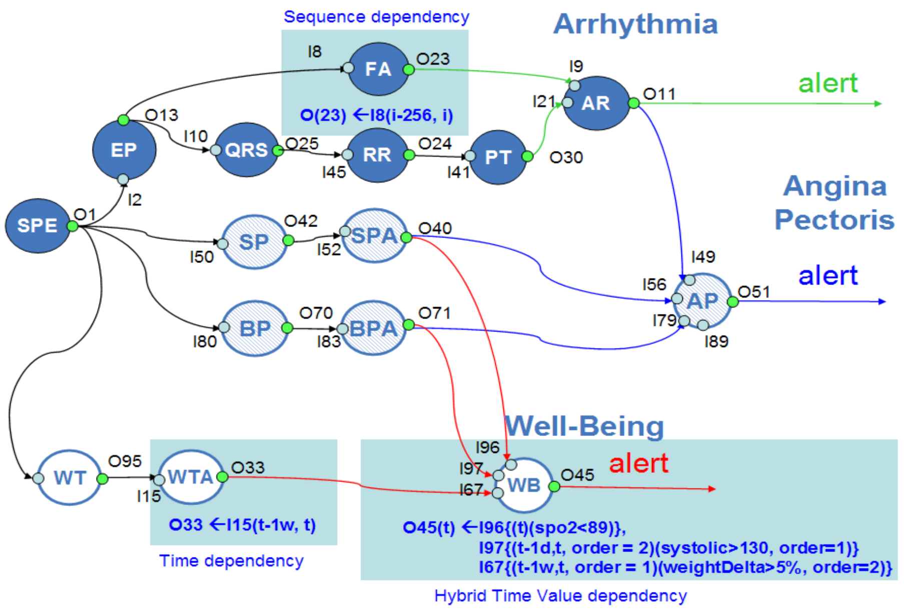

Fig. 2. Graph of a Representative Arrythmia Monitoring Application in Century. (The TVC-based dependency relationship for two PEs is explicitly highlighted.)

```
RetrieveCausativeData(Event e) {
ts= e.Timestamp; oport= e.phyOutputPort
{PE, logOutPort}= lookupDynamic(oport); // find the logical (PE,port) pair
$tvcTerms = lookupTVC(PE, logOutPort)$; // find statically specified TVC terms
for ($i \in InputPorts$) {
    dataElements= retrieveElements(i); // retrieve incoming data elements
    /*use TVC to identify the causative subset */
    causativeInput= filter(tvcTerms, dataElements);
    causList.add(causativeInput);
} return causList;
}
```

Fig. 3. Data Provenance Reconstruction Algorithm

Assuming that all elements of all data streams are persisted, deriving the set of input causative data sample is a fairly straightforward process captured by the simple high level pseudo-code in Figure 3. Recursive application of this pseudo-code enables the reconstruction of data dependencies at progressively upstream points in the processing graph.

2.1 Challenges in the Practical Application of Model-Based Provenance

Applying the *generic* TVC description above to an actual stream computing environment, however, gives rise to two practical challenges:

- *Intermediate Stream Persistence and the Resulting Storage Load:* Streaming systems supporting data provenance require the data elements of *each stream* to be persisted. The TVC framework is no exception. Let $\overline{O_m, I_n}$ denote the stream flowing between output port m and input port n. To reconstruct the entire data provenance along the entire path for ECG in Figure 2, the system must store both the incoming ECG samples ($\overline{O_1, I_2}$) and all the intermediate streams ($\overline{O_{13}, I_8}$, $\overline{O_{13}, I_{10}}$, $\overline{O_{25}, I_{45}}$, $\overline{O_{24}, I_{41}}$). The persistence of high volume data streams is already known to be a potential performance bottleneck for state of the art database systems. In [12], the authors show both analytically and experimentally that the persistence of Electrocardiogram data streams with a state of the art database system could only scale up to a few hundreds of patients; capturing data provenance further acerbates the problem by causing a multiplicative increase in the stream insert rate on the backend storage system. Conceptually, we require an enhanced solution that can eliminate this requirement for storing every intermediate data stream.
- *Granularity Mismatch of Output and Input units for a Data Stream:* Consider a pair PE_1 and PE_2 of PEs where the output of PE_1 results in a stream that is one of the input streams of PE_2. In loosely-typed or extensible systems, it is entirely possible that the *granularity at which PE_2 consumes streaming elements differs from the granularity at which PE_1 generates streaming elements.* This difference may occur for two distinct reasons:
  - The ‘data type’ of the elements produced by PE_1 and those consumed by PE_2 need not be identical. In many systems that permit type extensions and

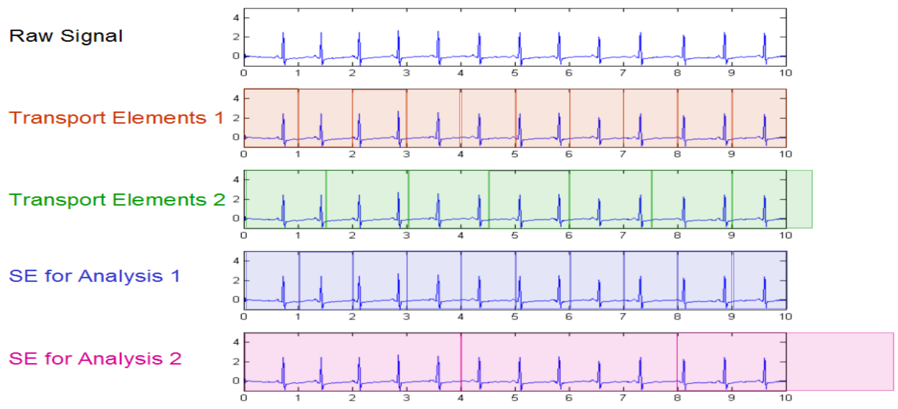

Fig. 4. Role of TEs vs. SEs in a generic stream-based analytic infrastructure. (The $TE \rightarrow SE$ mapping can be either one-one, one-many or many-one.)

inheritance (e.g., Tribeca [14]), and where stream bindings are based upon type-based subscriptions (e.g., SPC [2]), downstream PEs may bind to any output PE that produces the specific data type or its super-type (i.e., the consumed data is only *part* of the produced data element.) In most cases, the child PE merely consumes a sub-set of the data elements produced by a parent PE. As an example, PE_1 may be producing a person's 'vitalsigns' data type (which contains the elemental types: blood pressure (BP), heart rate and SpO2), while PE_2 may be using only the BP values; data reconstruction for a provenance query should then expose only the BP data.

- PE_1 may package multiple elements of a given data type into a single, larger transport element (TE), as this promotes more efficient transport of data within the processing runtime, by amortizing the transport-layer overhead over multiple data elements[1], especially when an individual data element may correspond to a sample of only a few bits. Figure 4 illustrates this for ECG signals that are collected in variable-sized TEs. This results in a potential incompatibility between the units of data produced by PE_1 and the unit of data consumed by PE_2. As an example of this, PE_1 may be an ECG PE that produces TEs containing a variable number of ECG 'samples', while PE_2 is a QRS detector PE that produces a QRS value based on the ECG samples in the last 60 seconds. Let's assume that PE_2 assumes an input rate of 1 TE (comprising 5 ECG samples) every 5 seconds, and therefore 12 TEs are used. Then, the TVC rule $O(t) \leftarrow I(i-12, i)$ captures the provenance of QRS outputs in PE_2. What if we replace PE_1 with a PE_1' that uses a different rate, generating, say, one TE (comprising 1 sample) every 1 sec? How does this *innocent* change affect provenance? It is not hard to see that our TVC rule would now need to

[1] This issue does not arise in more restrictive systems, such as Aurora [1], where data units are both defined and transported as fixed tuples.

change to $O(t) \leftarrow I(i-60,i)$! Ideally, the provenance design should allow the TVC relationship specification to remain invariant of the specific granularity at which the data elements arrive at its input ports (as different ‘parent’ PEs can provide varying transport encapsulations of the data elements).

The above examples demonstrate that, in terms of data provenance, it does not suffice to focus our attention solely on models that describe the output/input dependencies *within* a single PE. Additional techniques are needed to capture the discrepancies that might arise between the units in which the data is produced by the parent PE and in which the data is consumed by the receiving PE.

3 Looking towards the Future: The CMIR Data Provenance Framework

We now propose a novel approach to provenance for stream-based environments that preserves the explicit model-based dependency specification of the TVC approach, yet does not require the persistence of all intermediate streams (but perhaps, only a smaller set of streams). The new approach, called *Composite Modeling With Intermediate Replay (CMIR)*, aggregates a cluster of PEs into a *virtual PE*, such that only streams that act as either input to or are output by the virtual PE are persisted. Moreover, the TVC relationships are then defined in terms of the output and input streams of the virtual PE, thus enabling the set of causative elements of input streams (of the virtual PE) to be determined for any given output stream element. The individual ‘real’ PEs, and their associated bindings, within such a virtual PE, are opaque to this model-based provenance framework, which treats the virtual PE as a ‘black box’. The greater the size of the cluster, the smaller the number of streams that become ‘external’ to the virtual PE, thereby reducing the storage burden. Figure 5 illustrates the concept of CMIR-based provenance–in this case, the provenance relationships are captured over the *output and input streams of* PE_{V1}, a virtual PE defined by aggregating the ‘real’ PEs, $\{PE_1, PE_2\}$.

The process of virtualizing a group of PEs must also be supplemented by a mechanism that recreates, *on-demand*, the streams internal to the virtual PE, since data provenance inherently demands the reconstruction of data elements along the entire path of a specific processing graph. Our approach for this involves the use of a *replay mechanism*. To achieve this, one firstly requires the knowledge of the internal structure of the virtual PE, including the various real PE instances and the associated stream bindings. The dynamic provenance information must be extended to capture the association between the virtual PEs and the ’real’ PEs.

The bigger challenge arises from the potential *statefulness* of the real PEs; such statefulness implies that the set of output stream objects produced by a PE will depend not only on its *fixed* processing logic, but also its current internal or external state. For CMIR, each individual PE must be *provenance-aware*–i.e., it must be responsible for checkpointing its internal state to the provenance store, and, conversely, for recreating its internal state based on such retrieved historical data. In the TVC model, the state of each individual PE is captured in provenance metadata externalized to the provenance infrastructure (typically, by annotating the state within the output stream elements).

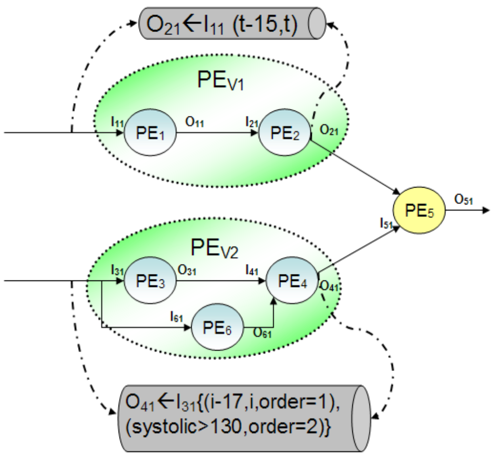

Fig. 5. The CMIR Framework and the Use of Virtual PEs

However, in the CMIR model, PEs internal to a virtual PE are not externalizing that metadata, so relevant state must be externalized in this way. In addition to such checkpointing, a CMIR based provenance system must also have a Replay component that dynamically instantiates, within the runtime, the set of PEs (along with their corresponding state evolution) corresponding to a virtual PE.

3.1 Challenges in CMIR-Based Provenance System Design

While the application of a CMIR-based solution for Century is still in its initial design phase, we are already aware of a few challenges that we must address. In particular, two very interesting open challenges are:

– *Models for Persisting State:* To support accurate replay of a PE's internal logic, the CMIR framework requires the persistence of the PE's internal state. Our initial thoughts are to have the provenance system treat this 'state information' as an opaque byte-stream, implying that each PE has the freedom to generate its own custom representation of its own state. It is, however, likely that the state information of the vast majority of PEs is likely to contain some common objects (examples of such likely state objects include command line arguments, the PE's load, the IDs of the individuals whose streams are being monitored, etc.); in such a situation, it may be worthwhile to define a more structured format for the object state. Moreover, it may also be desirable that this state representation lend itself easily to partial changes (as state change is often incremental), thereby allowing a PE to express its evolving state to the Provenance storage infrastructure in a more efficient fashion. *The issue of appropriate representation formats for such state information, which balance efficient storage and easy reconstruction, appears to be an open research question.*

– *Techniques for Composing Provenance Dependencies:* The use of virtual PEs within the CMIR framework implies the need for the system to be aware of the output-input dependency relationships at the virtual PE-level. Virtual PEs are, however, merely a runtime artifact of the provenance system; the basic TVC-style relationships will continue to be expressed for each individual PE (as individual PE developers shall specify the dependency logic of only their authored PEs). The provenance system must thus programmatically cascade the TVC relationships of individual PEs to derive the 'macro' dependency relationships of the virtual PE.

  An interesting question that arises here relates to what types of dependencies are composable and what aren't. As a simple example, a time interval-based dependency primitive is composable in a fairly-straightforward fashion. If PE_1 has a time based relationship $O_{11}(t) \leftarrow I_{11}(t-10,t)$ and PE_2 has also a timed based relationship $O_{21}(t) \leftarrow I_{21}(t-5,t)$, then as shown in Figure 5 the composed rule for the virtual PE PE_{v1} is $O_{21}(t) \leftarrow I_{11}(t-15,t)$. However, other primitives of the basic TVC model do not lend themselves to such relationship cascading. For example, if PE_2 has a value-dependent relationship, such that $O_{21}(t)$ depends on the last 10 values generated by PE_1 with '$attr1 > 10$', while PE_1 has the same time based relationship as before, then the input-output relationship of the virtual PE *can no longer be expressed* using the primitives of the basic TVC model.

  This example illustrates the central role that the choice of primitives in the dependency model have on the feasibility of deriving dependency relationships for the virtual PEs. Accordingly, we need to develop an enhanced *composable* provenance dependency model, such that its primitives, while being adequate expressive, are *'closed' (in set-theoretic terms)* under the operation of cascading. The issue of cascading is further complicated by the fact that, in many applications and scenarios, provenance is not used simply for backward reconstruction of data elements in a processing graph, but for *forward reconstruction* as well. As an example based on our own experiences with Century, a medical stakeholder who detects a faulty 'arrhythmia' analysis for a given patient may need to look 'downstream' and cleanse the system of faulty alerts generated as a result of this incorrect intermediate value. To support such 'forward provenance' semantics, the primitives of the provenance specification language must also be *reversible* (even if they are not very precise). *Overall, we believe that the development of a set of expressive provenance primitives, with the necessary composable and reversible properties, constitutes an important open problem for stream-based provenance.*

4 Resolving Granularity Differences between Stream Data Producers and Consumers

In Section 2, we illustrated how discrepancies in the granularity of stream elements produced by PEs, and the elements consumed by other PEs, directly influence our ability to accurately apply model-based provenance across PEs. One alternative to address this problem has already being hinted in Section 2. Instead of associating a single TVC rule for a particular PE, one can associate a set of rules, one for each output stream granularity (of the parent PE) that is known *a priori*. Unfortunately, this is a bad design choice for extensible stream systems, where new PEs, data types or stream encapsulations (containing the data type desired by a consuming PE) may become part of the stream computing infrastructure at any point; in such systems, the behavior of potential suppliers of specific data types cannot be predicted at PE design time. There are two other alternative, and better, design choices available:

– We may require the data types (and super-type) definitions to be *externalized* in a global type repository, with stream consumption by PEs being rigidly enforced to observe such type definitions. In such a system, a PE must indicate the *exact data type*, say DT_c, that it consumes on any input port, and the *runtime* must then ensure that this particular PE is able to receive only that exact data (i.e., for a parent PE that generates data elements belonging to data type DT_s that is a super-type of DT_c, the runtime must eliminate all extraneous data attributes and fields in DT_s, before making only DT_c available to the consuming PE). Such a strongly-typed system may become cumbersome for an open and extensible streaming infrastructure, where different organizations may define their own PEs, each of which may utilize multiple elements/fields within, or straddling different, data 'types'.
– Alternately, we can require the specification of a separate set of 'mapping functions' that perform the conversion between data elements of an output port and the data elements consumed by an input port. For flexibility, such mapping functions must be user-definable, thus supporting arbitrary mappings. Each stream binding (i.e., output, input) port combination is associated with one such function. Conceptually, such a mapping can itself be viewed as a TVC-style dependency rule, applied to an 'invisible PE' that simply transforms the data output by the stream's source to the data elements consumed by the stream's sink. This mapping function captures the discrepancy arising out of either inexact matches between the data 'types' or different *TE* encapsulations at the transport layer.

Either approach allows us to separate the TVC provenance logic (which uniquely captures the internal data dependencies of an individual PE) from the data element conversion logic (which is a function of the data formats and encapsulation, rather than a PE's *processing* logic). However, an implementation of either approach must choose between proactive vs. reactive conversion: the mapping from output to input element granularity may be performed either proactively (when elements are transported within the runtime) or reactively (in response to data provenance queries). Both approaches involve tradeoffs between the processing load and the resulting complexity of the data storage system, and thus require further investigation.

4.1 Granularity Resolution in Current Century Implementation

Century's current implementation is based on the second solution, namely the use of 'mapping functions' that convert output elements transported by the SPC runtime to input elements consumed by downstream PEs. In SPC, data is transported within the runtime in units known as Stream Data Objects (SDOs)–each SDO thus corresponds to a single TE. The provenance (TVC) specifications are themselves defined in terms of the elements (which we call *Stream Elements (SEs)*) consumed by a PE. Note that an individual SDO can contain both multiple elements of the same type (e.g., a batch

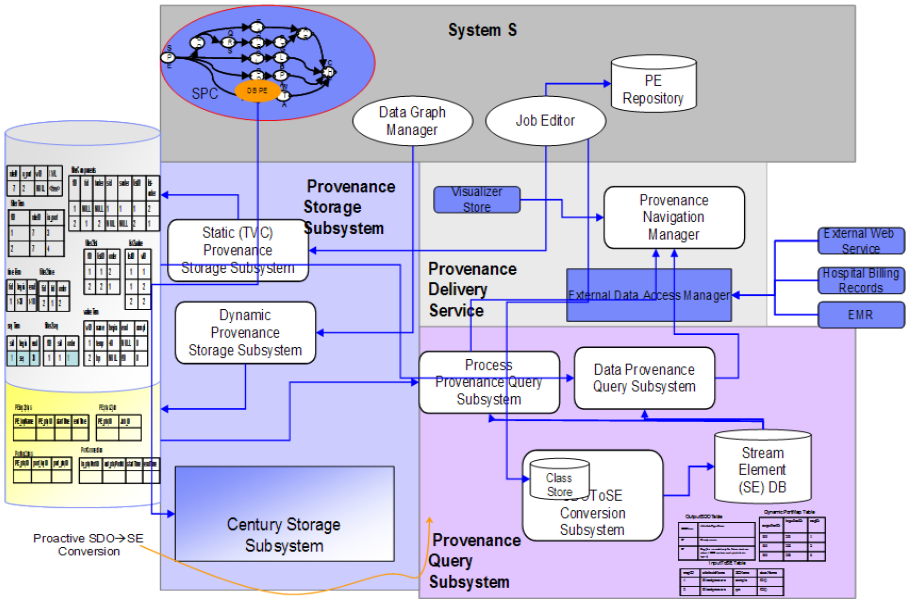
System S
SPC
DS PE
Data Graph Manager
Job Editor
PE Repository
Provenance Storage Subsystem
Static (TVC) Provenance Storage Subsystem
Dynamic Provenance Storage Subsystem
Century Storage Subsystem
Visualizer Store
Provenance Navigation Manager
Provenance Delivery Service
External Data Access Manager
External Web Service
Hospital Billing Records
EMR
Process Provenance Query Subsystem
Data Provenance Query Subsystem
Class Store
SDOToSE Conversion Subsystem
Stream Element (SE) DB
Provenance Query Subsystem
Proactive SDO→SE Conversion

Fig. 6. Century's current TVC-based Provenance Architecture. Currently, provenance is tracked for a specified subset of PEs, and TEs are proactively converted to SEs prior to storage.

of ECG samples) or elements belonging to different data types (e.g., carry both 'QRS', 'ECG' and 'BP' data in the same SDO). Figure 6 shows the resulting component level architecture of Century's current provenance architecture design. To provide the needed SDO→SE conversion, Century currently requires the use of developer-specified 'conversion classes', stored in the class store.

5 Related Work

Provenance support for workflow-based systems has been investigated relatively recently, primarily in the context of scientific workflows. The Karma provenance framework [11] uses a publish-subscribe architecture for capturing and propagating process and data provenance for data-centric workflows in computational grids. Similarly, the PreServ provenance solution [9] provides a service for explicitly documenting and storing the process provenance in scientific experiments. More recently, the CoMaD provenance framework [4] for scientific applications presented an annotation-based approach reduces the volume of provenance information recorded for a workflow, by allowing provenance annotations on collections to cascade to child elements. All of these approaches involve explicit provenance annotations and are thus geared towards transactional systems, where events between workflow components have a much lower rate.

Data provenance has been explored more actively in the context of databases. The overview paper [15] classifies existing works into two categories, namely, the *annotation* [8,13] vs. the *non-annotation* [6] approaches, based on whether, or not, additional *meta-data* are required to compute the provenance of data. The data provenance problem without the use of annotations has also been studied by Cui et al. [7], Buneman et al. [5], and Widom [18]. However, none of the works mentioned here considers streaming environments or the associated scalability issues.

The relatively limited work on scalable provenance for stream-oriented computing systems includes an efficient *process* provenance solution in [16], which focuses on identifying and storing dependencies among streams (by encoding, as a tree, the IDs of ancestor streams of a derived stream), rather than the data dependencies for individual stream elements. Our earlier work in [17] was one of the first to explore a model-based solution for *data* provenance in stream computing platforms.

6 Conclusions

We have described the initial implementation of a model-based data provenance solution (using TVC primitives) within Century, an extensible, high-performance stream processing system we are building to support online health analytics over medical sensor streams. While a TVC based approach incurs much lower overhead than annotation-based approaches, its scalability is limited by the resulting need to store elements of *all data streams* in persistent storage. To overcome this practical limitation, we proposed a new provenance architecture, called *CMIR*, which implements model-based provenance over PE clusters, and uses data replay to recreate stream elements within the cluster. To support CMIR, the Provenance system has to implement new functions such as state persistence and recovery, cascaded replay of data streams and automated composition of provenance specifications for virtual PEs. This architecture also requires technical innovations for *a)* creating useful provenance primitives that are cascadable and reversible, and *b)* for mediating differences in the granularity of production and consumption of data stream elements. We are addressing these challenges in ongoing work.

References

1. Abadi, D., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: A New Model and Architecture for Data Stream Management. VLDB Journal 2(2), 120–139 (2003)
2. Amini, L., Andrade, H., Bhagwan, R., Eskesen, F., King, R., Selo, P., Park, Y., Venkatramani, C.: SPC: A Distributed, Scalable Platform for Data Mining. In: SIGKDD 2006 Workshop on Data Mining Standards, Services, and Platforms, pp. 27–37 (August 2006)
3. Blount, M., Davis II, J.S., Ebling, M., Kim, J.H., Kim, K.H., Lee, K., Misra, A., Park, S., Sow, D.M., Tak, Y.J., Wang, M., Witting, K.: Century:Automated Aspects of Patient Care. In: 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007) (August 2007)
4. Bowers, S., McPhillips, T., Ludascher, B.: Provenance in Collection-Oriented Scientific Workflows. Concurrency and Computation: Practice & Experience, special issue on the First Provenance Challenge (in press, 2007)

5. Buneman, P., Khanna, S., Tan, W.C.: On propagation of deletions and annotations through views. In: Proceedings of the ACM PODS Conference (2002)
6. Chiticariu, L., Tan, W.C.: Debugging Schema Mappings with Routes. In: Proceedings of the VLDB Conference (2006)
7. Cui, Y., Widom, J., Wiener, J.L.: Tracing the lineage of view data in a warehousing environment. ACM Trans. Database Syst. 25(2) (2000)
8. Geerts, F., Kementsietsidis, A., Milano, D.: MONDRIAN: Annotating and Querying Databases through Colors and Blocks. In: Proceedings of the International Conference on Data Engineering (ICDE) (2006)
9. Groth, P., Luck, M., Moreau, L.: A protocol for recording provenance in service-oriented grids. In: Higashino, T. (ed.) OPODIS 2004. LNCS, vol. 3544, pp. 124–139. Springer, Heidelberg (2005)
10. Hildrum, K., Douglis, F., Wolf, J.L., Yu, P.S., Fleischer, L., Katta, A.: Storage optimization for large-scale distributed stream-processing systems. ACM TOS 3(4), 1–28 (2008)
11. Simmhan, Y.L., Plale, B., Gannon, D., Marru, S.: Performance Evaluation of the Karma Provenance Framework for Scientific Workflows. In: International Provenance and Annotation Workshop (IPAW) (May 2006)
12. Sow, D., Lim, L., Wang, M., Kim, K.H.: Persisting and querying biometric event streams with hybrid relational-XML DBMS. In: Proceedings of the International Conference on Distributed Event-Based Systems (DEBS), pp. 189–197 (June 2007)
13. Srivastava, D., Velegrakis, Y.: Intensional associations between data and metadata. In: Proceedings of the ACM SIGMOD Conference, pp. 401–412 (June 2007)
14. Sullivan, M., Heybey, A.: Tribeca: A System for Managing Large Databases of Network Traffic. In: Proceedings of the 1998 USENIX Annual Technical Conference (June 1998)
15. Tan, W.C.: Provenance in Databases: Past, Current, and Future. IEEE Data Eng. Bull. 30(4), 3–12 (2007)
16. Vijayakumar, N., Plale, B.: Towards Low Overhead Provenance Tracking in Near Real-Time Stream Filtering. In: International Provenance and Annotation Workshop, IPAW (May 2006)
17. Wang, M., Blount, M., Davis, J., Misra, A., Sow, D.: A Time-and-Value Centric Provenance Model and Architecture for Medical Event Streams. In: ACM HealthNet Workshop, pp. 95–100 (June 2007)
18. Widom, J.: Trio: A system for integrated management of data, accuracy, and lineage. In: Proceedings of CIDR (2005)