Lahar Demonstration: Warehousing Markovian Streams

Julie Letchner
University of Washington
letchner@cs.washington.edu

Christopher Ré
University of Washington
chrisre@cs.washington.edu

Magdalena Balazinska
University of Washington
magda@cs.washington.edu

Matthai Philipose
Intel Research Seattle
matthai.philipose@intel.com

ABSTRACT

Lahar is a warehousing system for *Markovian streams*—a common class of uncertain data streams produced via inference on probabilistic models. Example Markovian streams include text inferred from speech, location streams inferred from GPS or RFID readings, and human activity streams inferred from sensor data. Lahar supports OLAP-style queries on Markovian stream archives by leveraging novel approximation and indexing techniques that efficiently manipulate stream probabilities.

This demonstration allows users to interactively query a warehouse of imprecise text streams inferred automatically from audio podcasts. Through this interaction, the demonstration introduces users to the challenges of Markovian stream processing as well as technical contributions developed to address these challenges.

1. INTRODUCTION

People and computers worldwide generate exabytes of audio, video, text, GPS, RFID, and many other types of sensor/monitoring data daily—and because disk storage is cheap, most of this data is warehoused for future use [1]. Much of this archived data shares two properties: first, it is *sequential*, either because it is audio-visual or because it is temporal; and second, it is only *indirectly useful* to applications. For example, raw audio files can be searched and indexed by web crawlers only after they have been parsed into text. Similarly, smart-home sensor streams can be used to infer a resident's daily activities, but only the activity sequences themselves are used by applications to provide residents with alerts, reminders or daily activity logs. In order to be useful, warehouses for these huge archives must therefore expose views that allow queries to directly reference the high-level information of interest (*e.g.* words, activities) while hiding the details of the raw, low-level data (*e.g.* raw audio streams, RFID readings, *etc.*).

The process of computing high-level attributes from uncertain, low-level data is called *inference*. A data warehouse can expose inferred, high-level attributes to queries through views. Importantly, because of noise in the raw data sequences or ambiguity in the inference process (or both), the resulting view-level sequences are *uncertain*. Speech recognition systems, for example, often cannot exactly translate a particular phrase; instead they return a set of several guesses, each with a different probability. Such uncertain sequences are commonly represented as *Markovian streams*. A sample Markovian stream inferred from a news podcast is shown in Figure 1; its relational schema is shown in Figure 5. A Markovian stream is a compact representation of a probability distribution over an exponential number of possible sequences. These sequences represent spoken phrases, location trajectories, activity sequences, *etc.* depending on the domain. We refer readers to a recent overview of Markovian streams for additional details [2].

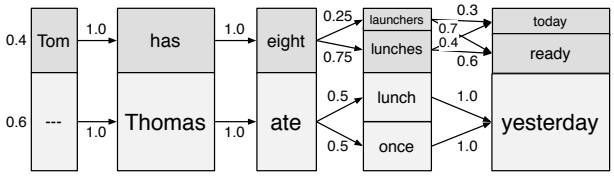

Figure 1: Visualization of a Markovian stream inferred from a podcast snippet. Boxes show the likelihoods of each word at each instant. Arrows show the conditional probabilities of each word *given* the previous word. Transitions with zero probability are omitted.

In this demonstration we present Lahar, a Markovian stream warehousing system that allows users to interactively query large Markovian stream archives. Queries are expressed to a front-end application: for example, "Find all occurrences of the phrase 'president Barack Obama' in January 2008 podcasts." Lahar computes the query in real time and returns results to the front-end for display. The results of the above query are a list of timestamps at which Lahar detects the phrase "president Barack Obama", as well as the probability of the detection (because Markovian streams are imprecise). The application interface links each timestamp in the result set to the corresponding audio snippet in the raw data for immediate playback, allowing users to explore the quality of Lahar's results. A mock-up of the demonstration setup is shown in Figure 2. In addition to occurrence-seeking queries, Lahar also supports aggregate queries (*e.g. "How many times did the word 'economy' appear across all podcasts from January 2008?"*); we discuss Lahar queries in more detail in Section 2.3.

The purpose of the Lahar demonstration is to introduce users to Markovian streams, to the importance and challenges of query processing on these streams, and to an array of techniques that make this processing feasible. We have curated a warehouse of of Markovian streams derived from audio data, on which users can pose

VLDB '09, August 24-28, 2009, Lyon, France

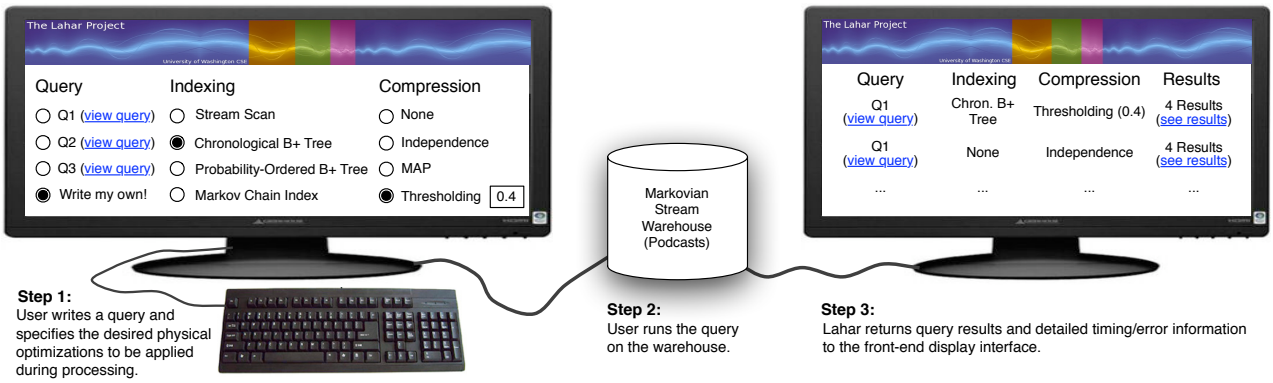

Figure 2: Mock-up of the demonstration setup.

arbitrary SQL-like Lahar queries. We also provide a set of pre-written example queries that highlight the expressive power of Lahar's event and OLAP queries, as well as some of the performance challenges posed by these queries. The demonstration is thus a showcase for the first Markovian stream warehousing system (Lahar). The demonstration allows users to explore and evaluate the effects of various physical query optimizations by interactively activating and de-activating these optimizations and observing the effect on both performance and, when approximations are leveraged, query accuracy.

Challenges

The primary challenge of Lahar is to process uncertain Markovian streams efficiently. This is difficult for three reasons: First, Markovian streams are large, because of both uncertainty and temporal interpolation. The size and rich semantics of Markovian streams dramatically slows processing time. Second, precise manipulation of Markovian stream correlations requires that streams be processed sequentially and from the beginning. This reduces the applicability of standard speedup techniques like B+ trees that extract small, disconnected stream intervals. Third, Markovian streams are *non-summarizable* (*i.e.* aggregate query values cannot be correctly computed from lower-level aggregate queries, but must be computed directly on the base Markovian streams). This invalidates the standard cube-based warehousing model for this data.

The Lahar system overcomes these challenges using physical optimizations described in detail in Section 2.3. These optimizations represent Lahar's technical contributions. More generally, the contributions of the demonstration are as follow:

1. We present a **system for processing OLAP queries on Markovian stream warehouses**. A front-end application interface receives user queries and displays query results (audio snippets) and meta-information (timing, approximation error, etc.).

2. The novel contribution of this demonstration is the full integration into a single system of the techniques introduced in prior work [5, 3, 4]. In particular, this demonstration represents the first time that Markovian stream indexing and compression techniques are leveraged simultaneously.

2. SYSTEM OVERVIEW

Lahar is a system that efficiently answers event and OLAP queries on Markovian stream warehouses. Its architecture is shown in Figure 4 and mirrors that of a traditional DBMS. Markovian streams are generated outside of the system (Section 2.1) and loaded into Lahar via a bulk loader (Section 2.2) which performs several indexing and compression tasks on each stream before writing the stream to disk. These loaded streams, together with dimension tables on both their certain and uncertain attributes, comprise a Markovian stream warehouse. At query time (Section 2.3), a parser and optimizer transform a SQL-like Lahar query into a plan which is executed by the execution engine. Results are returned to the appropriate application—in this case, the front-end display interface.

Lahar efficiently manages the uncertainty of Markovian stream warehouses using indexing [3] and approximation [4]; both techniques are introduced in prior work. Indexes allow the processing engine to skip irrelevant portions of each Markovian stream, even though event query processing requires sequential stream access. Approximation techniques allow the processing engine to trade accuracy for efficiency, depending on the requirements of a given query. Both of these technical components are described in further detail in Sections 2.2 and 2.3.

2.1 Data Preprocessing

The preprocessing (called *inference*) that transforms a raw sequence into a Markovian stream is not technically part of the Lahar system but is instead applied to data externally. The resulting output, a Markovian stream, is then loaded into Lahar. This allows Lahar to manage all Markovian streams, independently of their derivation. Such flexibility is important because different data domains require different modeling techniques and inference processes.

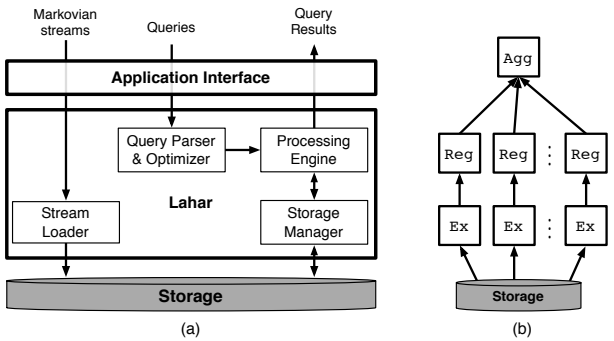

Figure 4: System architecture (a) and sample query plan (b).

```
1. SELECT <EXISTS | INSTANTS | COUNT>                                  // Temporal aggregation semantics
2. FROM Podcasts P                                                    // Identify the stream schema
3.    WITHKEY P.topic = 'News'                                        // Stream ID predicates
4.    WINDOW P.seqID=00:00 TO P.seqID=01:00                           // Temporal predicates
5. EVENT E1 NEXT E2                                                   // Event pattern structure
6.    WHERE E1.word = 'economic'                                      // Event predicates
7.    AND E2.word = 'downturn'                                        ...
8. GROUP BY P.SourceNetwork USING <STREAMEXISTS | STREAMCOUNT>        // Stream aggregation predicates/semantics
```

Figure 3: Query syntax. Lines 1-4 specify the relevant streams (podcasts), temporal interval (first minute), and temporal aggregation semantics (`EXISTS, INSTANTS, COUNT`). Line 5 specifies the event pattern (here, a two-word phrase). Lines 6-7 specify the pattern predicates (here, 'economic' then 'downturn'). Line 8 specifies the stream aggregation semantics (`STREAMEXISTS, STREAMCOUNT`) and grouping criteria (here, by podcast source).

The audio-based Markovian streams used in this demonstration warehouse are generated using the HTK speech recognition toolkit. We refer readers to a recent overview article [2] for a more broad discussion of Markovian stream generation.

2.2 Data Loading

Lahar leverages two key physical optimizations—indexing and approximation—to efficiently answer event and OLAP queries on Markovian streams. When a Markovian stream is loaded into the warehouse, Lahar computes and stores the information necessary to allow it to leverage these physical optimizations at query time.

2.2.1 Index Construction

Lahar constructs several different index structures on each Markovian stream. These structures index *uncertain* stream attributes (those that are associated with probabilities, as opposed to certain attributes like the stream ID). These indexes include [3]:

- **Chronological B+ Tree (BT_C)**: This is a standard B+ Tree on search key $\langle U_0, ..., U_n, t\rangle$, where U_i is the i^{th} uncertain attribute and t is the stream's sequence ID (*e.g.* timestamp). On the `Podcast` schema, the BT_C search key is $\langle word, t\rangle$.
- **Probability-Ordered B+ Tree (BT_P)**: This is another standard B+ Tree. Its search key is $\langle U_0, ..., U_n, p, t\rangle$ where the new identifier p represents the marginal probability of the specified uncertain state $U_0, ..., U_n$ at time t. On the `Podcast` schema, the BT_P search key is $\langle word, p, t\rangle$.
- **Markov Chain Index (MC)**: This is a hierarchical index structure that "summarizes" correlations between distant stream time steps. An example entry of the MC index on the `Podcast` schema might, for example, give the probability that the hundredth word in the podcast is 'finance' given that the tenth word is 'economy'.

At query time, Lahar's optimizer chooses an access method that leverages zero or more of these index structures.

2.2.2 Precomputing Approximations (Compression)

Lahar uses approximation (lossy compression) in addition to indexes to improve performance. Lossy compression is used instead of standard lossless techniques (*i.e.* run-length or dictionary encoding) for two reasons: First, lossless techniques achieve poor compression on the probability-heavy data of Markovian streams. Second and more importantly, Lahar's approximations alter the structure of the uncertainty represented by each stream. In many cases the simpler, approximate structure can be processed orders of magnitude more quickly than the original stream. Of course, these simplified structures lose some expressive power relative to the original stream. Thus, in general, each approximation technique represents a different point in the trade-off space between accuracy and efficiency. Experimental results on real-world data have shown that these techniques often achieve performance speedups of several orders of magnitude with minimal error [4].

When a new Markovian stream is loaded into the warehouse, Lahar materializes many copies of the stream. Each copy is approximated using a different lossy compression technique. At query time, Lahar's optimizer chooses the materialization that provides the optimal accuracy/efficiency trade-off for the given query (and accuracy requirements). Below we describe several of the approximation techniques employed by Lahar; for a full list we refer readers to our recent technical report [4].

- **Non-**compression does not alter a Markovian stream in any way; in otherwords, Lahar always retains a copy of the original stream.
- **Independence** compression simply drops (does not store) temporal correlations. Uncertainty about the state within each stream element, however, is retained. This reduces query processing time by roughly an order of magnitude. The effect on result quality depends on the strength of correlations in the data as well as on the event pattern length.
- **MAP** compression determinizes a Markovian stream into the single most likely (**m**aximum **a p**osteriori) sequence. This reduces processing time by several orders of magnitude but also dramatically reduces result quality in some cases. MAP compression can work well, however, for aggregate queries or on data containing only small amounts of uncertainty.
- **Thresholding** drops all correlations with probability below a threshold T—thus the parameter T offers direct control over the accuracy/efficiency tradeoff. While the optimal threshold depends on both the stream and query, relatively low values (*e.g.* $T = 0.1$) tend to greatly speed processing with minimal impact on result quality [4]. Thresholding can be used in combination with independence compression.

Because all of Lahar's compression techniques apply to individual stream elements, these compressed representations are easily indexed by the structures described in Section 2.2.1.

2.3 Runtime

At runtime, Lahar accepts a query in the SQL-like syntax shown in Figure 3. This syntax provides mechanisms for specifying stream selection predicates (line 3), temporal selection predicates (line 4), event sequences (line 5) and associated predicates (lines 6-7), temporal aggregation semantics (Line 1), and cross-stream aggregation semantics (line 8).

Marginal distributions

StreamID	SeqID	Word	p
...	...	...	
NPR	3	eight	0.4
NPR	3	ate	0.6
NPR	4	launchers	0.1
NPR	4	lunches	0.3
NPR	4	lunch	0.3
NPR	4	once	0.3
...	...	...	
CNN	0	Iran	
...	...	...	
NBC	0	the	
...	...	...	

Conditional distributions (correlations)

StreamID	SeqID	$Word_{PREV}$	$Word_{NEXT}$	p
...	...	...	...	
NPR	3	eight	launchers	0.25
NPR	3	eight	lunches	0.75
NPR	3	ate	lunch	0.5
NPR	3	ate	once	0.5
NPR	4	launchers	ready	0.7
NPR	4	launchers	today	0.3
NPR	4	lunches	ready	0.6
NPR	4	lunches	today	0.4
NPR	4	lunch	yesterday	1.0
NPR	4	once	yesterday	1.0
...	...	...	...	

a) Markovian stream **Podcast** in relational format

StreamID	Media	Topic	Length
CNN	TV	World	10 min
NBC	TV	U.S.	5 min
NPR	Radio	Technology	5 min

b) StreamID dimension table

Word	Part of Speech	Location
lunches	plural noun	--
the	article	--
Iran	proper noun	Middle East

c) Uncertain attributes dimension table

Figure 5: Markovian stream schema (a) and dimension tables (b)-(c). The stream snippet represented here is the stream shown in Figure 1.

All Lahar queries contain an event query at their core (lines 5-7). This event query specifies a pattern that will be detected in all streams included in the query. Like the stream selection predicates (line 3), the pattern predicates (lines 6-7) can reference the warehouse's dimension tables.

The results of this event query can be aggregated temporally within a single stream using one of three semantics: `EXISTS` (*"Did my phrase occur?"*), `INSTANTS` (*"Find all times when my phrase occurred."*), or `COUNT` (*"How many times did my phrase occur?"*). Temporally-aggregated results from each stream can be further aggregated across streams using either a `STREAMEXISTS` semantics (*"Did my phrase occur in any stream?"*) or a `STREAMCOUNT` semantics (*"How many times did my phrase occur across these streams?"*). Our technical report contains additional details about Lahar's query semantics and syntax [4].

The optimizer converts a parsed query into a query plan (Figure 4(b)). Lahar query plans always comprise trees of three operators in sequence: `Extract`, `Reg`, and `Agg`. The `Extract` operator (one per stream) retrieves stream elements from disk; this is where indexing and compression optimizations are leveraged. The `Reg` operator (one per stream) computes the core event query on the resulting extracted time steps, and also performs temporal aggregation. Finally, the single `Agg` operator aggregates per-stream results into a single result set.

3. DEMONSTRATION CONTENT

This demonstration allows users to interact with the Lahar Markovian stream warehouse by writing their own queries or submitting one of several pre-written examples. In response, Lahar returns not only result tuples, but also information about the query plan (including any indexes and compressed stream materializations that were leveraged), timing results, and accuracy results when appropriate. This information is displayed in a front-end interface similar to the mock-up shown in Figure 2. For queries whose results include timestamps, the front-end allows immediate playback of the audio snippets identified by these timestamps. This feedback allows users to more intuitively understand result quality.

The warehouse is loaded with a Markovian stream warehouse inferred from audio streams derived from various sources. All of the Markovian streams in the warehouse adhere to the `Podcast` schema in Figure 5.

In order to encourage users to explore the tradeoffs of various access methods, we have constructed this demonstration in the context of a simple game. In this game the user plays the role of the Lahar query optimizer. His or her goal is to find the optimal access method for a particular query and a particular weighting of time-vs.-accuracy. In this format, the demonstration walks users through the following steps:

1. Query submission is done by selecting one of several pre-written examples or by writing a query from scratch.

2. Time/accuracy valuation is chosen by the user from one of several pre-selected values. These values represent various scenarios including interactive data exploration (time is highly valued), model learning (accuracy is highly valued), *etc.*

3. Compressed stream view selection is expressed by radio-button selection of one of the lossily-compressed stream materializations described in Section 2.2.2.

4. Access method selection is also done via radio-button selection, but in this case the selection is of one of the four indexes described in Section 2.2.1.

5. Query execution is user-triggered after steps 1-4.

6. Result playback allows users to evaluate the success of their chosen query plan. The demonstration provides a single quality score based on the time/accuracy valuation chosen by the user. This score is the quantitative expression of the quality of a particular access method on a particular query. Users are encouraged to explore (via audio playback) the results of their queries to better understand the performance/quality tradeoffs.

4. CONCLUSION

This demonstration presents Lahar, a fully-functional system for querying Markovian stream warehouses. The demonstration is supported by a front-end game application in which users play the role of the query optimizer, and a Markovian stream back-end data store derived from audio streams. Through this demonstration we hope to familiarize users with the challenges involved in probabilistic stream processing as well as some of the interesting open challenges in this area.

Acknowledgements

This work was partially supported by NSF Grants IIS-0713123, CNS-0454425, IIS-0513877, IIS-0627585, and CRI-0454394; by gifts from Intel Research; and by M. Balazinska's Microsoft Research New Faculty Fellowship. J. Letchner is supported by an NSF graduate fellowship.

5. REFERENCES

[1] IDC. The expanding digital universe: A forecast of worldwide information growth through 2010. An IDC White Paper sponsored by EMC., March 2007.

[2] J. Letchner, C. Re, M. Balazinska, and M. Philipose. Challenges for event queries over markovian streams. *IEEE Internet Computing*, 12(6):30–36, 2008.

[3] J. Letchner, C. Re, M. Balazinska, and M. Philipose. Access methods for markovian streams. In *25th International Conference on Data Engineering*, 2009.

[4] J. Letchner, C. Re, M. Balazinska, and M. Philipose. Supporting olap queries on markovian streams. In *Technial Report. University of Washington*, 2009.

[5] C. Ré, J. Letchner, M. Balazinska, and D. Suciu. Event queries on correlated probabilistic streams. In *SIGMOD Conference*, pages 715–728, 2008.