Temporal Analytics on Big Data for Web Advertising

Badrish Chandramouli, Jonathan Goldstein, Songyun Duan
Data sources, assets, feeds, stock tickers, sensors, etc.

The M3 Cycle

Monitor → Data Store → Mine → Manage

Input Streams: Monitor → Input Streams → Data Store

Output Streams: Input Streams → Output Streams

Common paradigm across scenarios
- Call-center analytics
- Financial risk analysis
- Fraud detection
- Web advertising
The M3 Cycle

Example: Behavior-targeted Web advertising
- Observe user activity (e.g., searches) & deliver relevant ads
- Example: visit to carfax.com indicates interest in buying cars

(1) Monitor
- User searches
- URLs visited
- Ad clicks
- Ad impressions

(2) Mine
- Build models to relate recent user activity to ad click likelihood
- Use map-reduce

(3) Manage
- Maintain real time per-user activity info.
- Score user using model
- Deliver relevant ad

That’s three separate pieces of complex custom software
- Transitions between them are not smooth
Aren’t the model and its exploitation somehow related?
How can we leverage the commonality?

Core Observations
- The input data is temporal
- The queries are temporal (time is central)
  - Example: Generation of training data
    <user history, ad click/no click>
- True for both manage and mine phases
A Simple Example

- **Mine**: Compute the number of clicks (or average CTR) for each ad in a 6-hour window, varied over a 30-day dataset.

- **Manage**: Report – in real time – the number of clicks (or average CTR) for each ad in a 6-hour sliding window.

Difference is in *setting*, not *expression*
- They are both temporal in nature
- Mining has all the data available
- Mining is more resource intensive
Our Solution

- Use a DSMS language to express both
  - Easier to express time-oriented queries
- Processing: use DSMS in manage phase
- How to process temporal queries on offline data during mine phase?
  - Build and use a distributed DSMS?
    - Complicated, solves a much harder problem
  - Leverage today’s map-reduce systems that are perfect for resilient big-data analytics
User writes declarative temporal queries
  E.g., StreamInsight LINQ or StreamSQL
TiMR processes queries on offline data
Interface unmodified map-reduce cluster and unmodified DSMS
Use M-R for scale-out
  Automatically generate M-R jobs
Run DSMS inside reducers, in each data partition
  Each DSMS runs a part of the original query
Benefits

- Works with today’s infrastructure and software artifacts (DSMS, map-reduce)
- Language makes temporal reasoning much simpler
- Time is a first-class citizen: some processing becomes more efficient (vs. set-oriented)
  - Self-join vs. temporal join to correlate clicks with corresponding impression
- Real-time queries can be back-tested on large offline data
- Side-effect: Our analytics queries are “real-time-ready”
Declarative Temporal Query

StreamInsight LINQ Query

```csharp
var clickCount = from e in inputStream
                 where e_STREAMId == 1 // filter on some column
                 group e by e.AdId into grp // group-by, then window
                 from w in grp.SlidingWindow(TimeSpan.FromHours(6))
                 select new Output { ClickCount = w.Count(), .. };
```
Partitioning by application time
- Useful when no grouping key, windowed operations by time
- Automatically choose partitioning key
  - \{ UserId, Keyword \} \rightarrow \{ UserId \}
- Can use Cascades-style query optimizer

Application-time-based stream processing
- Real-time & offline queries are “compatible”
We perform a case study for behavioral targeted Web advertising implemented using ~20 LINQ queries. Implemented using ~20 LINQ queries. Easier than customized reducers.

User Behavior Profile

Input data Schema:

<table>
<thead>
<tr>
<th>Time:long</th>
<th>StreamId:int</th>
<th>UserId:string</th>
<th>KwAdId:string</th>
</tr>
</thead>
</table>

Is this a Practical Solution?
Example 1: Bot Elimination

Eliminate users with too many clicks or keyword searches in a short duration
Example 2: Feature Selection

- Preserve relevant keywords w.r.t. ad clicks
- We use statistical hypothesis-testing
  - For each \{ad, keyword\}, score the relevance of keyword for ad
  - Retain top K keywords for each ad
- For each \{ad, keyword\}, we need 4 counters:
  - #clicks and #impressions with/without keyword
- Easily implemented as temporal queries
- Incremental dimensionality reduction
Example 2: Feature Selection

UserDefinedFunction: StatisticalTest()

TemporalJoin (UserId)

GroupApply (AdId)
  - Count
    - Window
  - Filter StreamId=0

GroupApply (AdId)
  - Count
    - Window
  - Filter StreamId=1

GroupApply (AdId, KW)
  - Count
    - Window
  - Filter StreamId=0

GroupApply (AdId, KW)
  - Count
    - Window
  - Filter StreamId=1

Input-1

GenTrainData
Implemented TiMR to work with
- Microsoft StreamInsight DSMS
- SCOPE/Cosmos M-R system

One week of logs in Cosmos
- Separate into training and test data

Ten ad classes

250M unique users, 50M keywords
Evaluating TiMR

- Lines of code
  - Order of magnitude lower than custom code
  - Declarative & temporal
- Performance not affected significantly

![Bar chart showing comparison between Custom and TiMR time taken in hours.](chart.png)
Time-based Partitioning

- Partitions overlap at time-boundaries
  - Small partitions → too much redundant work
  - Large partitions → not enough parallelism

![Graph showing runtime vs. partition width](image-url)
## Keyword Elimination: Case I

<table>
<thead>
<tr>
<th>Highly Positive Keyword</th>
<th>Score</th>
<th>Highly Negative Keyword</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>celebrity</td>
<td>11.0</td>
<td>verizon</td>
<td>-1.3</td>
</tr>
<tr>
<td>icroly</td>
<td>6.7</td>
<td>construct</td>
<td>-1.4</td>
</tr>
<tr>
<td>tattoo</td>
<td>8.0</td>
<td>service</td>
<td>-1.5</td>
</tr>
<tr>
<td>games</td>
<td>6.5</td>
<td>ford</td>
<td>-1.6</td>
</tr>
<tr>
<td>chat</td>
<td>6.5</td>
<td>hotels</td>
<td>-1.8</td>
</tr>
<tr>
<td>videos</td>
<td>6.4</td>
<td>jobless</td>
<td>-1.9</td>
</tr>
<tr>
<td>hannah</td>
<td>5.4</td>
<td>pilot</td>
<td>-3.1</td>
</tr>
<tr>
<td>exam</td>
<td>5.1</td>
<td>credit</td>
<td>-3.6</td>
</tr>
<tr>
<td>music</td>
<td>3.3</td>
<td>craigslist</td>
<td>-4.4</td>
</tr>
</tbody>
</table>

**Ad = Deodorant Ad**
## Keyword Elimination: Case II

<table>
<thead>
<tr>
<th>Highly Positive</th>
<th>Highly Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Keyword</strong></td>
<td><strong>Score</strong></td>
</tr>
<tr>
<td>dell</td>
<td>28.6</td>
</tr>
<tr>
<td>laptops</td>
<td>22.8</td>
</tr>
<tr>
<td>computers</td>
<td>22.8</td>
</tr>
<tr>
<td>Juris</td>
<td>21.5</td>
</tr>
<tr>
<td>toshiba</td>
<td>12.7</td>
</tr>
<tr>
<td>vostro</td>
<td>12.6</td>
</tr>
<tr>
<td>hp</td>
<td>9.1</td>
</tr>
</tbody>
</table>

**Ad = Laptop Ad**
Summary

- Data & queries often temporal in nature
  - use temporal language for both mining & managing
  - unified user model for temporal analytics

Two main contributions:

- TiMR Framework: process temporal queries over large offline datasets
  - uses unmodified DSMS & M-R

- Case study for Behavioral Targeted ads
  - temporal LINQ makes analytics easier
Microsoft

Be what’s next.