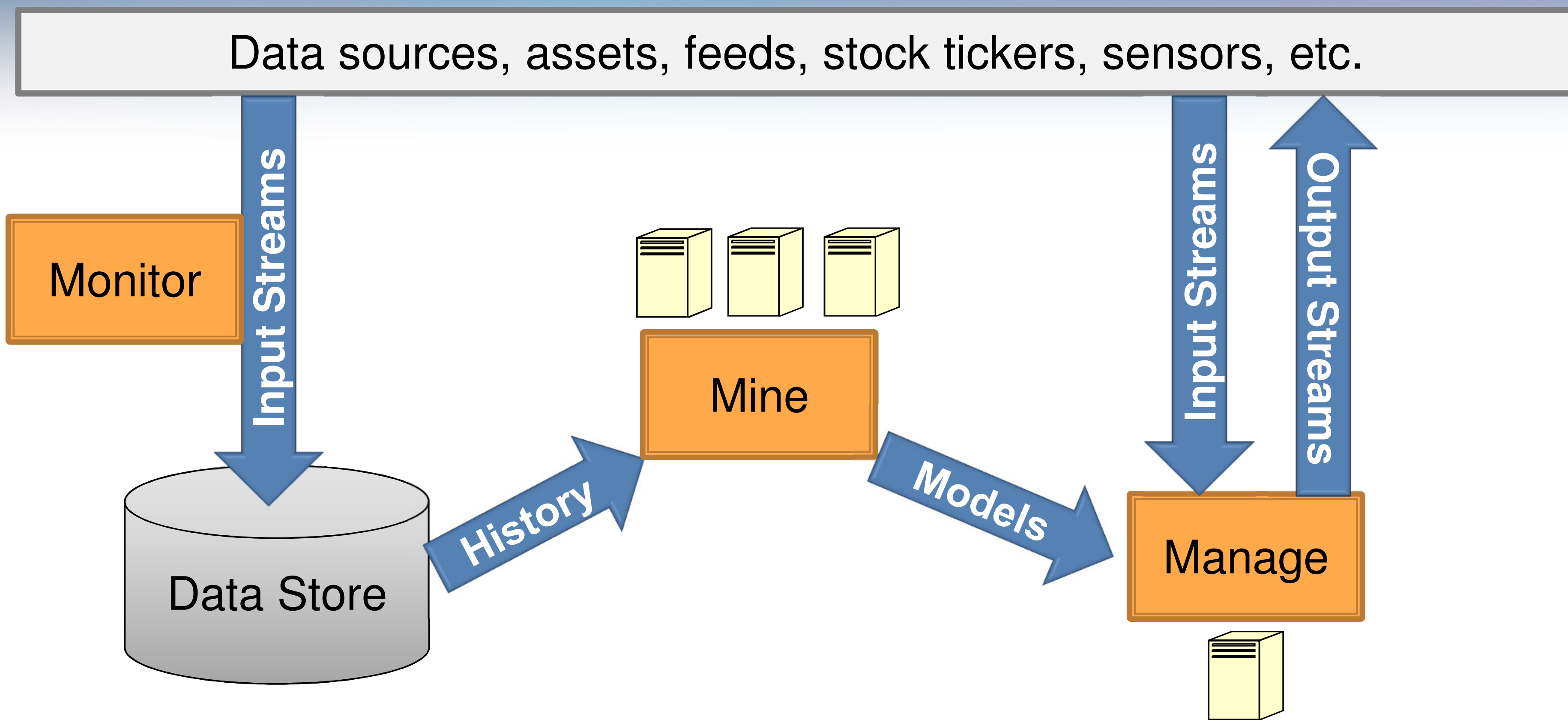
Temporal Analytics on Big Data for Web Advertising

Badrish Chandramouli, Jonathan Goldstein, Songyun Duan

Microsoft Research

Microsoft® StreamInsight™

The M3 Cycle



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

What is common?

- 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

The TiMR Framework

- 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

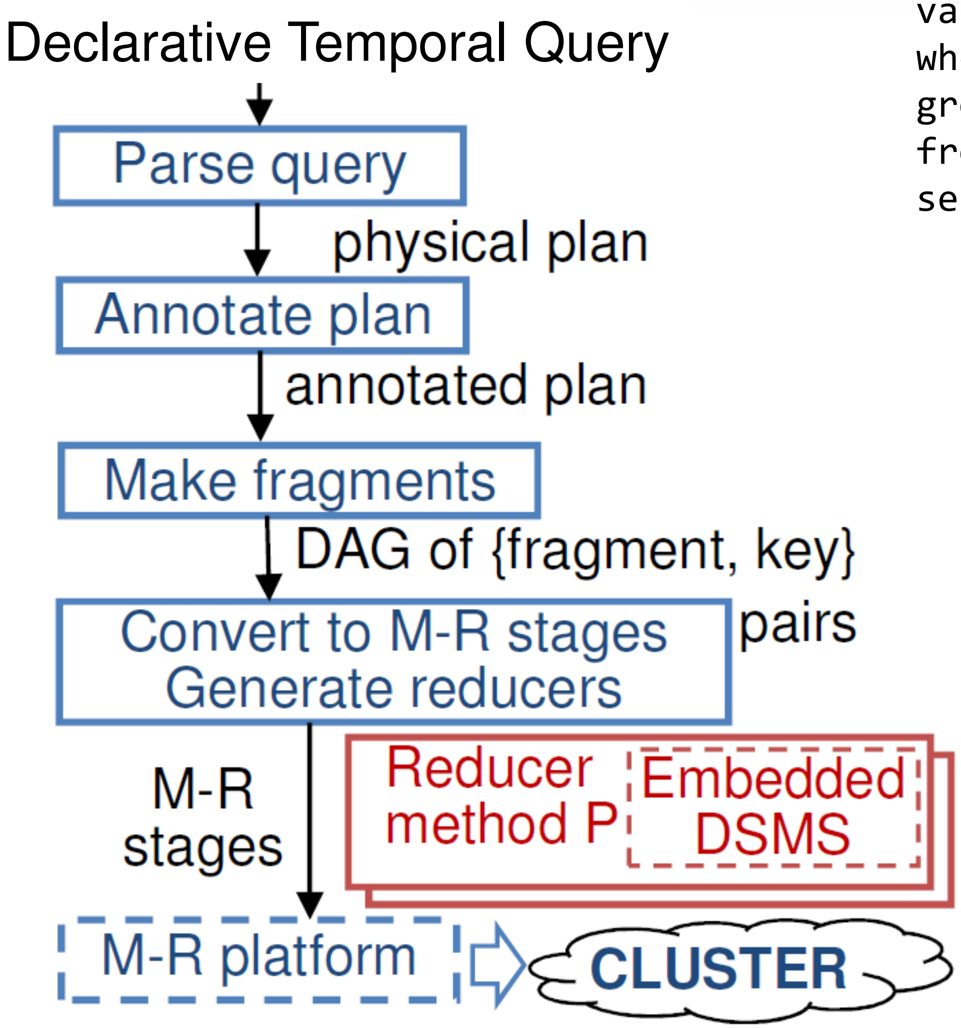
Benefits

- 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"

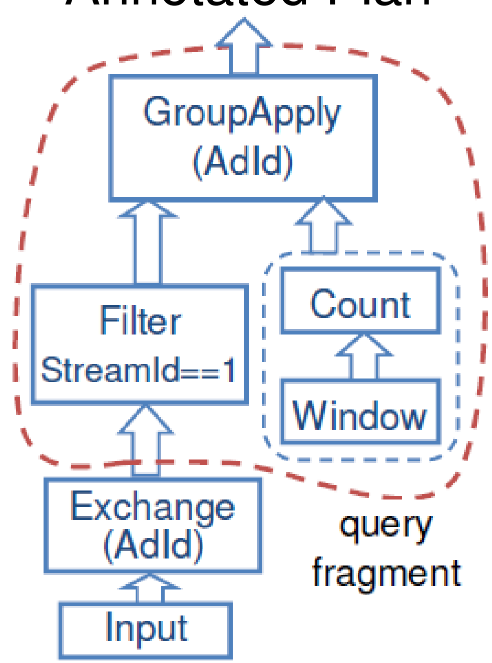
TiMR Workflow

StreamInsight LINQ Query

```
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(), .. };
```



Annotated Plan

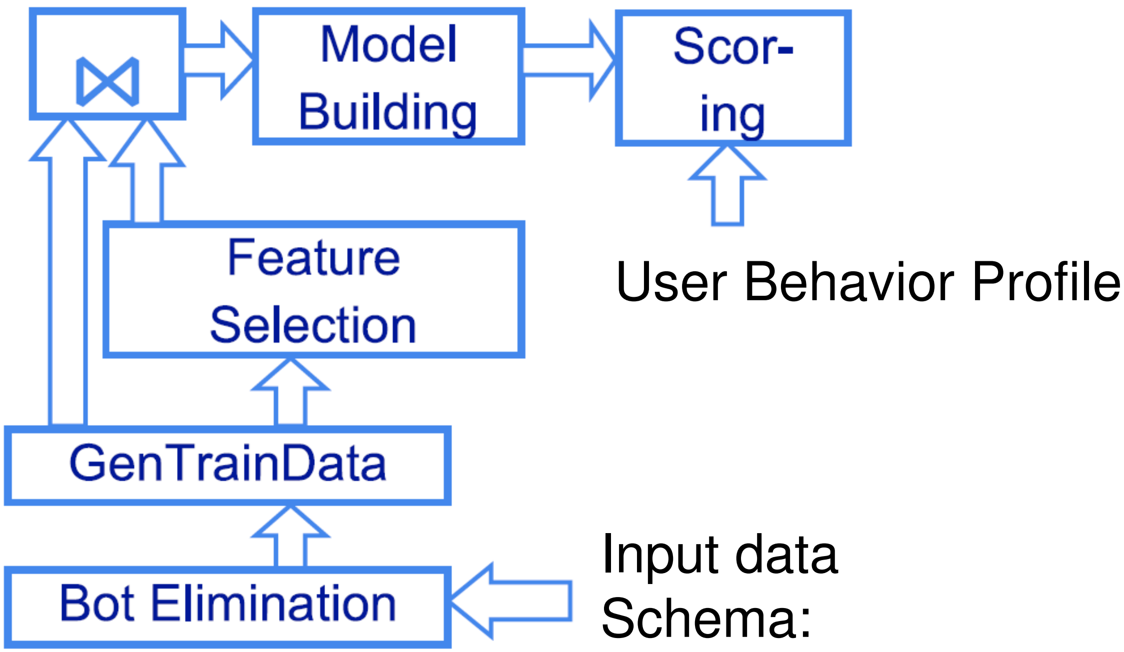


Discussion

- Partitioning by application time
  - Useful when no grouping key, windowed operations by time
- Automatically choose partitioning key
  - { UserId, Keyword } → { UserId }
  - Can use Cascades-style query optimizer
- Application-time-based stream processing
  - Real-time & offline queries are “compatible”

Is this a Practical Solution?

- We perform a case study for behavioral targeted Web advertising

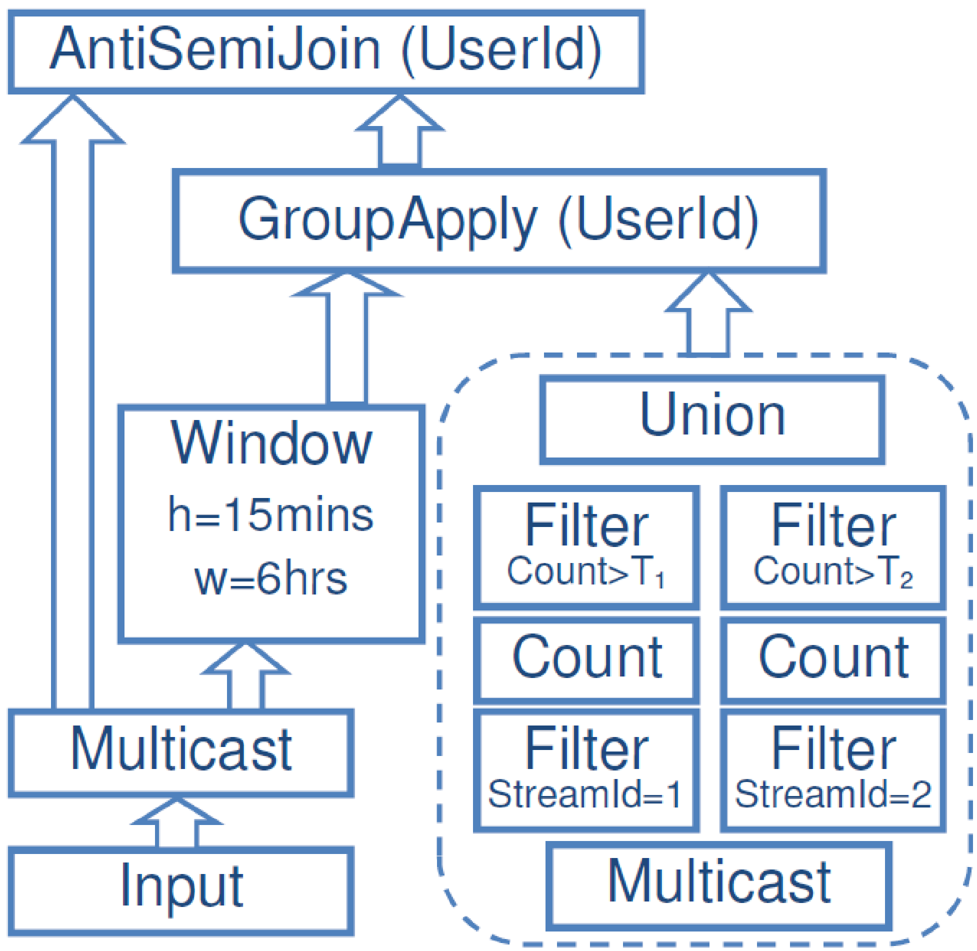


Time:long	StreamId:int	UserId:string	KwAdId:string

- Implemented using ~20 LINQ queries
  - Easier than customized reducers

Example 1: Bot Elimination

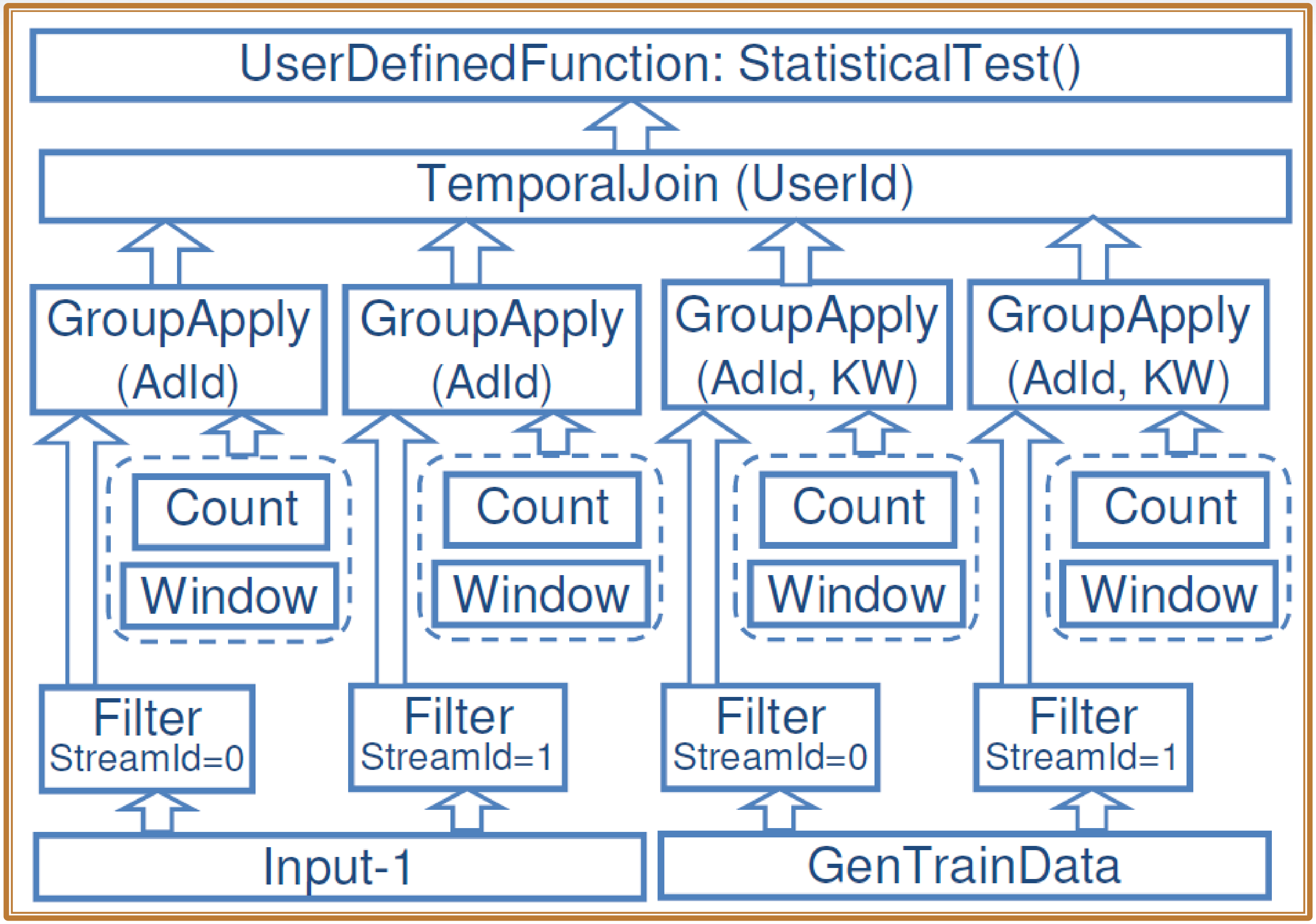
- Eliminate users with too many clicks or keyword searches in a short duration



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

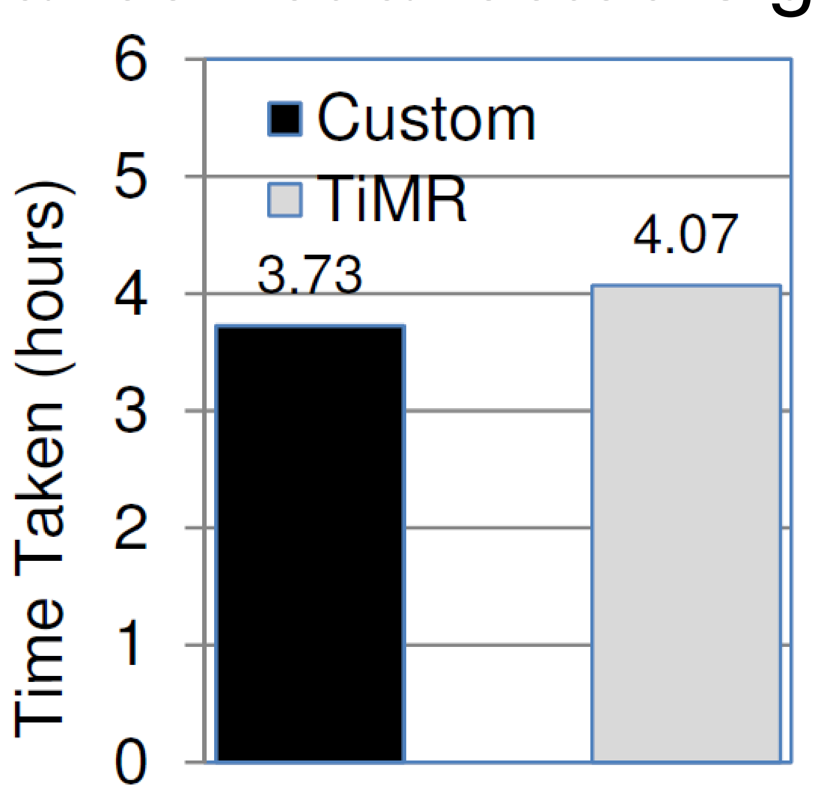


Implementation & Setup

- 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

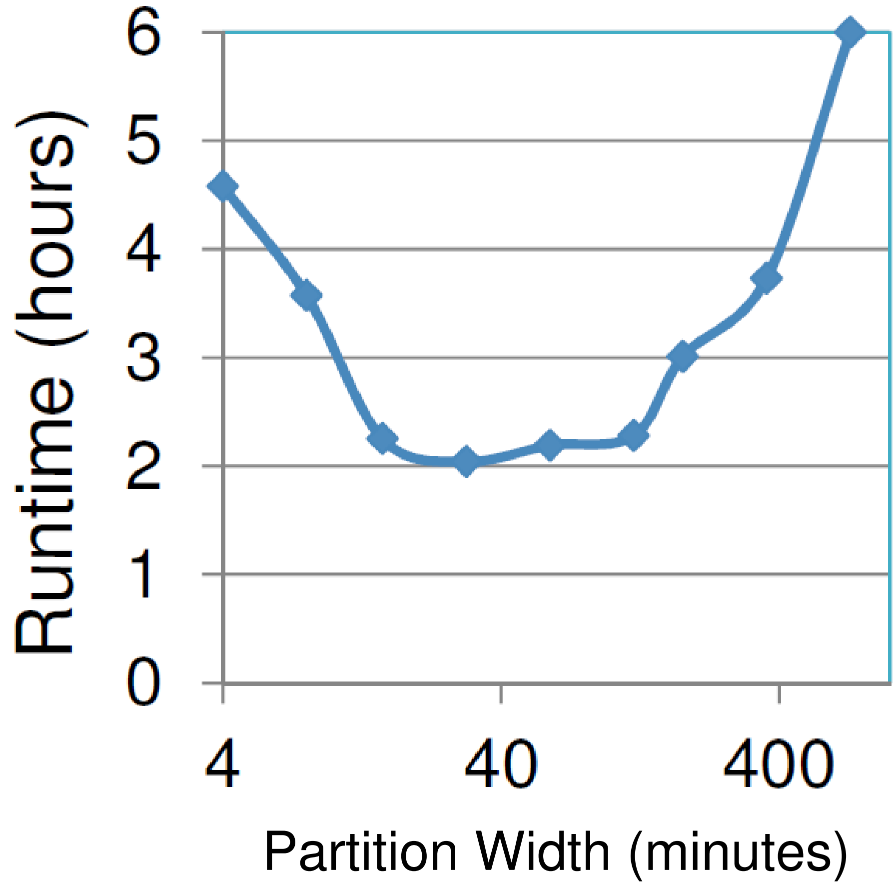
Evaluating TiMR

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



Time-based Partitioning

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



Keyword Elimination: Case I

Highly Positive		Highly Negative	
Keyword	Score	Keyword	Score
celebrity	11.0	verizon	−1.3
icarly	6.7	construct	−1.4
tattoo	8.0	service	−1.5
games	6.5	ford	−1.6
chat	6.5	hotels	−1.8
videos	6.4	jobless	−1.9
hannah	5.4	pilot	−3.1
exam	5.1	credit	−3.6
music	3.3	craigslist	−4.4

Ad = Deodorant Ad

Keyword Elimination: Case II

Highly Positive		Highly Negative	
Keyword	Score	Keyword	Score
dell	28.6	pregnant	–2.9
laptops	22.8	stars	–4.0
computers	22.8	wang	–4.2
Juris	21.5	vera	–4.2
toshiba	12.7	dancing	–4.2
vostro	12.6	myspace	–8.0
hp	9.1	facebook	–8.6

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 Research

Microsoft

Be what's next.