Triton: A Continuous Query Translation Engine for Trident/Storm

Presented by

Zhiheng Li

Advisor: Amarnath Gupta

Table of Contents

Introduction

- Background
- Motivation

Background

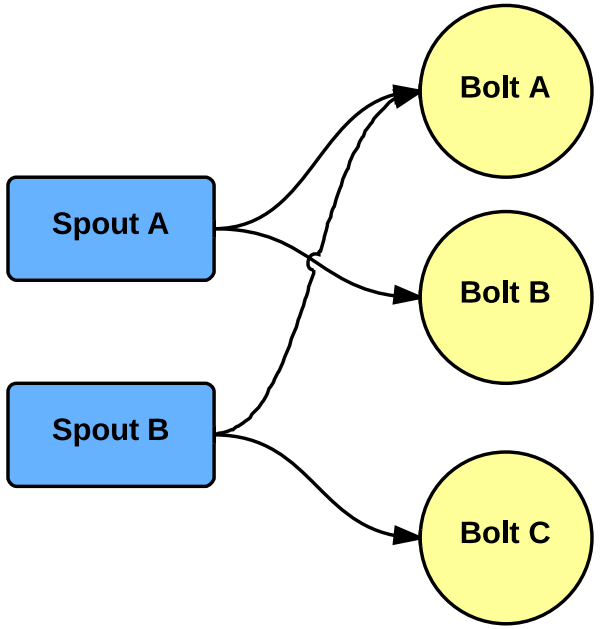
- Continues Query Language (CQL)
  - Sliding window
- Data Stream Management System (DSMS)
  - STREAM
- Complex Event Processing (CEP)
  - Esper (In memory)
- Distributed streaming computation
  - Storm/Trident

Sliding window

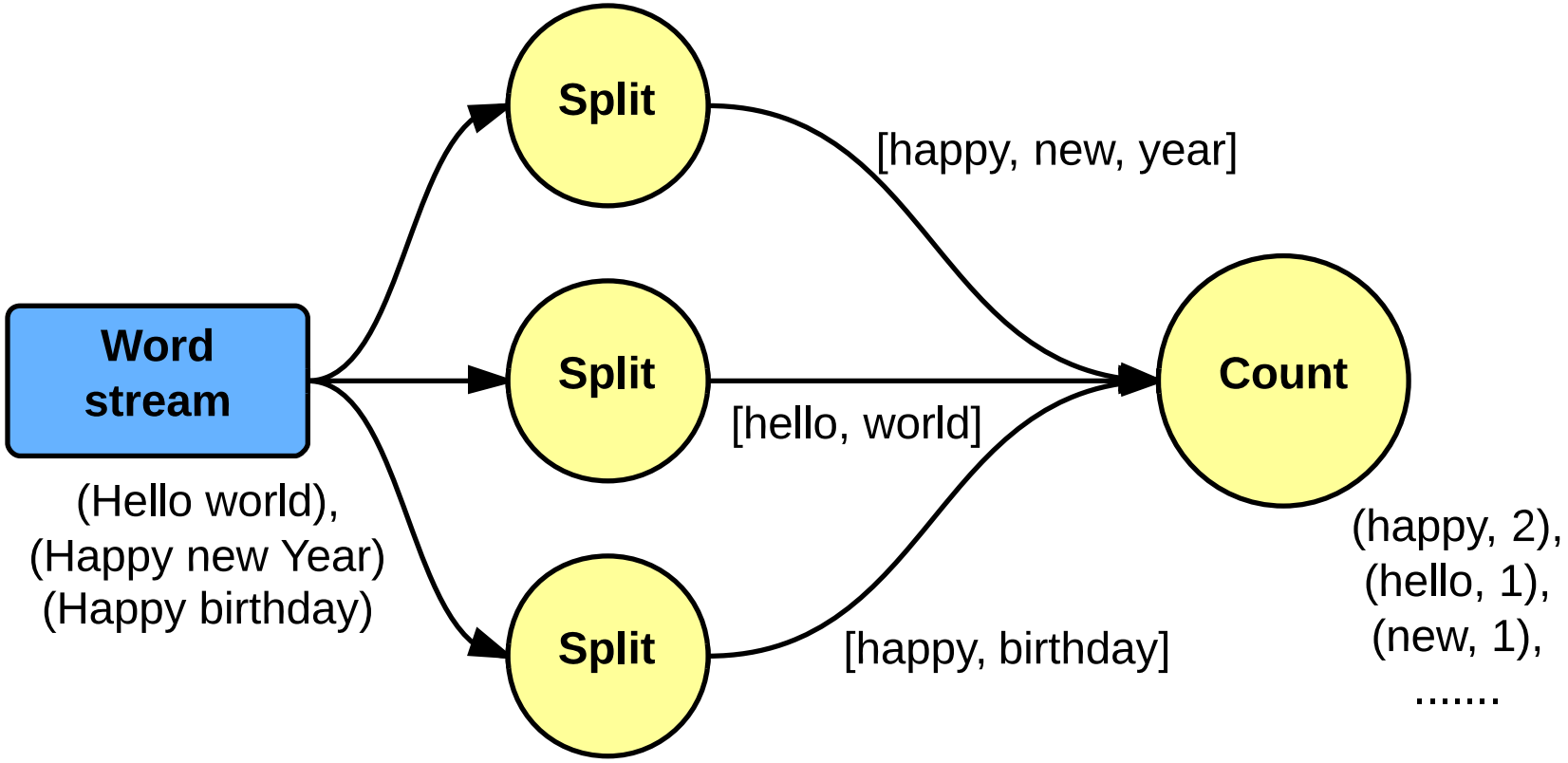
- Row based
  - A window only contains last *N* tuples
- Time based
  - A window contains tuples in last *N* (sec/min/hr)
- Time batch
  - A window that updates every *N* (sec/min/hr)

Storm Overview

- Topology: a graph of computation.
- Stream: key abstraction of data flow.
- Spout: source of stream.
- Bolt: computation node.



A Concrete Example of Storm Topology



Storm and Trident

- Storm: A distributed real-time computation system.
- Trident: A high-level abstraction on Storm to easy the development of Storm program.
- Problem:
  - Lack of high-level query language like CQL in STREAM.
  - No sliding window support.

Motivation

- Pig on top of Hadoop.
- A system that combine the advantage of Esper and Storm.
- The Triton translation engine
  - Compiles the *TQL* query into native Trident program written in JAVA.
- *TQL*: a query language for the Triton system.

Table of Contents

Triton Overview

- Design Decision
- System Overview

Design Decisions

- System architecture
- Compliable and readable code generation

System Overview

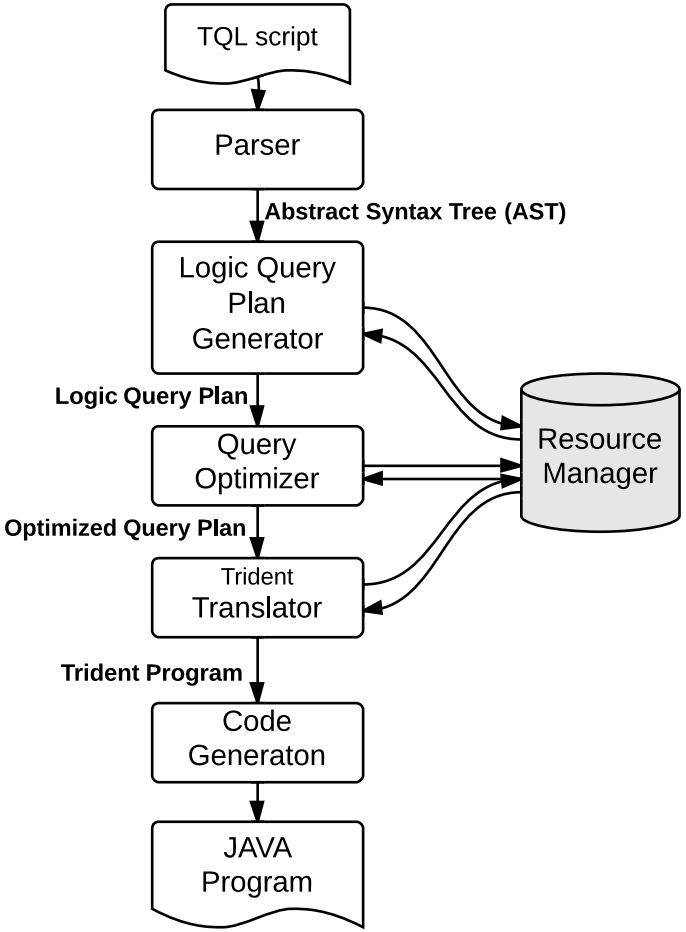


Table of Contents

Triton Query Language (TQL)

- Syntax Overview
  - An example

Syntax Overview

- An extension to standard SQL
- Stream Registration

```
REGISTER STREAM s(id int,
                  word string)
FROM file("word.dat")
```

- Sliding window definition

```
stock[code='GOOG'].win:time(1 minute)
```

Syntax Overview

- Query definition

```
SELECT word, count(word) as c_w
FROM wordStream:win:time(1 minute)
GROUP BY word;
```

- Named query definition
  - Query definition can appear on the `FROM` clause of the stream registration.

A complete example

- Trending topic

```
# register a word stream
REGISTER STREAM wordStream(word string)
FROM file("data/word.dat");

# compute word count for past 1 min.
REGISTER STREAM wordCountStream(word String,
wordCount int) FROM
   SELECT word, count(word) AS wordCount
   FROM wordStream.win:time(1 minute) AS s
   GROUP BY word;

# compute top 10 word
SELECT word FROM wordCountStream
ORDER BY wordCount DESC LIMIT 10;
```

Table of Contents

Implementation Details

- Parser
- Meta-data management
- Logic query plan
- Optimization
- Code generation

Parser

- JJTree and JavaCC
  - BNF grammar -> Abstract Syntax Tree (AST)
- Left-recursion elimination

- Usage of LOOKAHEAD
  - select a
  - select a, b

Meta-data Management

- Script level
  - Stream definition management.
- Query level
  - Stream/attribute Renaming.
  - Query dependencies.
  - Named/Anonymous query.

Logic Query Plan

- Introduce the window operator

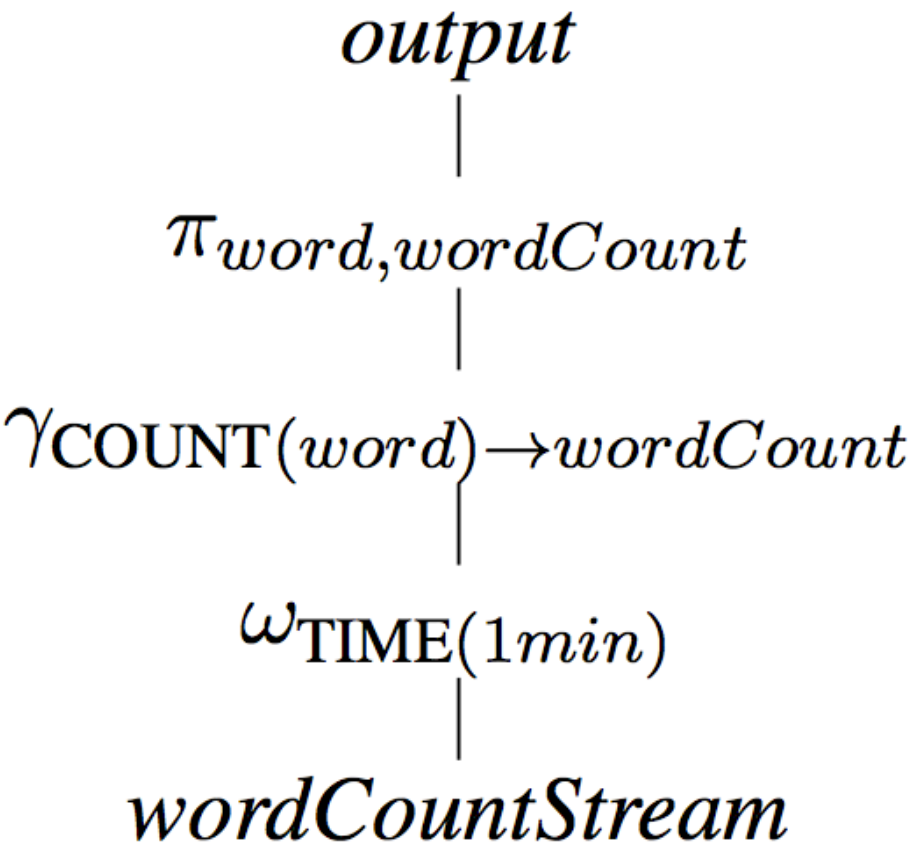
$$\omega_{TIME(1min)}\left(\sigma_{symbol='\mathrm{GOOG}'}\right)$$

- Translation from AST to logic query plan
  - simple case.
  - multiple streams involved.

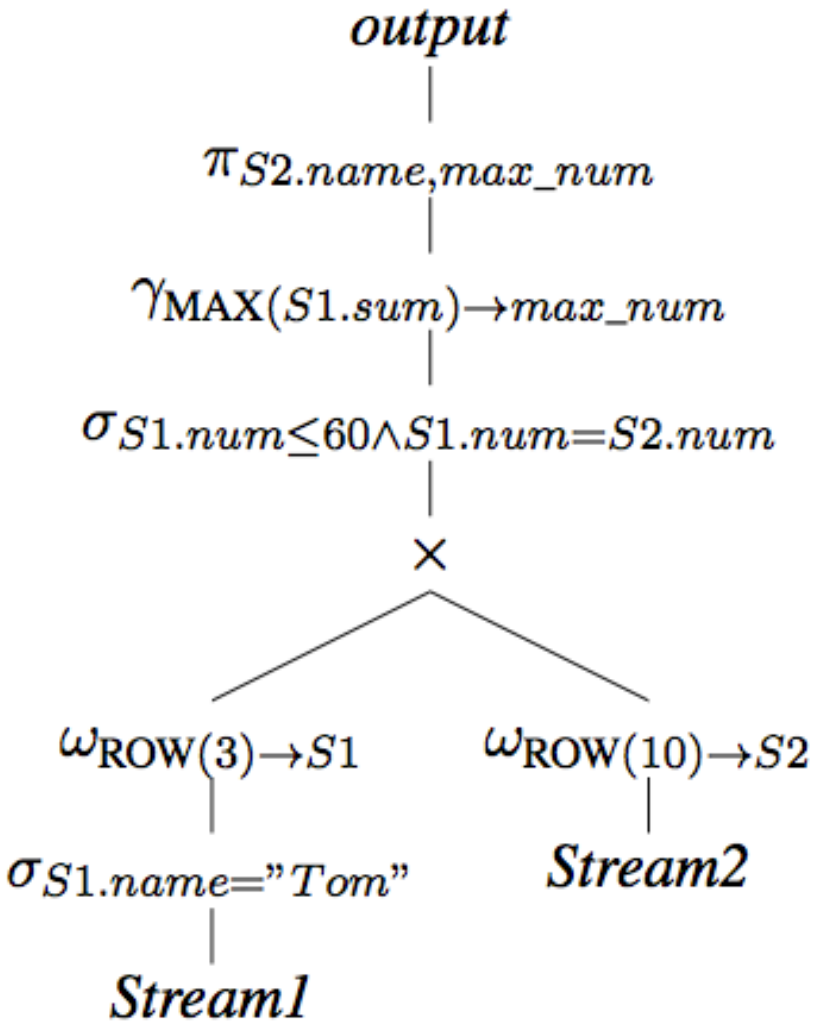
Simple case

- A simple logic plan

```
SELECT word,
count(word) AS
wordCount
FROM wordCountStream
.win:time(1 minute)
GROUP BY word;
```



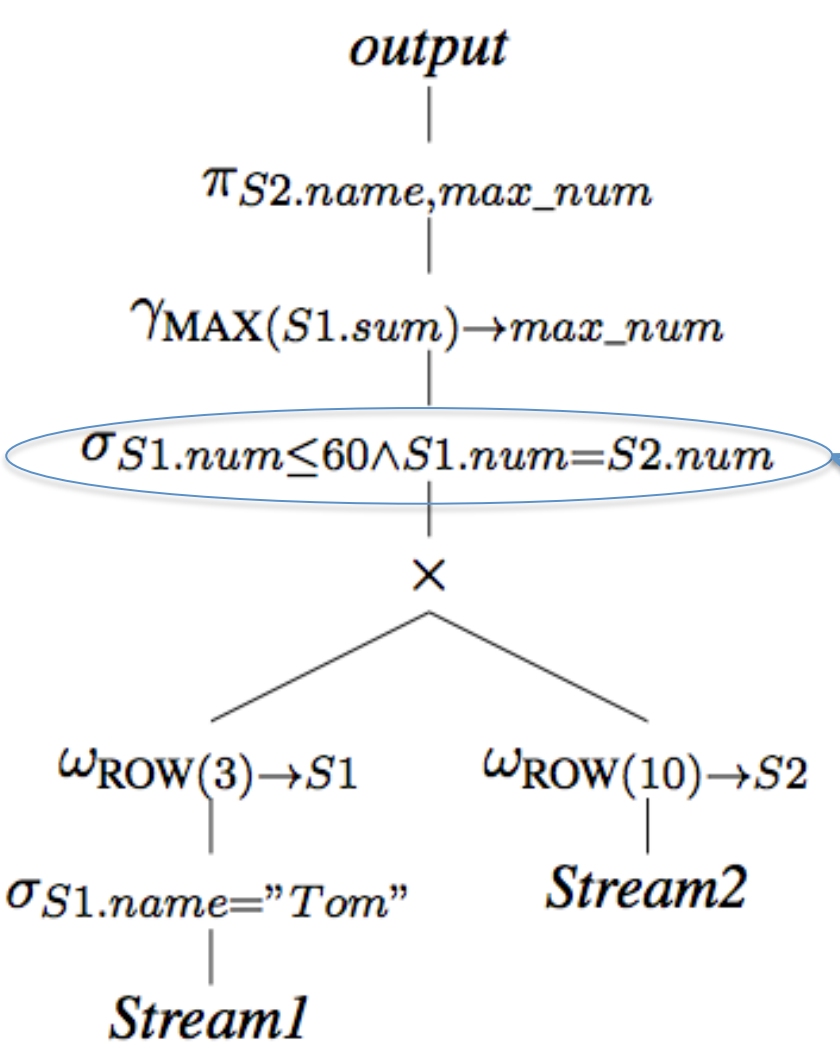
Multiple stream



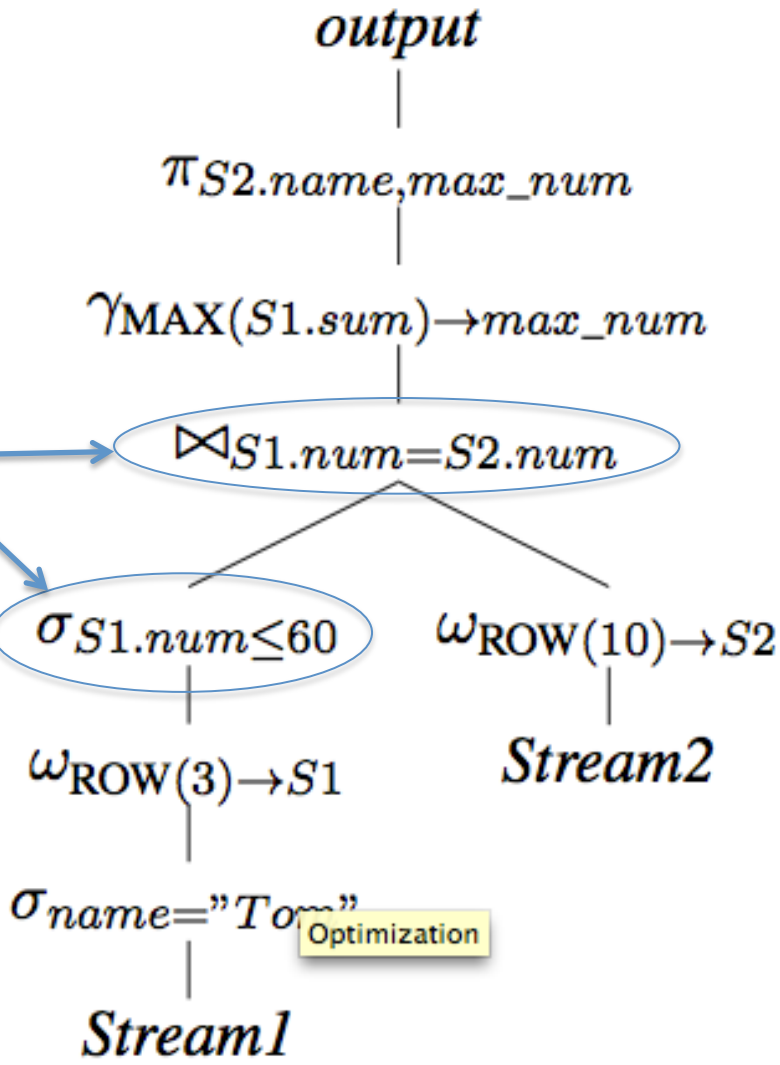
Optimization

- Selection push-down
- Join rewritten
- Optimization techniques
  - Decompose the logic expression into a list of AND connected small expressions
  - S1.num ≤ 60 ⋀ S1.num = S2.num

Original query plan



Optimized query plan



Code Generation

- Trident API Overview
  - newStream
  - each
    - function
    - filter
  - groupBy

Trident API Overview

- partitionPersistent
- aggregate
- applyAssembly
- newValueStream

Translation of Window Operator

- Eviction policy based ring buffer
- A combine of three Trident APIs
  - partitionPersistent
  - newValueStream
  - groupBy
- An example

Translation of Window Operator(cont.)

```
.newStream("wordCountStreams”, wordStream)
.partitionPersist(
   new TimeSlidingWindow.Factory(60),
   new Fields("wordStream.word"),
   new SlidingWindowUpdater(),
   new Fields("windowId", "wordStream.word"))
.newValuesStream()
.groupBy(
   new Fields("windowId”, "wordStream.word"))
```

Table of Trident API and logic operator mapping

Logic Operator	Trident API
Selection	each with Filter class
Projection	each with BaseFunction class project
InputStream	newStream
OutputStream	each
Aggregation	groupBy aggregation persistentAggregation
Window	partitionPersistent newValueStream groupBy
Join	join
OrderBy	applyAssembly(firstN)

JAVA Code Generation

- Query dependencies
  - Topological sort
- Import package management
  - Include a default import package list
- Building script
  - Include a default MAVEN script for compilation and execution

Table of Contents

Future Work

- Query plan optimization
- Built-in sliding window support
- User Defined Function support(UDF)
- Performance benchmark on cluster

Table of Contents

Conclusion

- A query translation engine
- Real-time streaming process ecosystem

Demo

- Triton compiler
- Trending topic application

Thanks

Q & A