Towards Efficient Stream Processing in the Wide Area

Matvey Arye
Siddhartha Sen, Ariel Rabkin, Michael J. Freedman

Princeton University
Our Problem Domain
Also Our Problem Domain
Use Cases

• Network Monitoring
• Internet Service Monitoring
• Military Intelligence
• Smart Grid
• Environmental Sensing
• Internet of Things
The World of Analytical Processing

Real-Time

Streaming

Historical

OLAP Databases
The World of Analytical Processing

Real-Time

Streaming

Simpler queries
Standing queries
Real-time answers

Historical

OLAP Databases

High ingest time
Fast query time

Borealis/Streambase
System-S, Storm

Oracle, SAP, IBM

Single Datacenter
Data Transfer
Trends in Cost/Performance 2003-2008

[CPU (16x)]
[Storage (10x)]
[Bandwidth (2.7x)]

[Above the Clouds, Armbrust et al.]
Aggregate At Local Datacenters
The World of Analytical Processing

JetStream = Real-time + Historical + Wide Area
Large Caveat

- Preliminary work
- We want feedback and suggestions
Challenges

• Query placement and scheduling
• Approximation of answers
• Supporting User Defined Functions (UDFs)
• Queries on historical data
• Adaptation to network changes
• Handling node failures
Motivating Example

• “Top-K domains served by a CDN”
  – Recall CDN is globally distributed
  – Services many domains

• Main Challenge: Minimize backhaul of data
How Is the Query Specified

Union → Count → Sort → Limit
Problems

- Single aggregation point
- Runs on a single node

Diagram:
- Union
- Count
- Sort
- Limit
Aggregate at local DC

DC1
- Count Partial

DC2
- Count Partial

DC3
- Union
- Count
- Sort
- Limit

Less Data
Count Partials

Count Partial

Union

Count

(Google, 1)

(Google, 1)

(Google, 1)

(Google, 1)

(Google, 1)
Non-Distributed Computation

DC1 -> Union -> Count -> Sort -> Limit

DC2

DC3
Split Count

DC1

DC2

DC3

Union

Count A-H

Count I-M

Count N-Z

Sort

Limit
Do Partial Sort

DC1

Load Bal.

DC2

Load Bal.

DC3

Count A-H → Sort Partial

Count I-M → Sort Partial

Count N-Z → Sort Partial

Sort

Limit
Distributed Version

DC1

Load Bal.

Count A-H

Sort Partial

Limit

DC2

Load Bal.

Count I-M

Sort Partial

Limit

Count N-Z

Sort Partial

Limit

Sort

Limit

DC3

Single Host
What Is New

• Previous streaming systems
  – User guided transformations (System-S, Storm)
  – Simple transforms (Aurora)

• JetStream
  – More complex transforms
  – Transformation is network aware
  – Annotations for user defined functions
Joint Problems

• Transformations
  – Choosing which ones

• Placement
  – Network constrained
  – Heterogeneous nodes
  – Resource availability

• Decision has to be made at run-time
Tackling the Joint Problems

• Using heuristics

• Split into increasingly more local decisions
  – Global decisions are coarse grained
    • Example: Assign operators to DCs
  – Localized decisions
    • Operate only on local part of subgraph
    • Have more current view of available resources
    • Do not affect other parts of query graph placement
Bottlenecks Still Possible

Use Approximations when necessary
Adjusting Amount of Approximation
As a reaction to network dynamism

If bottleneck goes away, return to exact answers
Approximation Challenges

• How to quantify error for approximations?
  – Uniform across approximation methods
  – Easy to understand
  – Integrates well with metrics for source/node failures

• How do we allow UDF approximation algorithms
  – Which exact operators can they replace
  – Quantifying the tradeoffs
  – Placement & Scheduling
If we approximate count, how does that error affect sort & final answer?
Do we need to approximate link DC1-DC3 if we approximate link DC2-DC3?
Discovering data trends?

• How has top-k changed over past hour?

• Current streaming systems don’t answer this
  – Except by using centralized DBs.

• JetStream proposes using storage at the edges
Hypercube Data Structure

<table>
<thead>
<tr>
<th>Data Size</th>
<th>Google</th>
<th>Yahoo</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;5Kb</td>
<td>(10, 5ms)</td>
<td>(100, 20ms)</td>
</tr>
<tr>
<td>50Kb-1Mb</td>
<td>(0, 0ms)</td>
<td>(1, 4ms)</td>
</tr>
<tr>
<td>&gt;1Mb</td>
<td>(5, 10ms)</td>
<td>(5, 30ms)</td>
</tr>
</tbody>
</table>

Minute: 1 ... 60
Hypercube Data Structure

- Month
  - 01...12
- Day
  - 1...31
- Hour
  - 1...24
- Minute
  - 1...60
Hypercube Data Structure

Month
  Day
    Hour
      Minute

Aggregate

<table>
<thead>
<tr>
<th>Google</th>
<th>Yahoo</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;5Kb</td>
<td>(90, 9ms)</td>
</tr>
<tr>
<td>50Kb-1Mb</td>
<td>(0, 0ms)</td>
</tr>
<tr>
<td>&gt;1Mb</td>
<td>(5, 10ms)</td>
</tr>
</tbody>
</table>
Query: “Last Hour and a half”
(without materializing intermediate nodes)
Query: “Last Hour and a half”
by materializing intermediate nodes

Month
- 01, ..., 12

Day
- 1, ..., 31

Hour
- 1, 2, ..., 

Minute
- 1, 60, 1, 30
Historical Queries

• Hypercubes have been used before
  – In the database literature

• What’s Novel
  – Storage at the edges (and in the network)
  – Time hierarchy
Challenges we talked about

- Query placement and scheduling
- Approximation of answers
- Supporting User Defined Functions (UDFs)
- Queries on historical data
- Adaptation to network changes
- Handling node failures
Conclusion

JetStream Explores...
+ Stream Processing
+ Historical Data / Trend Analysis
+ Wide Area

Thanks!
arye@cs.princeton.edu