Optimizing Grouped Aggregation in Geo-Distributed Streaming Analytics

Benjamin Heintz, Abhishek Chandra
University of Minnesota

Ramesh K. Sitaraman
UMass Amherst & Akamai Technologies
Wide-Area Streaming Analytics

• **Geo-distributed Streams**
  – Video streaming: client logs
  – CDN: server logs
  – Smart homes: sensor readings

• **Real-time Analytics**
  – Audience behavior
  – Web application usage
  – Energy consumption
Wide-Area Streaming Analytics

- **Geo-distributed Streams**
  - Video streaming: client logs
  - CDN: server logs
  - Smart homes: sensor readings

- **Real-time Analytics**
  - Audience behavior
  - Web application usage
  - Energy consumption

  - Hour-by-hour, how many unique users have streamed each video?
  - How many bytes are served for each content provider every minute?
  - Every 15 minutes, show me the average electrical demand by zip code.
Wide-Area Streaming Analytics

• **Geo-distributed Streams**
  – Video streaming: client logs
  – CDN: server logs
  – Smart homes: sensor readings

• **Real-time Analytics**
  – Audience behavior
  – Web application usage
  – Energy consumption

• **Windowed Grouped Aggregation**

**Questions**

- **Hour-by-hour**, how many unique users have streamed **each video**?
- **How many bytes** are served for each content provider every minute?
- **Every 15 minutes**, show me the average electrical demand by zip code.
**Windowed Grouped Aggregation**

- **Example**
  - *For each* 1 minute *Window*
  - *For each* content_provider_id
  - *Sum* bytes_transferred

- **Widely used**

- **General aggregation**
  - sum, count
  - average, standard deviation
  - (approximate) sets, counts, ...

Heintz, Chandra, Sitaraman | HPDC 2015
System Model

- *Hub-and-Spoke Infrastructure*
- Compute at edges and center
- Queries served from center
Metrics

• **WAN Traffic**
  – *System-level* measure of *cost*
  – Major OPEX component

• **Staleness**
  – *User-level* measure of *quality*
  – Delay in getting final results
  – Often an SLA component
Our Goal

- Algorithms to *jointly minimize* staleness & traffic
- Algorithm
  - Input: *sequence of arrivals*
  - Output: *sequence of updates*
- Traffic: total number of updates
- Staleness: delay until last update reaches center
Naïve Algorithms

• Pure streaming
  – Edge immediately flushes upon arrival
  – Excessive traffic

• Pure batching
  – Edge flushes only after end of window
  – Excessive staleness
A Running Example

- Count words in the stream
  
  "fast data is fast"

- Network: 1 aggregate / second
- Window length: 8 seconds
Pure Streaming

**fast**

**Time:** 0

**Edge**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Center**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pure Streaming fast

Time: 0

![Diagram showing key-value pairs and time intervals]

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Flush immediately

Heintz, Chandra, Sitaraman | HPDC 2015
Pure Streaming

**fast**

Time: 1

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
</tbody>
</table>

Heintz, Chandra, Sitaraman | HPDC 2015
Pure Streaming data

Time: 2

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
</tbody>
</table>
Pure Streaming

Time: 3

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Value</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Heintz, Chandra, Sitaraman | HPDC 2015
Pure Streaming

Key Value
Edge

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
</tbody>
</table>
### Pure Streaming

![Diagram showing Pure Streaming](image)

#### Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

**Time: 5**

---

Heintz, Chandra, Sitaraman | HPDC 2015
Pure Streaming

fast

Time: 6

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>
### Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>
Pure Streaming

Traffic: 4

Staleness: 0

Time: 8

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

Heintz, Chandra, Sitaraman | HPDC 2015
Pure Batching

**fast**

Time: 0

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pure Batching

data

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pure Batching

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pure Batching

fast

Time: 6

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pure Batching

Time: 8

Key Value

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pure Batching

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
</tbody>
</table>

Heintz, Chandra, Sitaraman | HPDC 2015
Pure Batching

Time: 10

Key Value

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
</tbody>
</table>

Traffic

Time
Pure Batching

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>
Pure Batching

Time: 11

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge</td>
<td>Center</td>
</tr>
</tbody>
</table>

**Staleness:** 3

**Traffic:** 3

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>this</td>
<td>1</td>
</tr>
</tbody>
</table>
Pure Batching

Time: 11

Naïve Algorithms: Staleness vs. Traffic
Roadmap

• Naïve algorithms
• *Optimal offline algorithms*
• Practical online algorithms
• Experimental evaluation
Eager Offline Optimal

- **Idea**: flush immediately after last arrival
- Properties:
  - Flushes exactly once per key (traffic optimal)
  - Flushes at the earliest possible time (staleness optimal)
Eager Offline Optimal

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Eager Offline Optimal

**Time: 2**

**Data**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
</tbody>
</table>

**Edge**

**Center**
### Eager Offline Optimal

**Data**

- **Time:** 2

![Firefighter](image)

#### Flush

**Immediately** upon last arrival

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Chart

<table>
<thead>
<tr>
<th>Time</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

**Heintz, Chandra, Sitaraman | HPDC 2015**
Eager Offline Optimal

![Diagram showing data flow between Edge and Center with key-value pairs]

**Edge**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
</tbody>
</table>

**Center**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1</td>
</tr>
</tbody>
</table>

Heintz, Chandra, Sitaraman | HPDC 2015
Eager Offline Optimal

Time: 4

Is

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1</td>
</tr>
</tbody>
</table>

Heintz, Chandra, Sitaraman | HPDC 2015
Eager Offline Optimal

Key Value
fast 1

data 1
is 1

Time: 5
Eager Offline Optimal

fast

Time: 6

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>
Eager Offline Optimal

Time: 7

Edge

```
<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Center

```
<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>
```
Lazy Offline Optimal

• **Idea**: flush as late as possible after last arrival
• Properties
  – Still one flush per key (traffic optimal)
  – Staleness optimal by construction
Lazy Offline Optimal

Time: 0

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Heintz, Chandra, Sitaraman | HPDC 2015
Lazy Offline Optimal

Key Value
fast 1
data 1

Edge

Center

Heintz, Chandra, Sitaraman | HPDC 2015
Lazy Offline Optimal

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time: 4

Traffic vs. Time

<table>
<thead>
<tr>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

Key Value is 1
```
<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Flush *as late as possible* after last arrival
Lazy Offline Optimal

Time: 6

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1</td>
</tr>
</tbody>
</table>
Lazy Offline Optimal

Time: 6

Key
Value

fast
2

is
1

Edge

Center

Key
Value

data
1
Lazy Offline Optimal

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

Heintz, Chandra, Sitaraman | HPDC 2015
Lazy Offline Optimal

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>
Lazy Offline Optimal

Edge

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Center

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

Staleness: 0
Traffic: 3

Time: 8
Family of Optimal Algorithms

![Graph showing traffic vs. time comparing eager and lazy algorithms. The eager algorithm starts with a sharp rise in traffic, while the lazy algorithm shows a gradual increase.](image-url)
Roadmap

• Naïve algorithms
• Optimal offline algorithms
• *Practical online algorithms*
• Experimental evaluation
Caching Analogy

• **Insight**: Treat the edge as a cache
  
• For staleness
    – **When** to flush?
    – Dynamic sizing policy

• For traffic
  – **What** to flush?
  – Eviction policy

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fast</td>
<td>2</td>
</tr>
<tr>
<td>data</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
</tbody>
</table>

Heintz, Chandra, Sitaraman | HPDC 2015
Practical Online Algorithms

• Dynamic sizing
  – Emulate the offline optimal cache sizes
  – Eager: *predict* how many keys will arrive again
  – Lazy: *predict* how late we can defer flushing
Practical Online Algorithms

• Dynamic sizing
  – Emulate the offline optimal cache sizes
  – Eager: *predict* how many keys will arrive again
  – Lazy: *predict* how late we can defer flushing

• Eviction
  – Don’t reinvent the wheel
  – Leverage existing work (LRU, LFU, ...)
Hybrid Online Algorithm

- **Eager**: better to overestimate size
  - Many early evictions
  - Risk: *incorrect eviction decisions*
- **Lazy**: better to underestimate size
  - Risk: *bandwidth misprediction*
  - Be pessimistic
- **Hybrid**: linear combination of eager, lazy cache sizes
  \[ c(t) = \alpha \cdot c_l(t) + (1 - \alpha) \cdot c_e(t) \]
Roadmap

• Naïve algorithms
• Optimal offline algorithms
• Practical online algorithms
• *Experimental evaluation*
## Experiments: Data Set

Month-long Akamai download analytics beacon log

<table>
<thead>
<tr>
<th>Name</th>
<th>Key</th>
<th>Value</th>
<th>Query Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>(cpid, bw)</td>
<td>bytes downloaded (sum)</td>
<td>$O(10^2)$ groups</td>
</tr>
<tr>
<td>medium</td>
<td>(cpid, bw, country_code)</td>
<td>bytes downloaded (moments)</td>
<td>$O(10^4)$ groups</td>
</tr>
<tr>
<td>large</td>
<td>(cpid, bw, url)</td>
<td>client ip (HyperLogLog)</td>
<td>$O(10^6)$ groups</td>
</tr>
</tbody>
</table>
Experiments: Implementation

- Implemented algorithms in Apache Storm
- One Storm cluster at center, each edge
- Seven total PlanetLab sites
Experimental Evaluation: Single-Edge

**Traffic**

- Batching
- Hybrid(0.25)
- Streaming
- Optimal

**Staleness**

- Batching
- Hybrid(0.25)
- Streaming
- Optimal
Experimental Evaluation: Single-Edge

**Traffic**
- Batching
- Hybrid(0.25)
- Streaming
- Optimal

**Staleness**
- Batching
- Hybrid(0.25)
- Streaming
- Optimal

- Near-optimal traffic
- Significant staleness reduction
Experimental Evaluation: Multi-Edge (3 edges)

**Traffic**

- **Normalized Mean Traffic**
  - Batching
  - Hybrid (0.25)
  - Streaming
  - Optimal

**Staleness**

- **Normalized 95th %-ile Staleness**
  - Batching
  - Hybrid (0.25)
  - Streaming
  - Optimal
Experimental Evaluation: Multi-Edge (6 edges)

Traffic

<table>
<thead>
<tr>
<th></th>
<th>Batching</th>
<th>Hybrid(0.25)</th>
<th>Streaming</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Staleness

<table>
<thead>
<tr>
<th></th>
<th>Batching</th>
<th>Hybrid(0.25)</th>
<th>Streaming</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Related Work

• **Systems**
  – Twitter Heron (SIGMOD ‘15)
  – Apache Spark Streaming (SOSP ‘13)
  – Google MillWheel (VLDB ‘13)
  – Muppet (VLDB ’12)

• **Wide-area Computing**
  – WANalytics (CIDR ‘15)
  – JetStream (NSDI ‘14)
  – Pietzuch et al. (ICDE ’06)

• **Optimization Trade-offs**
  – BlinkDB (VLDB ‘12)
  – LazyBase (EuroSys ‘12)
Conclusion

• Geo-distributed windowed, grouped aggregation
• Naïve algorithms: low staleness or low traffic
• Optimal offline algorithms to jointly minimize both
• Practical algorithms via caching
  – Dynamic sizing policies
  – Reuse existing eviction policies