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***

Hour-by-hour, ***how many unique users*** have streamed ***each video***?

How many bytes are served ***for each content provider every minute***?

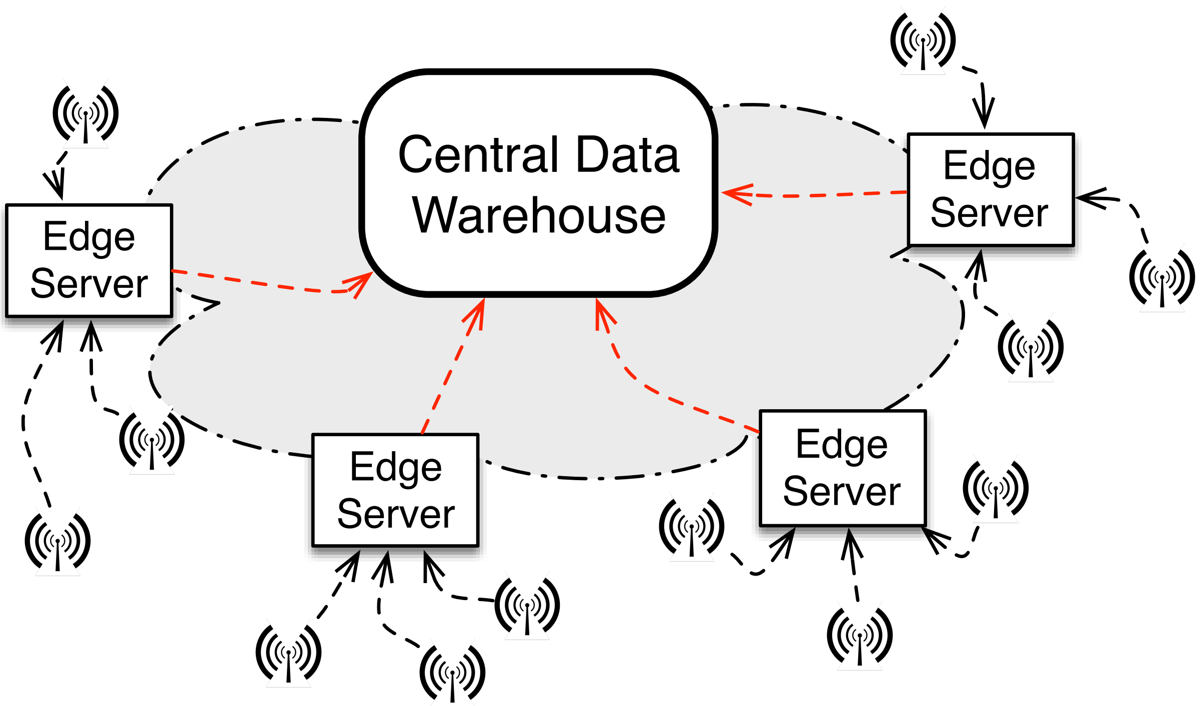
Every 15 minutes, show me the ***average electrical demand by zip code***.

Windowed Grouped Aggregation

- Example
  - *For each* `1 minute` *Window* ← windowed
  - *For each* `content_provider_id` ← grouped
  - *Sum* `bytes_transferred` ← aggregation
- Widely used
- General aggregation
  - sum, count
  - average, standard deviation
  - (approximate) sets, counts, …

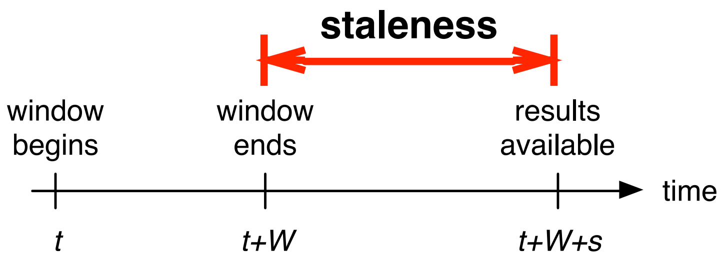
System Model

- *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

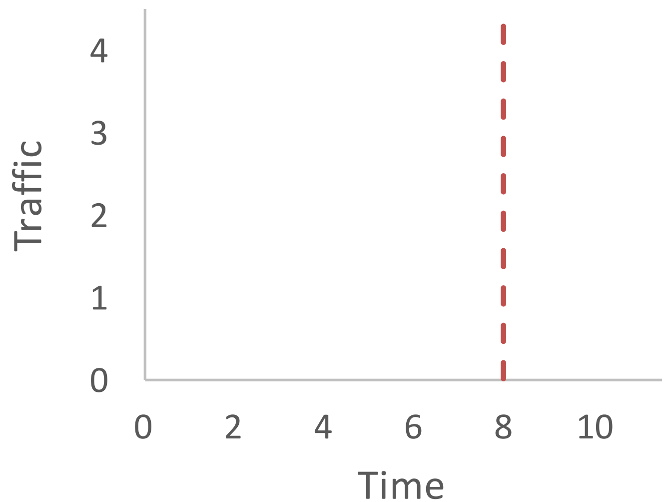
"fast data is fast"

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

Pure Streaming

Time: 0

fast



Edge

Key	Value

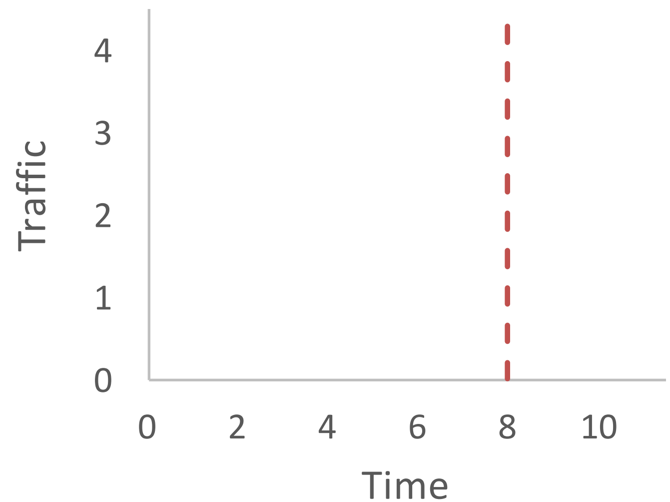
Center

Key	Value

Pure Streaming

Time: 0

fast



Edge

Key	Value

Flush
immediately

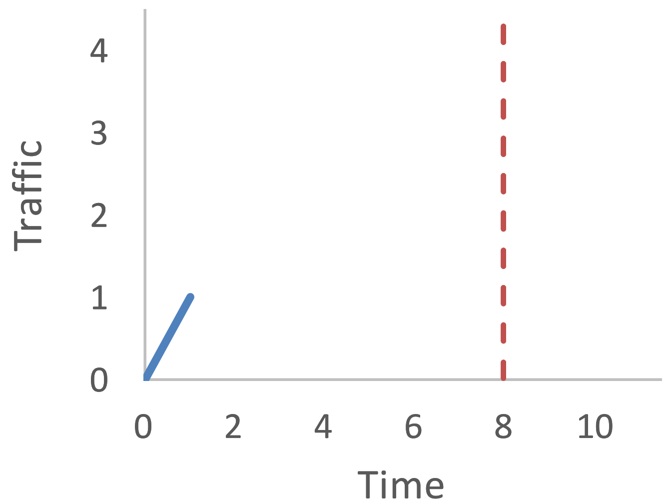
Center

Key	Value

Pure Streaming

Time: 1

fast



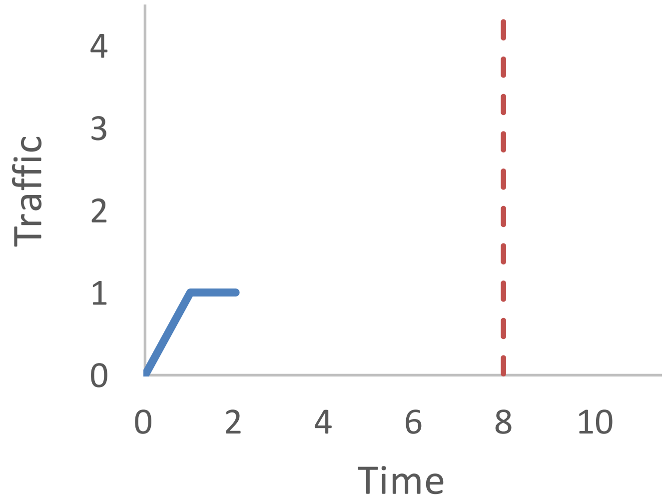
Edge

Key	Value

Center

Key	Value
fast	1

Pure Streaming



Edge

Key	Value

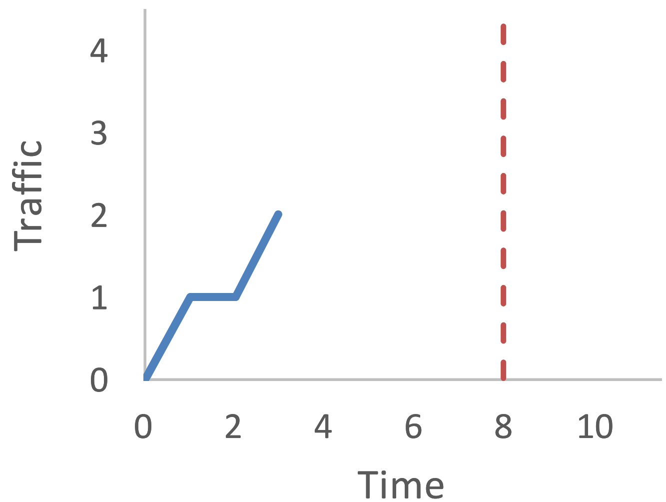
Center

Key	Value
fast	1

Pure Streaming

Time: 3

data



Edge

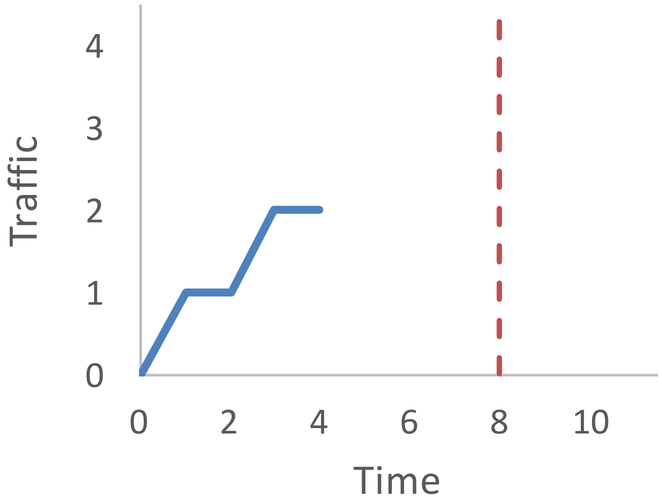
Key	Value

Center

Key	Value
fast	1
data	1

Pure Streaming

Time: 4



Edge

Key	Value

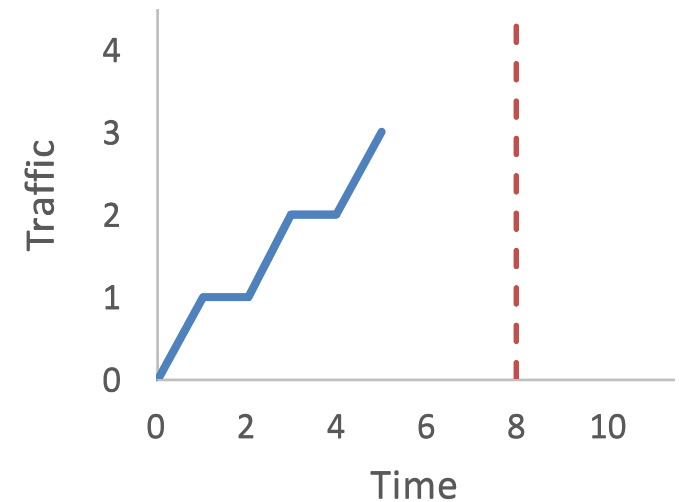
Center

Key	Value
fast	1
data	1

Pure Streaming

is

Time: 5



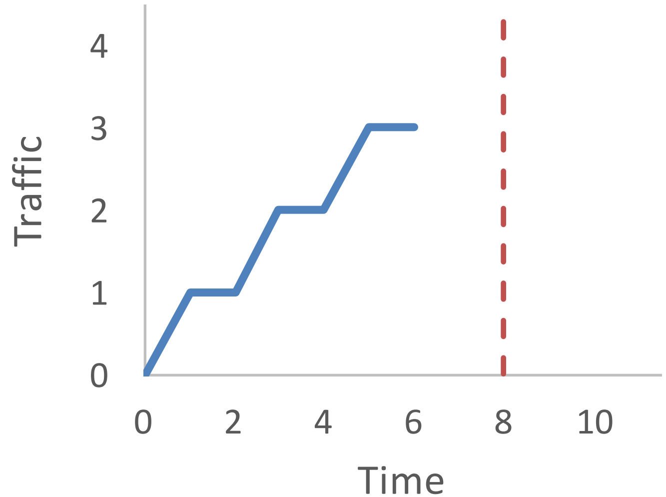
Edge

Key	Value

Center

Key	Value
fast	1
data	1
is	1

Pure Streaming



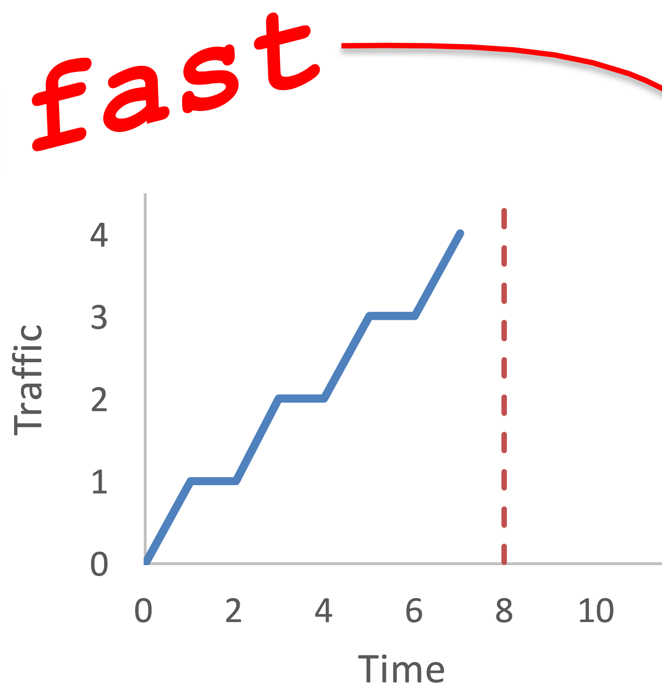
Edge

Key	Value

Center

Key	Value
fast	1
data	1
is	1

Pure Streaming



Edge

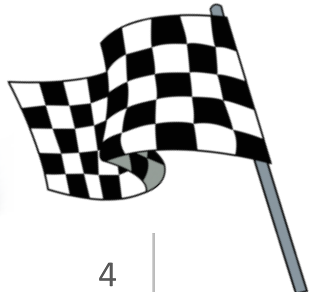
Key	Value

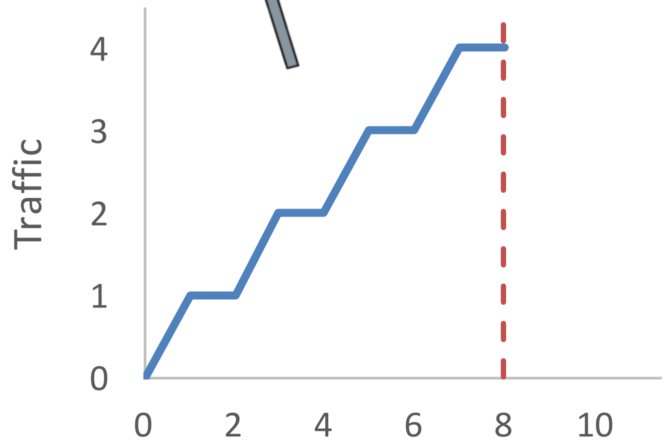
Center

Key	Value
fast	2
data	1
is	1

Pure Streaming

Time: 8





Edge

Key	Value

Staleness: 0

Traffic: 4

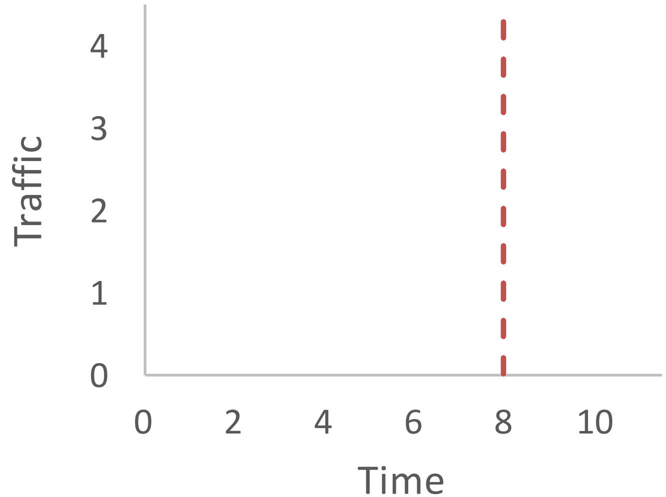
Center

Key	Value
fast	2
data	1
is	1

Pure Batching

Time: 0

fast



Edge

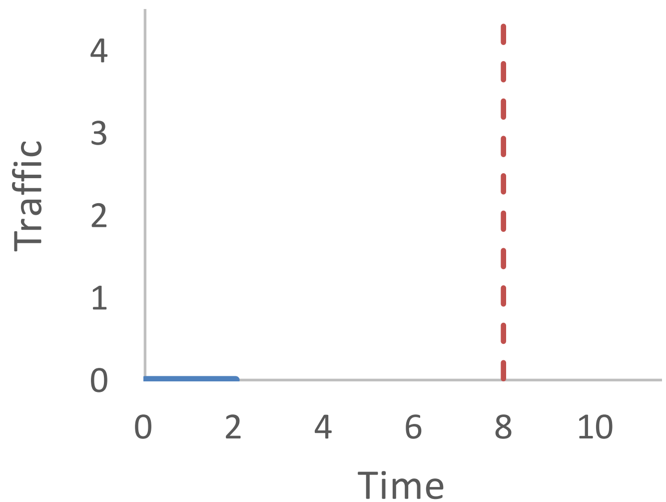
Key	Value
fast	1

Center

Key	Value

Pure Batching

Time: 2



Edge

Key	Value
fast	1
data	1

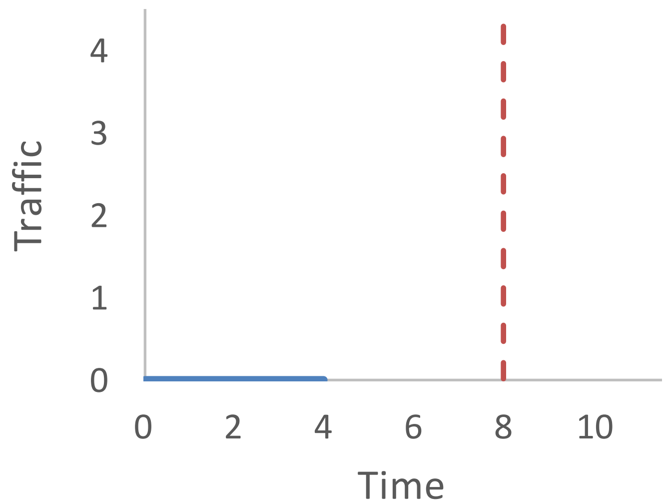
Center

Key	Value

Pure Batching

Time: 4

is



Edge

Key	Value
fast	1
data	1
is	1

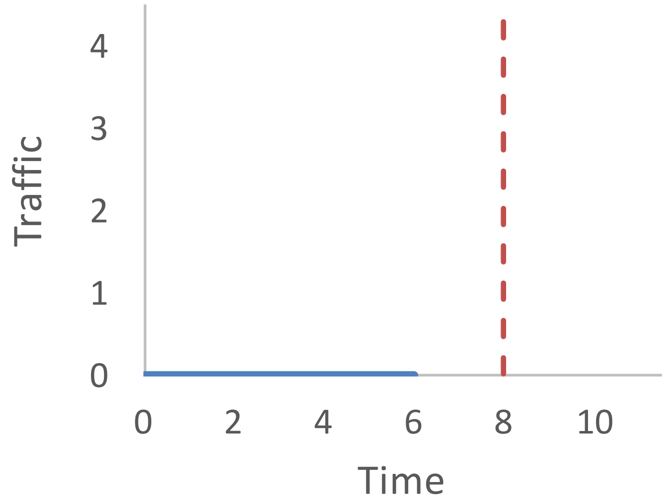
Center

Key	Value

Pure Batching

Time: 6

fast



Edge

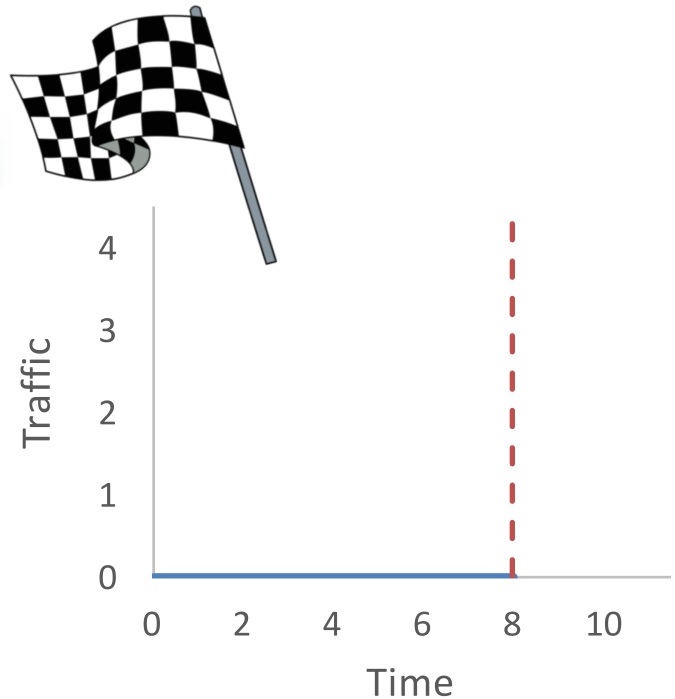
Key	Value
fast	2
data	1
is	1

Center

Key	Value

Pure Batching

Time: 8



Edge

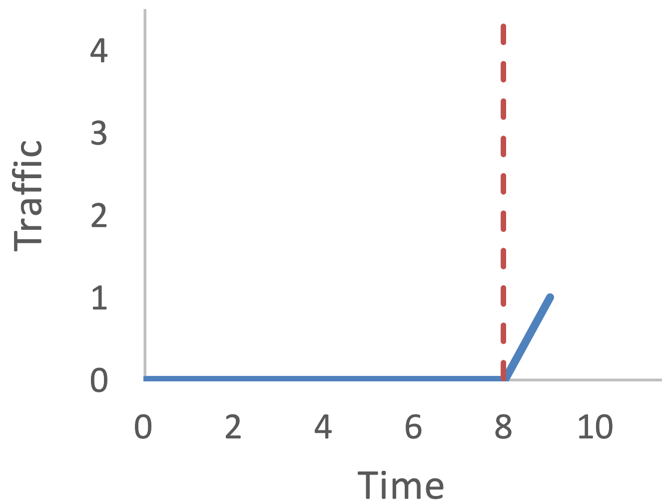
Key	Value
fast	2
data	1
is	1

Center

Key	Value

Pure Batching

Time: 9



Edge

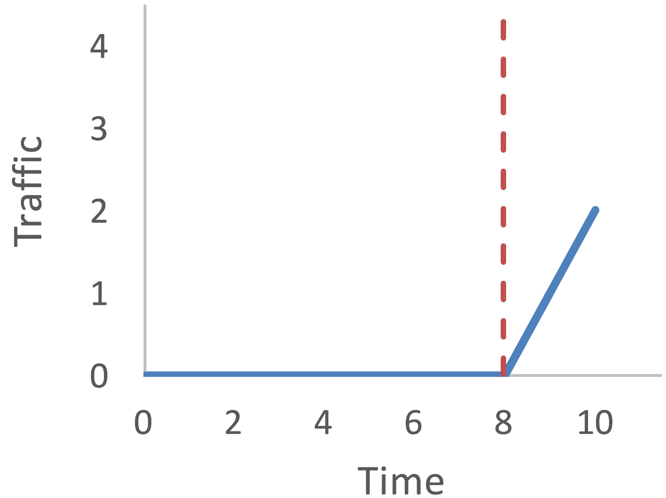
Key	Value
data	1
is	1

Center

Key	Value
fast	2

Pure Batching

Time: 10



Edge

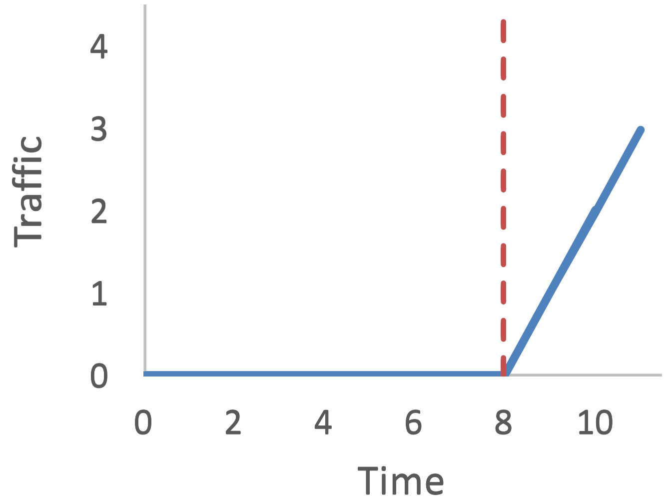
Key	Value
is	1

Center

Key	Value
fast	2
data	1

Pure Batching

Time: 11



Edge

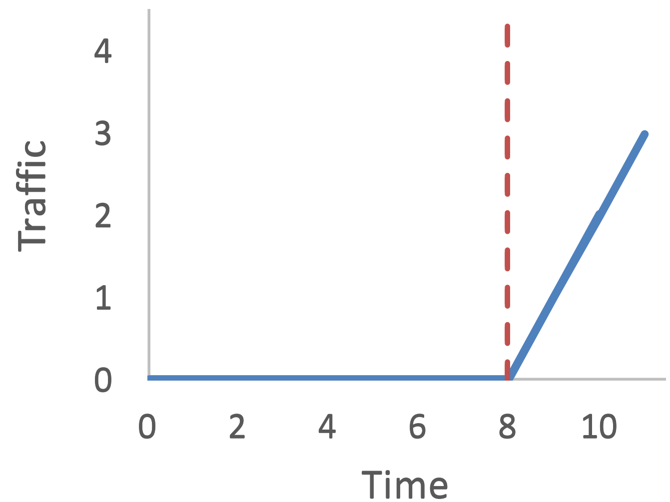
Key	Value

Center

Key	Value
fast	2
data	1
is	1

Pure Batching

Time: 11



Edge

Key	Valu

Staleness: 3

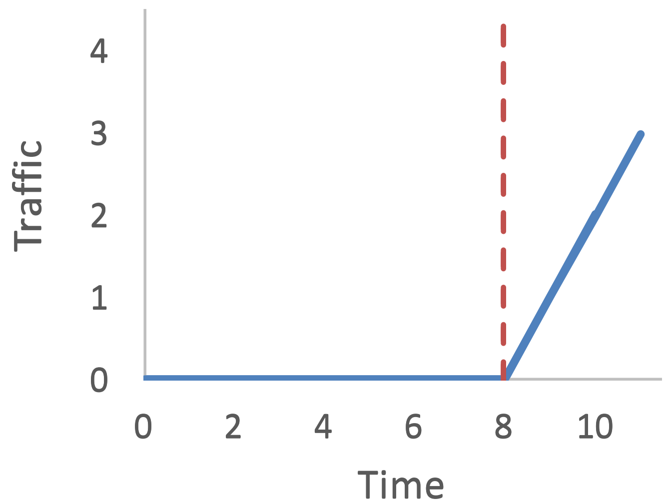
Traffic: 3

Center

Key	Value
fast	2
data	1
is	1

Pure Batching

Time: 11



Naïve Algorithms: Staleness vs. Traffic

Roadmap

- Naïve algorithms
- ***Optimal offline algorithms***
- Practical online algorithms
- Experimental evaluation

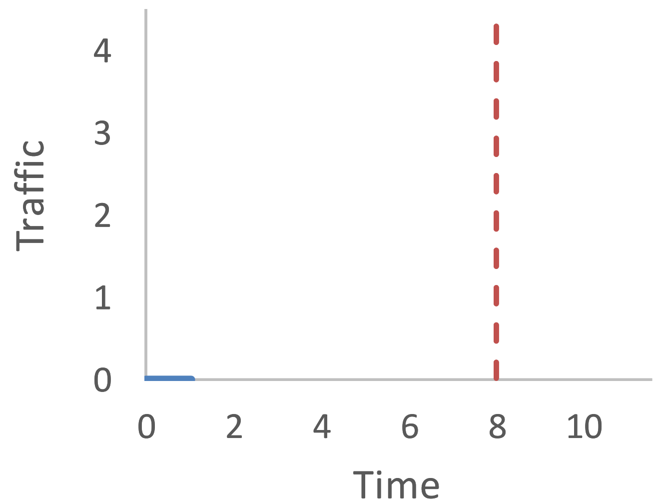
Eager Offline Optimal

- ***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

Time: 0

fast



Edge

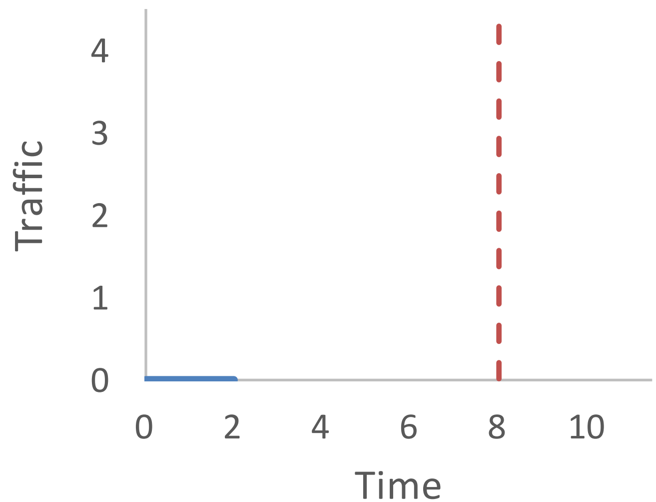
Key	Value
fast	1

Center

Key	Value

Eager Offline Optimal

Time: 2



Edge

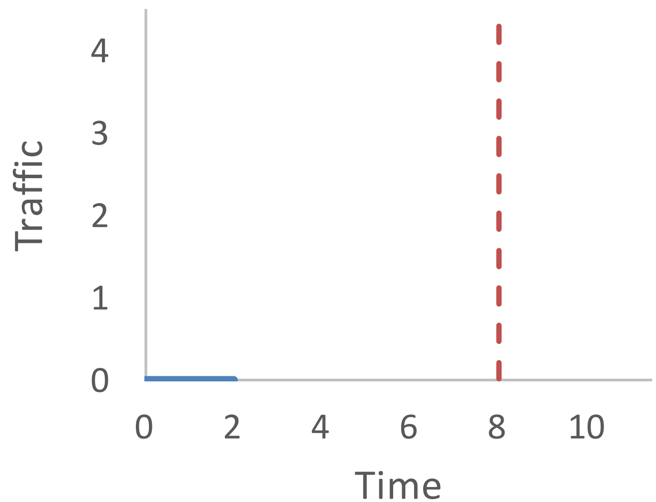
Key	Value
fast	1

Center

Key	Value

Eager Offline Optimal

Time: 2



Edge

Key	Value
fast	1

Flush *immediately* upon last arrival

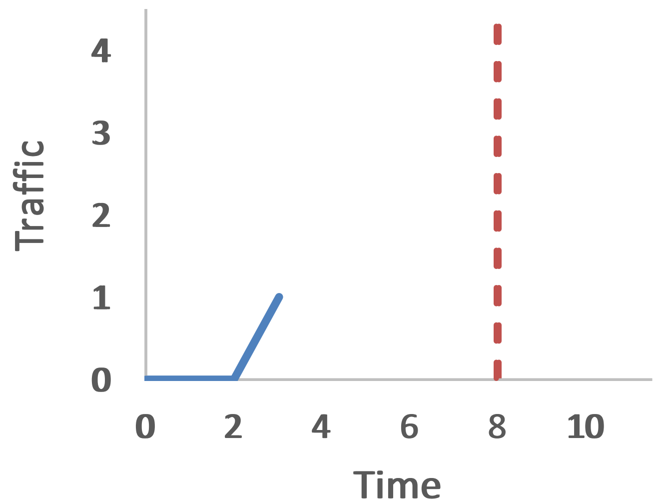
Center

Key	Value

Eager Offline Optimal

Time: 3

data



Edge

Key	Value
fast	1

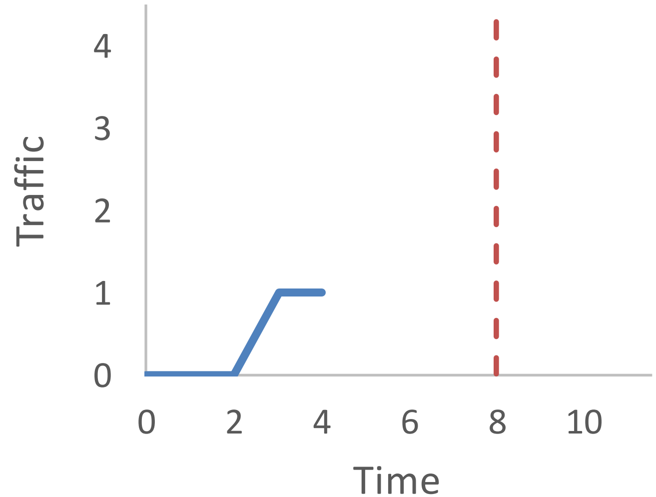
Center

Key	Value
data	1

Eager Offline Optimal

Time: 4

is



Edge

Key	Value
fast	1

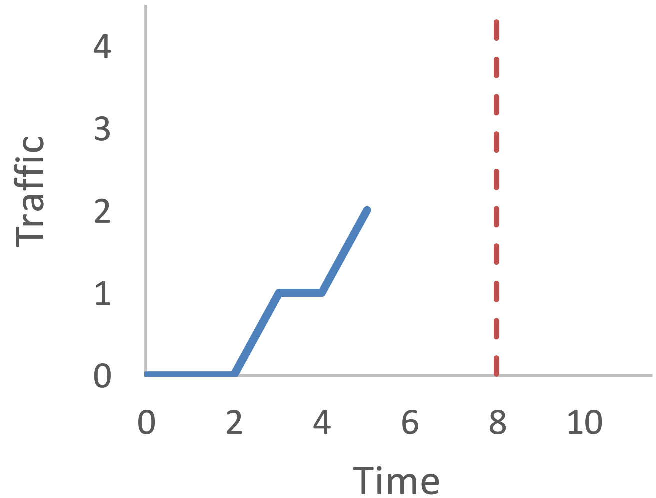
Center

Key	Value
data	1

Eager Offline Optimal

Time: 5

is



Edge

Key	Value
fast	1

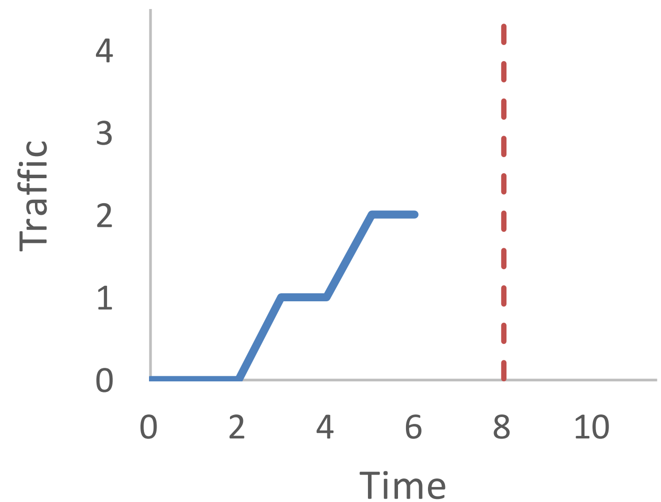
Center

Key	Value
data	1
is	1

Eager Offline Optimal

Time: 6

fast



Edge

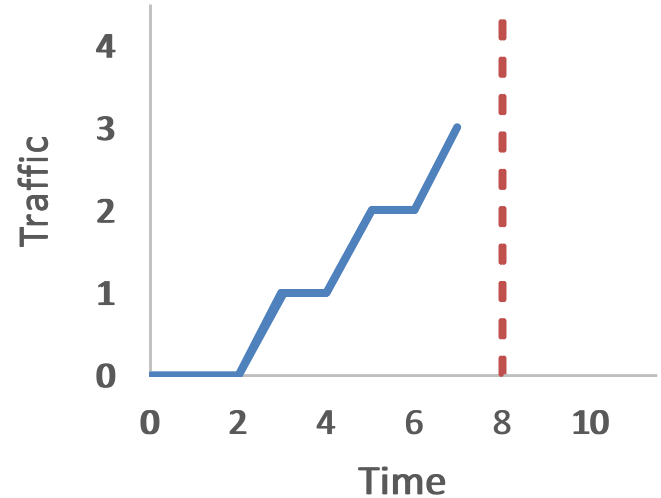
Key	Value
fast	2

Center

Key	Value
data	1
is	1

Eager Offline Optimal

Time: 7



Edge

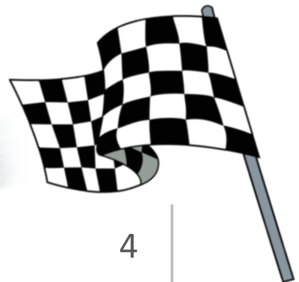
Key	Value

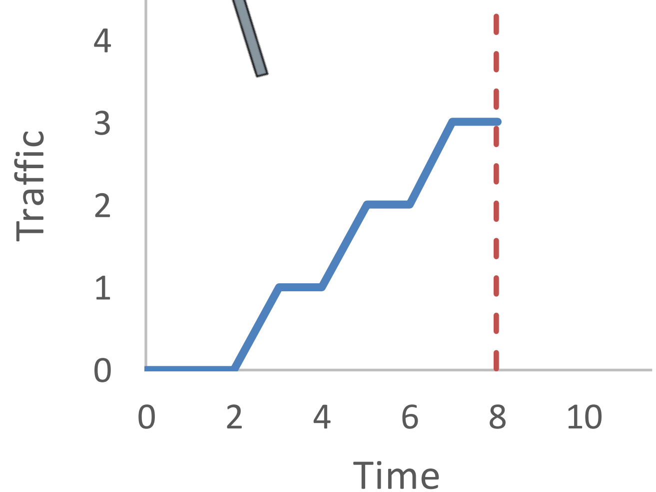
Center

Key	Value
fast	2
data	1
is	1

Eager Offline Optimal

Time: 8





Edge

Key	Value

Staleness: 0

Traffic: 3

Center

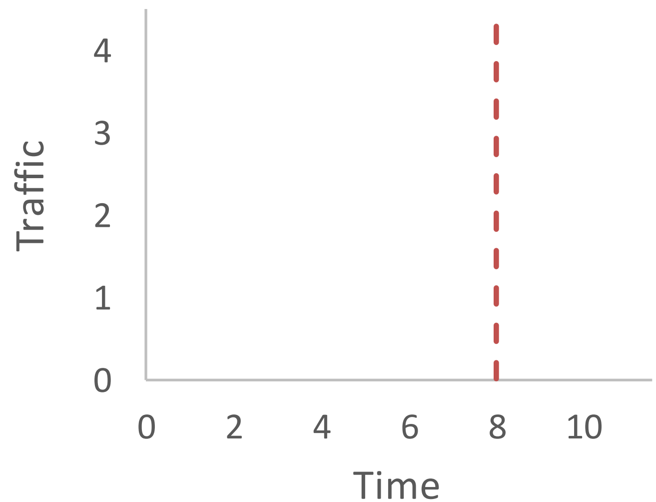
Key	Value
fast	2
data	1
is	1

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

Key	Value
fast	1

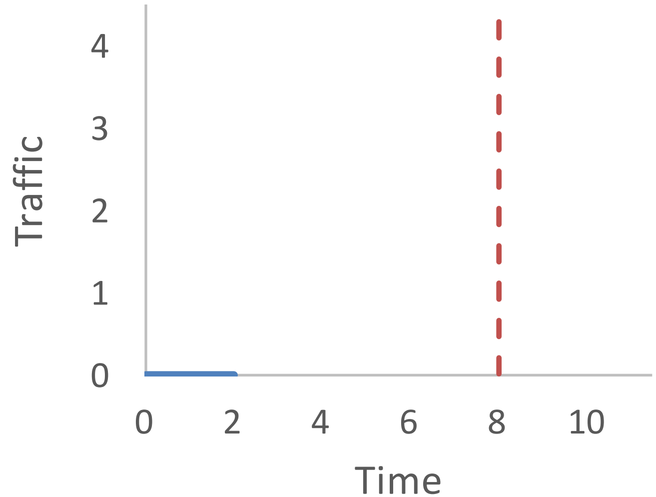
Center

Key	Value

Lazy Offline Optimal

Time: 2

data



Edge

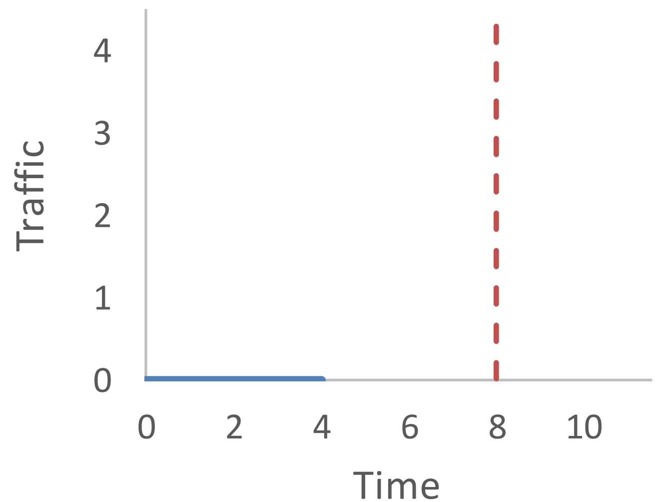
Key	Value
fast	1
data	1

Center

Key	Value

Lazy Offline Optimal

Time: 4



Edge

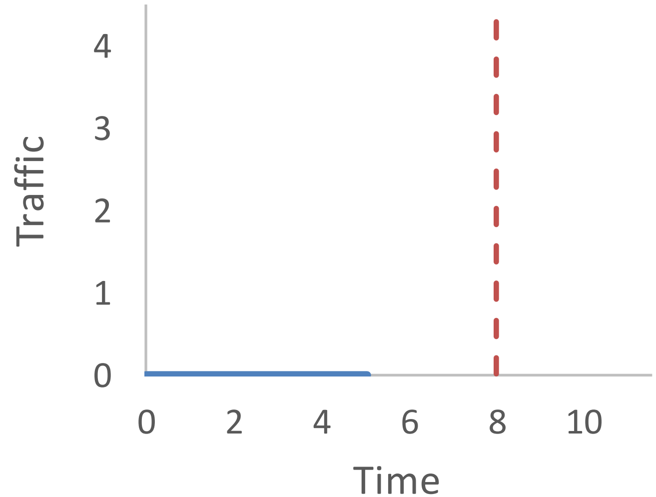
Key	Value
fast	1
data	1
is	1

Center

Key	Value

Lazy Offline Optimal

Time: 5



Edge

Key	Value
fast	1
data	1
is	1

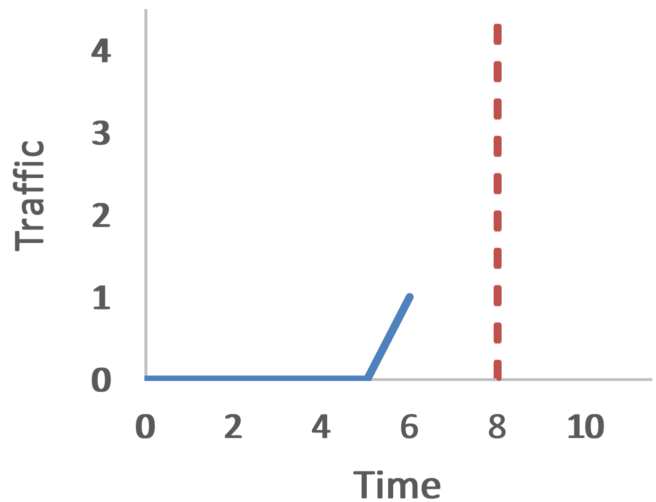
Center

Key	Value

Flush
as late as possible
after last arrival

Lazy Offline Optimal

Time: 6



Edge

Key	Value
fast	1
is	1

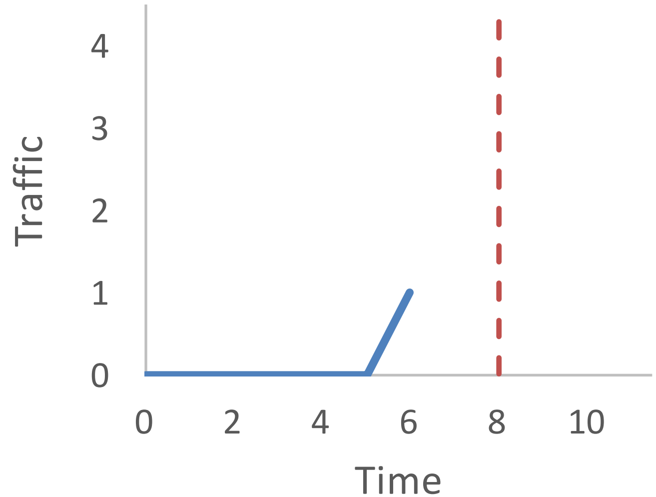
Center

Key	Value
data	1

Lazy Offline Optimal

Time: 6

fast



Edge

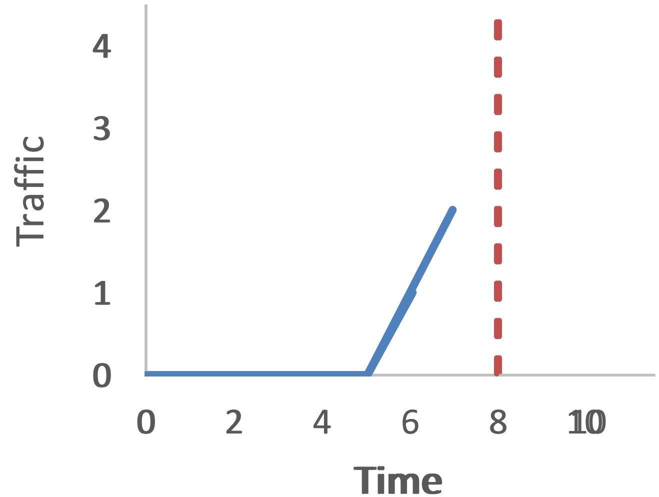
Key	Value
fast	2
is	1

Center

Key	Value
data	1

Lazy Offline Optimal

Time: 7



Edge

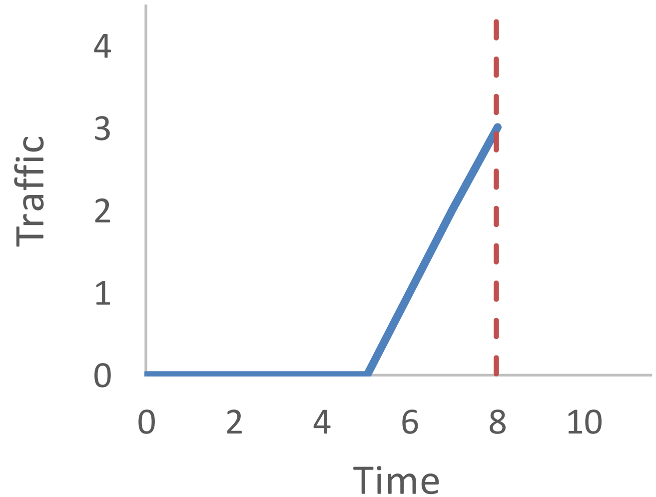
Key	Value
fast	2

Center

Key	Value
data	1
is	1

Lazy Offline Optimal

Time: 8



Edge

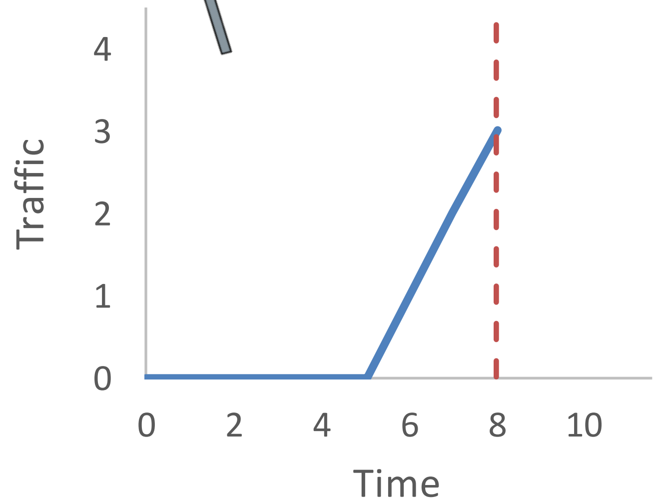
Key	Value

Center

Key	Value
fast	2
data	1
is	1

Lazy Offline Optimal

Time: 8



Edge

Key	Value

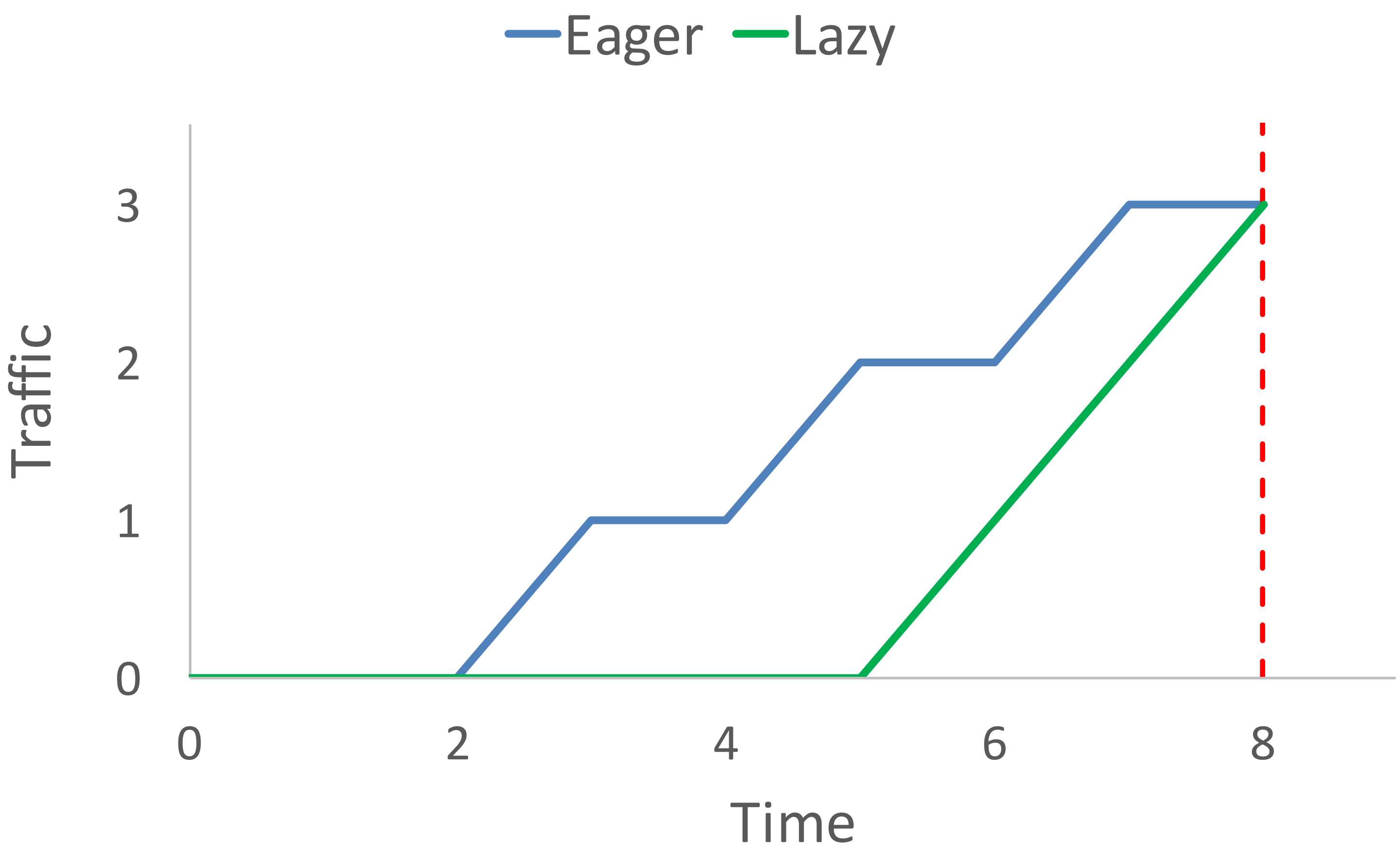
Staleness: 0

Traffic: 3

Center

Key	Value
fast	2
data	1
is	1

Family of Optimal Algorithms



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

Edge

Key	Value
fast	2
data	1
is	1

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***

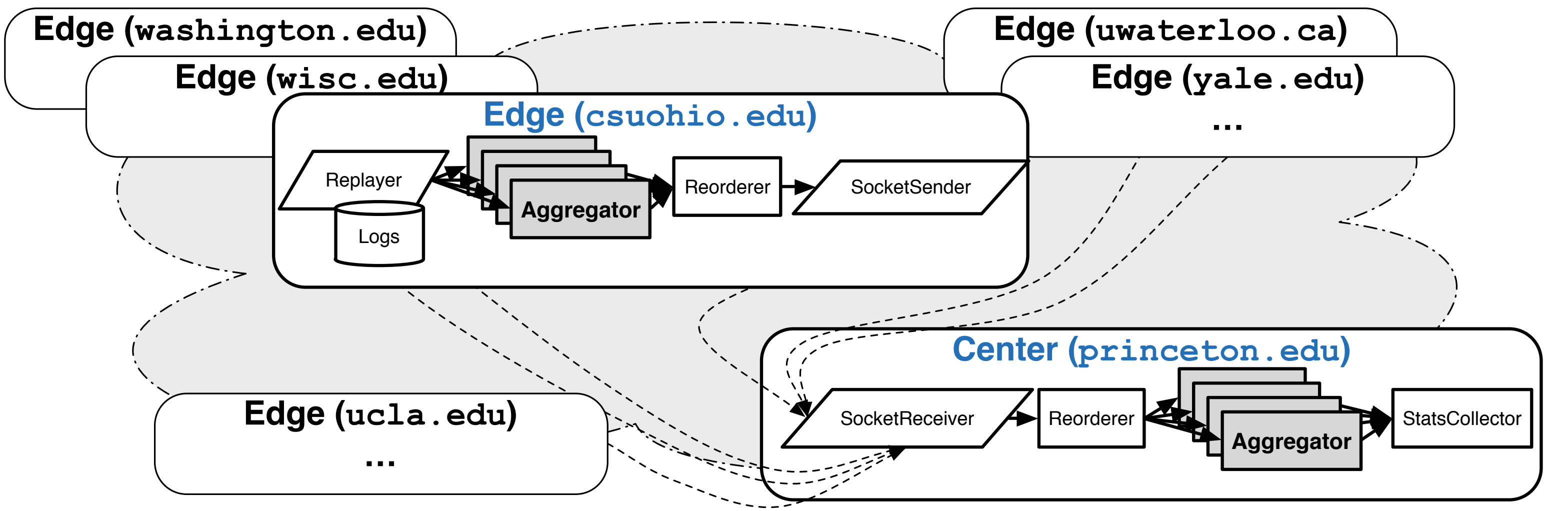
Experiments: Data Set

Month-long Akamai download analytics beacon log

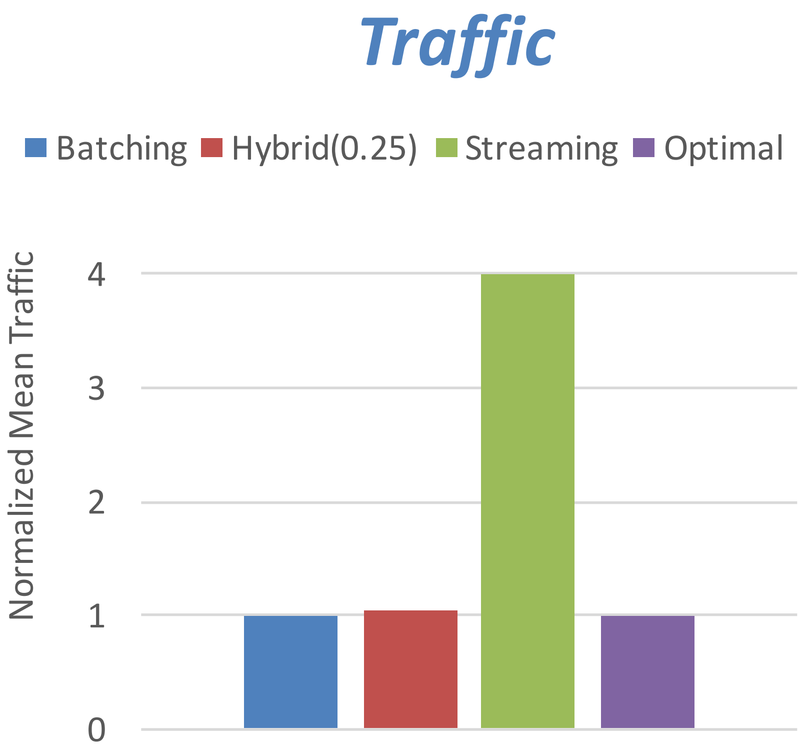
Name	Key	Value	Query Size
small	(cpid, bw)	bytes downloaded (**sum**)	$O(10^2)$ groups
medium	(cpid, bw, country_code)	bytes downloaded (***moments***)	$O(10^4)$ groups
large	(cpid, bw, url)	client ip (***HyperLogLog***)	$O(10^6)$ groups

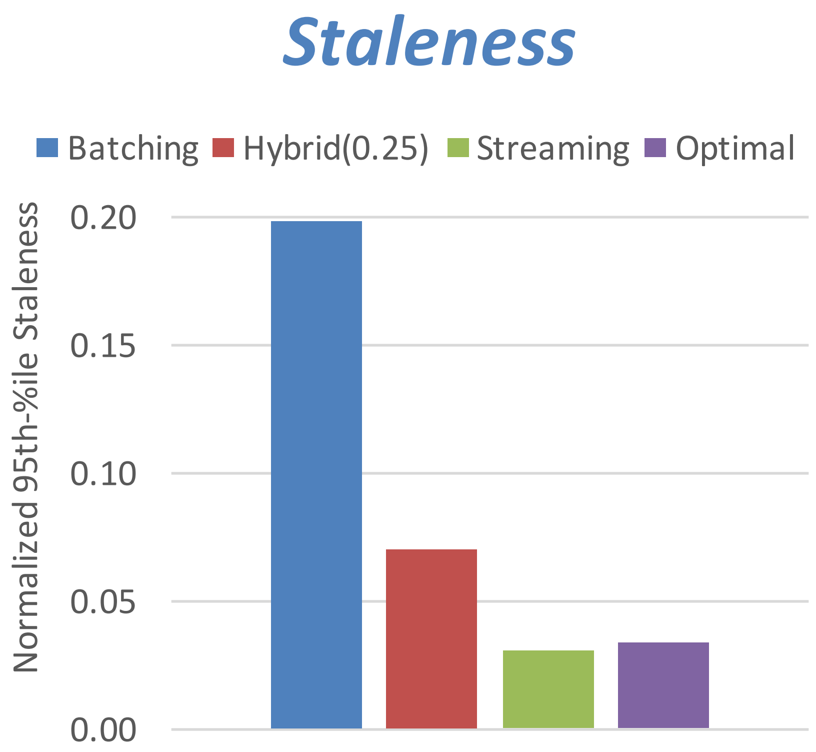
Experiments: Implementation

- Implemented algorithms in Apache Storm
- One Storm cluster at center, each edge
- Seven total PlanetLab sites

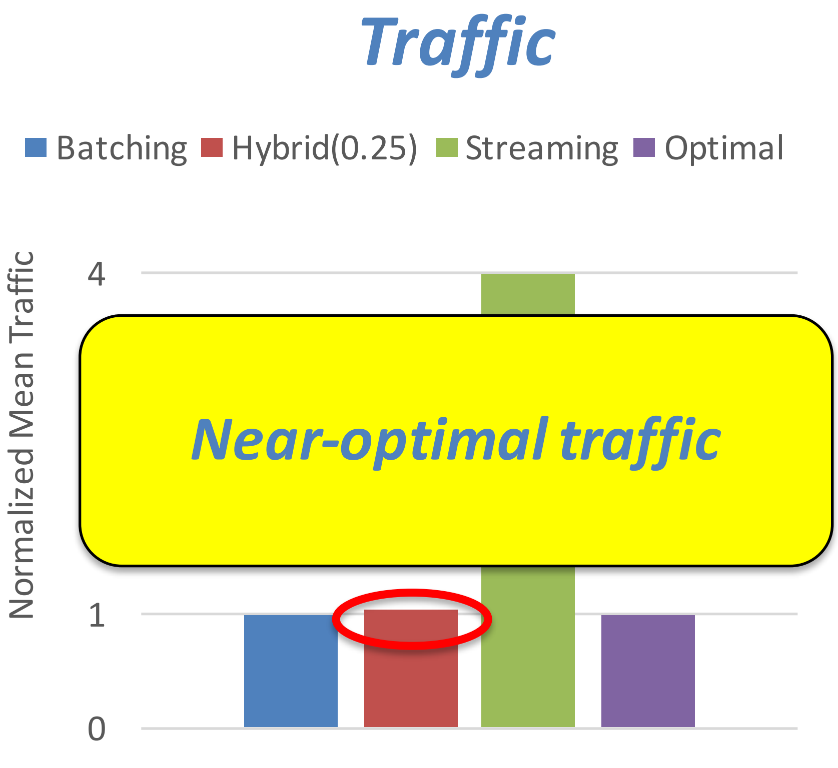


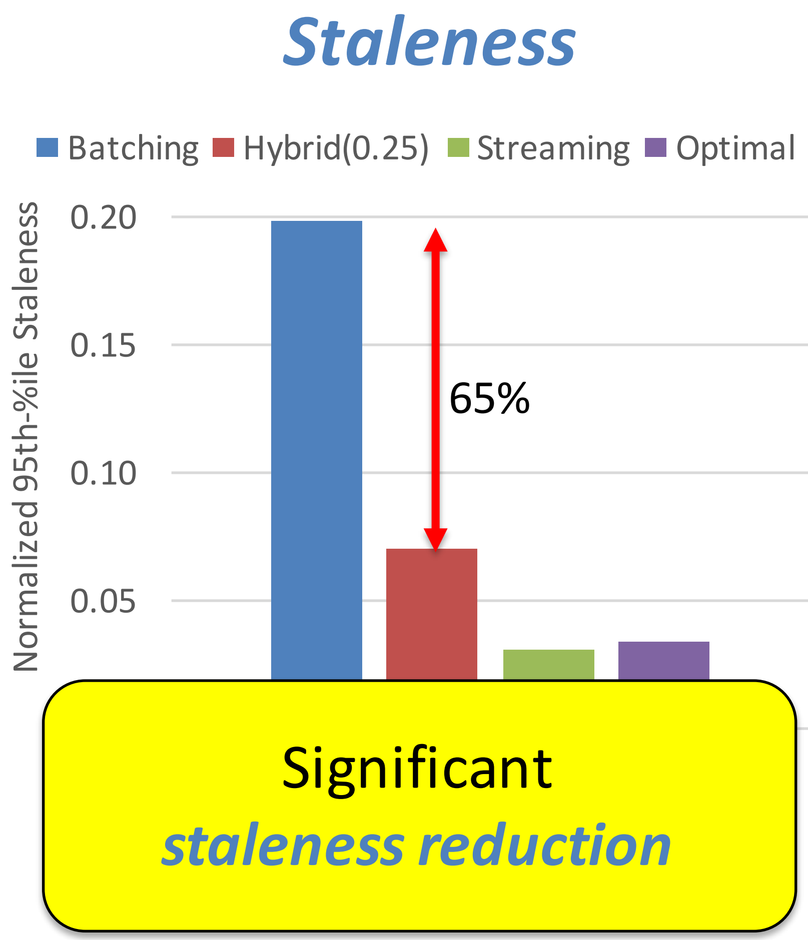
Experimental Evaluation: Single-Edge





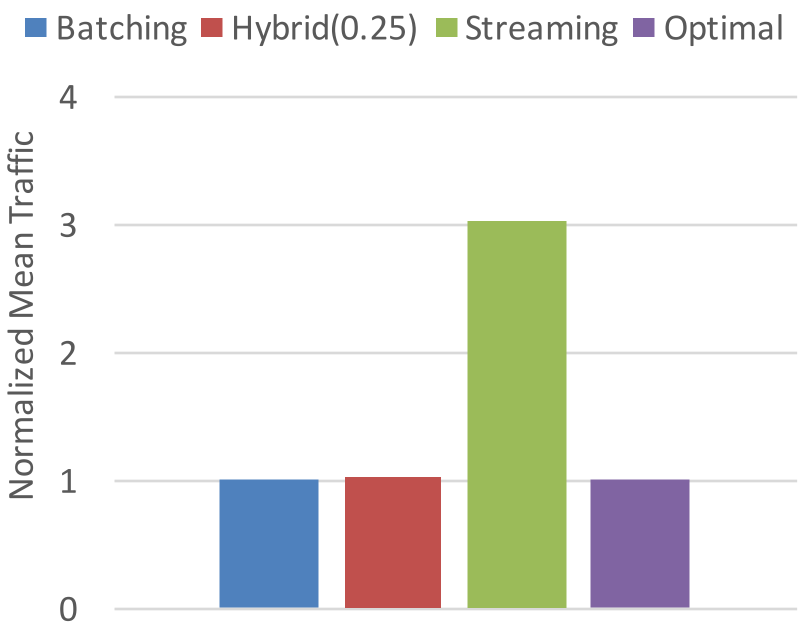
Experimental Evaluation: Single-Edge



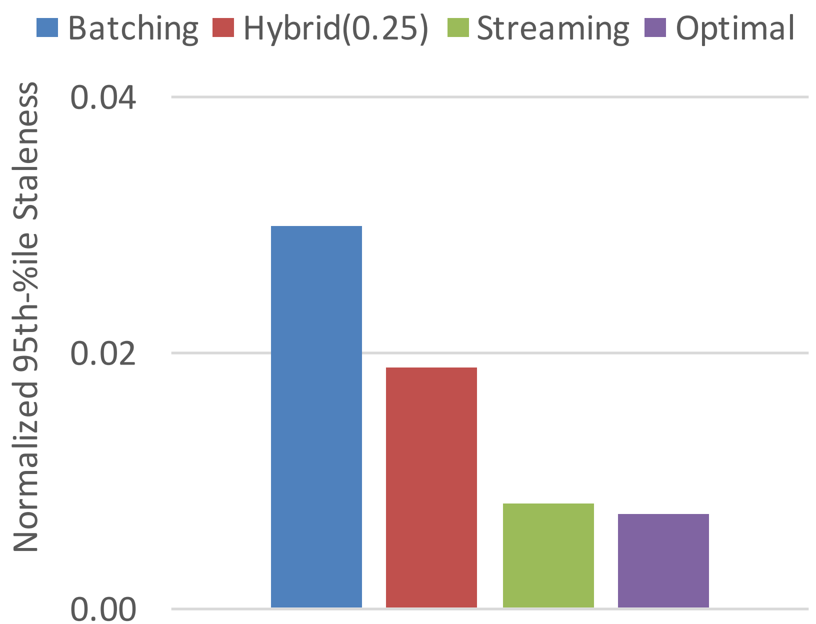


Experimental Evaluation: Multi-Edge (3 edges)

Traffic

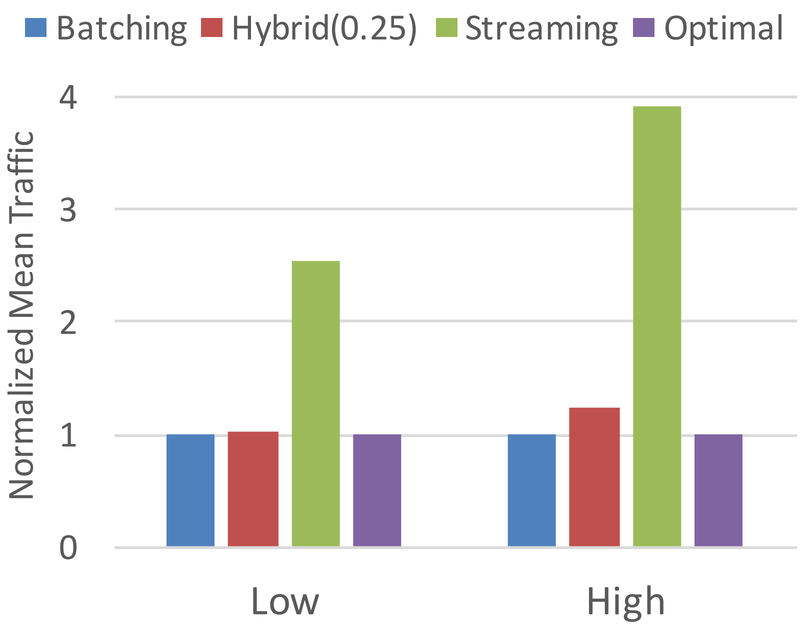


Staleness

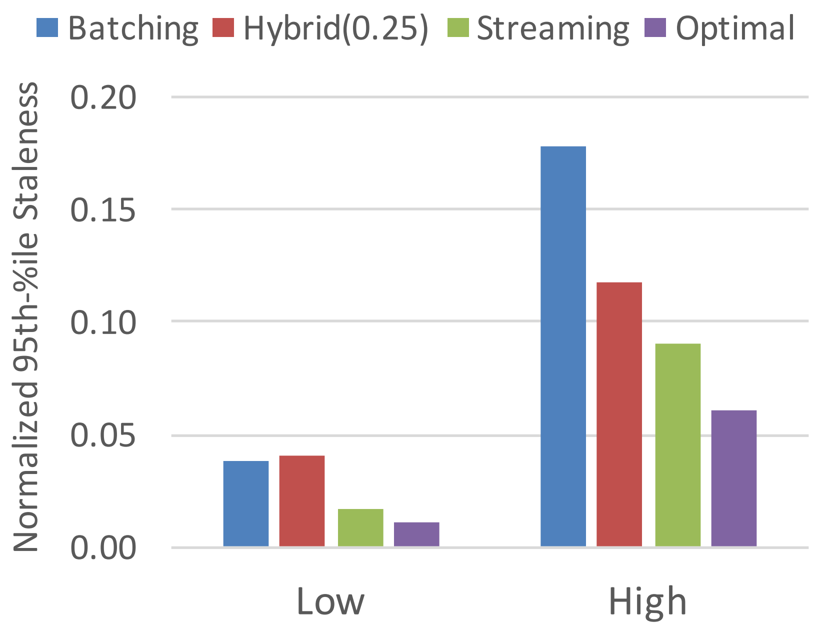


Experimental Evaluation: Multi-Edge (6 edges)

Traffic



Staleness



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