Data Streaming

UNIVERSITY OF
WATERLOO

uwaterloo.ca

Lukasz Golab

lgolab@uwaterloo.ca

engineering.uwaterooo.ca/~lgolab

Outline

- Context
- Relatively slow streams
- Relatively fast streams

Big Data

- Every 2 days the world creates as much information as it did up to 2003
  - (Eric Schmidt, Google CEO)

UNIVERSITY OF WATERLOO

Why Now?

- 1. Easier/cheaper to generate data
  - Sensors, smart devices
  - Internet of Things
  - Social software
  - Web data

Source: Abadi et al., The Beckman Report on Database Research, SIGMOD Record 43(3)

Why Now?

- 2. Easier/cheaper to process data
  - Cheap hard drives and SSDs
  - Cheap commodity hardware

Source: Abadi et al., The Beckman Report on Database Research, SIGMOD Record 43(3)

Why Now?

- 3. Data Democratization
  - Anyone can get involved in data, not just database people
  - Open-source software
  - Cloud computing
  - Open data initiatives

Source: Abadi et al., The Beckman Report on Database Research, SIGMOD Record 43(3)

3 Vs of Big Data

- Volume
- Velocity -> data streams
- Variety

Data Streams

- Many interesting data arrive over time
- Think of the schema as
  - (key, timestamp, other attributes)
- Or maybe new keys trickle in
  - data extraction

Data Processing

- Typical big data workflow
  - Collect all data, prepare, load, process, repeat if necessary
- Typical streaming workflow
  - Process as data are coming in
  - Reduce the time “from ingest to insight”

Slow vs. Fast Streams

- Slow
  - ..enough that you can use a DBMS
  - maybe one file every 5 minutes (batch)
  - don’t need to do real-time processing
- Fast
  - Thousands/Millions of records per second

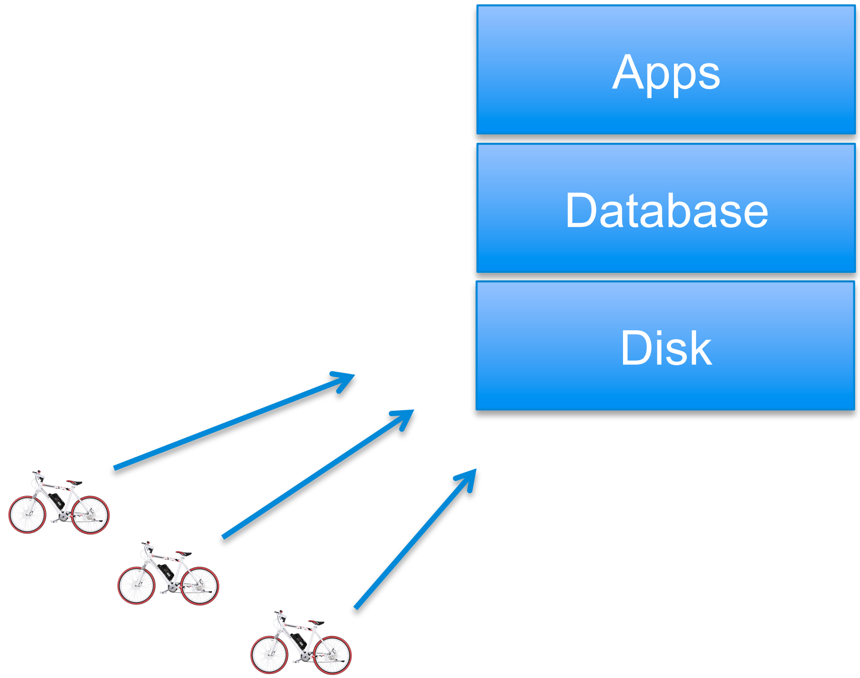
Outline

- Relatively slow streams

Application: WeBike

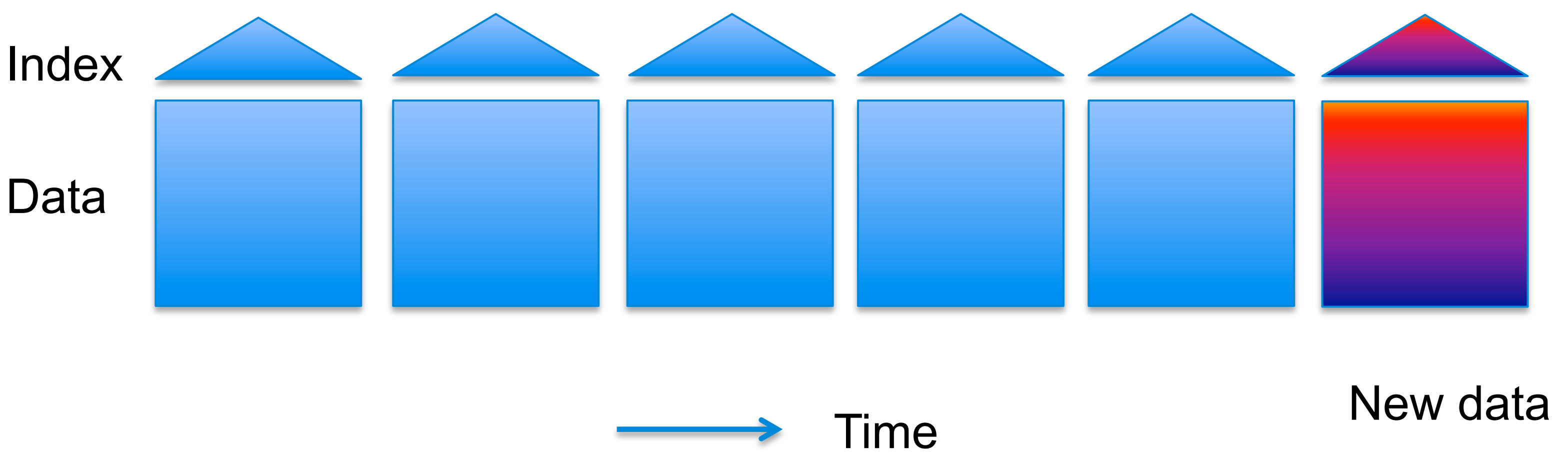
UNIVERSITY OF
WATERLOO

Data Flow



Data Layout

- Partition by time



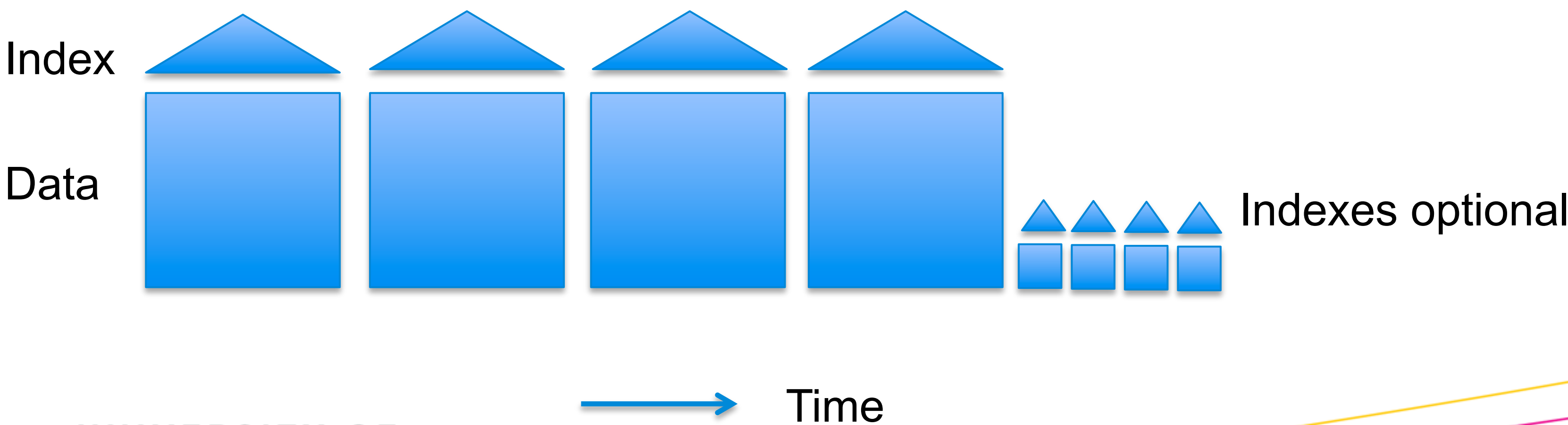
UNIVERSITY OF WATERLOO

Data Layout

- New data loaded to new partition; existing partitions are not touched
  - Except out-of-order data
- Logically one table, physically many tables
  - Index on the table directory

Data Layout Optimization

- How big should each partition be?
  - Small partitions: easy to add new data, but queries spanning a long history will be slow
- Solution: merge partitions as they age



Out-of-order Data

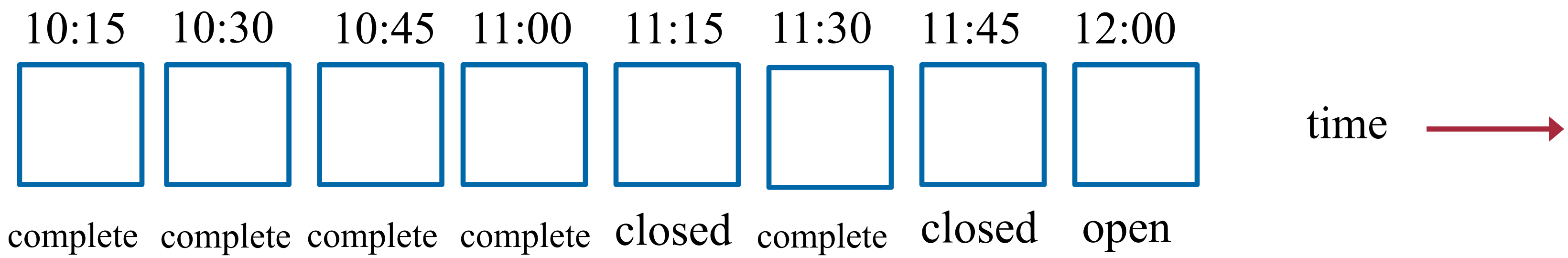
- Different data sources have different time lags and different likelihoods of late data
- How do I know when my data are stable enough to query?

Out-of-order Data

- Assign labels to each partition
  - Open = more data may be added
  - Closed = no more data expected
  - Complete = Closed and all expected data have arrived (i.e., no data permanently lost)
  - …

Example

- Closed up to 11:45
- Note: completeness not always contiguous



UNIVERSITY OF WATERLOO

Partition Labels

- Of course, this works only if we can verify closed-ness and completeness
  - E.g., each of our 30 e-bikes produces a file every minute and keeps it for a day

Queries over Slow Streams

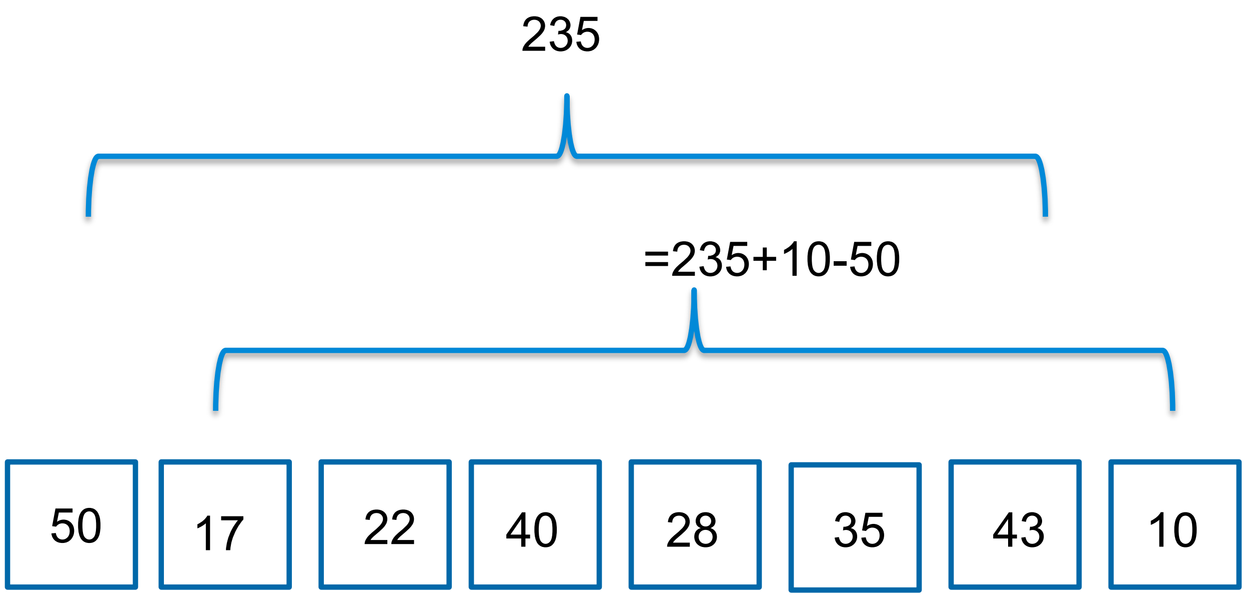
- Traditional database: query workload usually not known ahead of time
- Streaming: users ask the same queries over time

UNIVERSITY OF WATERLOO

Incremental Query Processing

- E.g., what was the total riding distance of each person within the last 7 days?
- Naïve approach: every day, recompute the query
- Faster approach: every day, incrementally update the query
  - But have to store extra information

Incremental Query Processing



UNIVERSITY OF WATERLOO

Also…

- If we know (some of) the queries, we can try to do shared processing
  – Or reorder them for better cache performance

Recap

- Handling relatively slow streams/ real-time response not needed
  - Can use a regular DBMS
  - Consider partitioning by time to speed up insertions
  - Consider keeping extra information to enable incremental query processing

For More Information

- Golab, Johnson, Seidel, Shkapenyuk, Stream Warehousing with DataDepot, SIGMOD 2009
- Golab, Johnson, Consistency in a Stream Warehouse, CIDR 2011
- Golab, Johnson, Shkapenyuk, Scalable Scheduling of Updates in Streaming Data Warehouses, TKDE 2012
- Baer, Golab, Ruehrup, Schiavone, Casas, Cache-Oblivious Scheduling of Shared Workloads, ICDE 2015

UNIVERSITY OF WATERLOO

Outline

- Relatively fast streams
  - … too fast to use a traditional DBMS
  - So we need to design a new system
  - Call it DSMS

Simple Example

- Network firewall
- Streaming input -> drop packets that fail some criteria -> streaming output
- Simple SELECT FROM WHERE streaming query

Streaming Queries

- At any point in time, returns the same answer as an equivalent SQL query over a relation consisting of the stream seen so far

How Does it Work

- No time to “load” the data
- Quickly look up the attribute of interest (e.g., port number or source IP address) in each packet
- Drop or pass on to the output stream
- Move on to the next packet

Simple DSMS

- Simple WHERE predicates
- Pre-defined queries
- Pre-defined stream schema
  - Need to tell the system where to find each attribute
  - But not all fields inside an IP packet are fixed-offset
  - And may want to filter on payload contents

More Complex Example

(timestamp,
src/dest,
bytes)

SELECT timestamp/60, src, dest, sum(bytes)
FROM IP_STREAM
GROUP BY timestamp/60, src, dest

Per-minute traffic
for each src/dest pair

How Does it Work

- Maintain a hash table on src/dest storing sum(bytes)
- At the end of each minute, output the sums for each src/dest pair and clear the hash table
  - GROUP BY condition must include the timestamp, which splits the stream into windows

What if the stream is really, really fast?

- Resort to approximate answers
  - Sampling
  - One-pass algorithms

Recap

- Data Stream Management Systems (DSMS)
  - SQL-like language (but not full SQL)
  - Stream-in -> Stream-out
  - Predefined queries
- Approximate one-pass stream algorithms for dealing with very high velocities

For More Information

- Cranor, Johnson, Spatscheck, Shkapenyuk, The Gigascope Stream Database, IEEE DE Bul, 26(3), 2003
- Golab, Johnson, Spatscheck, Prefilter: Predicate Pushdown at Streaming Speeds, SSPS 2008
- Golab, Ozsu, Data Stream Management, Morgan & Claypool, 2010

UNIVERSITY OF
WATERLOO

Summary

- Data Stream Processing
  - Batch-oriented vs real-time
  - Adapting existing data management technologies (slow)
  - Developing new systems (fast)

Open Problems

- Distributed/cloud stream processing
- Can help deal with very fast streams
  - Many DSMSs can process a stream in parallel
- Also helpful for slower streams
  - Already some work on incremental computation in Hadoop/MapReduce