

Time-Series Data Architectures and Use Cases

BY TED MALASKA

High Level Overview

- Day 1
 - What is time series
 - Why is time series so important
 - Ways we can look at time series data
 - Examples
 - Lead & Lag
 - Tumbling window
 - Sliding window
 - Session window
 - Respect to inflection point
 - Reflection on samples
 - Let's talk about Bucketing and sorting
 - CDC Change Data Capture
 - TTL

High Level Overview

- Day 2
 - Getting your first win
 - Working with stack holders
 - Setting up incentives
 - Deeper drive into machine learning
 - Machine learning inputs
 - Feature creation
 - Exercise weather golfing
 - Exercise Titanic
 - Exercise Movies
 - Sessionization
 - Network Relations
 - Talking to executives
 - Quantifying value

What is Time Series?

What is Time Series

- Events that happen with in time
- Events can be related to entities
- Events can be related to many entities

Web Session

- Initial Land
- Log on
- Viewing Ads and Promotions
- Clicks
- Exist or inactivity

Slot Machines Sessions

- Set down
- Interval between games
- Eye contact
- Reactiveness to the prices and the animation
- Get up from set

Devices Patterns

- Network communication
 - Comparative patterns
 - Like entities
 - Historic readings
 - Thresholds
 - Lateral movements
 - Play back

Questions

- ???

Why is time
Series so
Important

Time Series Use Cases

- Entity
 - People
 - Group or Team
 - Things
- Time
 - Gives context
 - Defines behavior/patterns
 - Relationships

Context

- If you don't have time what do you have
 - Snapshots
 - Aggregations
- Time tells the story
 - Beginning middle and end
 - Continues
 - Patterns with in a story
 - Patterns across stories
 - Outliers

Behaviors/Patterns

- Weekends from weekdays
- Night vs day
- Rhythm of life
 - Cycle of Economic Utilities
- Engagement Levels
- NGrams before events

Relationships

- Fraud
 - Abrupt behavior changes
- Relationships
 - Events with respect to over events from different entities
- Correlation
 - With promotions
 - With competition actions

Predictions

- Predicting load
- Predicting events
 - Churn
 - Phone company story
 - Death
 - Drug story
 - System failure
 - Drive story
 - ER Story

Clustering

- Machine Learning approach to clumping Entities together
- N dimensional relations
 - KMean
 - GMM
- Converting events and time to points in N dimensional space
 - Peaks
 - Valleys
 - Distances between peaks/valleys
 - Slopes
 - Activity density

Changing Centers

- How to change a nodes center
- Impact behavior

Questions

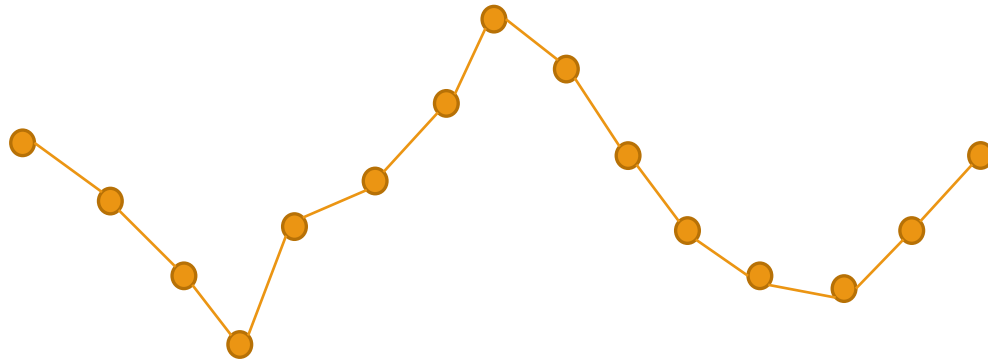
- ???

Ways we can
look at time
series data

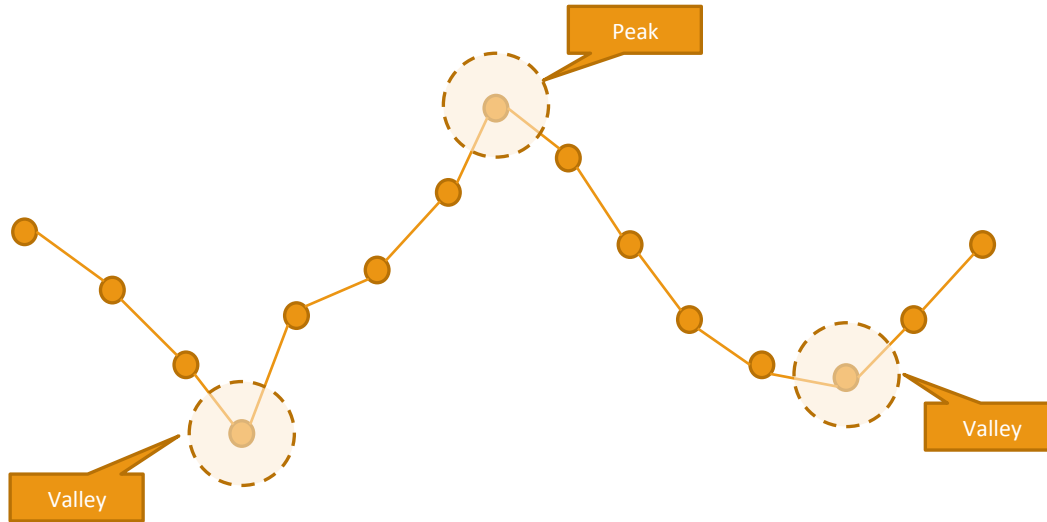
Ways to look at time Series Data

- Lead and Lag
- Tumbling windows
- Sliding windows
- Session windows
- Respect to Infection Point

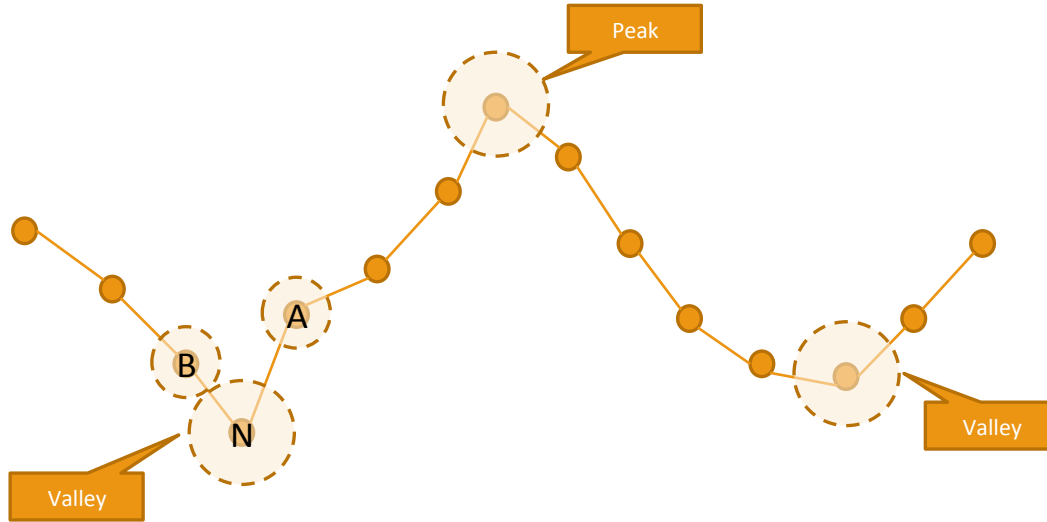
Lead and Lag



Lead and Lag

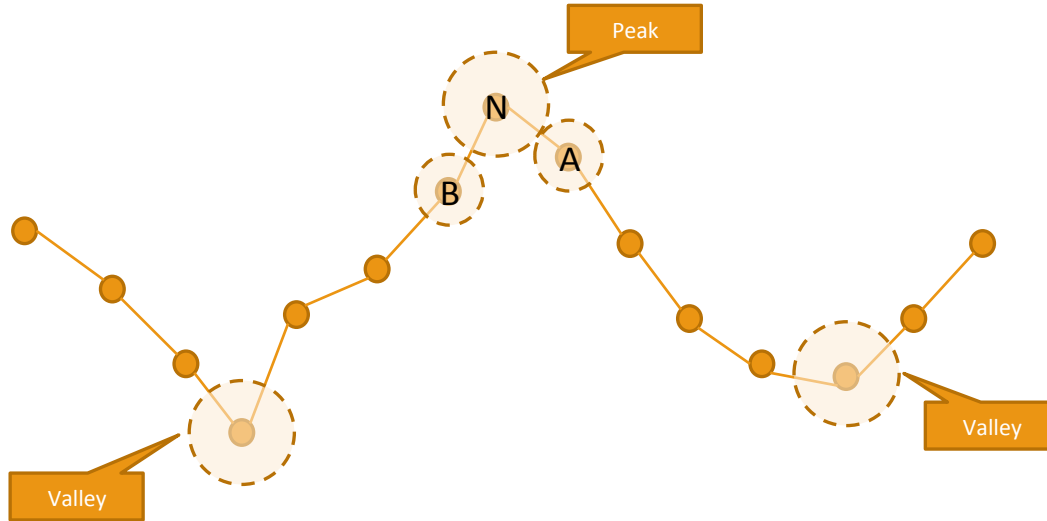


Lead and Lag



$N < B$ and $N < A \Rightarrow \text{Valley}$

Lead and Lag



$N > B \text{ and } N > A \Rightarrow \text{Peak}$

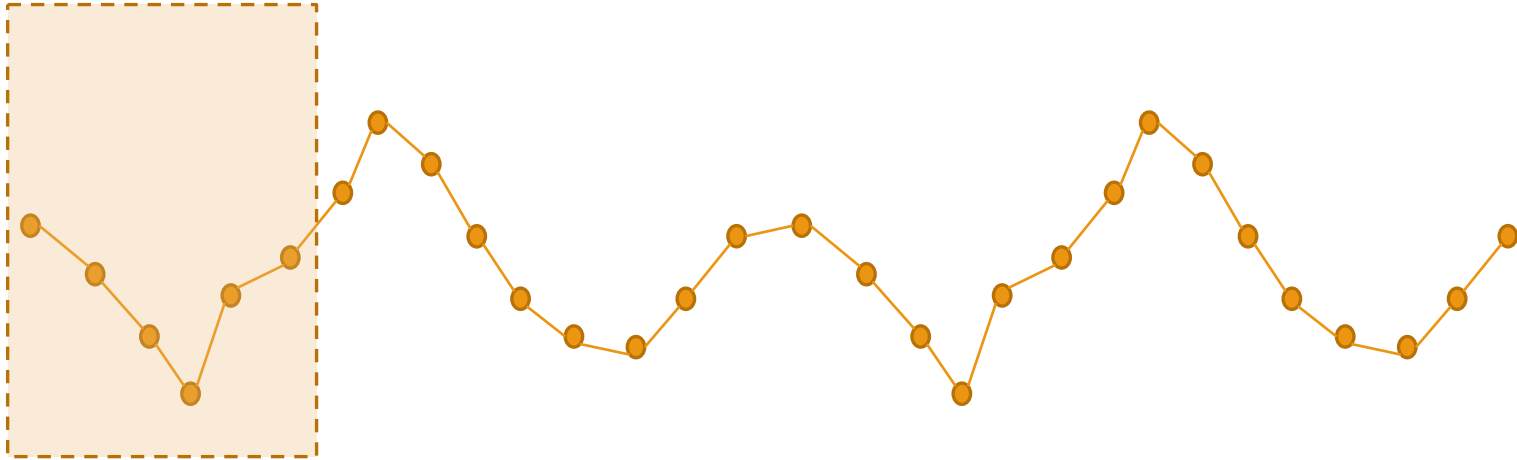
Lead and Lag

- Sorted
- Have multi records in context at once
- This is the simplest of time sessions operations

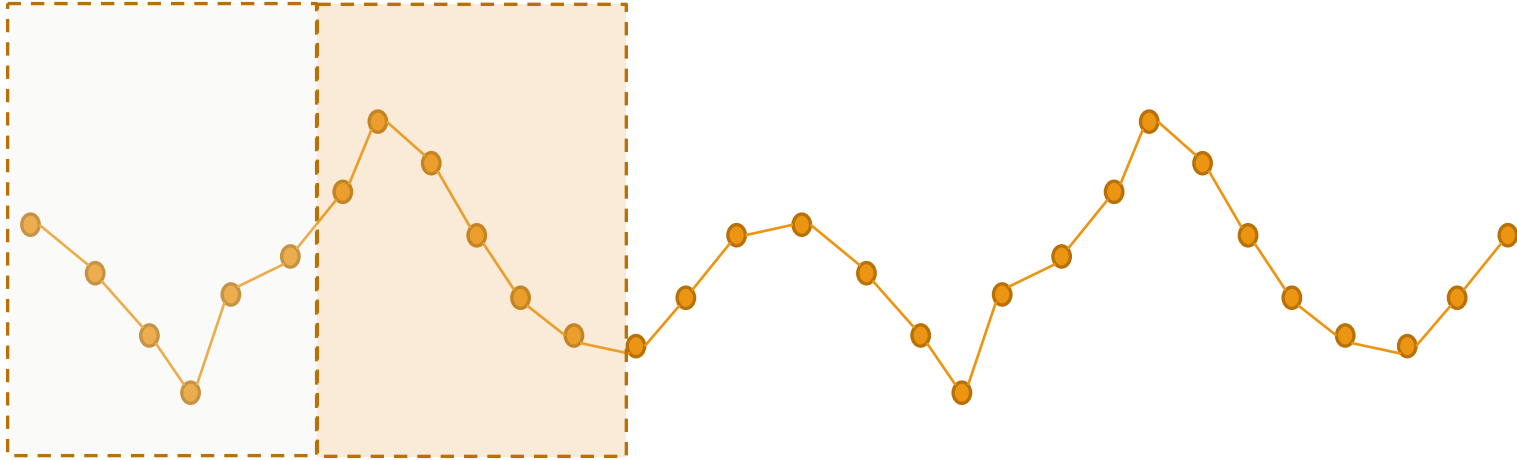
Tumbling Window

- Think of it like days or hours
- Hard cut offs

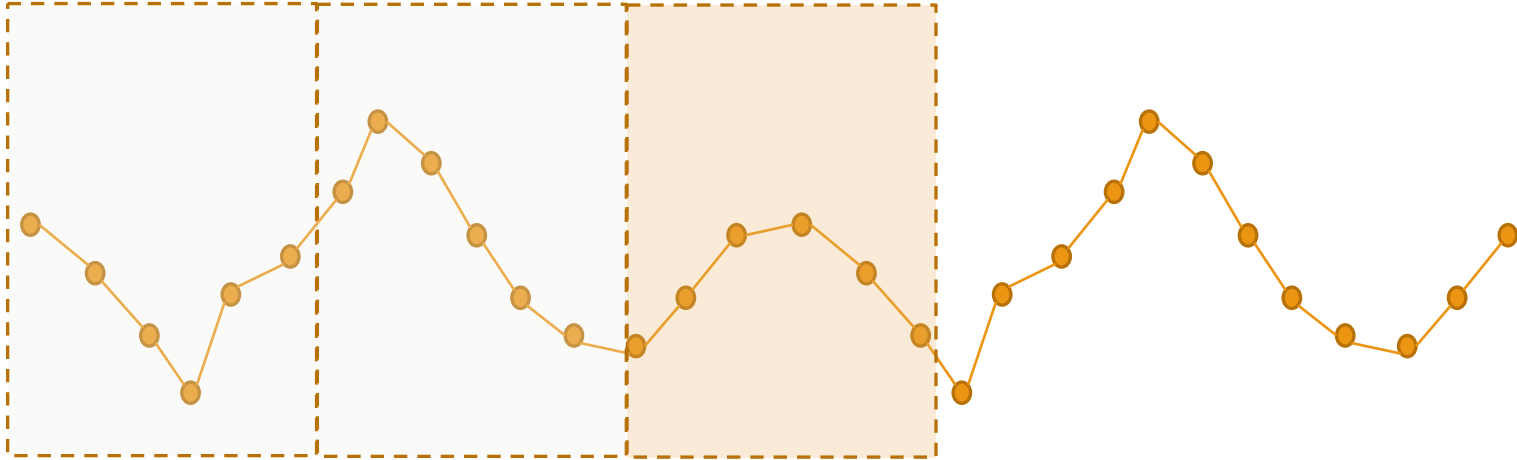
Tumbling Window



Tumbling Window



Tumbling Window



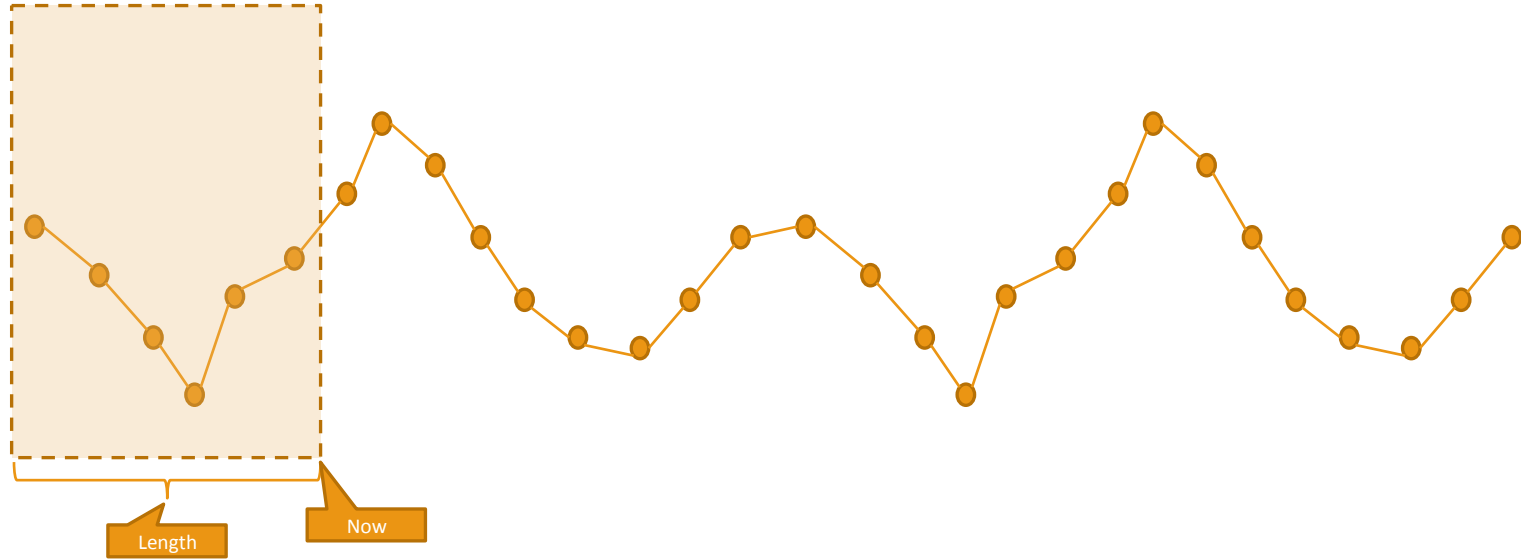
Tumbling Window

- Quarters in a American Football game
- Hard cut off aggregations
- Point based Interest

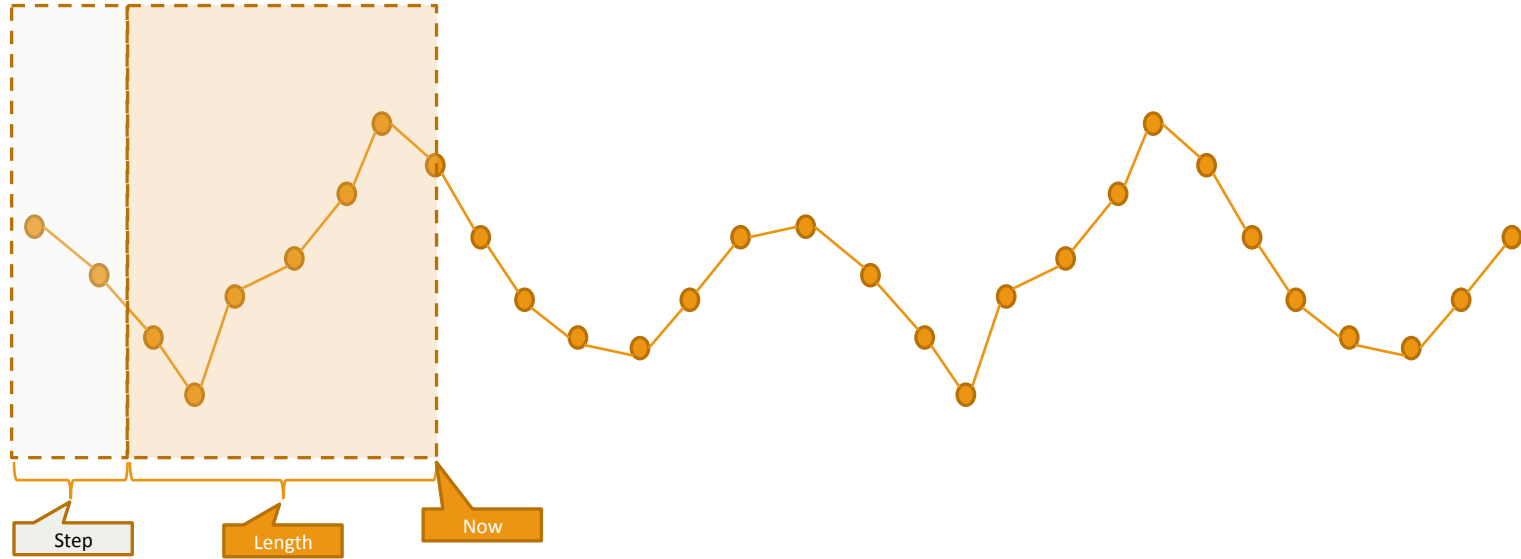
Sliding Windows

- Where Tumbling is hard cuts Sliding is a moving window
- Now
- Window Length
- Slide Interval

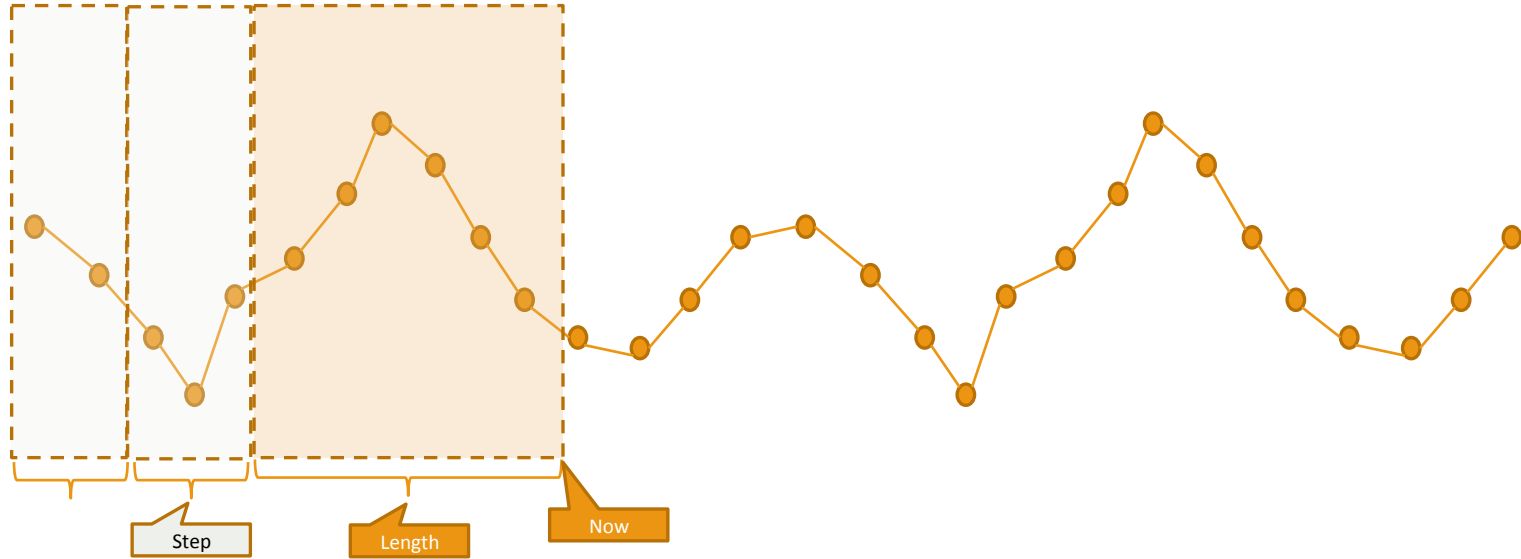
Sliding Window



Sliding Window



Sliding Window



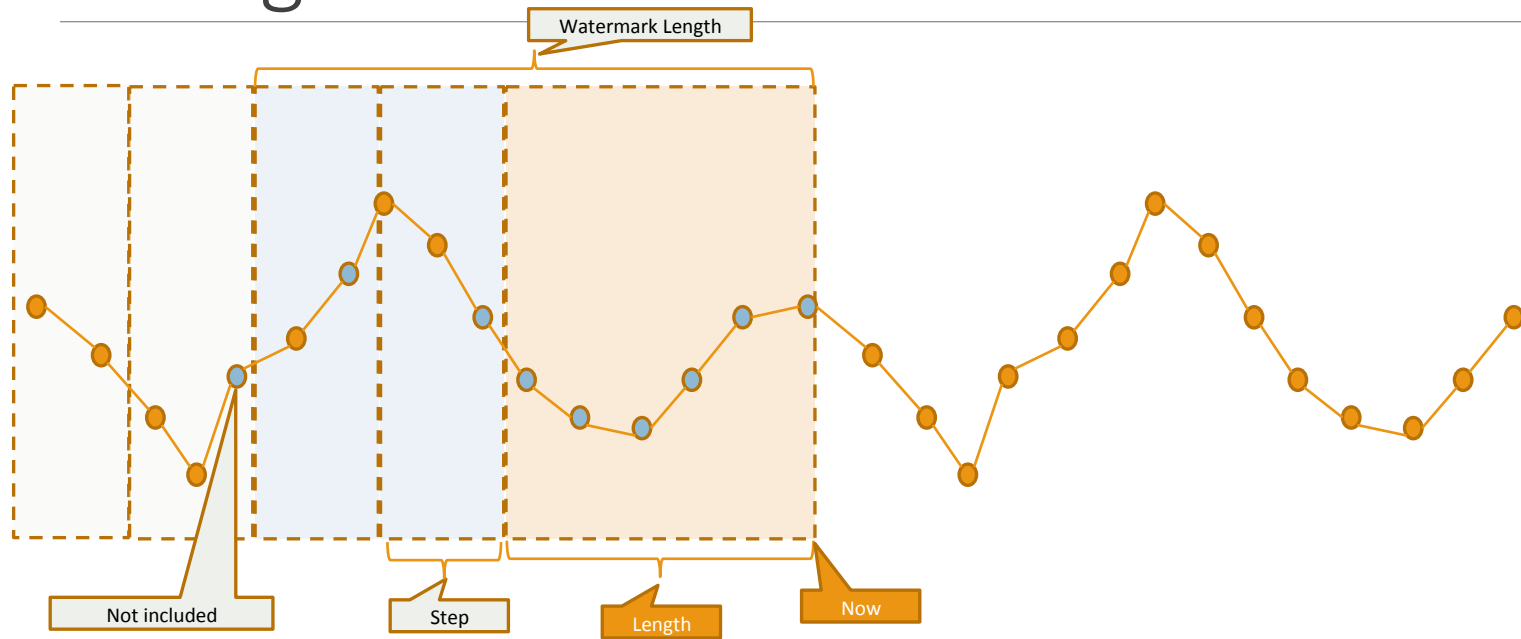
Event Time vs Process Time

- Process Time: is when the event hits your system
- Event Time: is when the event was created

Event Time vs Process Time

- Problem with Processing Time
 - Delay of delivery
- Problem with Event Time
 - Memory is not limitless
 - Reach back can be expensive
- Watermarks

Sliding Window: Event Time

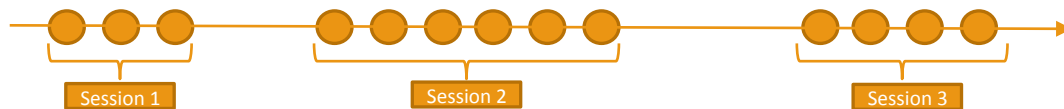


Session Windows

- **Tumbling and Sliding:** are global windowing techniques
- **Session:** is a entity specific window
- Type Of sessions Windows
 - Holding all the session
 - Holding aggregates of the session

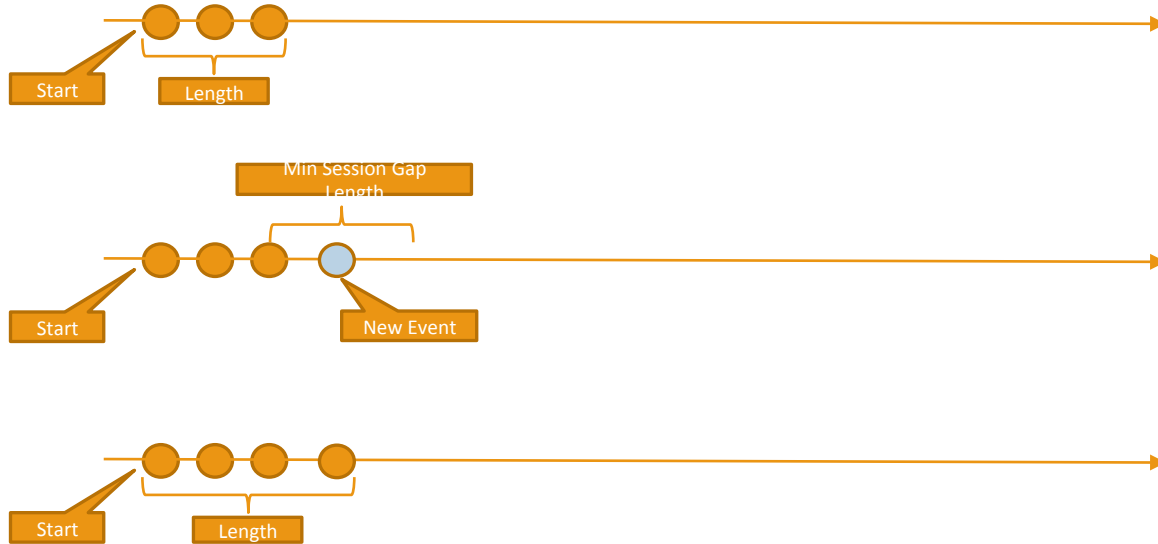
What to do with Sessions?

- Window of events defined by a gap

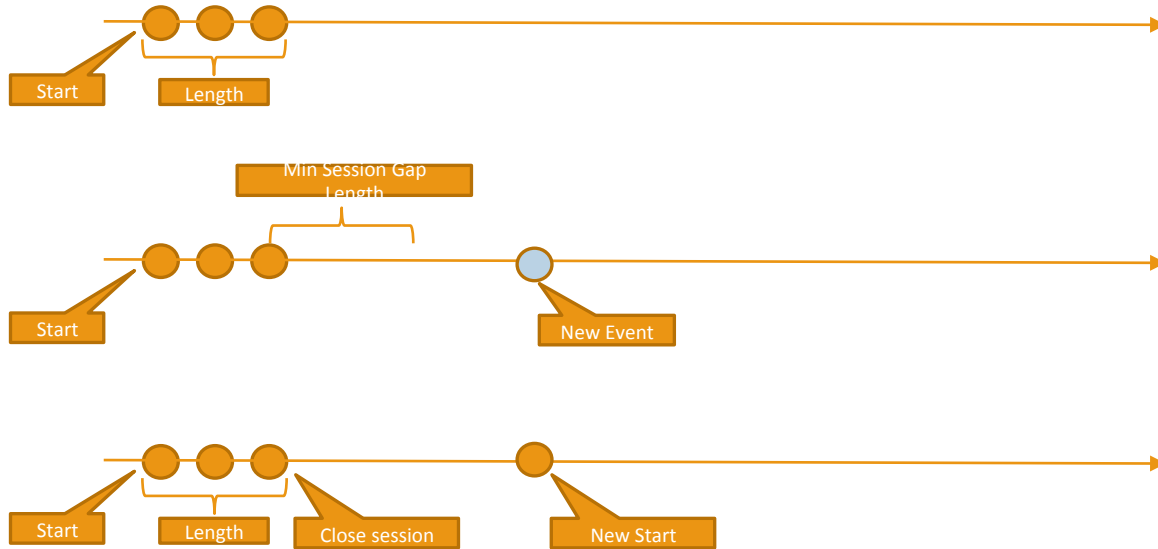


- What to do with them
 - Length of sessions
 - Number of sessions with in a day
 - Average separation between events in a session
 - Common orders with in sessions
 - Counts of types of events with in sessions

What to do with Sessions?



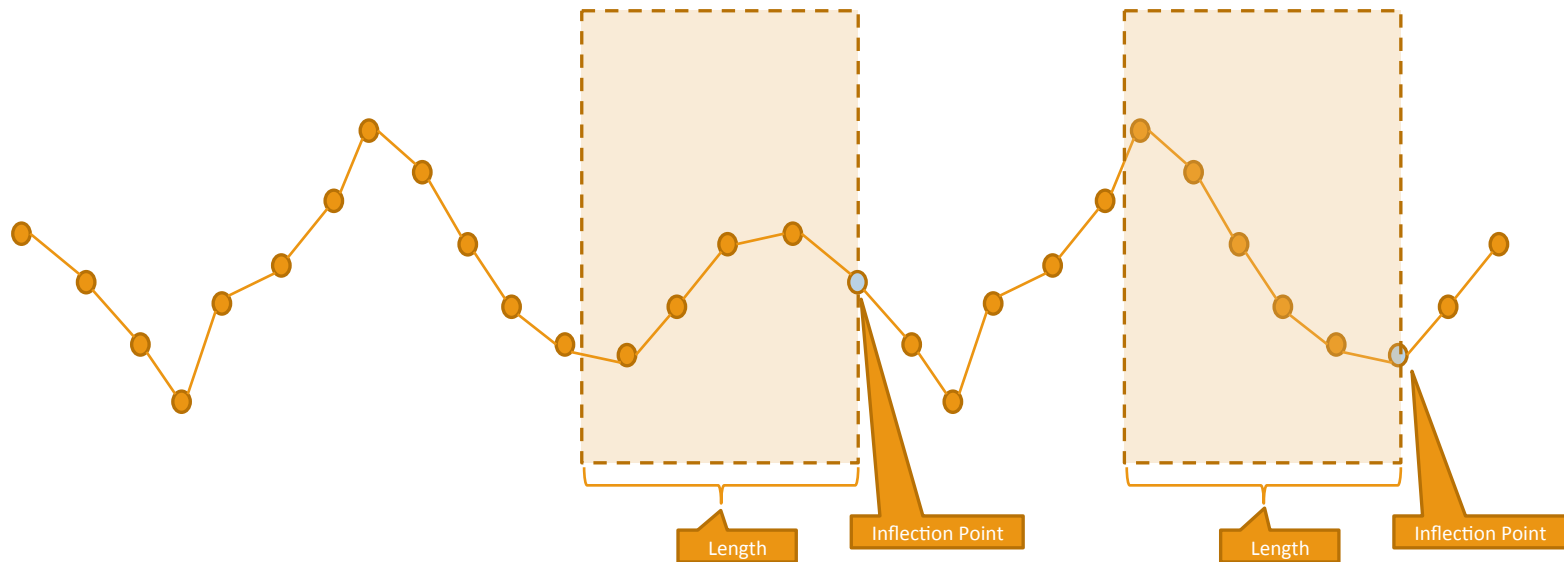
What to do with Sessions?



Respect to Infection Points

- Think about
 - Churn
 - Heart attacks
 - Death
- Infection point type
- Length before infection point

Inflection Point Windowing



Ways to look at time Series Data

- Lead and Lag
- Tumbling windows
- Sliding windows
- Session windows
- Respect to Infection Point

Questions

- ???

Before we do
Examples
Context

Types of Time Series Implementations

- Things we need to think about
 - Big and Small windowing
 - Stream vs batch

Big and Small

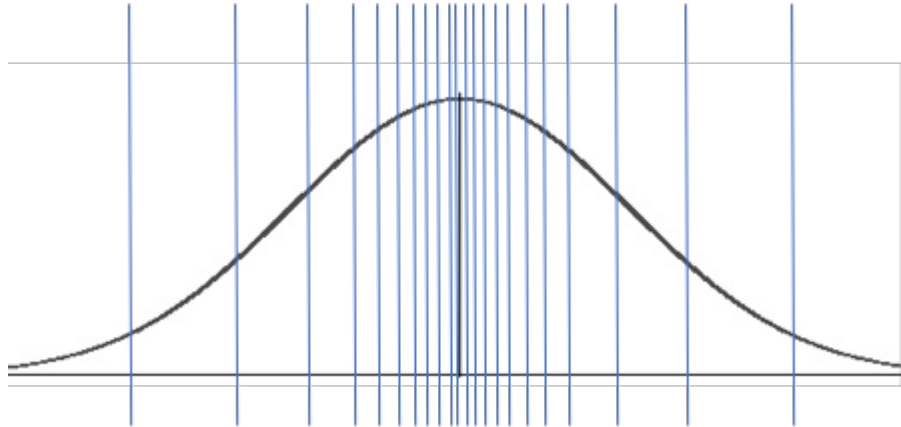
- Examples
 - Loan Payments
 - People Credit Card Statements
 - Stock Trades of Cloudera (Avg Volume 1 million)
 - Stock Trades of Starbucks (Volume 53 million on July 28)

Big and Small

- The real question
 - Do we trust that all events for our window can fit into the memory of one executor?
- If not we may want to split things up (Solutions)
 - Hash Partitioning ???
 - Time Range Partitioning
 - Sampling

Types of Range Partitioning

- Fixed Interval
 - Hours in a day
 - Days
- Normal curve distribution
- Unknown curve distributed
 - Code Example Soon



Big and Small

- Initially we will solve everything with the assumption of small
- We will solve big later in the course

Streaming vs Batch

- Stream -> continues feed
- Batch -> Interval feed

Streaming vs Batch

- Streaming is solved by making batches
 - Triggers
 - Evictors
 - Watermarks
- We will dig into streaming later
- For now everything will be batch

Questions

- ???

Lead & Lag Example

Use Case Pikes and Valleys

- Identifying pikes and valleys
 - SQL
 - Code
- Lets go to the code

Questions

- ???

Tumbling Windows

Use Case Role Ups Per Hour

- Identifying pikes and valleys
 - SQL
 - Code
- Lets go to the code

Questions

- ???

Sliding Window Example

Use Case Role Up Every 5 Minutes

- Identifying pikes and valleys
 - SQL
 - Code
- Lets go to the code

Questions

- ???

Session Windows

Use Case: Role Up Per Session

- Identifying pikes and valleys
 - Code
- Lets go to the code

Questions

- ???

Respect to
Inflection Point

Use Case: Role Up for N Minutes before Inflection Points

- Identifying pikes and valleys
 - SQL
 - Code
- Lets go to the code

Questions

- ???

Reflection on Samples

What was Common about Small Batch

- Some could be solved easily with SQL
- Some could not
- Simple Group by was a big part of the work
- We could do a lot of windowing operations with a single group by

What was Common about Small Batch

- Big problem is if we can't sort the data with one partition
 - Break it up
 - Or store it sorted

Questions

- ???

Let's talk about
bucketing and
sorting

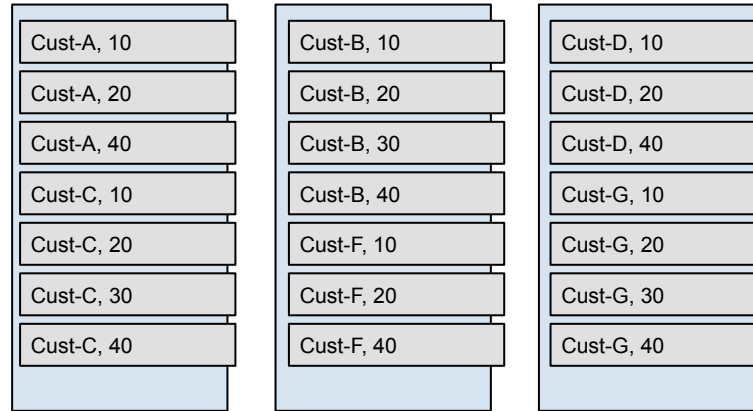
Use Case: Role Up for N Minutes before Inflection Points

- Identifying pikes and valleys
 - SQL
 - Code
- Lets go to the code

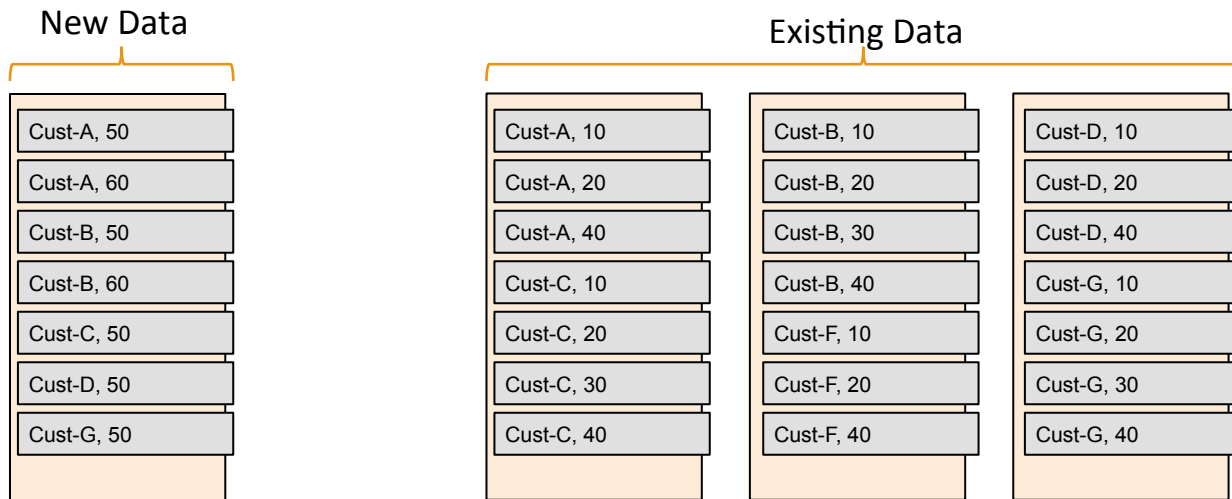
What is meant by Bucketing and Sorting

- Partitioning on a Key
- Then sorting on that key + another field(s)
- Example
 - User_id + Watch Event Time

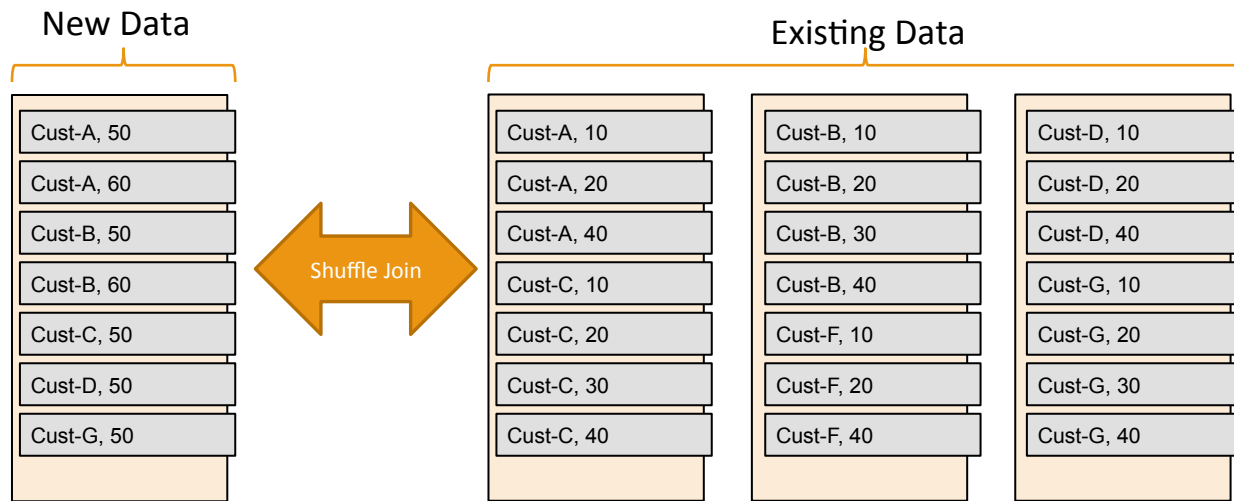
Example of Bucketed Sorted



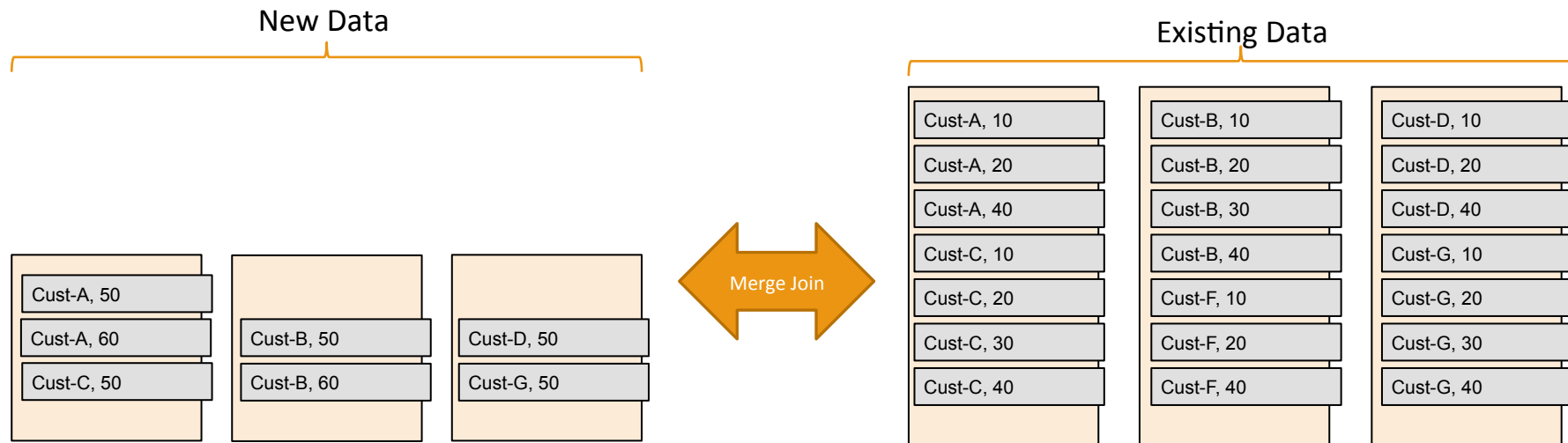
Adding to a Bucketed Sorted Dataset



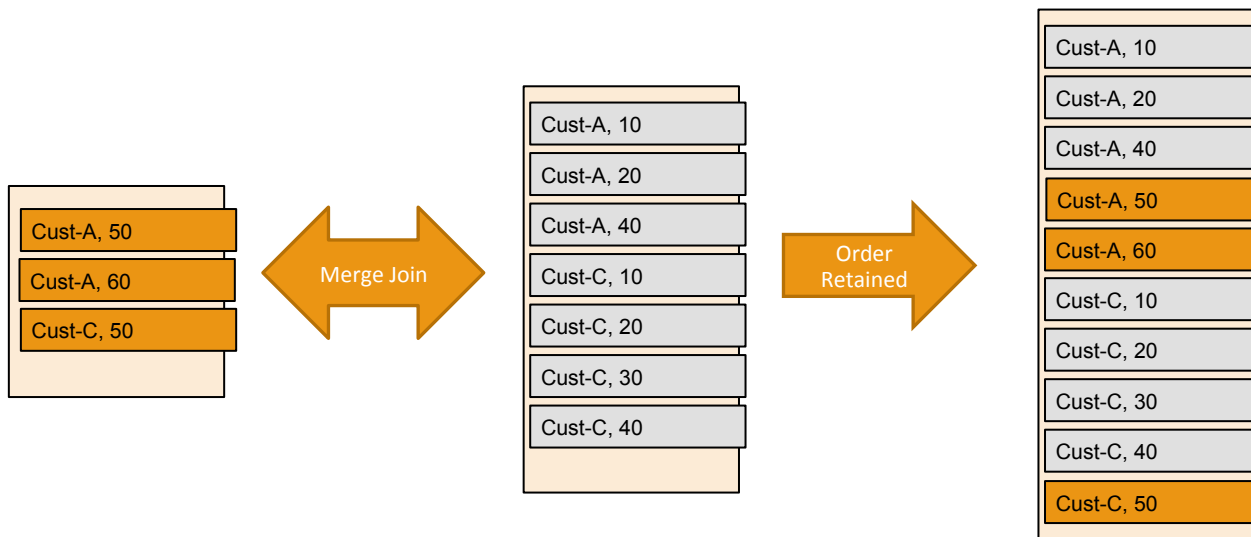
Adding to a Bucketed Sorted Dataset



Adding to a Bucketed Sorted Dataset



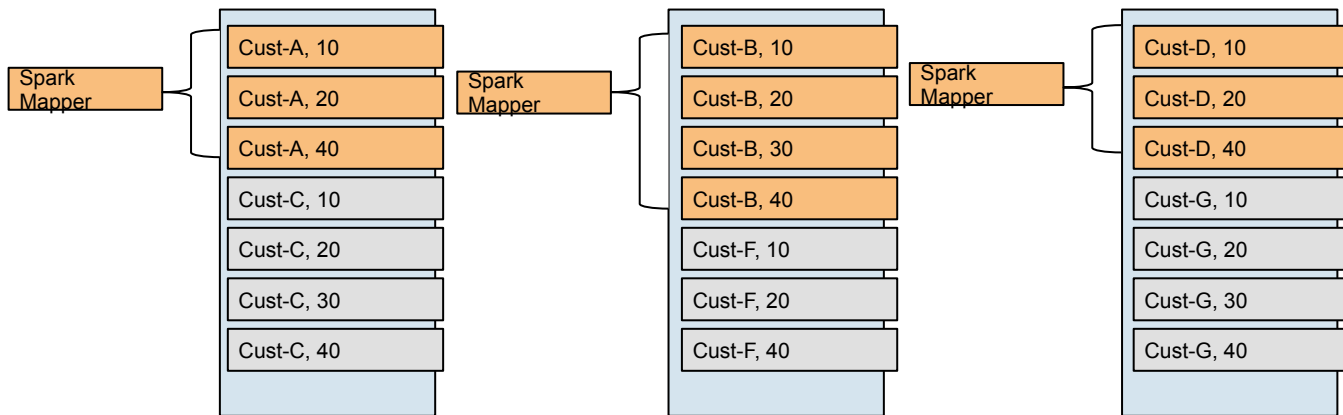
Adding to a Bucketed Sorted Dataset



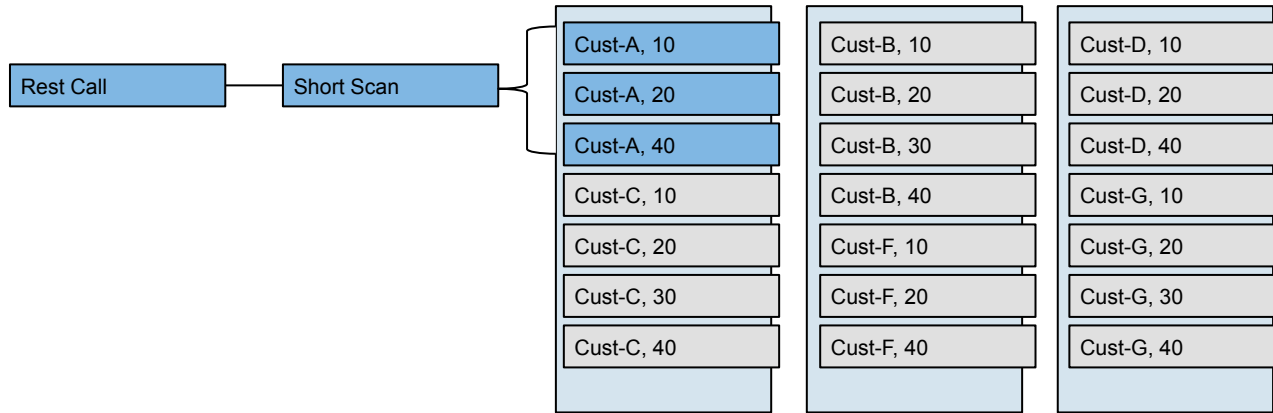
What else could be use Bucketing and Sorting for

- Windowing
- Point retrieval

Bucketed & Sorted for Windowing



Bucketed Sorted in a NoSQL



Questions

- ???

CDC Change Data Capture

Change Data Capture

- Events version snapshots
- Change Data Capture is the state of something with respect to time

Basic Example

- How do we make a Type 2 table with HDFS and a NoSQL

Doc_ID	Value	FromDt	ToDt
42	1	12/12/17	12/13/17
42	2	12/13/17	12/14/17
42	3	12/14/17	

HDFS & S3


- Partition on ToDt
- Always rewrite ToDt=now
- Append to ToDt=Yesterday

Doc_ID	Value	FromDt	ToDt
42	1	12/12/17	12/13/17
42	2	12/13/17	12/14/17
42	3	12/14/17	

HDFS & S3


- Partition on ToDt
- Always rewrite ToDt=now
- Append to ToDt=Yesterday

Sunday




Doc_ID	Value	FromDt	ToDt
A	A1	Saturday	Sunday
B	B1	Saturday	Sunday

Monday



Doc_ID	Value	FromDt	ToDt
A	A2	Sunday	Monday
C	C1	Saturday	Monday

Partition=Now



Doc_ID	Value	FromDt	ToDt
A	A3	Monday	
B	B2	Sunday	
C	C2	Saturday	

HDFS & S3

New Data

Doc_ID	Value
A	A4
B	B3

Sunday

Doc_ID	Value	FromDt	ToDt
A	A1	Saturday	Sunday
B	B1	Saturday	Sunday

Monday

Doc_ID	Value	FromDt	ToDt
A	A2	Sunday	Monday
C	C1	Saturday	Monday

Tuesday

Doc_ID	Value	FromDt	ToDt
A	A3	Monday	Tuesday
B	B2	Sunday	Tuesday

Now

Doc_ID	Value	FromDt	ToDt
A	A4	Tuesday	
B	B3	Tuesday	
C	C2	Saturday	

HDFS & S3

New Data

Doc_ID	Value
A	A4
B	B3

Sunday

Doc_ID	Value	FromDt	ToDt
A	A1	Saturday	Sunday
B	B1	Saturday	Sunday

Monday

Doc_ID	Value	FromDt	ToDt
A	A2	Sunday	Monday
C	C1	Saturday	Monday

Tuesday

Doc_ID	Value	FromDt	ToDt
A	A3	Monday	Tuesday
B	B2	Sunday	Tuesday

Now

Doc_ID	Value	FromDt	ToDt
A	A4	Tuesday	
B	B3	Tuesday	
C	C2	Saturday	

NoSQL

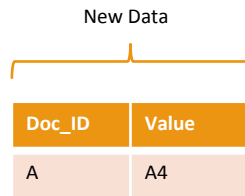
- Combination Key
 - Doc_Id, ToDt
- Append existing record with new ToDt as Yesterday
- Then rewrite existing record to have a new FromDt and new value

Doc_ID & ToDt	Value	FromDt
A	A3	12/14/17
A:12/13/17	A1	12/12/17
A:12/14/17	A2	12/13/17

NoSQL

- Combination Key
 - Doc_Id, ToDt
- Append existing record with new ToDt as Yesterday
- Then rewrite existing record to have a new FromDt and new value


New Data



Doc_ID	Value
A	A4

NoSQL


- Combination Key
 - Doc_Id, ToDt
- Append existing record with new ToDt as Yesterday
- Then rewrite existing record to have a new FromDt and new value



Doc_ID & ToDt	Value	FromDt
A	A3	12/14/17
A:12/13/17	A1	12/12/17
A:12/14/17	A2	12/13/17
A:12/15/17	A3	12/14/17

NoSQL

- Combination Key
 - Doc_Id, ToDt
- Append existing record with new ToDt as Yesterday
- Then rewrite existing record to have a new FromDt and new value



Doc_ID & ToDt	Value	FromDt
A	A4	12/14/17
A:12/13/17	A1	12/12/17
A:12/14/17	A2	12/13/17
A:12/15/17	A3	12/14/17

NoSQL – Two Tables

- Combination Key
 - Doc_Id, ToDt
- Append existing record with new ToDt as Yesterday
- Then rewrite existing record to have a new FromDt and new value

Current

Doc_ID & ToDt	Value	FromDt
A	A3	12/14/17

History

Doc_ID & ToDt	Value	FromDt
A:12/13/17	A1	12/12/17
A:12/14/17	A2	12/13/17

NoSQL – Two Tables

- Combination Key
 - Doc_Id, ToDt
- Append existing record with new ToDt as Yesterday
- Then rewrite existing record to have a new FromDt and new value

Current

Doc_ID & ToDt	Value	FromDt
A	A3	12/14/17

History

Doc_ID & ToDt	Value	FromDt
A:12/13/17	A1	12/12/17
A:12/14/17	A2	12/13/17
A:12/15/17	A3	12/14/17

NoSQL – Two Tables

- Combination Key
 - Doc_Id, ToDt
- Append existing record with new ToDt as Yesterday
- Then rewrite existing record to have a new FromDt and new value

Current

Doc_ID & ToDt	Value	FromDt
A	A4	12/15/17

History

Doc_ID & ToDt	Value	FromDt
A:12/13/17	A1	12/12/17
A:12/14/17	A2	12/13/17
A:12/15/17	A3	12/14/17

Cassandra Example

- Let's look at the code

Hive Example

- Let's look at the code

Questions

- ???

TTL

TTL

- Time to live
- Think of sliding windows
 - Now
 - Window Length

TTL (In Kafka)

- Image

TTL (In NoSQL)

- Image

TTL (In File Store or Object Store)

- Image

Questions

- ???

Recap

Questions

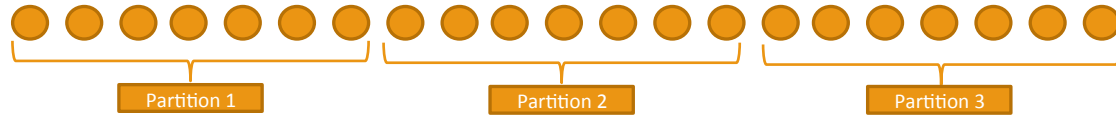
- What is time series
- Why is time series so important
- Ways we can look at time series data
- Examples
 - Lead & Lag
 - Tumbling window
 - Sliding window
 - Session window
 - Respect to inflection point
- Reflection on samples
- Let's talk about Bucketing and sorting
- CDC Change Data Capture
- TTL

Processing Big Windows in Batch

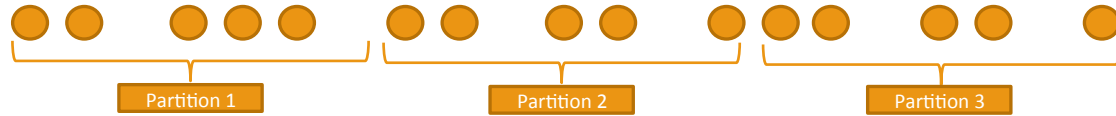
Processing Big Windows

- We will do fixed interval partitioning
- We will have two types of real time data
 - Fix interval events
 - Random interval events

Processing Big Windows: Fixed Interval Events



Processing Big Windows: Random Interval Events



Let's do some Peak and Valleys Again

- Spark Example

So what did we do

- Sorted the Data
- Read the start of every partition
- Sent the start of every partition and the partition number to the driver
- Broadcast the edge information to every partition
- Went through the data again with the edge information from the partition before

Questions

- ???

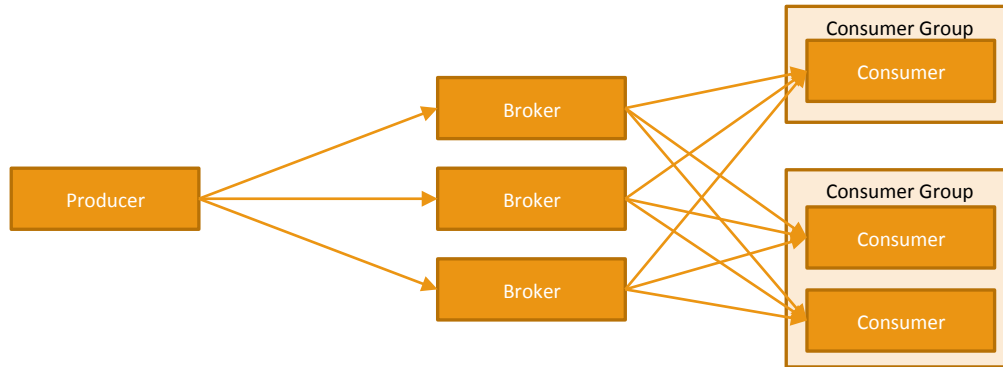
Kafka

- Producers
- Brokers
- Zookeeper
- Consumer Groups
- Consumers
- Topics
- Partitions
- Listeners
- Failure and Rebalancing

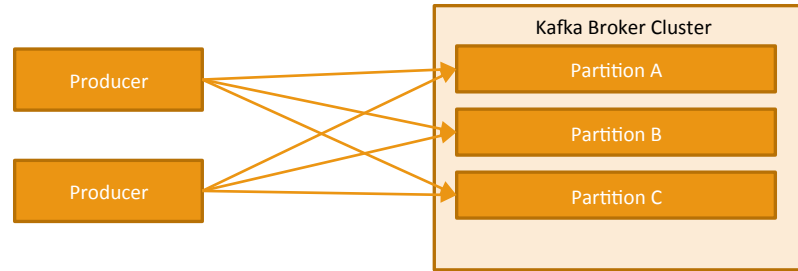
Distributed Log

- Pub-sub vs distributed log
- One to one
- One to many
- * Storage SQL and NRT

Basic Parts



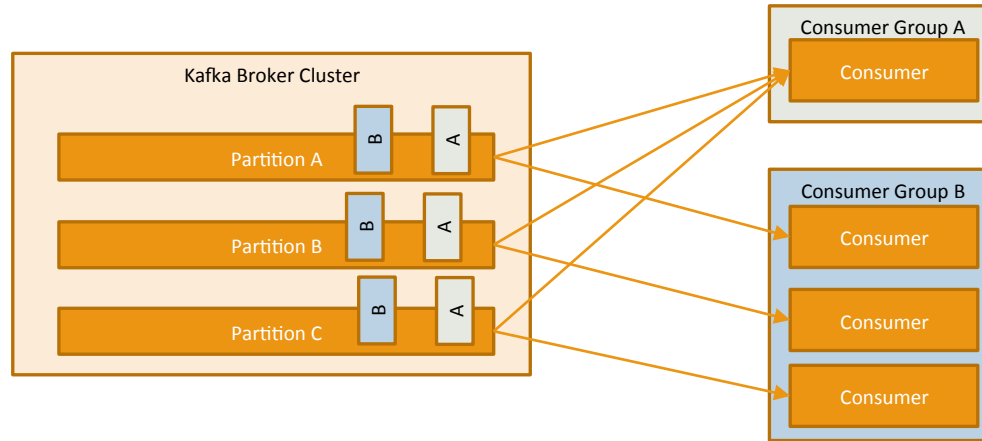
View from the Producer



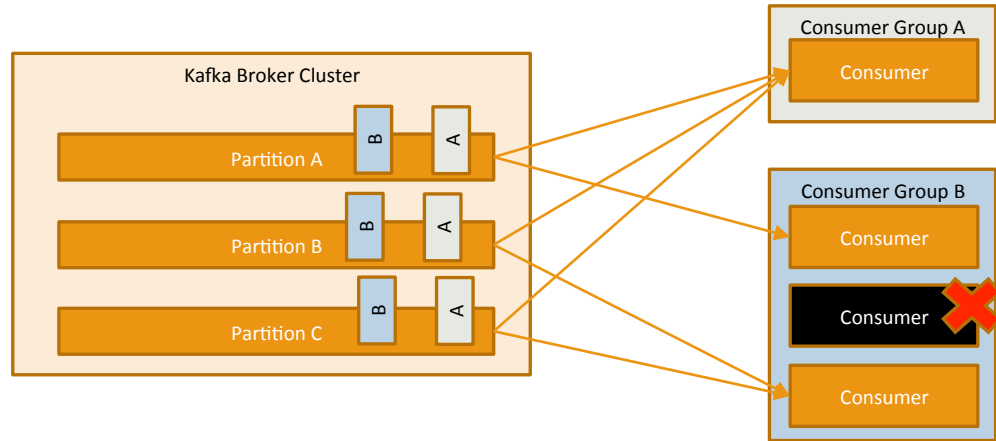
Partitioning

- Round Robin
- Lazy Round Robin
- Custom
 - Good and Bad
 - Avoid Skew

View from the Consumer

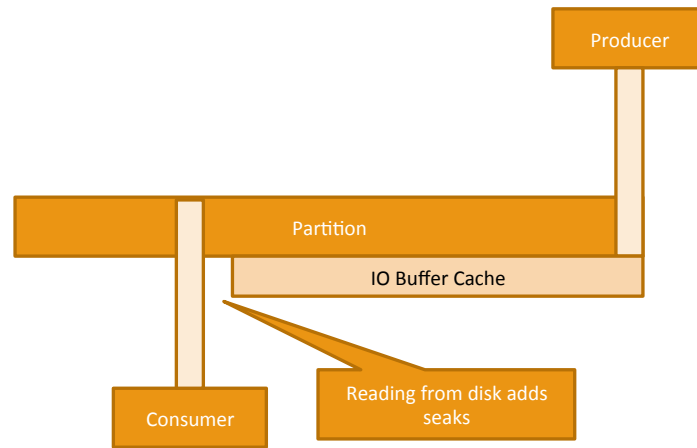
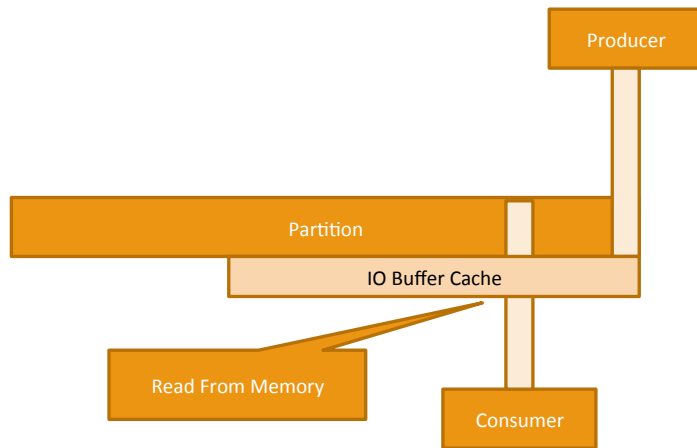


View from the Consumer



Kafka

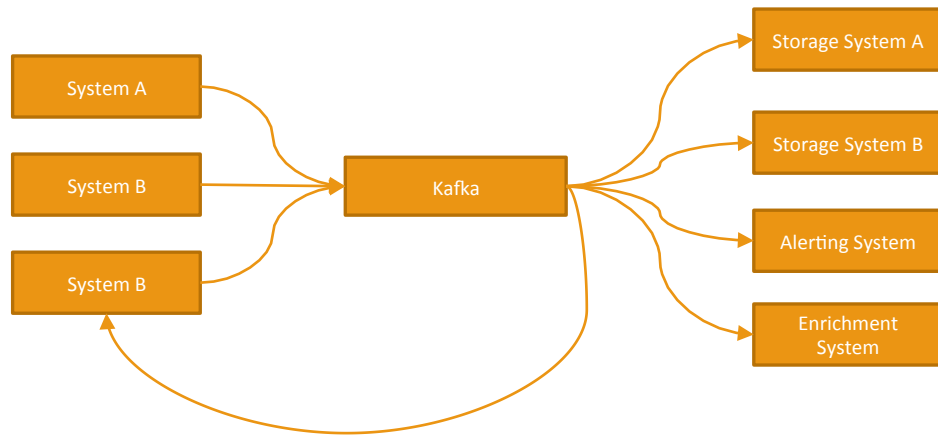
- Why is Kafka so Fast



Throughput and Latency

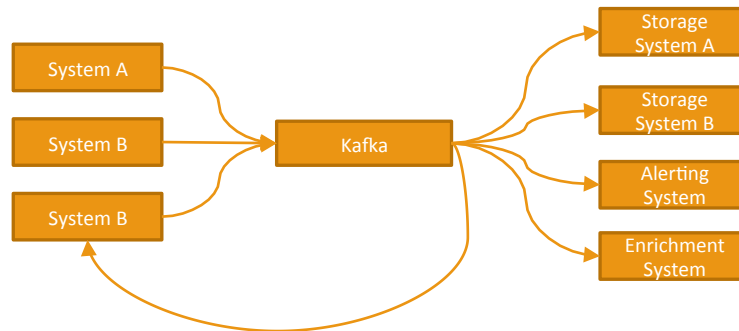
- Record Count
- Byte Count
- Linger Time

Golden Path (ESB)



Golden Path (ESB)

- Protected by replication
- Buffered for N amount of time
- Every one reads the same data in the same order



Summary & Q/A

Setting up Kafka

- Set up Kafka
- Create some topics
- Push some data to Kafka
- Read some data to Kafka
- Play around with Partitions a little

Setting up Kafka Locally

- We are going to use Docker Containers
- `docker pull wurstmeister/kafka`
- <https://hub.docker.com/r/wurstmeister/kafka/>
- [spotify/docker-kafka](https://hub.docker.com/r/spotify/docker-kafka/)
- <https://hub.docker.com/r/spotify/kafka/>

Create a Topic

- `bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test`
- `bin/kafka-topics.sh --list --zookeeper localhost:2181`

Test out out topic with the cmd line

- `bin/kafka-console-producer.sh --broker-list localhost:9092 --topic test`
- `bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning`

Setting Up Java Env

1. IntelliJ
2. Maven
3. Pom File

Java based producer

Lets look at the code

Java based Custom Partitioner

Let's look at the code

Java based Consumer

Let's look at the code

Java based Consumer Listeners

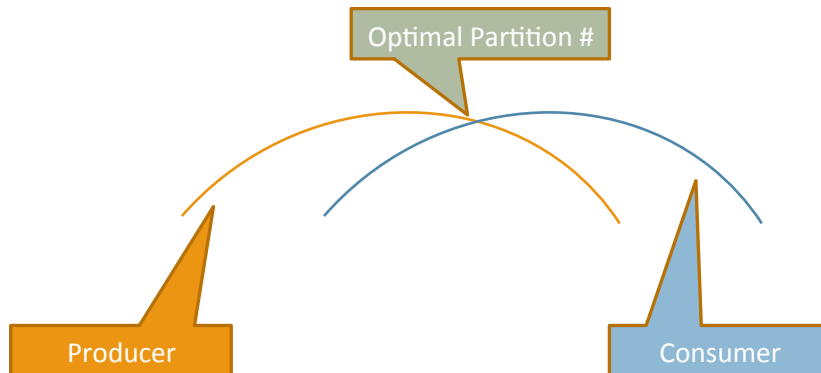
Let's look at the code

Configuring Things

1. Number of Partitions
2. Linger Times
3. Batch sizes
4. Message sizes

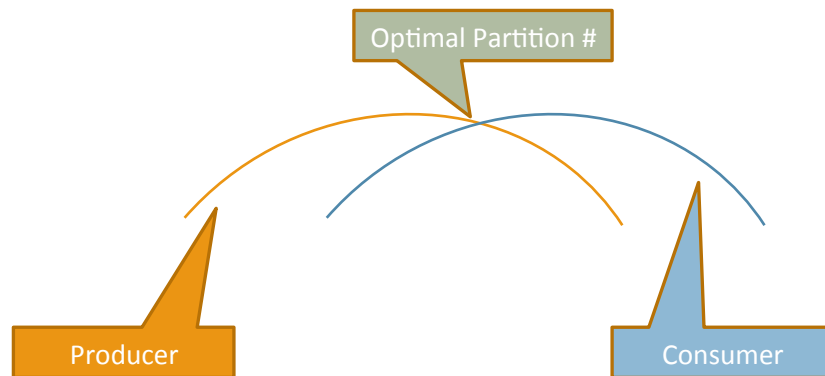
Configuring Things

1. Number of Partitions
 1. Benchmark for Producers
 2. Benchmark for Consumers
 3. Benchmark during failure
 4. Identify recovery time
 5. Look for the double hump



Configuring Things

1. Look for the double hump



Configuring Things

1. Linger Time
 1. Start with 0 and then try 1 millisecond

Configuring Things

1. Batch Sizes
 1. Start with a 1000 and look for the best option for you
 2. There will be a diminishing rate of return as you increase
2. Consider message size

Configuring Things

1. Message Size
 1. Try to stay under a 1KB
 2. If pre-compressed consider that also

Summary & Q/A

Why Kafka

- Throughput
- Partitions
- Order
- Replay ability
- TTL

Throughput

- Files per Partition
- Append only
- IO Buffer Stuff
- No required for the consumer guarantees

Partitions

- RePartition solves a boat load of processing problems
 - Joins
 - Distributed cache
- Custom Partitioning
- Listeners
- Custom Assignment

Order

- Determinist behavior is the key to failure recover
- Partition and Order can be used to simulate eventually consistent transactions

Replay Ability

- Order mixed with replay allows for failure recovery
- Testing
- Buffering
- Multi consumers

Replay Ability (Additional Call Out)

- Steaming Engine Internal State
 - Check pointing
 - Micro Batching
 - Barriers
- Relies on being able to replay

Replay Ability (Additional Call Out)

- Disaster protection
- 7 Day TTL
- < 7 Day check pointing

TTL

- Data will age out
- Be careful not to fill your drives
- Allows for a dumb simple way to remove data
- Satisfies most desires for consumer guarantees

TTL (Considerations)

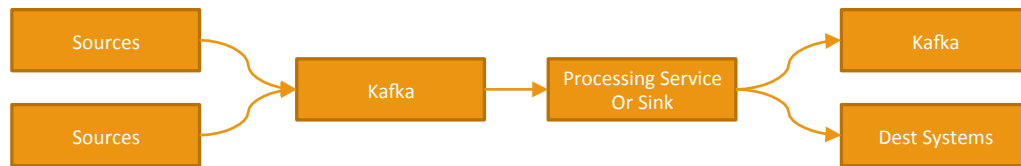
- Be mindful of disk space
- Don't use Kafka as a final data source

Summary & Q/A

Components of a Streaming Architecture

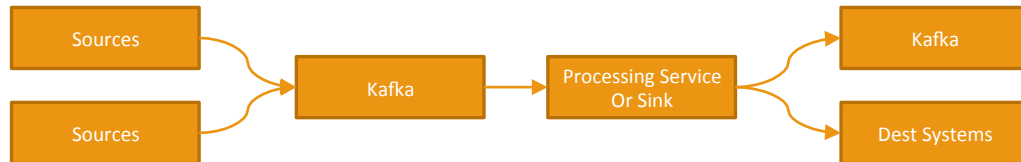
- Source
- Pipes
- Processing
- Ingestion

Basic Parts

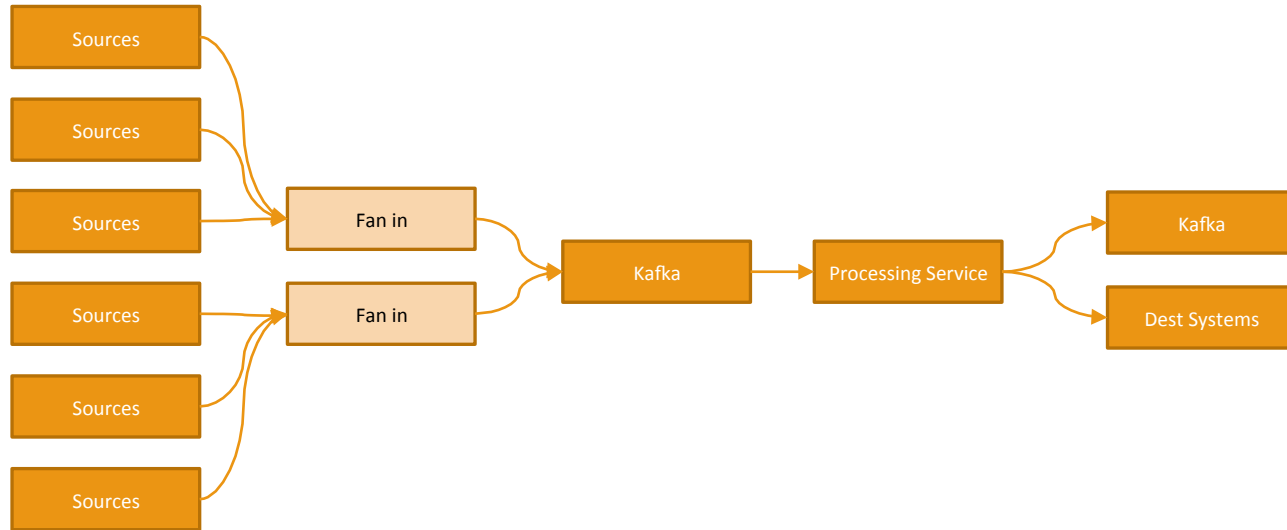


Basic Parts

1. Sources (Interface design)
2. Kafka (the Pipe)
3. Processing or Sink (Micro Service)
4. Destination

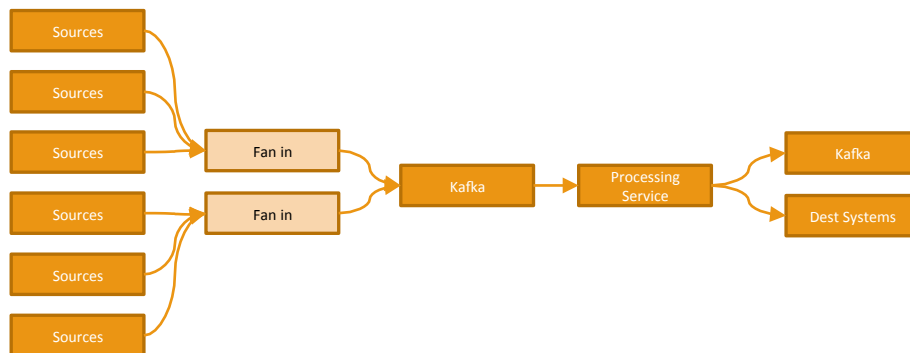


Advanced Parts (Fan In)

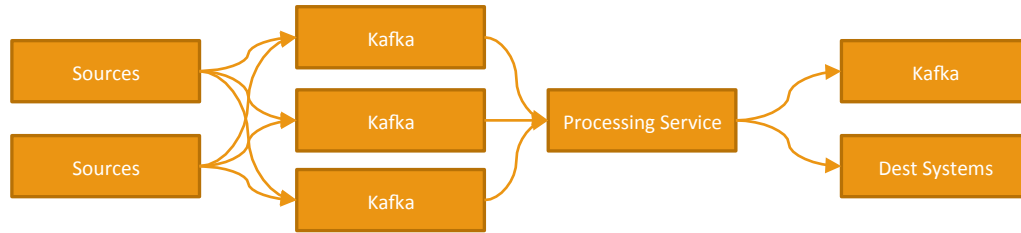


Advanced Parts (Fan In)

1. Protect Kafka
2. Abstract out Kafka
3. Concerns
 1. Data lose
 2. Performance
 3. Scalability

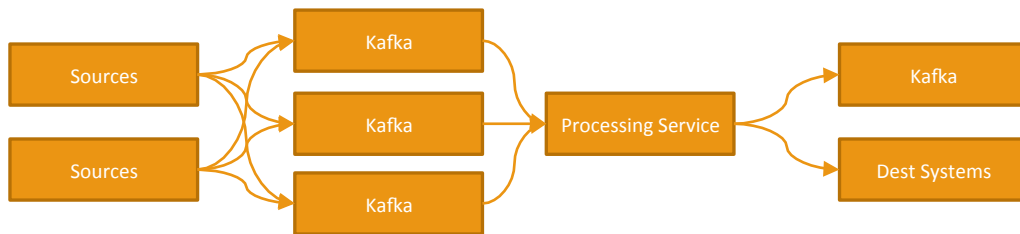


Advanced Parts (Multiple Kafkas)



Advanced Parts (Multiple Kafkas)

1. Localized Failure
2. Planned down time
3. Need for smart Producers and consumers



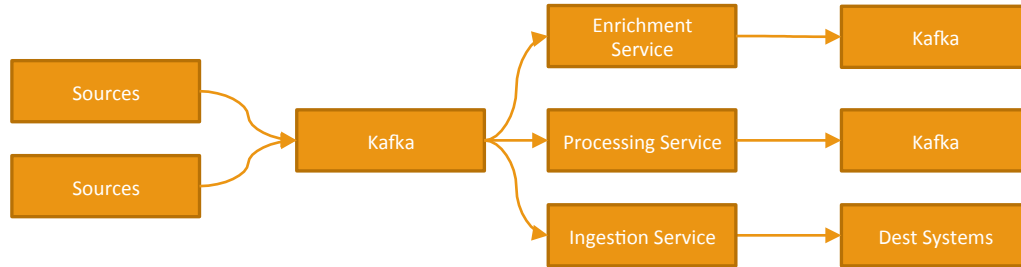
Advanced Parts (Switching Topics)

1. Message protocol
2. Repartitioning
3. Upgrades
4. Cluster switching

Advanced Parts (Switching Topics)

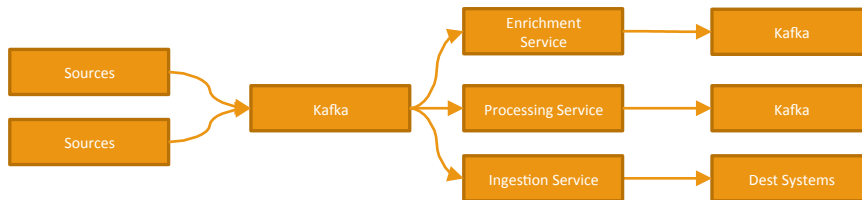
1. Switch Producers
2. Drain old topic
3. Then start reading from new topic
4. Delete old topic

Advanced Parts (Multiple Consumer Groups)

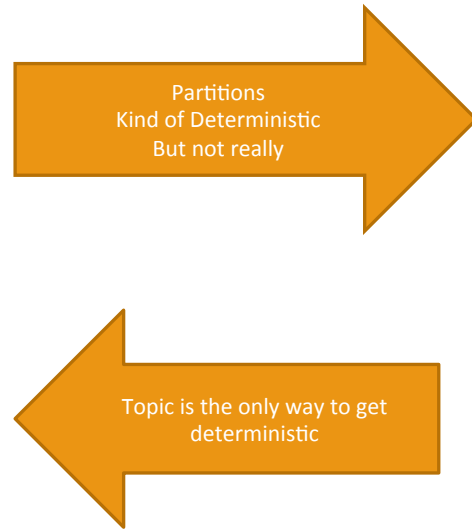


Advanced Parts (Multiple Consumer Groups)

1. Source of truth
2. Golden path



Advanced Parts (RPC)



Summary & Q/A

Streaming Engines and Market

Streaming Engines and Market

- Spark Streaming
- Spark Structured Streaming
- Flink
- Kafka Streams
- Storm
- Haren ???
- Apex
- Beam
- Micro Services
- Amazon Lambda

Streaming Engines and Market

- Spark Streaming
 - Same as ETL
 - High Latency
 - High Throughput
 - Very popular

Streaming Engines and Market

- Spark Streaming
 - Code

Streaming Engines and Market

- Spark Structured Streaming
 - Same as ETL
 - Low Latency
 - High Throughput
 - Very popular
 - New

Streaming Engines and Market

- Spark Structured Streaming
 - Code

Streaming Engines and Market

- Flink
 - Same as ETL
 - Low Latency
 - High Throughput
 - Very popular

Streaming Engines and Market

- Flink
 - Code

Streaming Engines and Market

- Kafka Streams
 - Same as ETL
 - Low Latency
 - High Throughput
 - Popular
 - Leans of Kafka a lot

Streaming Engines and Market

- Kafka Streams
 - Code

Streaming Engines and Market

- Storm
 - Old School
 - Low throughput
 - Low latency

Streaming Engines and Market

- Storm
 - Code

Streaming Engines and Market

- Haren ???
 - Storm 2.0
 - Backed by Twitter
 - Low latency
 - High throughput

Streaming Engines and Market

- Apex

Streaming Engines and Market

- Beam
 - Abstraction over
 - Spark
 - Flink
 - Dataflow

Streaming Engines and Market

- Beam
 - Code

Streaming Engines and Market

- Micro Services
 - Custom
 - Low latency
 - High throughput
 - May require Kafka Partitioning
 - In general dumber, more code, leaner, and faster

Streaming Engines and Market

- Apache Lamdba
 - Function as a Service
 - Easy
 - Flexable

Streaming Engines and Market

- Apache Lamdba
- Code

Summary & Q/A

Tables in a Stream

Windowing – Digging down

- Think about a WAL (write ahead log)
- Think about the table it creates

Windowing – Trigger and Evictors

- Trigger
 - When to execute a operation
- Evictor
 - When a event should be removed from a window

Windowing – Trigger and Evictors


- Any event adds to the current state
- Triggers run a query on the current state
- Evictors remove records from the current state

Windowing – Example

- Trigger => every 4 messages
- Evict => every 6 message
- Action => Sum

Windowing – Example

- Trigger => every 4 messages
- Evict => every 6 message
- Action => Group by Sum

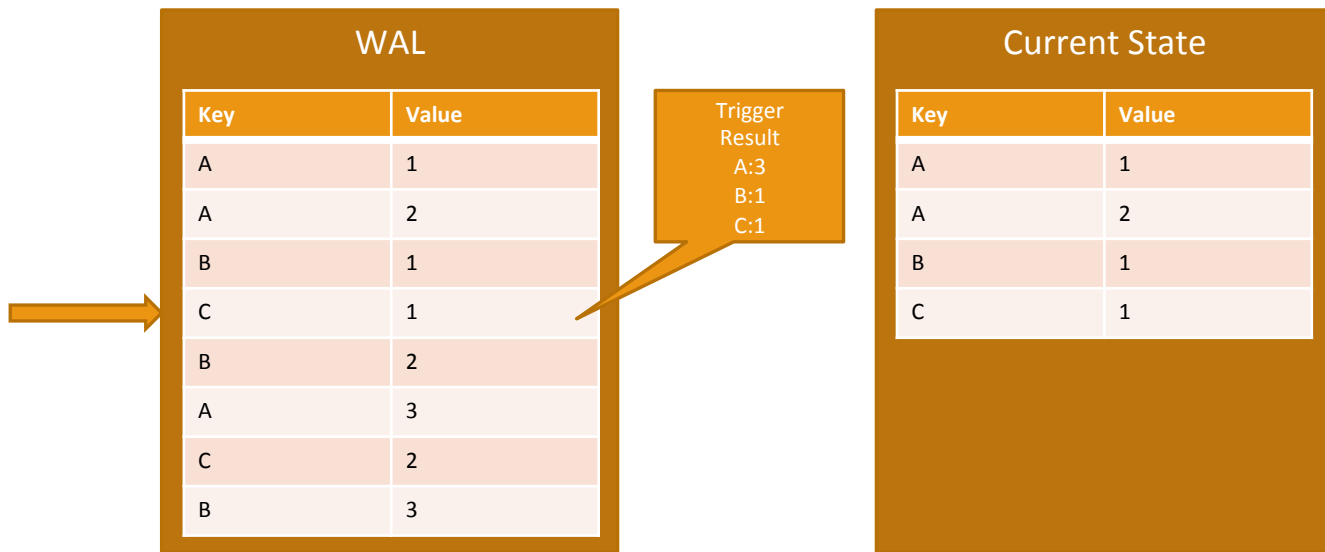


Key	Value
A	1
A	2
B	1
C	1
B	2
A	3
C	2
B	3

Key	Value
A	1
A	2
B	1


Windowing – Example

- Trigger => every 4 messages
- Evict => every 6 message
- Action => Group by Sum



Windowing – Example

- Trigger => every 4 messages
- Evict => every 6 message
- Action => Group by Sum

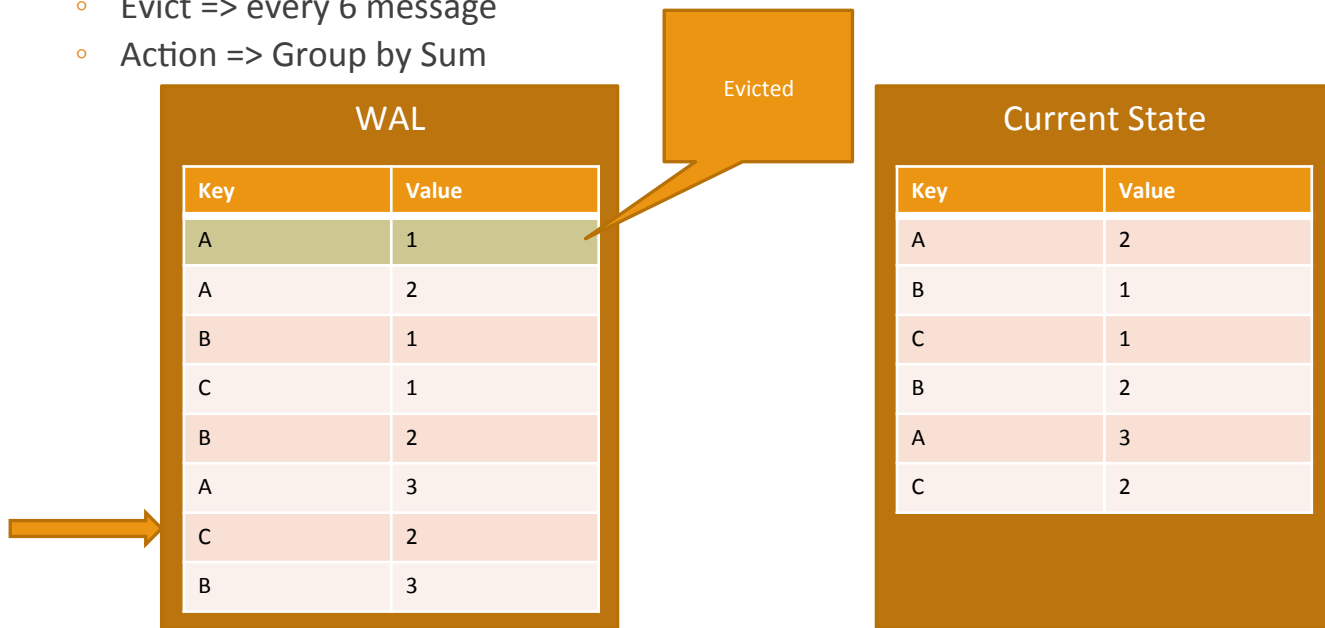


Key	Value
A	1
A	2
B	1
C	1
B	2
A	3
C	2
B	3

Key	Value
A	1
A	2
B	1
C	1
B	2
A	3

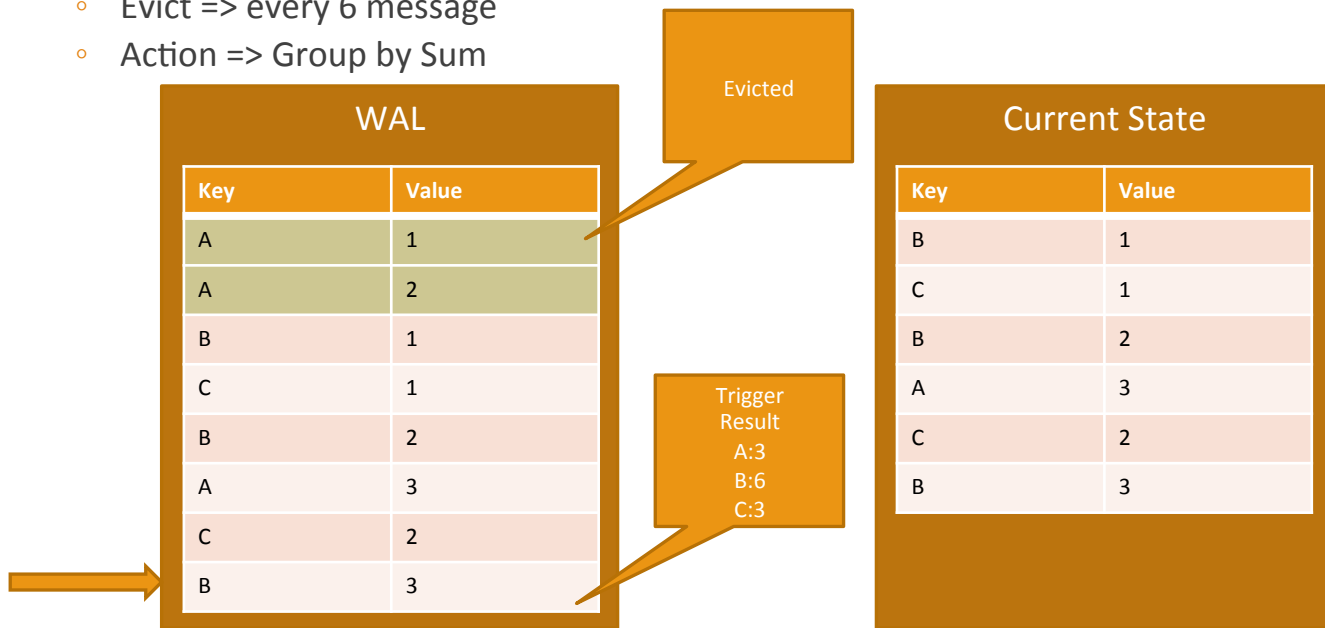
Windowing – Example

- Trigger => every 4 messages
- Evict => every 6 message
- Action => Group by Sum



Windowing – Example

- Trigger => every 4 messages
- Evict => every 6 message
- Action => Group by Sum



Windowing In Streams

Let's Implement with the following

- Spark Streaming
- Spark Structured Streaming
- Flink
- Kafka Streams

Use Case

- Peak and Valley finder
- Aggregation

Peak and Valley: Code example

- [Go to the code](#)

Aggregation

- [Go to the code](#)

Aggregation & Counting

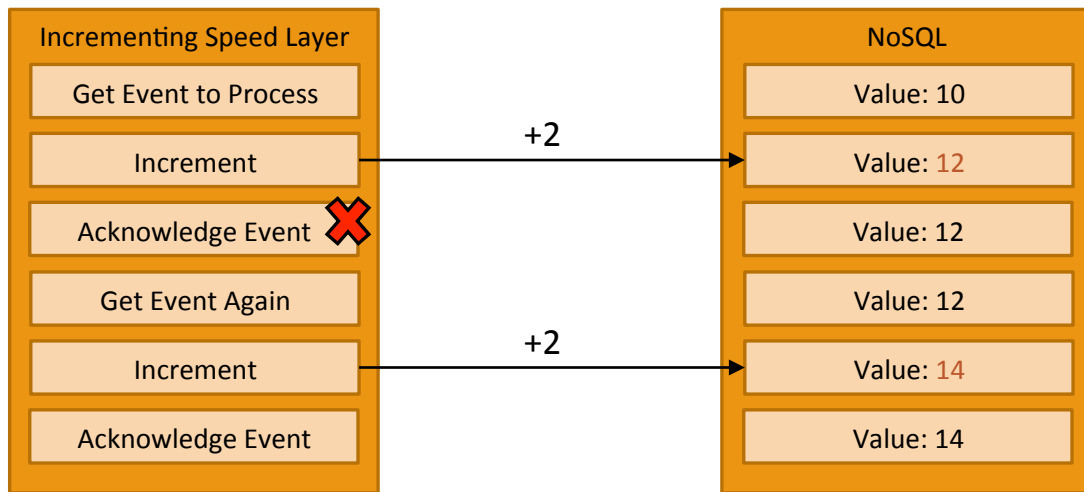
- Counting value with respect to an entity ID
- Counting totals and with respect to window times

Aggregation & Counting

- This is where we talk about Lambda
- There are many definitions
 - Common but not correct: Jobs that involve both Batch & Streaming
 - Correct: Perfect count is not possible with streaming so we use a combination of streaming and batch to show the right value

Aggregation & Counting

- Why is streaming not perfect

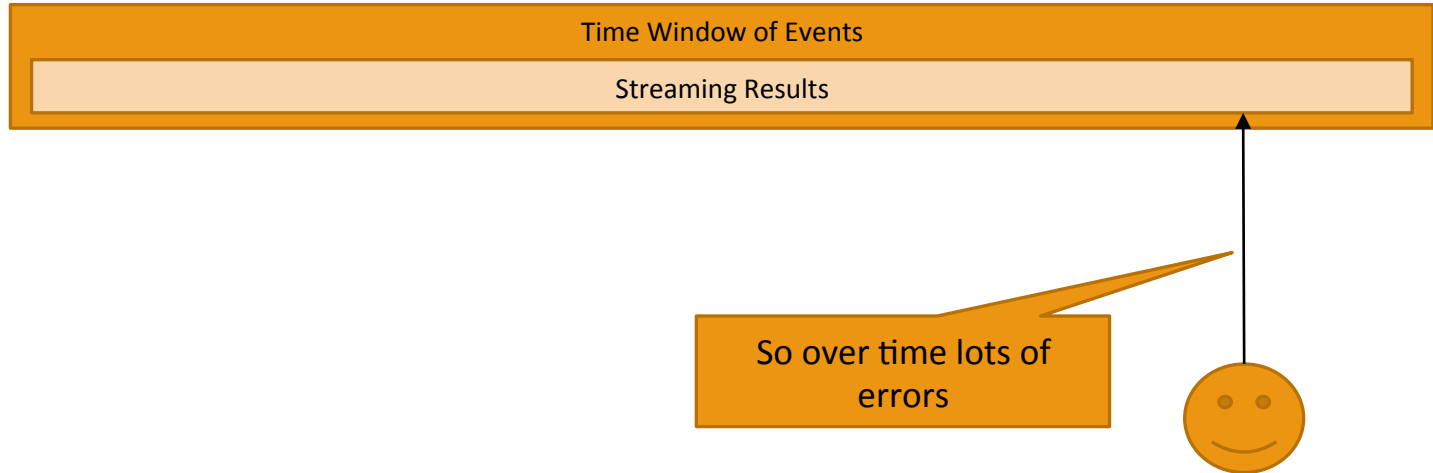


Aggregation & Counting

- Why is streaming not perfect (more)
 - Duping in general

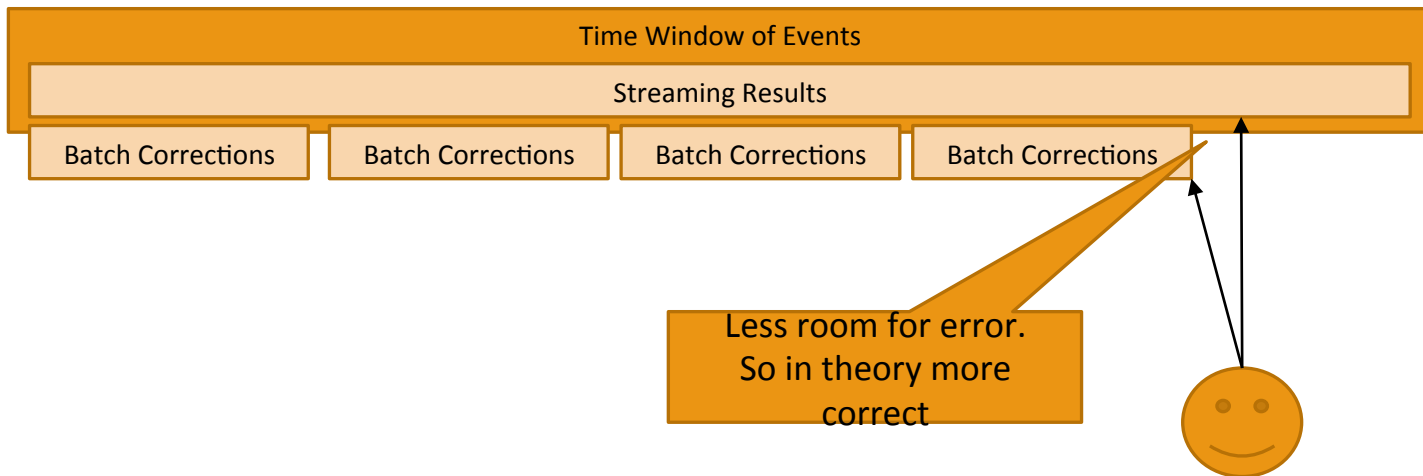
Aggregation & Counting

- What is Lambda



Aggregation & Counting

- What is Lambda

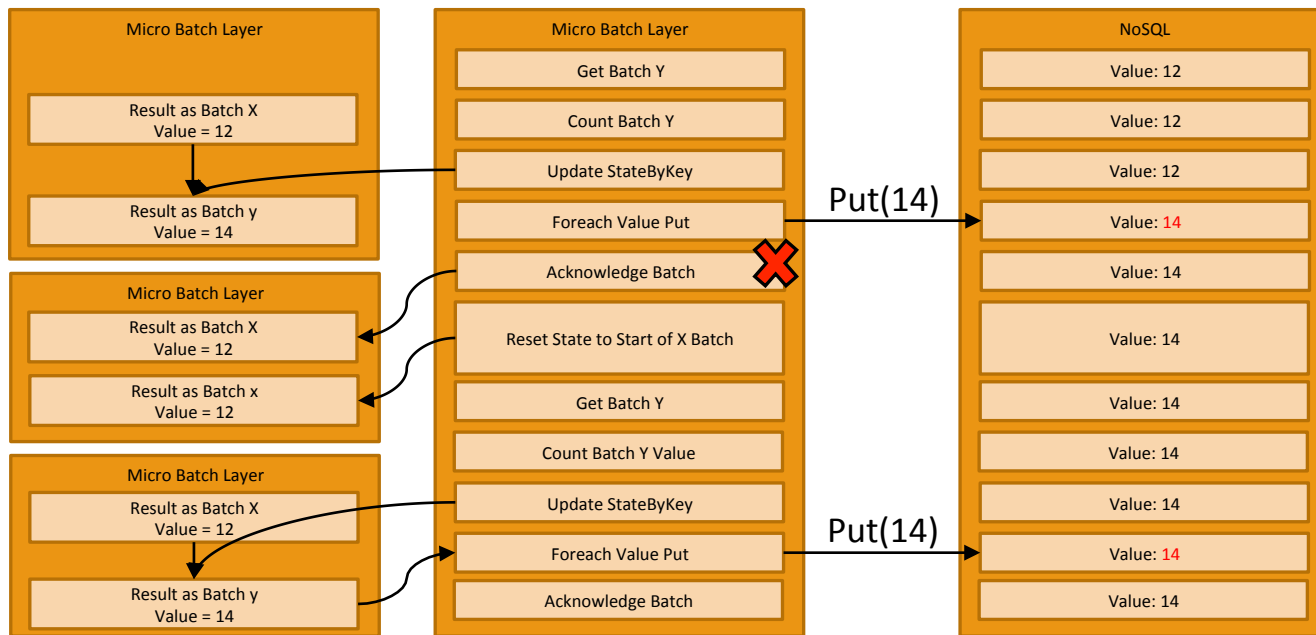


Aggregation & Counting

- How is lambda Resolved with out batch?
 - Failure problem
 - Deduping problem

Aggregation & Counting

- Failure Problem Solved by Adding internal State



Aggregation & Counting

- Deduping Problem Solved by
 - Some deduping can happen with in a batch
 - But this is not good enough
 - Source Sequence Numbers
 - With Order, Partitioning, and Internal State

Aggregation & Counting

- Deduping with Sequence Numbers



Source	Sequence	Value
A	1	10
B	1	100
A	2	10
B	2	100
A	3	10
B	2	100
B	3	100

Seq of A	Value of A	Seq of B	Value of B
1	10	-	-
1	10	1	100
2	20	1	100
2	20	2	200
3	30	2	200
3	30	2	200
3	30	3	300

Aggregation & Counting

- All the places in your pipe line where things can go wrong
- <image from streaming talk>

Aggregation & Counting

- Closing Thoughts on Lambda

Aggregation & Counting

- Why to do aggregation in a stream Vs aggregating at query time
 - When you know what you want to count
 - Which is more expensive

Aggregation & Counting

- Advantaged Aggregation Architecture Ideas
 - Pair with a time series database or NoSQL
 - Allow for pluggable aggregations