Distributed Time Series Analysis Framework For Spark

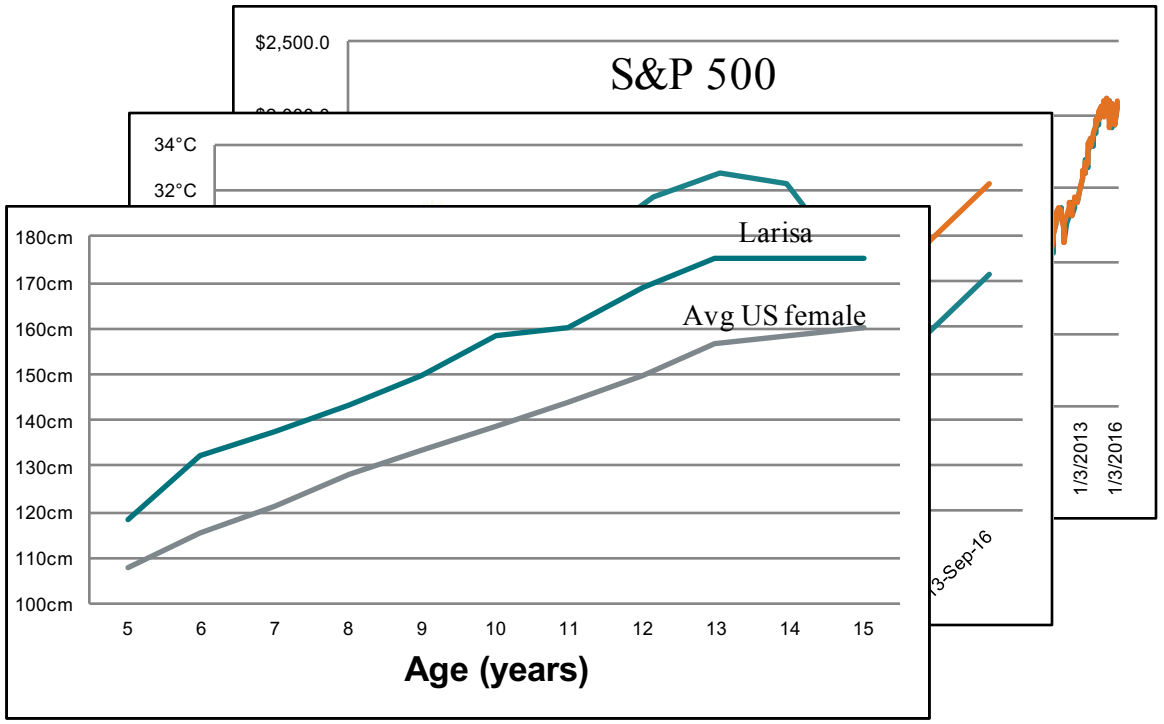
Larisa Sawyer
Two Sigma

Larisa Sawyer

TWO SIGMA

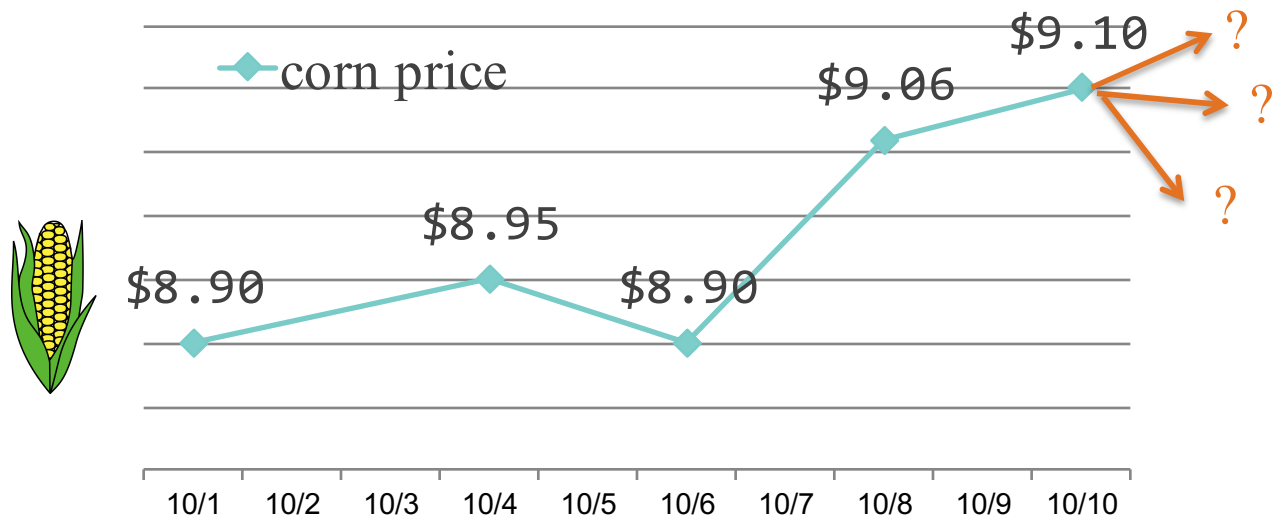
Time series examples

- Stock market prices
- Temperatures
- Height
- …

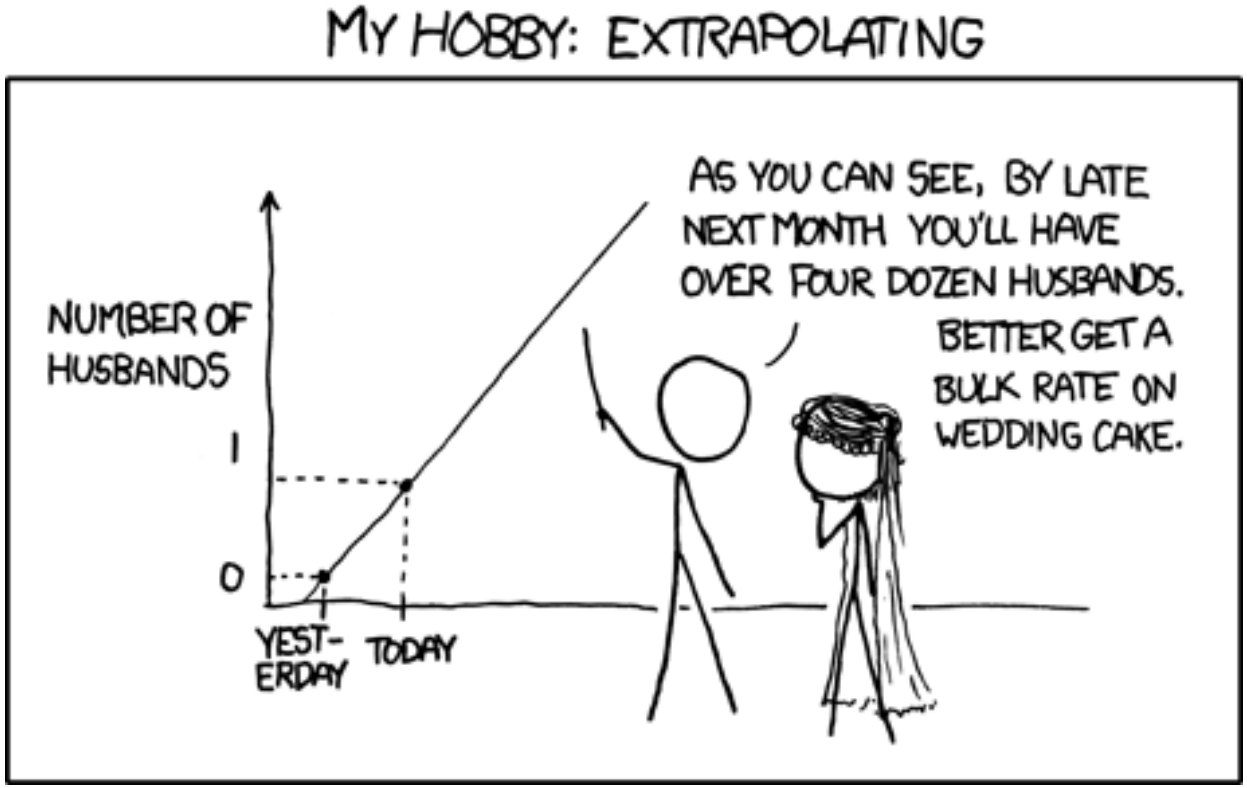


What do we do with time series data?

- Forecast future values given past observations

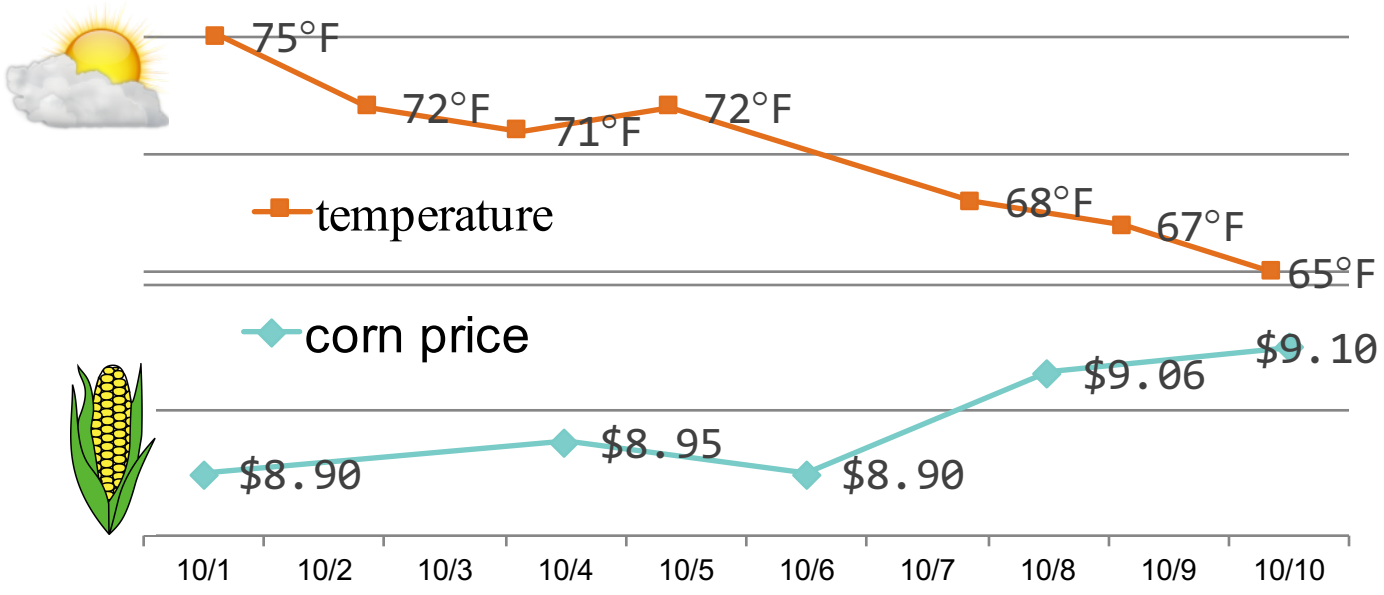


Univariate time series



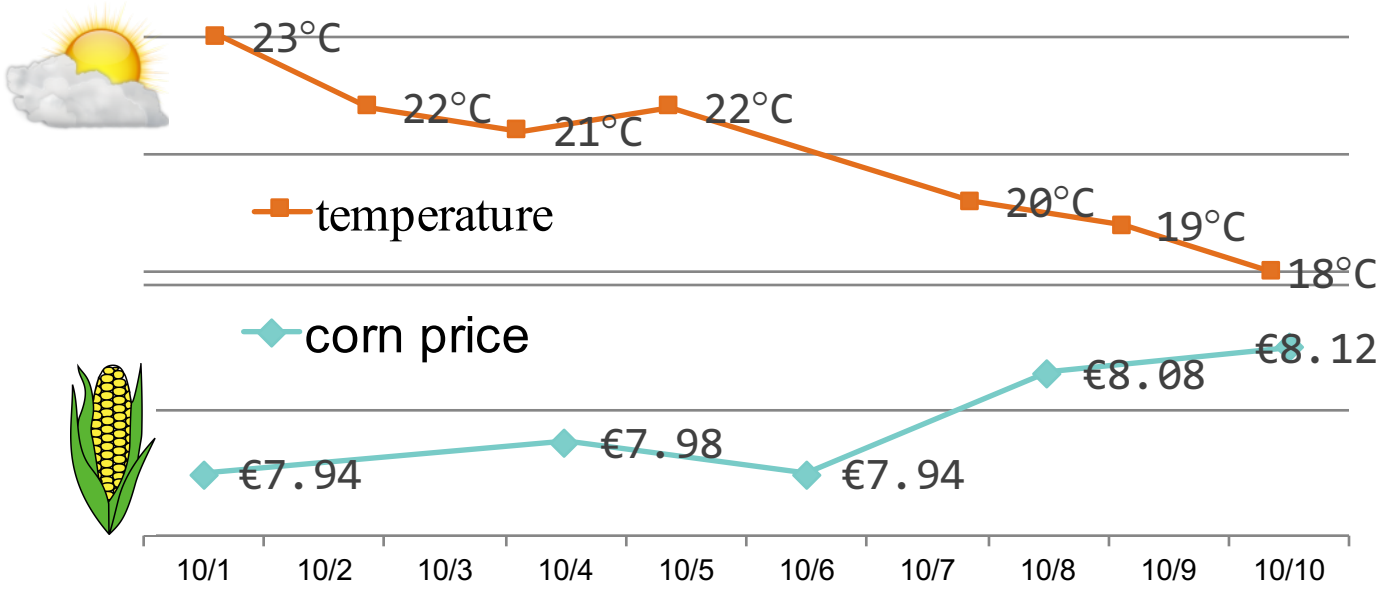
Multivariate time series

- We can forecast better by joining multiple time series
- Our framework enables fast distributed temporal join of large scale **unaligned** time series
- **Temporal join** is a fundamental operation for time series analysis



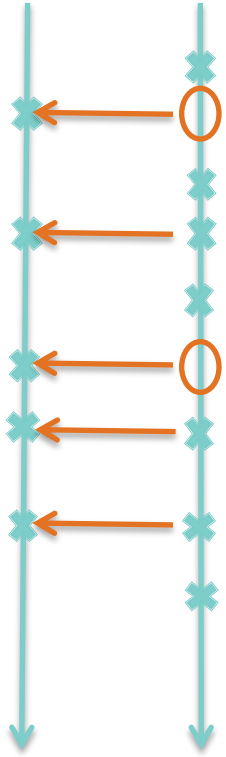
Multivariate time series

- We can forecast better by joining multiple time series
- Our framework enables fast distributed temporal join of large scale **unaligned** time series
- **Temporal join** is a fundamental operation for time series analysis



What is a left join?

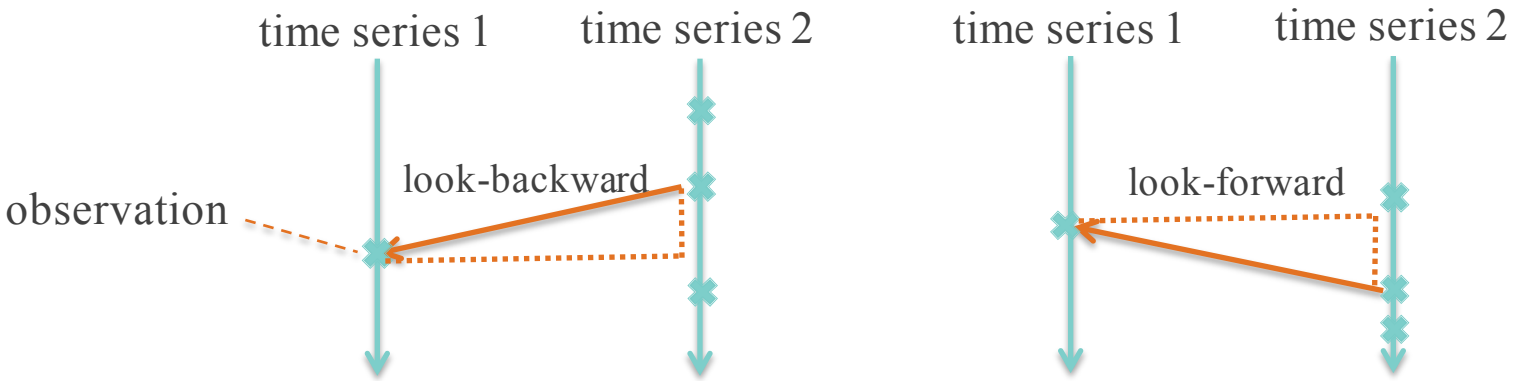
time series 1



time series 2

What is temporal join?

- A particular **join** function defined by a matching criteria over **time**
- Examples of criteria
  - **look-backward**
  - **look-forward**



Temporal join with look-backward criteria

time	tweets
08:00 AM	
10:00 AM	
12:00 PM	

time	BRK.A
08:00 AM	
11:00 AM	

Important Legal Information

The information presented here is offered for informational purposes only and should not be used for any other purpose (including, without limitation, the making of investment decisions). Examples provided herein are for illustrative purposes only and are not necessarily based on actual data. Nothing herein constitutes an offer to sell or the solicitation of any offer to buy any security or other interest. We consider this information to be confidential and not for redistribution or dissemination.

Temporal join with look-backward criteria

time	tweets
08:00 AM	
10:00 AM	
12:00 PM	

time	BRK.A
08:00 AM	
11:00 AM	

time	tweets	BRK.A
08:00 AM		
10:00 AM		
12:00 PM		

Temporal join with look-backward criteria

time	tweets
08:00 AM	
10:00 AM	
12:00 PM	

time	BRK.A
08:00 AM	
11:00 AM	

time	tweets	BRK.A
08:00 AM		
10:00 AM		
12:00 PM		

Temporal join with look-backward criteria

time	tweets
08:00 AM	
10:00 AM	
12:00 PM	

time	BRK.A
08:00 AM	
11:00 AM	

time	tweets	BRK.A
08:00 AM		
10:00 AM		
12:00 PM		

Temporal join with look-backward criteria

time	tweets
08:00 AM	
10:00 AM	
12:00 PM	

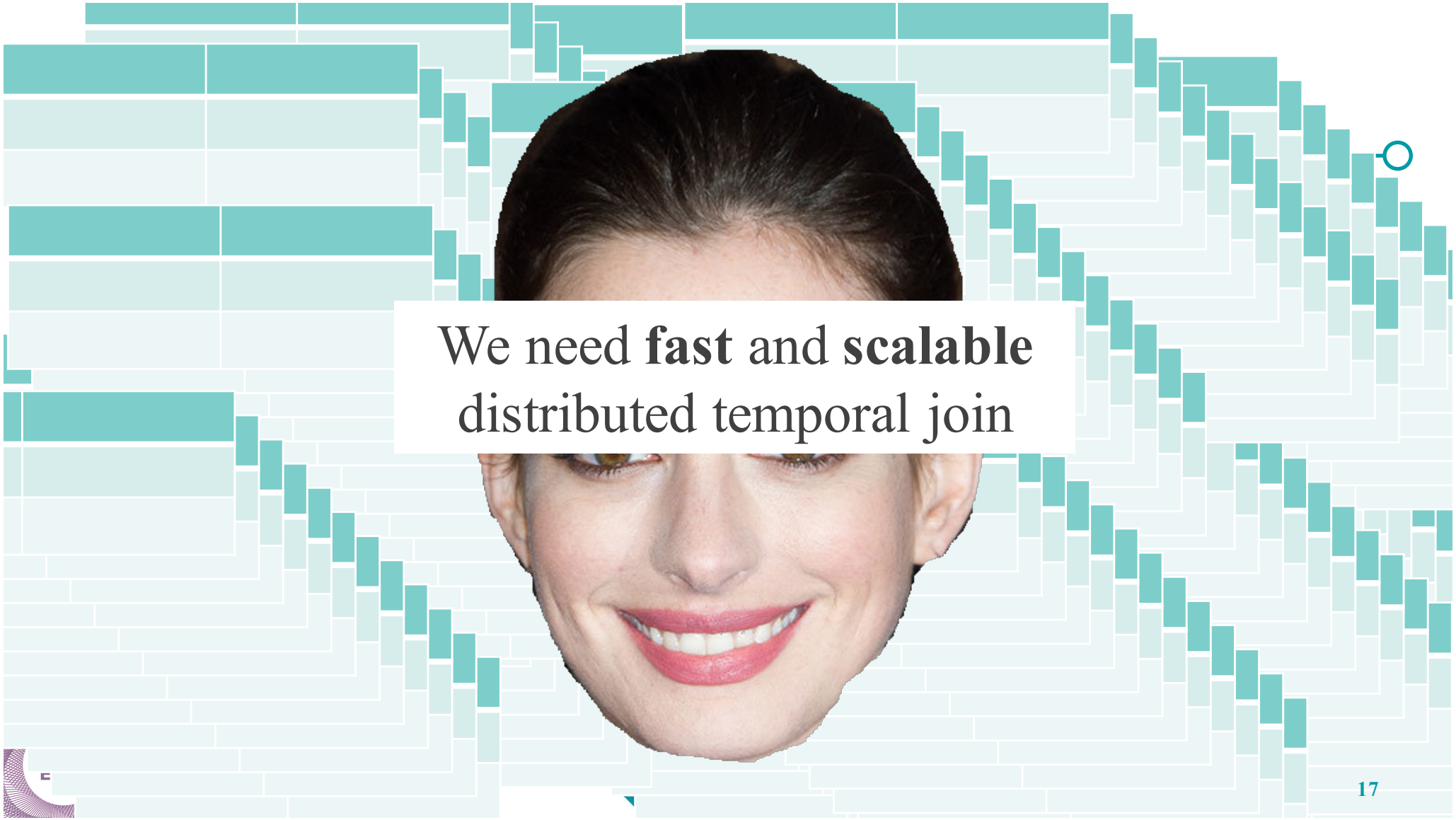
time	BRK.A
08:00 AM	
11:00 AM	

time	tweets	BRK.A
08:00 AM		
10:00 AM		
12:00 PM		

Temporal joins in practice

time	tweets
08:00 AM	
10:00 AM	
12:00 PM	

time	BRK.A
08:00 AM	
11:00 AM	


We need **fast** and **scalable**
distributed temporal join

Existing solutions

- Existing packages don't support temporal join or can't handle large time series
  - Pandas / R / Matlab
    - Limited to single machine
  - Spark
    - Does scale, but all data is unordered
  - spark-ts
    - Expects univariate time series to fit on single machine
    - Splits by col
    - Supports only snapshot data

Flint: A new time series library for Spark

- Goal
  - Provide a collection of functions to **manipulate** and **analyze** time series at scale
    - Group, temporal join, summarize, aggregate …
- How
  - Build a time series aware data structure
    - `TimeSeriesRDD extends RDD`
  - Optimize using temporal locality
    - Reduce shuffling
    - Reduce memory pressure by streaming

flint

What is a TimeSeriesRDD?

- `TimeSeriesRDD` vs `RDD`
  - Associate time range on each partition
  - Track partition time-ranges
  - Preserve temporal order

RDD

Raw Data

time	temperature
6:00 AM	60°F
6:01 AM	61°F
…	…
7:00 AM	70°F
7:01 AM	71°F
…	…
8:00 AM	80°F
8:01 AM	81°F
…	…

RDD

```
(6:00 AM, 60°F)
(6:01 AM, 61°F)
        …
(7:00 AM, 70°F)
(7:01 AM, 71°F)
        …
(6:58 AM, 64°F)
(6:59 AM, 65°F)
        …
(7:34 AM, 74°F)
(7:35 AM, 74°F)
        …
(7:58 AM, 76°F)
(7:59 AM, 77°F)
        …
(8:00 AM, 80°F)
(8:01 AM, 81°F)
        …
```

TimeSeriesRDD

Raw Data

time	temperature
6:00 AM	60°F
6:01 AM	61°F
…	…
7:00 AM	70°F
7:01 AM	71°F
…	…
8:00 AM	80°F
8:01 AM	81°F
…	…

RDD

(6:00 AM, 60°F)
(6:01 AM, 61°F)
…

(7:00 AM, 70°F)
(7:01 AM, 71°F)
…

(6:58 AM, 64°F)
(6:59 AM, 65°F)
…

(7:34 AM, 74°F)
(7:35 AM, 74°F)
…

(7:58 AM, 76°F)
(7:59 AM, 77°F)
…

(8:00 AM, 80°F)
(8:01 AM, 81°F)
…

TSRDD

(6:00 AM, 60°F)
(6:01 AM, 61°F)
…

[06:00 AM, 07:00 AM)

(7:00 AM, 70°F)
(7:01 AM, 71°F)
…

[07:00 AM, 8:00 AM)

(8:00 AM, 80°F)
(8:01 AM, 81°F)
…

[8:00 AM, ∞)

Group function

- A **group** function groups rows with exactly the same timestamps

time	city	temperature	
1:00 PM	New York	70°F	group 1
1:00 PM	Brussels	60°F	
2:00 PM	New York	71°F	group 2
2:00 PM	Brussels	61°F	
3:00 PM	New York	72°F	group 3
3:00 PM	Brussels	62°F	
4:00 PM	New York	73°F	group 4
4:00 PM	Brussels	63°F	

Group function

- A **group** function groups rows with nearby timestamps

time	city	temperature
1:00 PM	New York	70°F
1:00 PM	Brussels	60°F
2:00 PM	New York	71°F
2:00 PM	Brussels	61°F
3:00 PM	New York	72°F
3:00 PM	Brussels	62°F
4:00 PM	New York	73°F
4:00 PM	Brussels	63°F

group 1

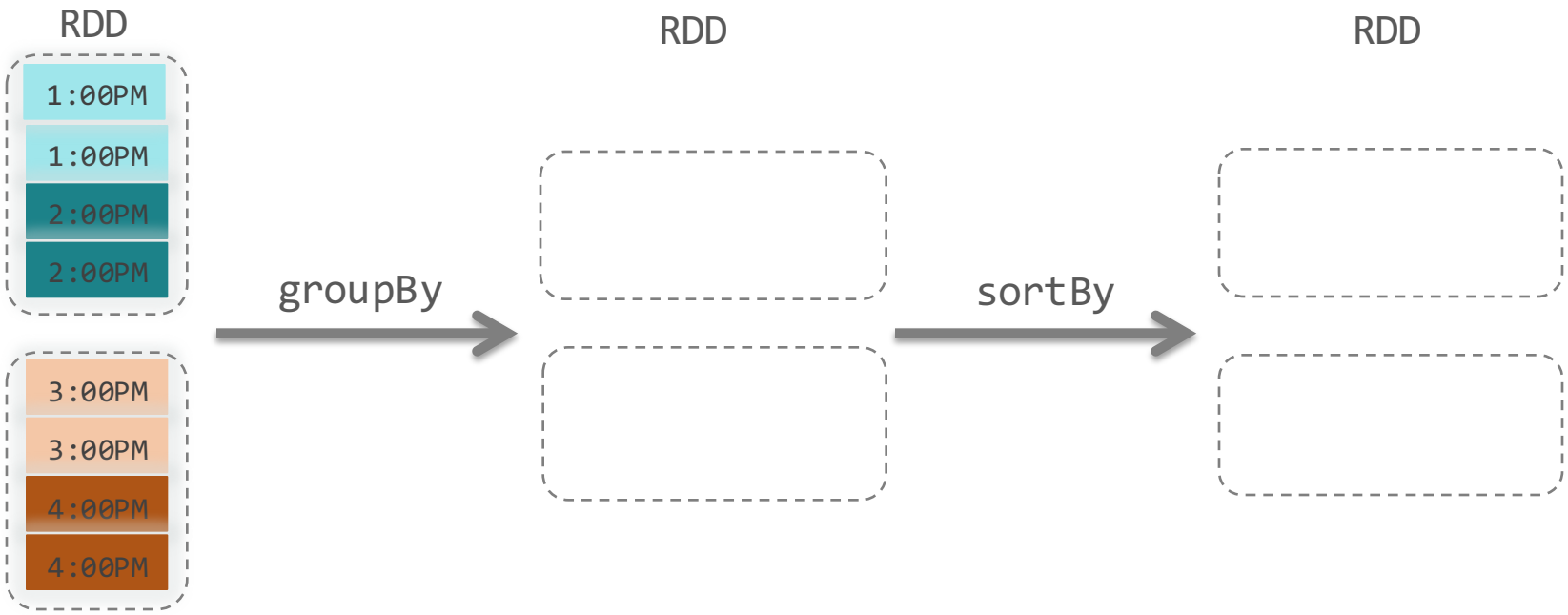
group 2

Group in Spark

- Groups rows with exactly the same timestamps

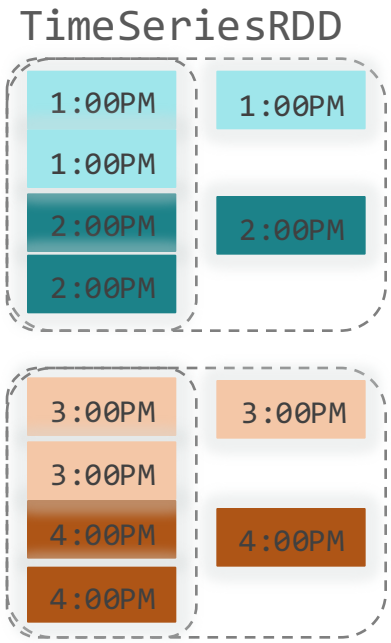
Group in Spark

- Back to Temporal Order
- Data is shuffled and materialized on the workers
- Temporal order is not preserved



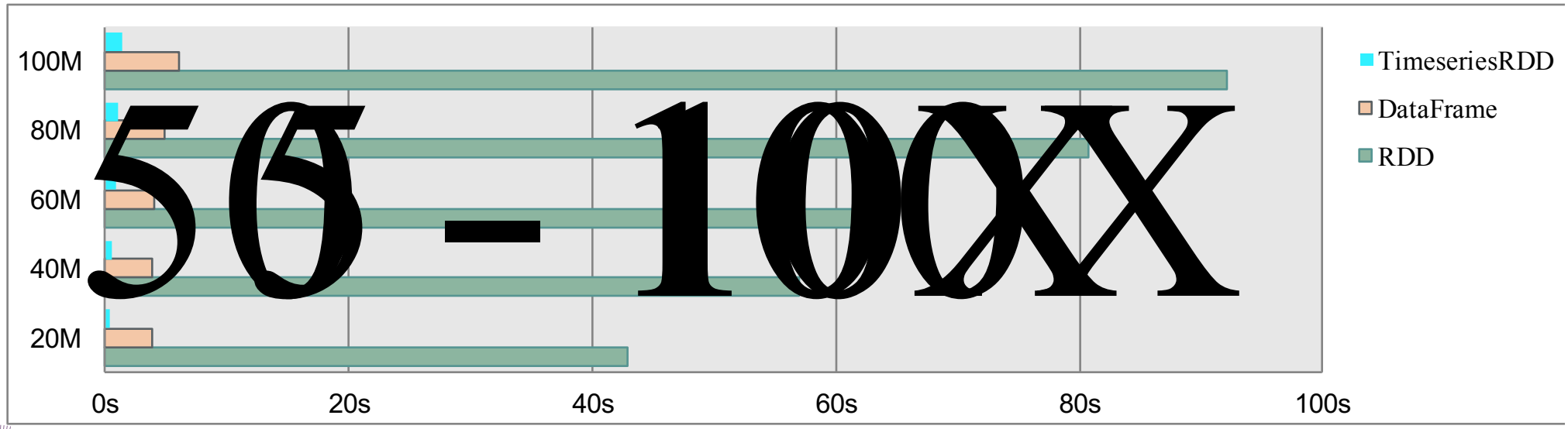
Group in `TimeSeriesRDD`

- Data is grouped per partition locally as streams



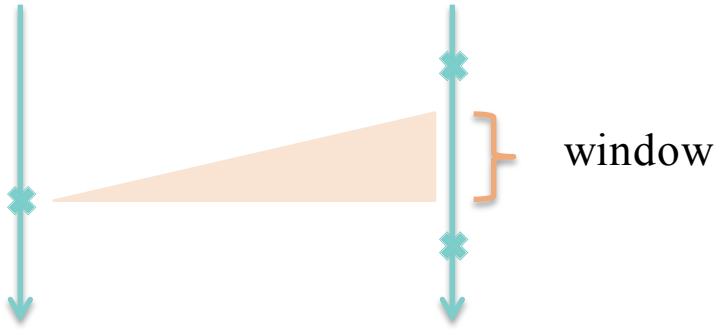
Benchmark for group + count

- Running time of **count** after **group**
  - 16 executors (10G memory and 4 cores per executor)
  - Data read from HDFS



Temporal join

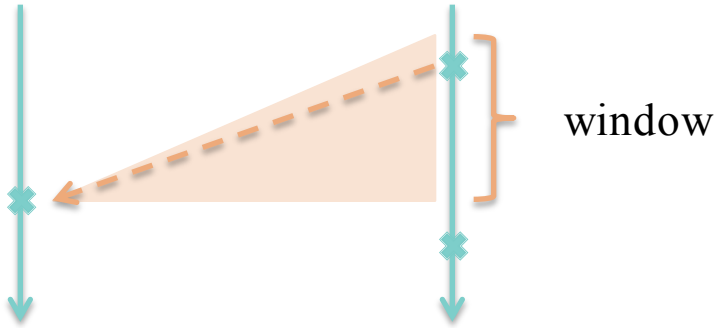
- A **temporal join** function is defined by a matching criteria over **time**
- A typical matching criteria has two parameters
  - **direction** – look-backward or look-forward
  - **window** – how much to look-backward or look-forward



look-backward temporal join

Temporal join

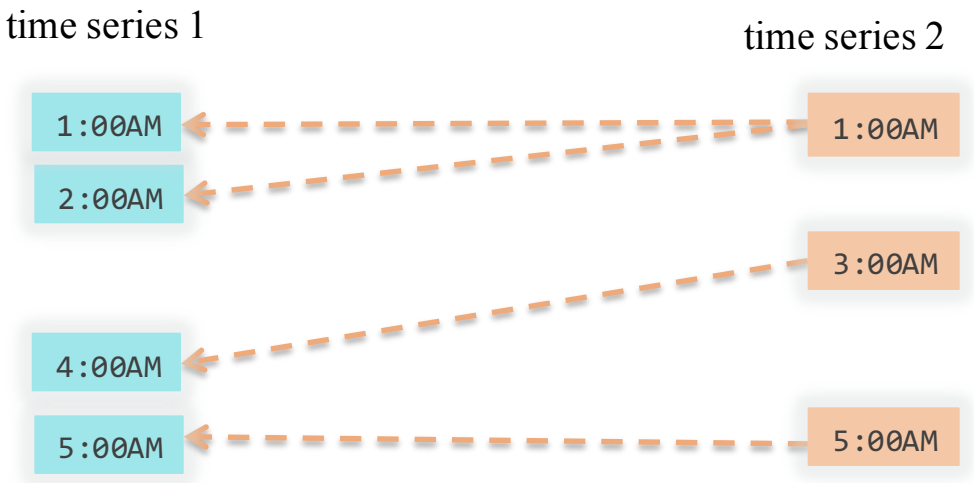
- A **temporal join** function is defined by a matching criteria over **time**
- A typical matching criteria has two parameters
  - **direction** – look-backward or look-forward
  - **window** – how much to look-backward or look-forward



look-backward temporal join

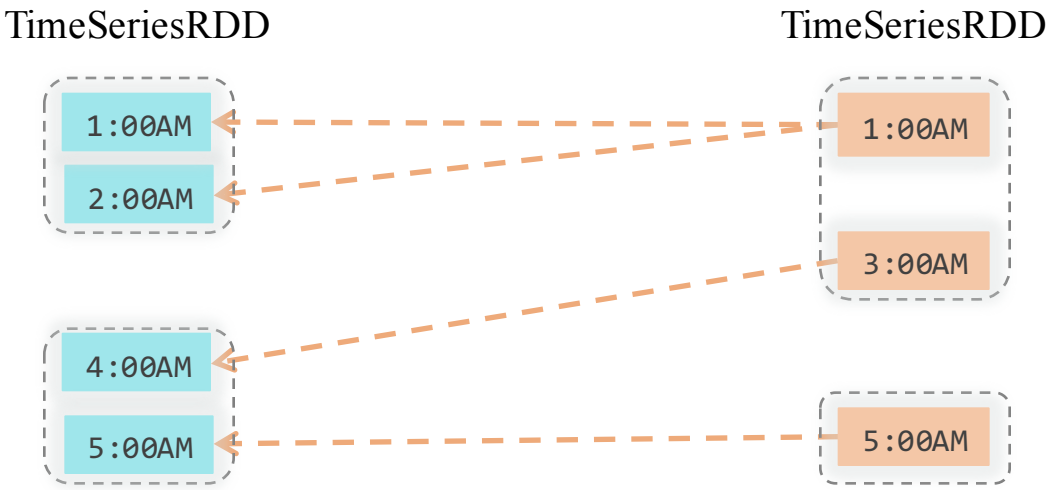
Temporal join

- Temporal join with criteria look-back and window of 1 hour



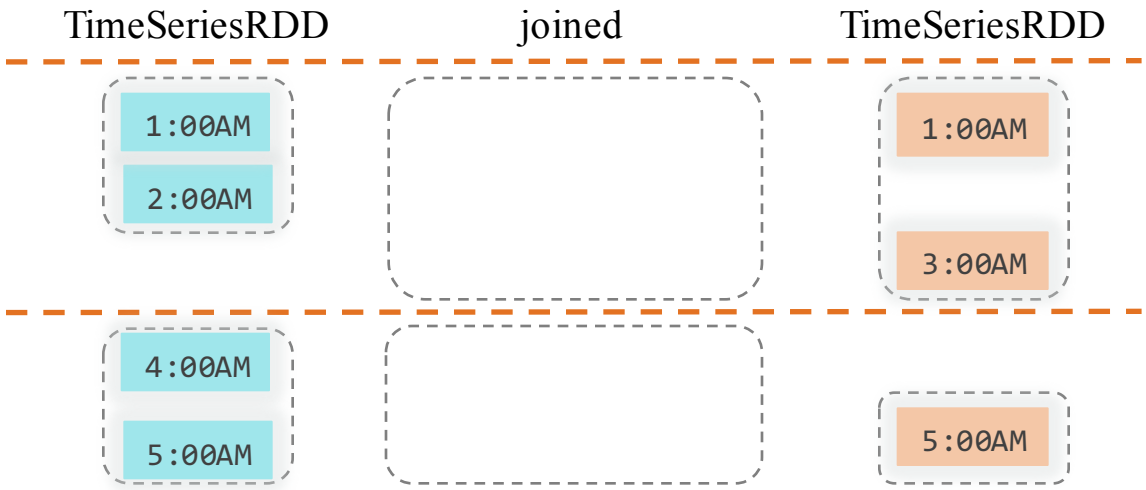
Temporal join

- Temporal join with criteria look-back and window of 1 hour
  - How do we do temporal join in TimeSeriesRDD?



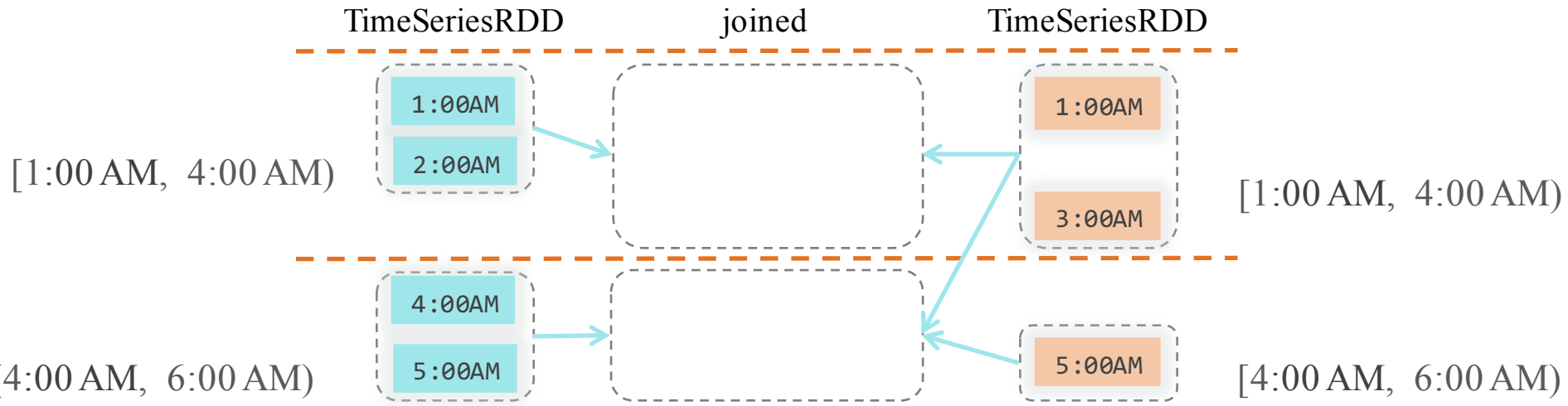
Temporal join in TimeSeriesRDD

- Temporal join with criteria look-back and window of 1 hour
  - partition **time** space into disjoint intervals



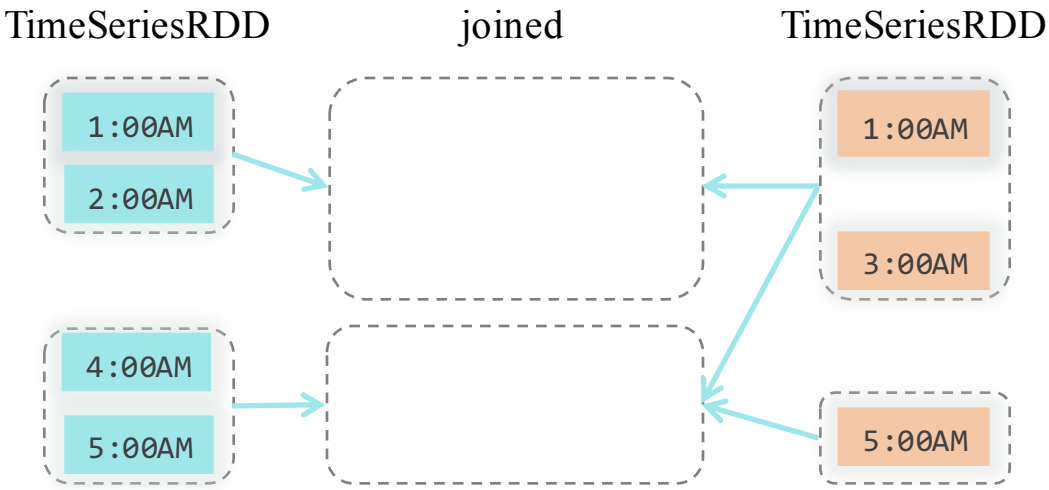
Temporal join in TimeSeriesRDD

- Temporal join with criteria look-back and window of 1 hour
  - Build dependency graph for the joined TimeSeriesRDD



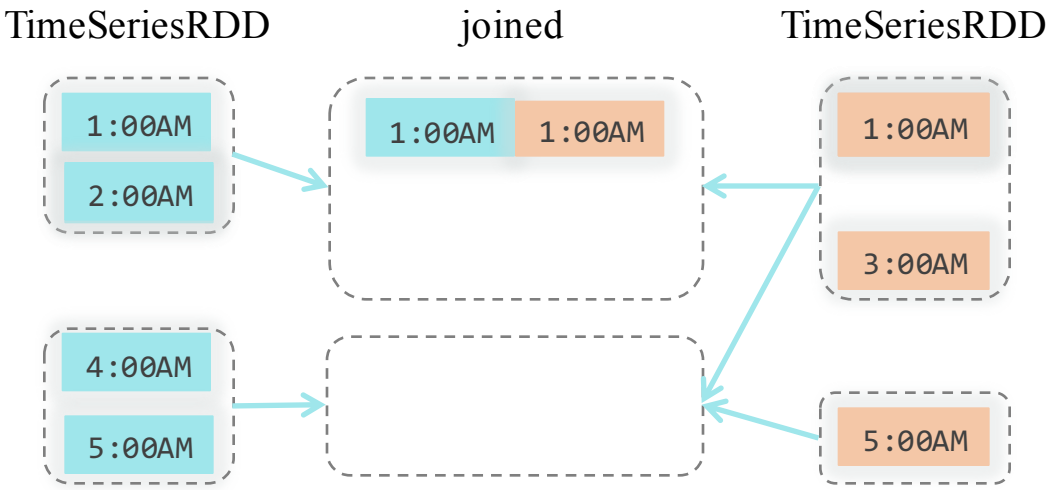
Temporal join in TimeSeriesRDD

- Temporal join with criteria look-back and window of 1 hour
  - Join data as streams per partition



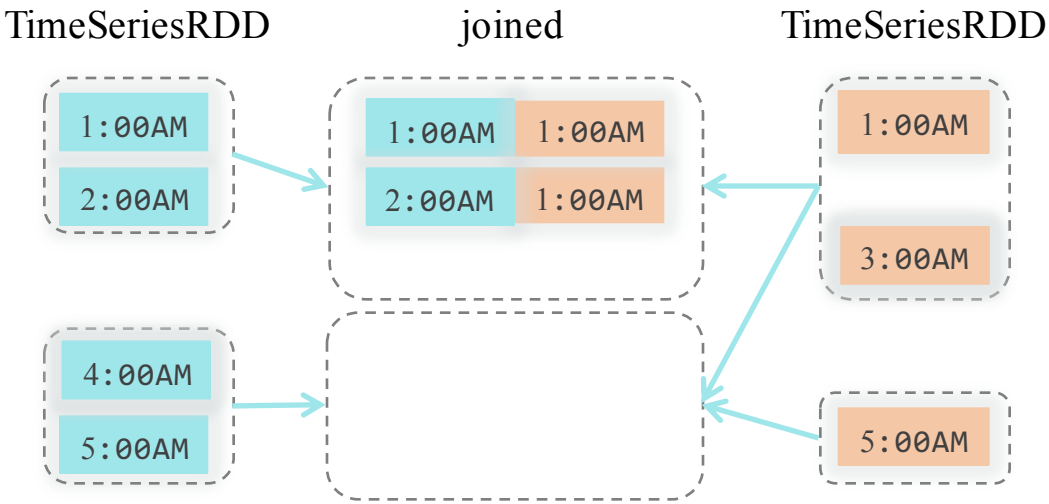
Temporal join in TimeSeriesRDD

- Temporal join with criteria look-back and window of 1 hour
  - Join data as streams



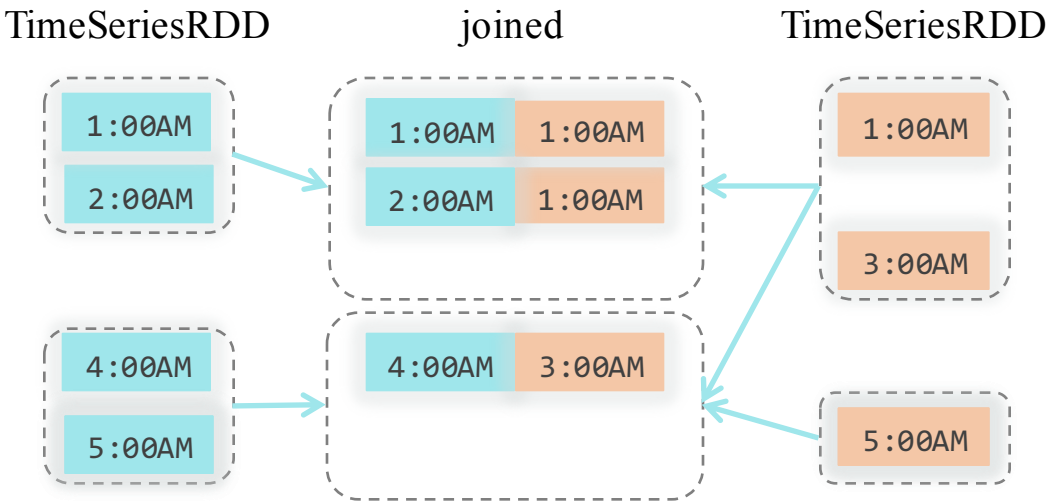
Temporal join in TimeSeriesRDD

- Temporal join with criteria look-back and window of 1 hour
  - Join data as streams



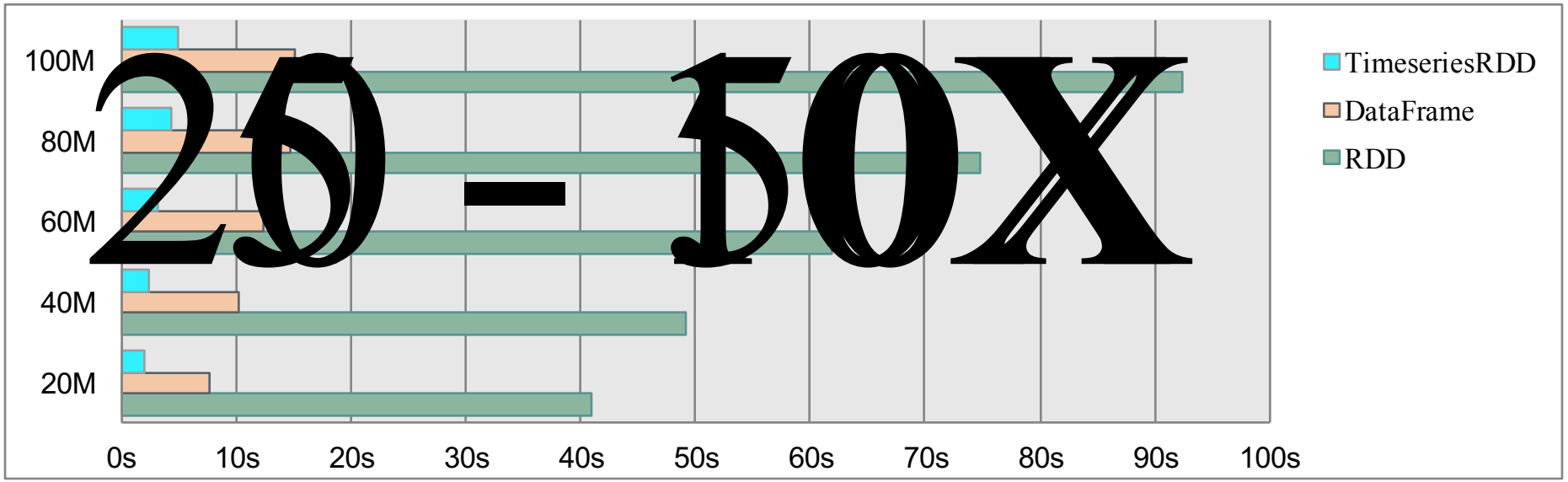
Temporal join in TimeSeriesRDD

- Temporal join with criteria look-back and window of 1 hour
  - Join data as streams



Benchmark for temporal join + count

- Running time of **count** after **temporal join**
  - 16 executors (10G memory and 4 cores per executor)
  - Data read from HDFS



Functions over TimeSeriesRDD

- Grouping functions
- Temporal joins such as look-forward, look-backward etc.
- Summarizers such as average, variance, z-score etc. over grouping functions

Open Source

- True!
- https://github.com/twosigma/flint

flint

What's next?

- TimeSeriesDataframe / TimeSeriesDataset
- Speed up
  - Richer APIs
- Python bindings
- Additional summarizers

Key contributors

- Christopher Aycock
- Yuri Bogomolov
- Jonathan Coveney
- Li Jin
- David Medina
- Julia Meinwald
- David Palaitis
- Larisa Sawyer
- Leif Walsh
- Wenbo Zhao

flint

Flint: Time Series For Spark

A library to solve for general time series analysis operations at massive scale

Anne Hathaway has nothing to do with Berkshire Hathaway

Check it out in open source, and contribute

https://github.com/twosigma/flint

flint

THANK YOU.

Larisa.Sawyer@twosigma.com

SPARK SUMMIT EUROPE 2016