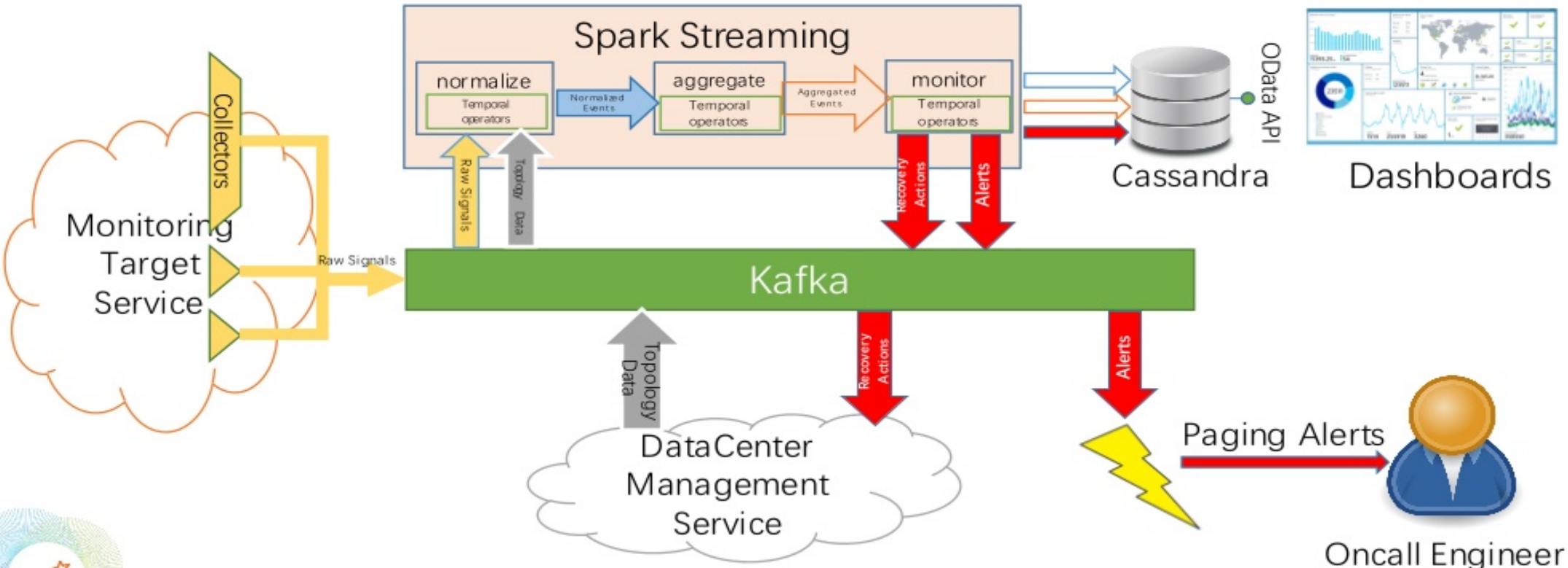
TEMPORAL OPERATORS FOR SPARK STREAMING AND ITS APPLICATION FOR OFFICE365 SERVICE MONITORING

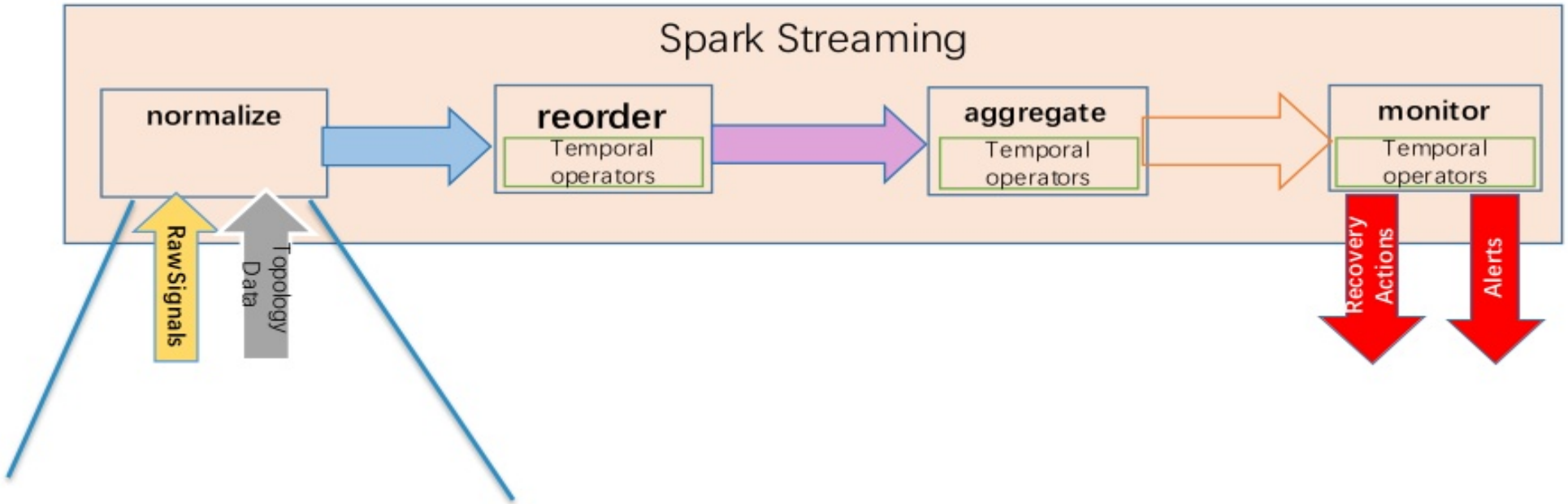
Jin Li, Wesley Miao
Microsoft

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

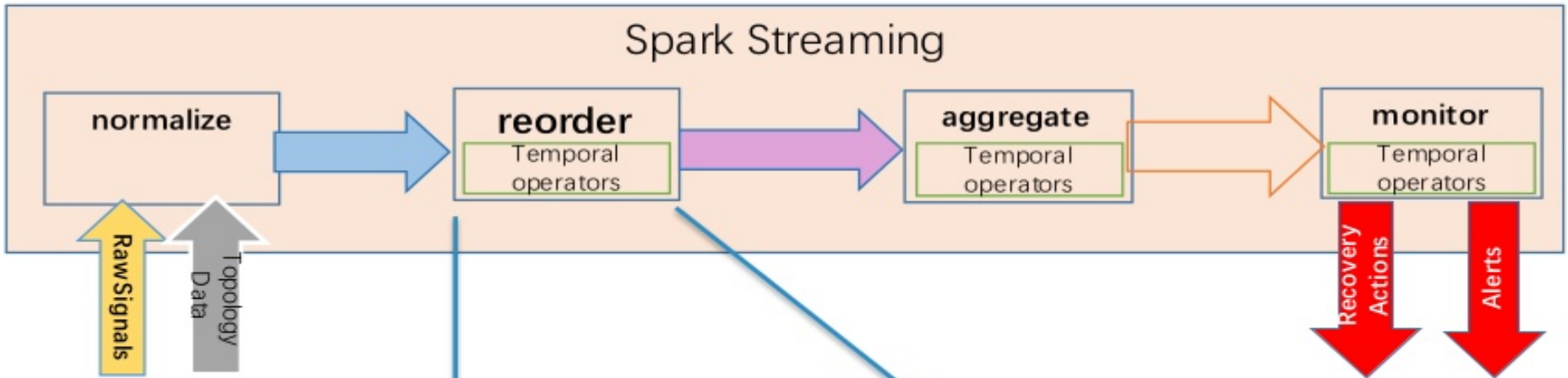

Spark Streaming
normalize
Temporal operators
Normalized Events
aggregate
Temporal operators
Aggregated Events
monitor
Temporal operators
OData API
Cassandra
Dashboards
Collectors
Monitoring Target Service
Raw Signals
Topology Data
Recovery Actions
Alerts
Kafka
Topology Data
Recovery Actions
Alerts
DataCenter Management Service
Paging Alerts
Oncall Engineer

Office365 Service Monitoring

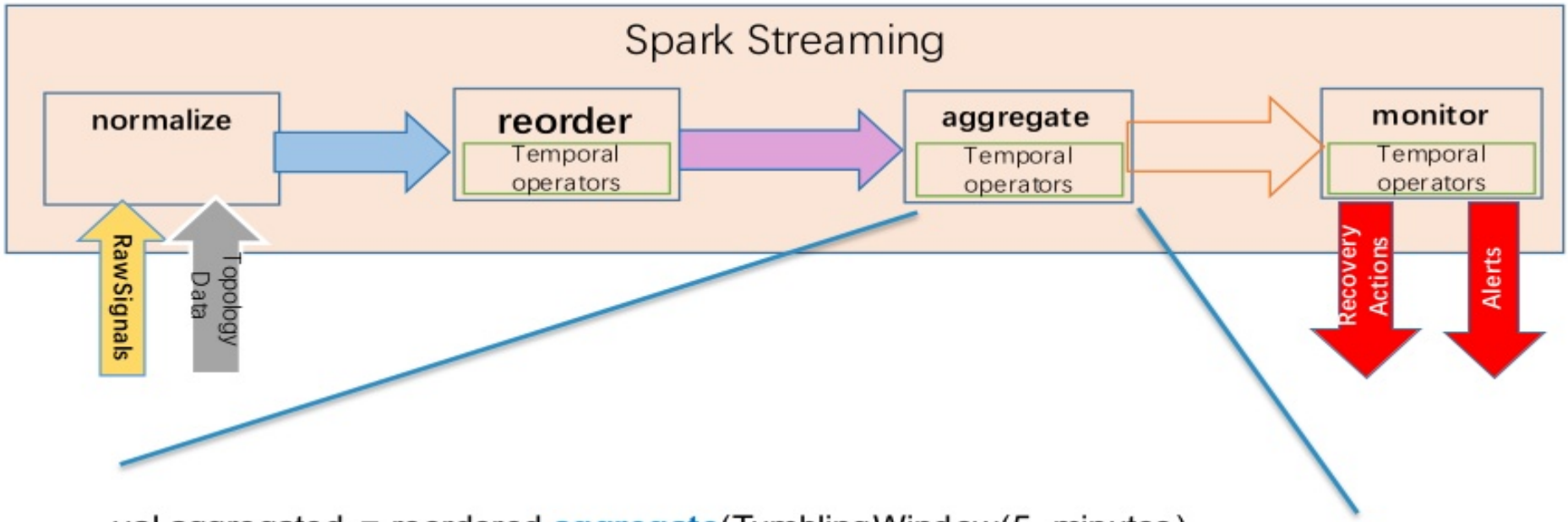
- Service Monitoring Signals
  - Local active monitoring
  - Schema: <TopologyScopeValue, IsSuccess, …>
  - TopologyScopeValue: grouping of hosts
    - e.g. server, rack, site, region
- Data may arrive out of order
- Application logic defined on data time



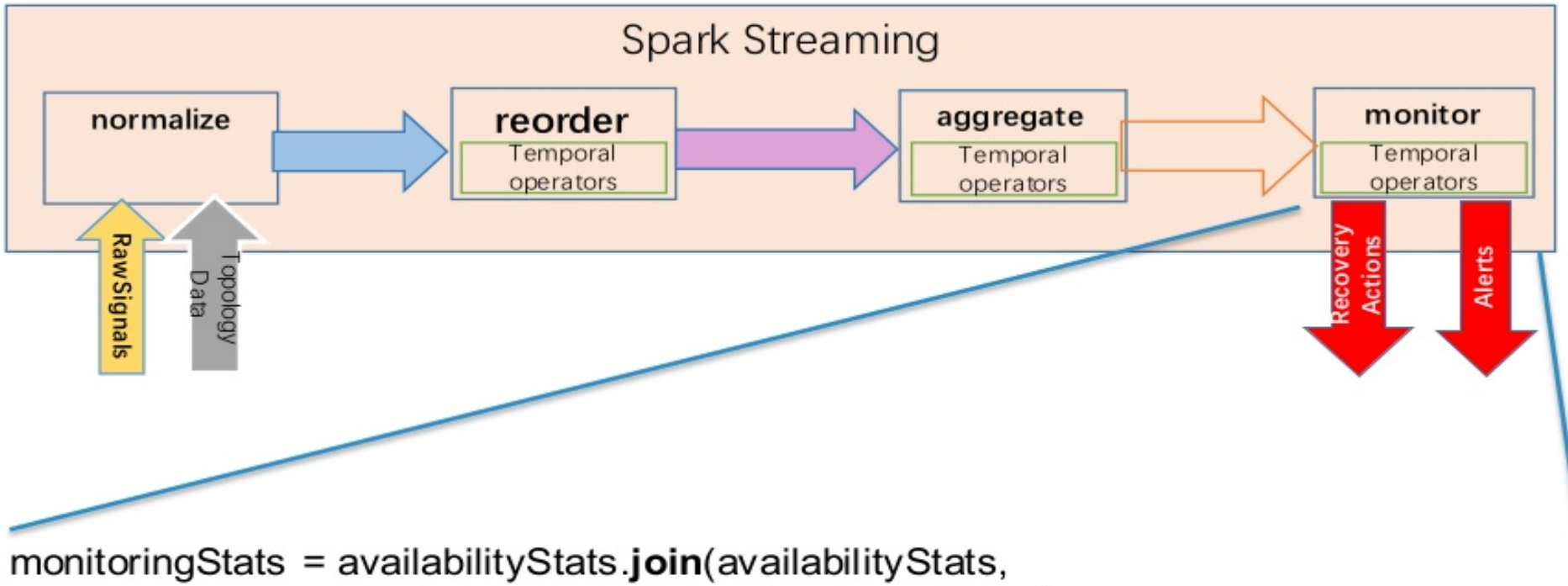

```
val rawSignals : DStream[T]
val topologyData:  DStream[T1]

val normalized =
rawSignals.join(topologyData,  …)
```




```
val reordered =
  normalized.reorder(Policy.ReorderAdjustAndDrop(Seconds(60),  Seconds(120)))
```




```
val aggregated = reordered.aggregate(TumblingWindow(5 minutes),
  groupBy(event.rack, event.site, event.region),
  Sum(event => event.isSuccess),
  Count(),
  Avg(event => event.latency)
)
val availabilityStats = aggregated.map{...}
```

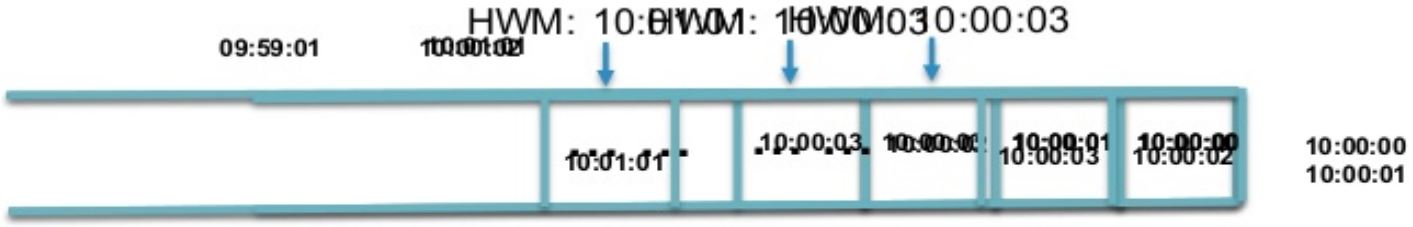



```
val monitoringStats = availabilityStats.join(availabilityStats,
                      left => JoinKey(left.topologyScopeValue),
                      right => JoinKey(right.topologyScopeValue),
                      (left, right) =>  left.availability < 0.99 &&
                                        right.availability >= 0.99
                      TimeDiff(Seconds(-300),  Seconds(0))
```

Temporal operators in depth

Temporal Operators – Reorder

- Reorder(policy)
  - inputStream
    .map { e => **StreamEvent(e.timestamp, e)** }
    .reorder(Policy.Reorder(Seconds(60)))



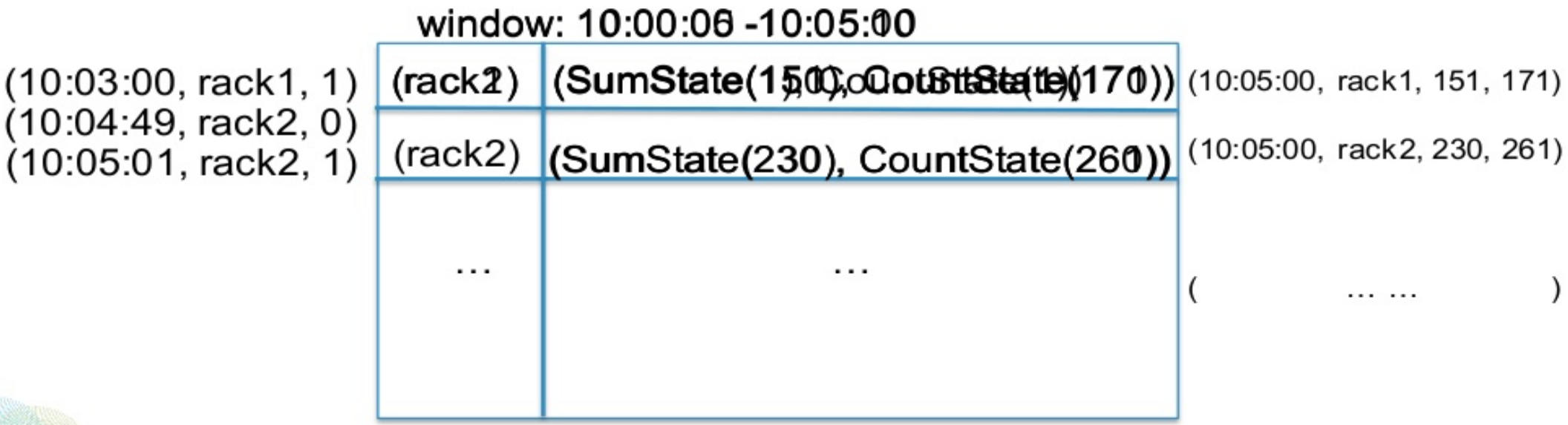
Temporal Operators – Event-time window aggregate

- Basic Availability Metric
  - SuccessEvents / TotalEvents for the every 5 minutes
    - Tumbling window sum
- Event-time window aggregate

```
val availabilityMetrics = input
.map { e => StreamEvent(e.timestamp, e) }
.reorder(Policy.ReorderAdjustAndDrop(Seconds(60), Seconds(120)))
.aggregate(
  TumblingWindow(Seconds(300)),
  event => GroupByKey((event.topologyScopeValue, "topologyScopeValue")),
  Sum(event => event.isSuccess, "sumOfSuccessEvent"),
  Count(event => event, "sumOfTotalEvent")
)
.map (...)
```

Temporal Operators – Event-time window aggregate

- Implementation



Temporal Operators – Join

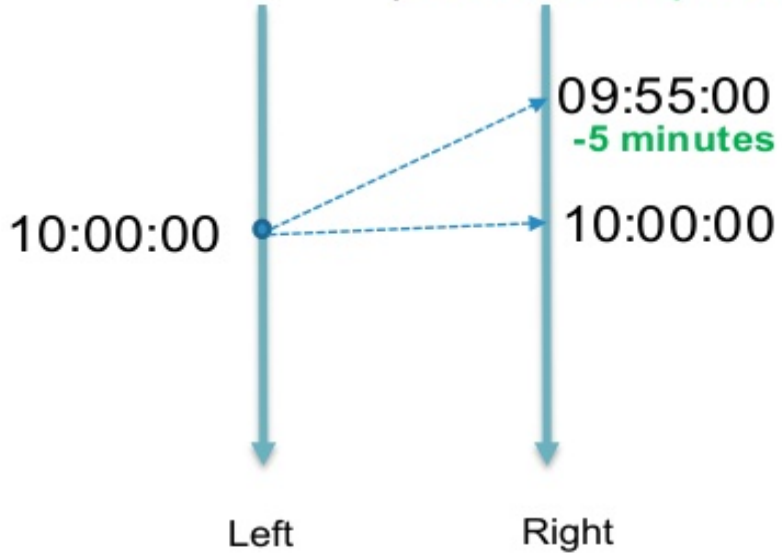
- Alarm
  - SuccessEvents/TotalEvents > threshold for the current 5 minutes, but not the previous 5 minutes
    - Temporal self join

- Temporal Join

```
availabilityMetrics.join(
 availabilityMetrics,
 left => JoinKey(x.topologyScopeValue, "leftTopologyScopeValue"),
 right => JoinKey(y.topologyScopeValue, "rightTopologyScopeValue"),
 (left, right) =>  left.availability < 0.99 && right.availability >= 0.99
 TimeDiff(Seconds(-300), Seconds(0))
)
```

Temporal Operators – Join

- Temporal condition for Join
  - TimeDiff(Seconds(-300), Seconds(0))



THANK YOU.

Jin Li (jliin@microsoft.com)

Wesley Miao (wemia@Microsoft.com)

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Questions?