



Building Structured Streaming Connector for Continuous Applications

Arijit Tarafdar

Nan Zhu

Azure HDInsight @ Microsoft

Who Are We?

- Nan Zhu
 - Software Engineer@Azure HDInsight. Work on Spark Streaming/Structured Streaming service in Azure. Committee Member of XGBoost@DMLC and Apache MxNet (incubator). Spark Contributor.
- Arijit Tarafdar
 - Software Engineer@Azure HDInsight. Work on Spark/Spark Streaming on Azure. Previously worked with other distributed platforms like DryadLinq and MPI. Also worked on graph coloring algorithms which was contributed to ADOL-C (<https://projects.coin-or.org/ADOL-C>).

Agenda

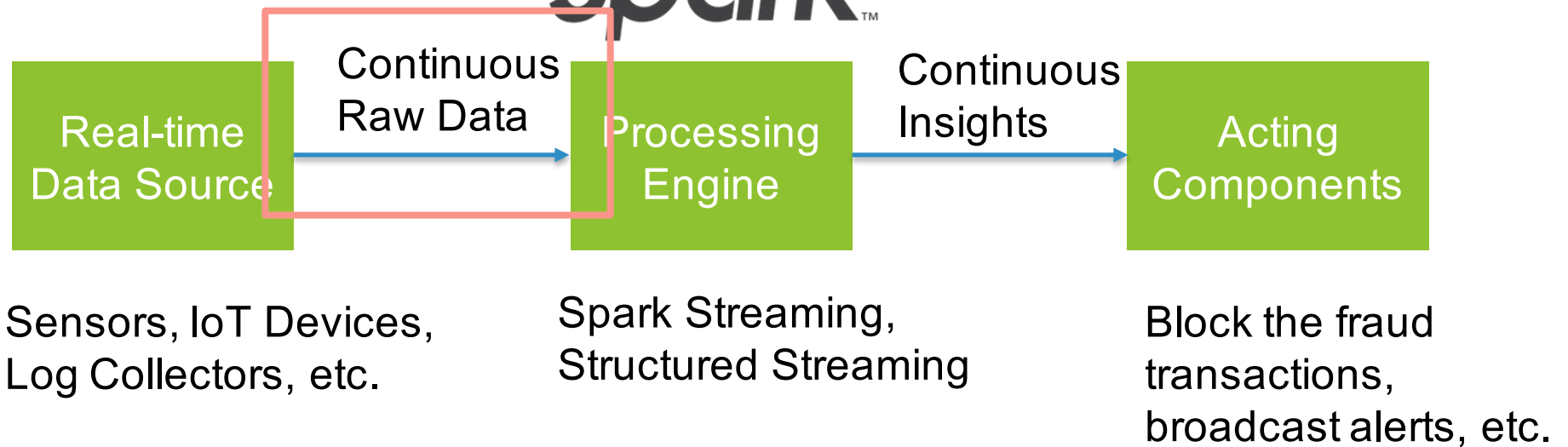
- What is/Why Continuous Application?
 - What are the challenges when we already have Spark in hand?
- Introduction of Azure Event Hubs and Structured Streaming
- Key Design Considerations in Building Structured Streaming Connector
- Test Structured Streaming Connector
- Summary

Continuous Application Architecture and Role of Spark Connectors (1)

- Not only **size** of data is increasing, but also the **velocity** of data
 - Sensors, IoT devices, social networks and online transactions are all generating data that needs to be **monitored constantly and acted upon quickly.**

Continuous Application Architecture and Role of Spark Connectors (2)

How to connect
Spark and Real-time
Data Sources?

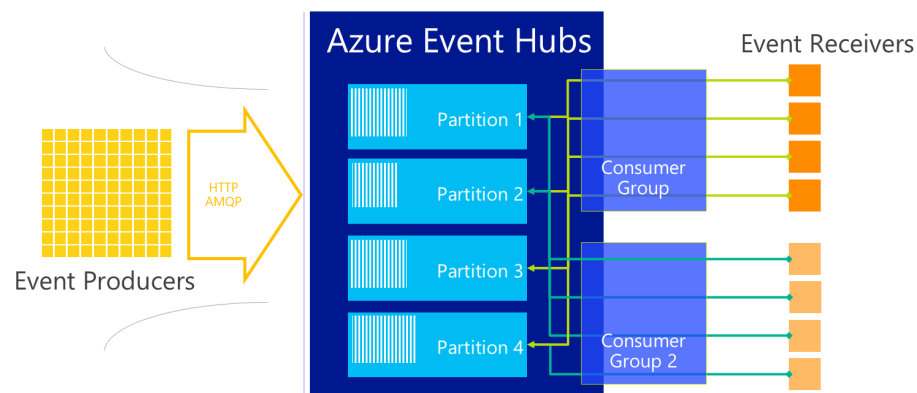


Building a Connector for *Real-time Data Source* and *Structured Streaming*

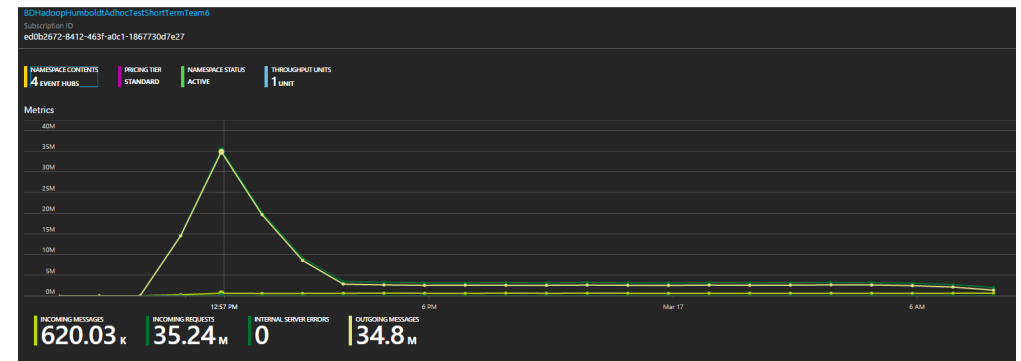
taking *Azure Event Hubs* as an example
(Comparing with Kafka Connector)

What is Azure Event Hubs?

Event Hubs conceptual architecture



Platform as a Service



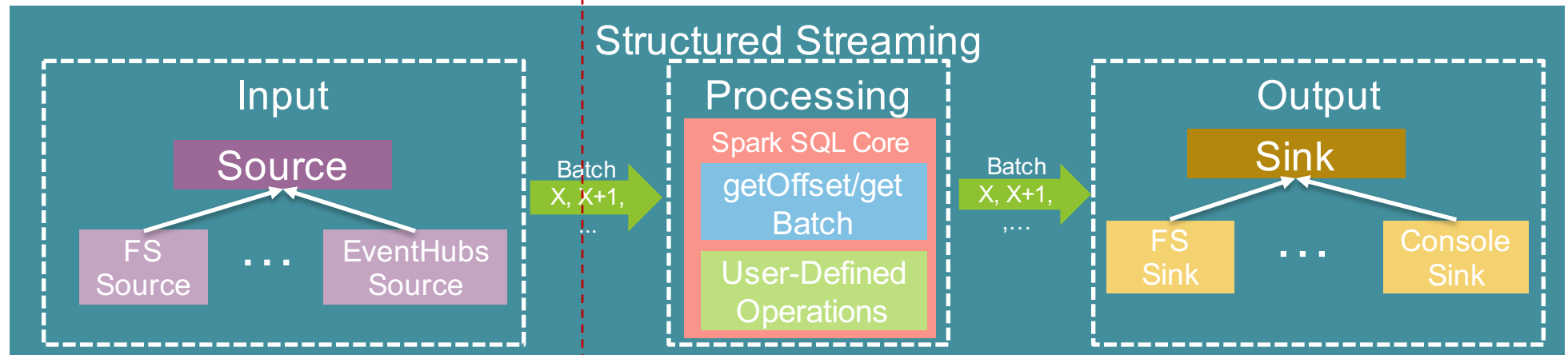
What is Structured Streaming?

- A relatively new component in Spark
 - Introduced in Spark 2.0
 - Keep evolving from Spark 2.0 – 2.2
- Streaming Analytic Engine for Structured Data
 - Sharing the same set of APIs with Spark SQL
 - Batching Engine
 - Running computation over continuously arriving data and keep updating results

Abstraction in Structured Streaming

(Spark 2.1
Implementation)

Still follow Micro-Batch streaming model



Describe Data Source:

1. **getOffset:** offset of the last message to be processed in next batch
2. **getBatch:** build DataFrame for the next batch

Define Data Transforming Task with Spark SQL APIs (Structured Streaming Internal)

Describe Data Sink

1. **addBatch:** evaluate DataFrame and save data to target system

Implementation of a “Source”

	Description	Implementation
getOffset()	Return the last offset of next batch	<i>EndOffsetOfLastBatch</i> “+” RateControlFunc(...)
getBatch(startOffset, endOffset)	Build DataFrame for next Batch	SS internal passes in startOffset and endOffset (results of getOffset)

1. How to represent ***Offset***
2. How to define ***Rate*** (To avoid tracking size of each message, rate is usually defined as # of messages)

Implementation of a “Source”

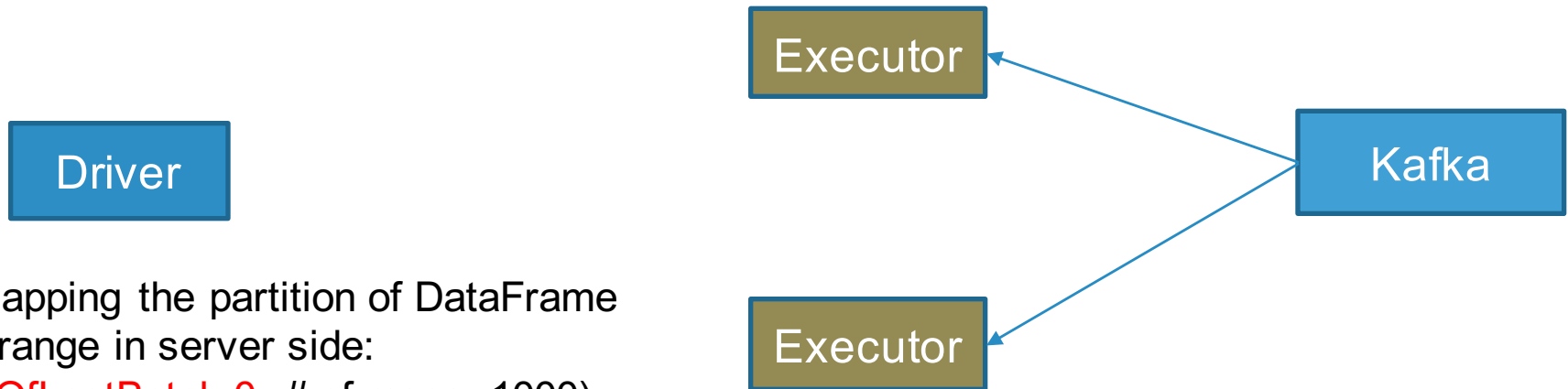
	Description	Implementation
getOffset()	Return the last offset of next batch	<i>EndOffsetOfLastBatch</i> “+” RateControlFunc(...)
getBatch(startOffset, endOffset)	Build DataFrame for next Batch	SS internal passes in startOffset and endOffset (results of getOffset)

1. **How to represent *Offset***
2. How to define ***Rate*** (To avoid tracking size of each message, rate is usually defined as # of messages)

Various Forms of Message Offset

- Consecutive Numbers
 - 0, 1, 2, 3, ...
 - Examples: Kafka, MQTT
 - Server-side index mapping an integer to the actual offset of the message in disk space
- Real offset
 - 0, `sizeof(msg0)`, `sizeof(msg0 + msg1)`, ...
 - Examples: Azure Event Hubs
 - No server-side index, passing offset as part of message to user

How it brings difference? - Kafka



Batch 0: Mapping the partition of DataFrame to a offset range in server side:

(**endOffsetOfLastBatch:0**, # of msgs: 1000)

Batch 1: How many messages are to be processed in next batch, and where to start?

(**endOffsetOfLastBatch: 999**, # of msgs: 1000)

How it brings difference? - Event Hubs

What's the offset of 1000th message???

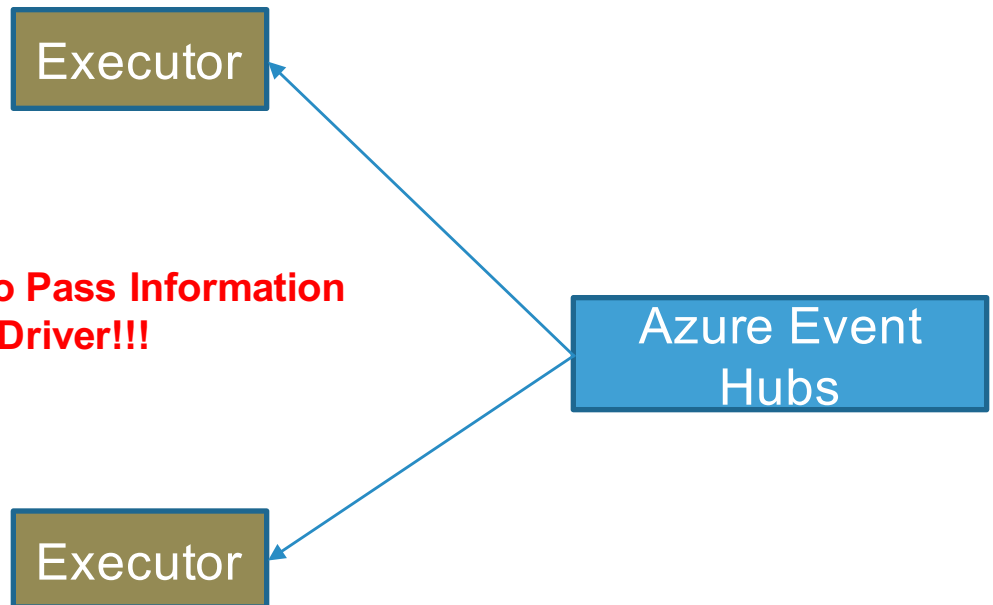
The answer appeared in **Executor** side
(when Task receives the message (offset as part of message))

Driver

**Build a Channel to Pass Information
from Executor to Driver!!!**

Batch 0: How many messages are to be
processed in next batch, and where to start?
(**endOffsetOfLastBatch:0**, # of Msgs: 1000)

Batch 1: How many messages are to be
processed in next batch, and where to start?
(**endOffsetOfLastBatch:?**, endOffset: 1000)



Difference of KafkaSource and EventHubsSource



Azure Event Hubs

getOffset

EndOffsetOfLastBatch (known
before the batch is finished)

Collect the ending
offsets of last batch
from executors

& Calculate targetOffset
of next Batch

getBatch(
startOffset,
endOffset)

StartOffset: EndOffsetOfLastBatch
(passed-in value from SS internal)

StartOffset: the collected values

How it brings difference? - Event Hubs

What's the offset of 1000th message???

The answer appeared in Executor side
(when Task receives the message (offset as part of message))

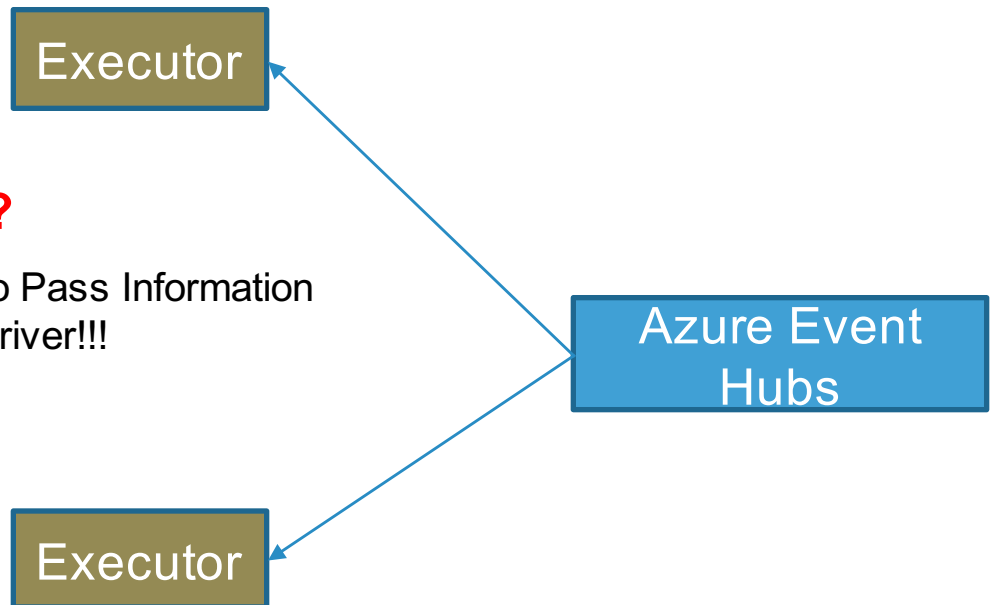
Driver

HOW?

Build a Channel to Pass Information
from Executor to Driver!!!

Batch 0: How many messages are to be
processed in next batch, and where to start?
(startOffset:0, # of Msgs: 1000)

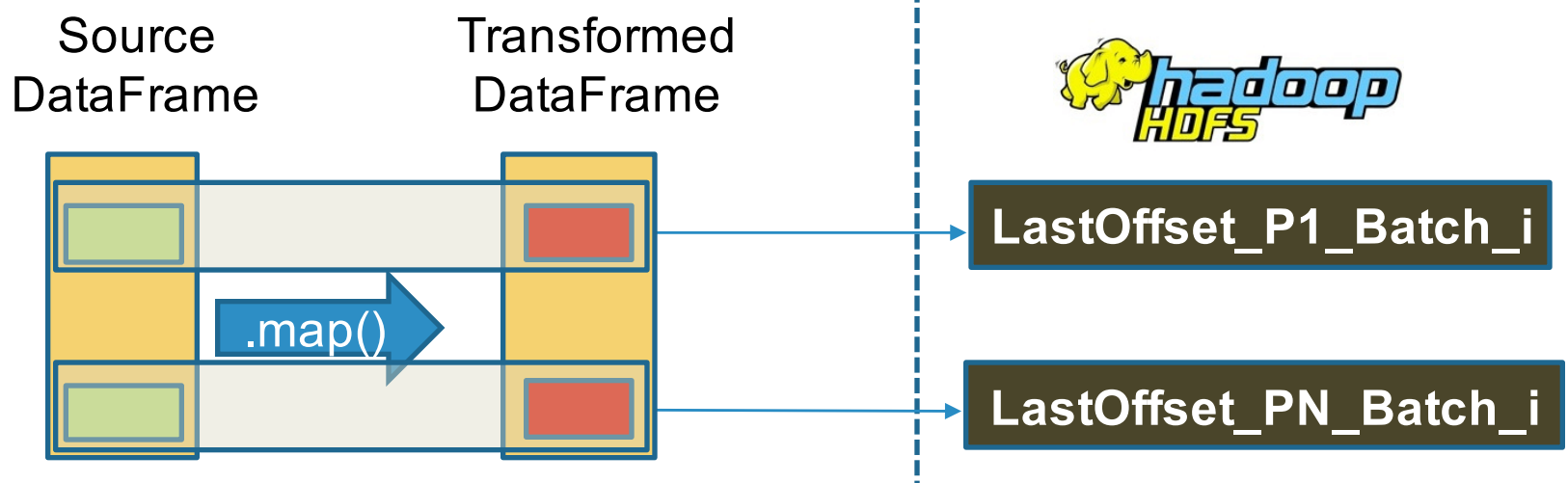
Batch 1: How many messages are to be
processed in next batch, and where to start?
(startOffset:?, endOffset: 1000)



HDFS-Based Channel

What's the next step??? Simply let Driver-side logic read the files?

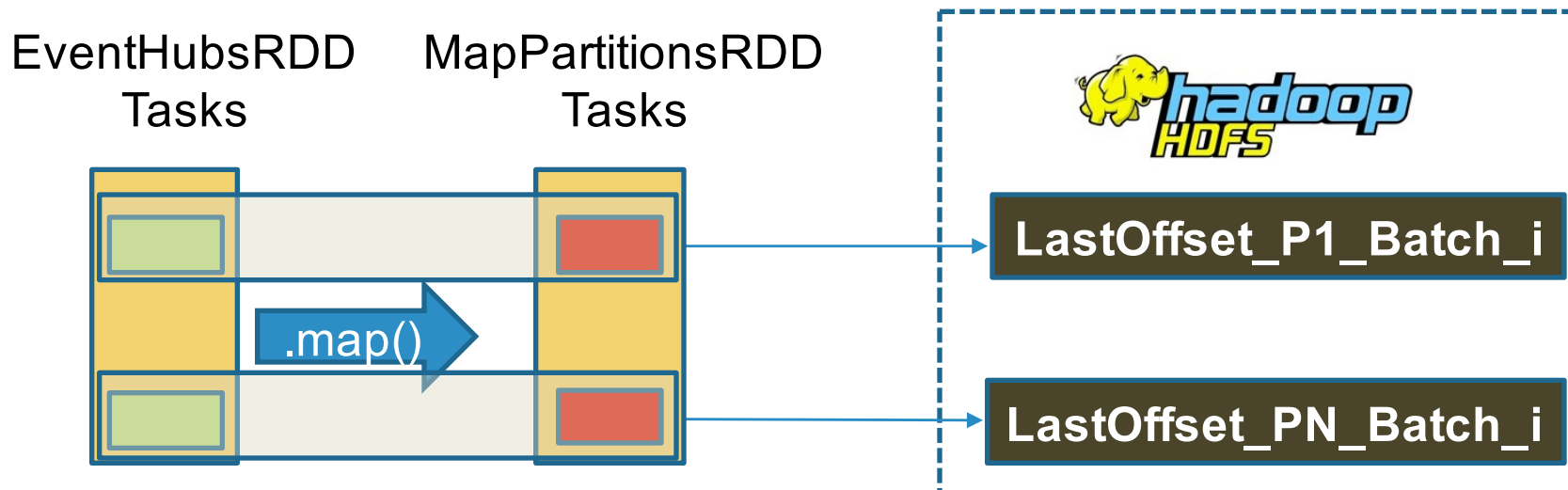
No!!!



- APIs like `DataFrame.head(x)` evaluates only some of the partitions... Batch 0 generate 3 files, and Batch 1 generates 5 files...
- You have to merge the latest files with the historical results and commit and then direct the driver-side logic to read

HDFS-Based Channel

- Ensure that all streams' offset are committed transactionally
- Discard the partially merged/committed results to rerun the batch



- APIs like `DataFrame.head(x)` evaluates only some of the partitions... Batch 0 generate 3 files, and Batch 1 generates 5 files...
- You have to merge the latest files with the historical results and **commit**...

Takeaways (1)

- Data Source is not only designed for structured streaming
- Accommodate Connector Implementation with Data Source Design
- **Message Addressing** (Offset) in data source side is the **Key Design Factor** to be considered
 - Requirement of Additional Information Sharing Channel between Executors and Driver
 - Design to ensure the correctness of the channel

Structured Streaming Unit Testing Framework

- “Action” based structured streaming test flow – StartStream, AddData, CheckAnswer, AdvanceClock, StopStream, etc.
- Kafka source unit tests use Kafka micro-service.
- Unit test framework not usable as is.

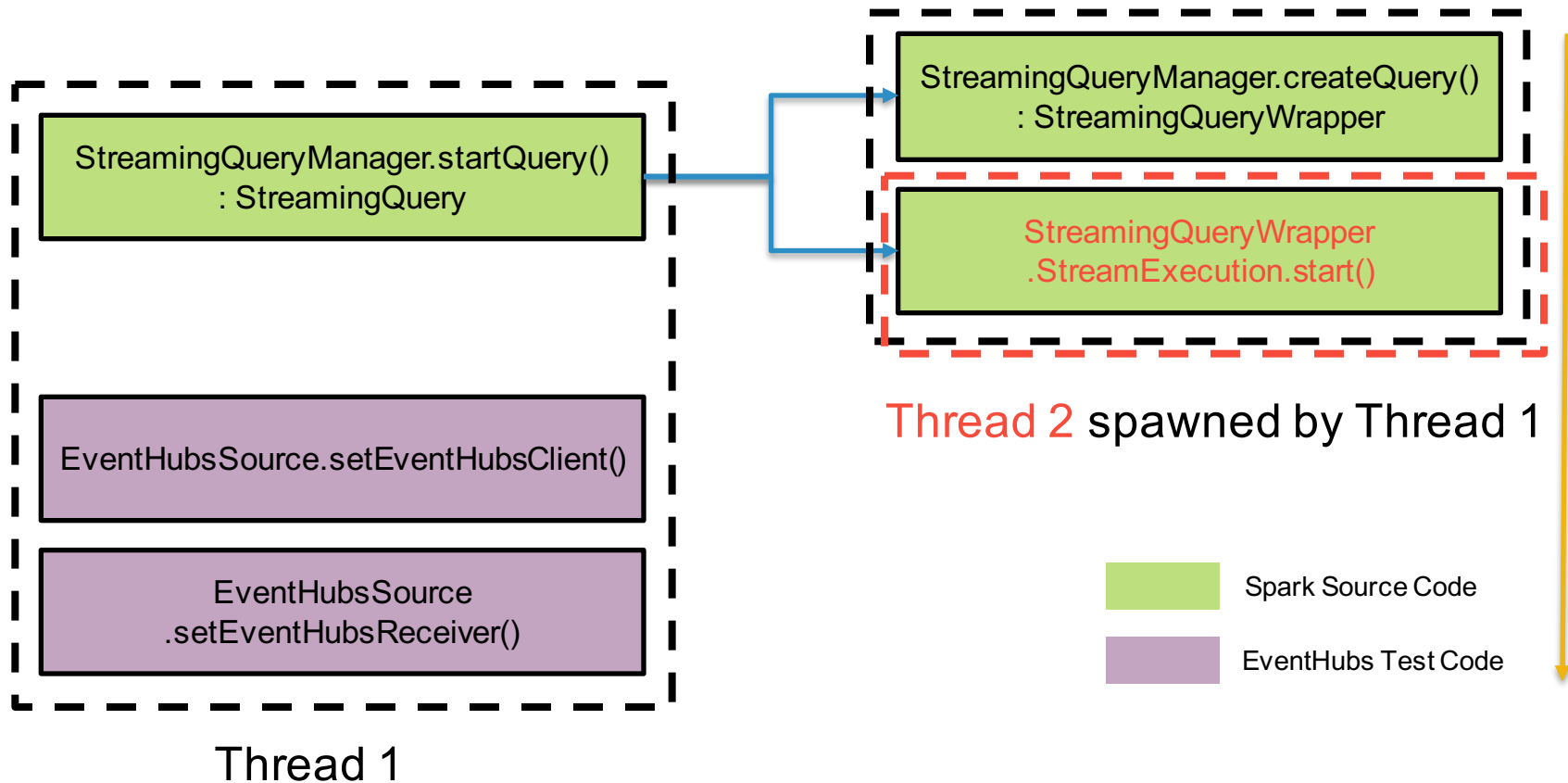
Unit Testing Event Hubs Source

- No Kafka type micro-service available for Event Hubs.
- Two simulated clients – Event Hubs AMQP and Event Hubs REST clients.
- Replace both clients before first call to the Event Hubs source.

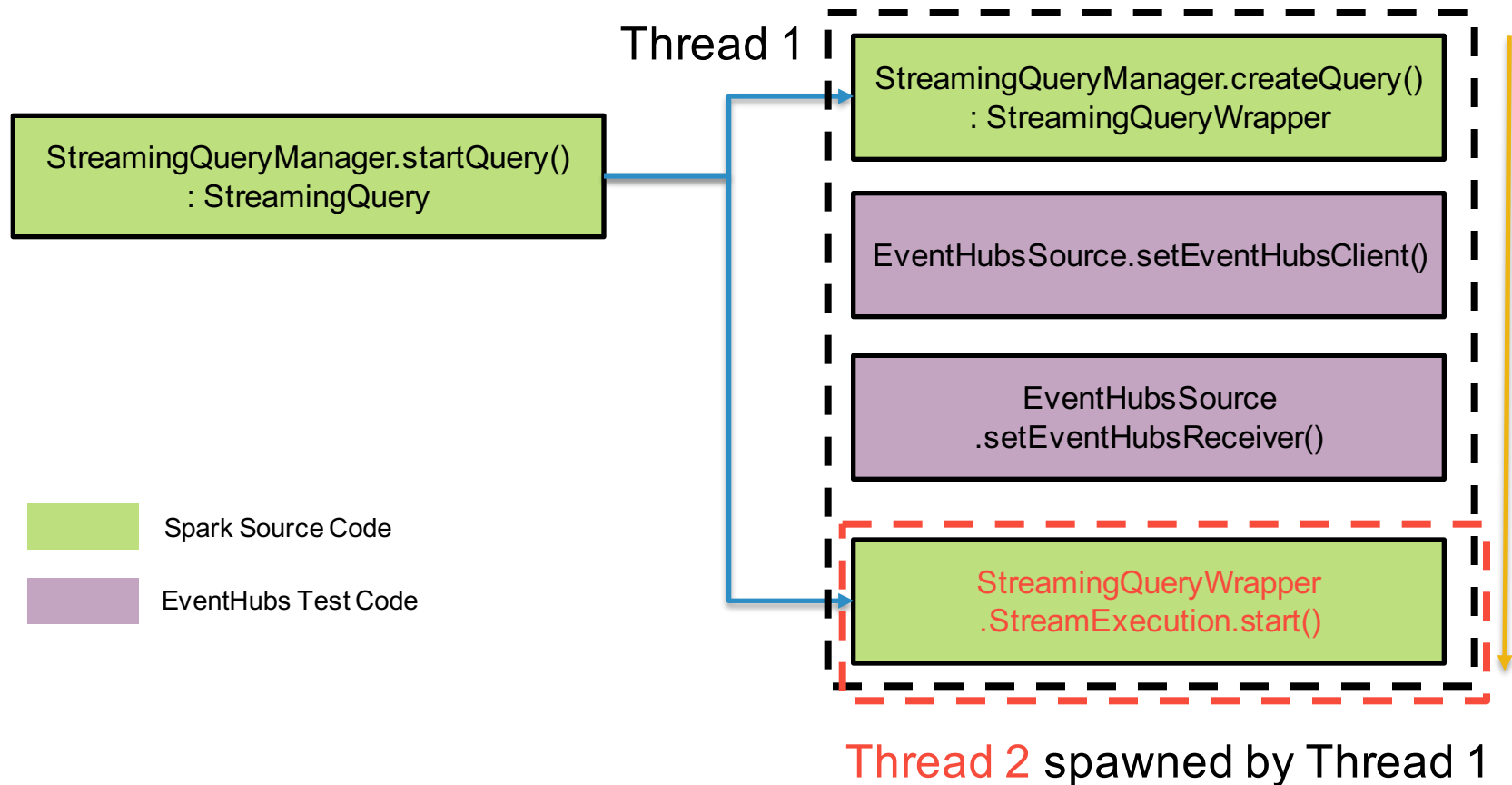
Issues With Unit Test Framework

- No separation between test setup and test execution.
- Unreliable update of the Event Hubs clients before asynchronous call to `StreamingQueryManager.startQuery()`.
- Task serialization issue with `AddData` as part of `StreamTest`.

What Is The Asynchrony Problem?



One Way To Solve Asynchrony Problem



Solving Asynchrony in Client Substitutions (1)

- **Invoke createQuery**

```
val createQueryMethod =  
sparkSession.streams.getClass.getDeclaredMethods.filter(m =>  
    m.getName == "createQuery").head  
createQueryMethod.setAccessible(true)  
currentStream = createQueryMethod.invoke(...)
```

- **Set as active query**

```
val activeQueriesField =  
sparkSession.streams.getClass.getDeclaredFields.filter(f => f.getName ==  
    "org$apache$spark$sql$streaming$StreamingQueryManager$$activeQueries").head  
activeQueriesField.setAccessible(true)
```

```
val activeQueries = activeQueriesField.get(sparkSession.streams).  
asInstanceOf[mutable.HashMap[UUID, StreamingQuery]]
```

```
activeQueries += currentStream.id -> currentStream
```

Solve Asynchrony in Client Substitutions (2)

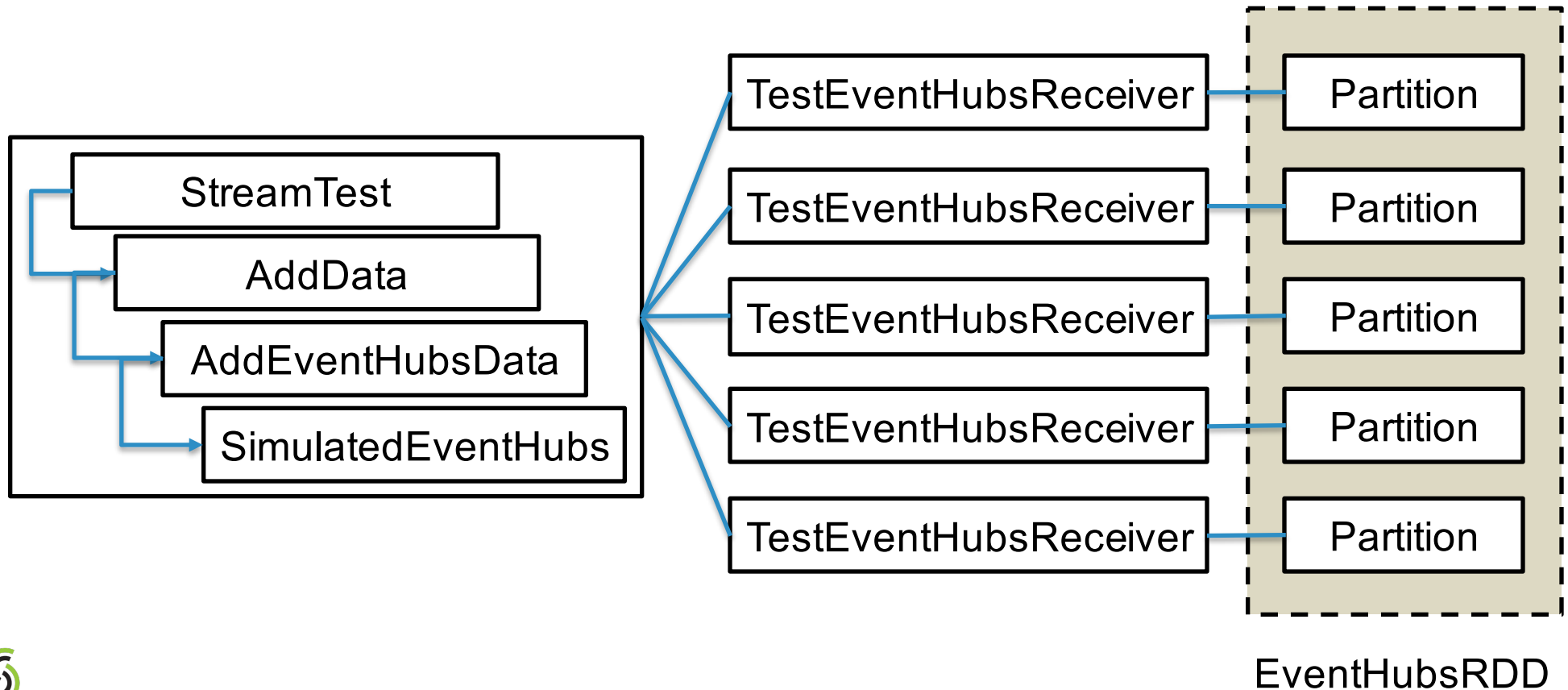
- **Substitute Event Hubs clients in source**

```
val eventHubsSource = sources.head
eventHubsSource.setEventHubClient(...)
eventHubsSource.setEventHubsReceiver(...)
```

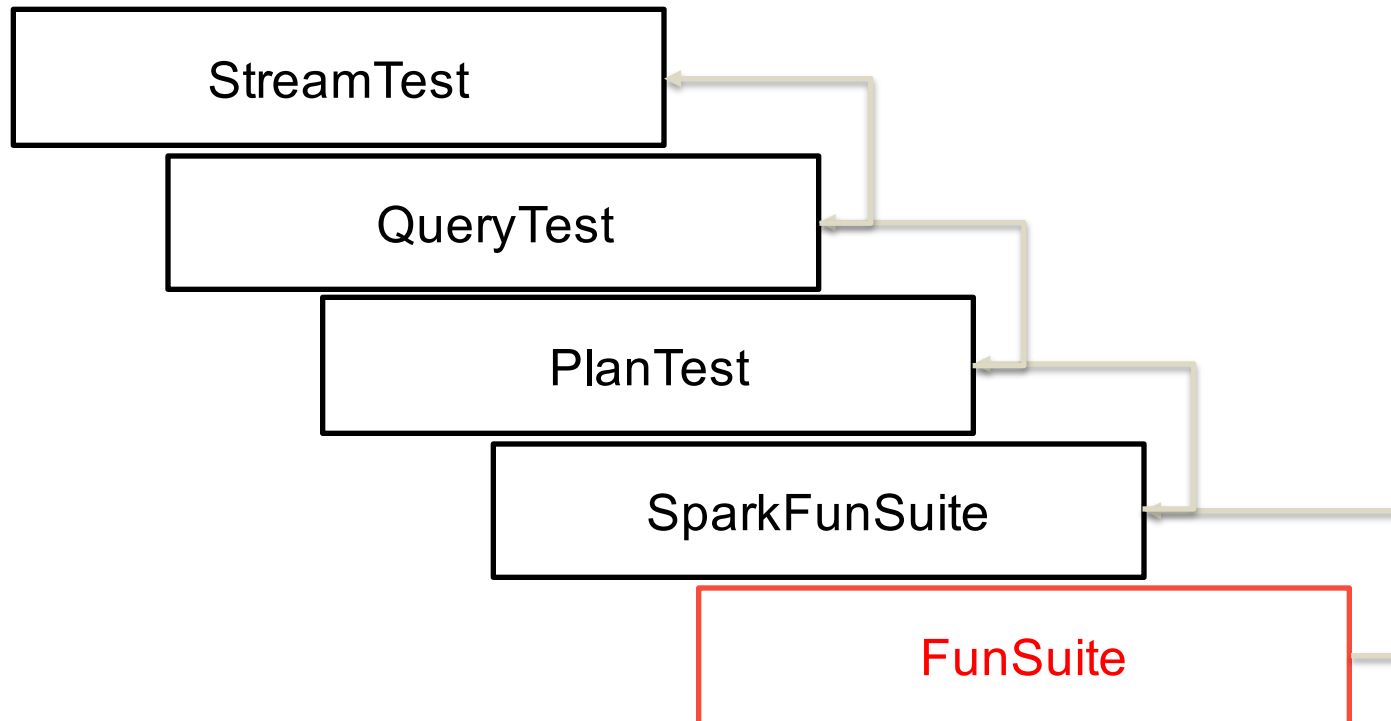
- **Start StreamExecution**

```
currentStream.start()
```

What Is The Task Serialization Issue?



What Is StreamTest Inheritance Model?



Solving Task Serialization Issue

- **Separate StreamTest into two traits:**

- **EventHubsStreamTest**

```
trait EventHubsStreamTest extends QueryTest with BeforeAndAfter with  
SharedSQLContext with Timeouts with Serializable { }
```

- **EventHubsAddData**

```
trait EventHubsAddData extends StreamAction with Serializable {}
```

After All These Changes...

```
testStream(sourceQuery) (  
    StartStream(...),  
    CheckAnswer(...),  
    AddEventHubsData(...),  
    AdvanceManualClock(...),  
    CheckAnswer(...),  
    StopStream,  
    StartStream(...),  
    CheckAnswer(...),  
    AddEventHubsData(...),  
    AdvanceManualClock(...),  
    AdvanceManualClock(...),  
    CheckAnswer(...))
```

Summary/Takeaway(2)

- We have a production grade Spark Streaming connector for Structured Streaming for the widely used Event Hubs as streaming source on Azure (<https://github.com/hdinsight/spark-eventhubs>)
- Design, implementation and testing of Structured Streaming Connectors
 - Accommodate Connector Implementation with Data Source Design (message addressing)
 - Unit Test with Simulated Service
 - Asynchrony – the biggest enemy to a easy test
 - Clear Boundary between test setup and test execution

Contributing Back To Community

- Failed Recovery from checkpoint caused by the multi-threads issue in Spark Streaming scheduler

<https://issues.apache.org/jira/browse/SPARK-19280>

One Realistic Example of its Impact: You are potentially getting wrong data when you use Kafka and reduceByWindow and recover from a failure

- Data loss caused by improper post-batch-completed processing

<https://issues.apache.org/jira/browse/SPARK-18905>

- Inconsistent Behavior of Spark Streaming Checkpoint

<https://issues.apache.org/jira/browse/SPARK-19233>



Thank You!!!

<https://github.com/hdinsight/spark-eventhubs>

<https://azure.microsoft.com/en-us/services/hdinsight/>