REX Real-time Data Pipeline:

Kafka Streams / Kafka Connect versus Spark Streaming

EL ARIB Abdelhamide , Data Engineer at ebiznext

OUTLINE

- The Challenge
- Brief presentation of Spark & Kafka
- Kafka vs Spark :
  1. New File Detection
  2. Processing
  3. Handling Failure
  4. Deployment
  5. Scaling
  6. Monitoring
- 10 points not to forget

THE CHALLENGE

HDFS is our source of truth

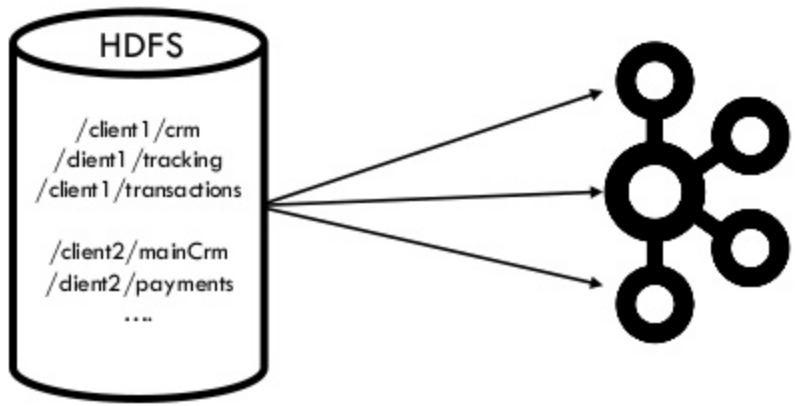
Need to stream very big data (parquet or Avro) into Kafka

Different schemas for each source

Need to scale up/down the speed of streaming

Exactly-Once Copy (Hdfs & Kafka)

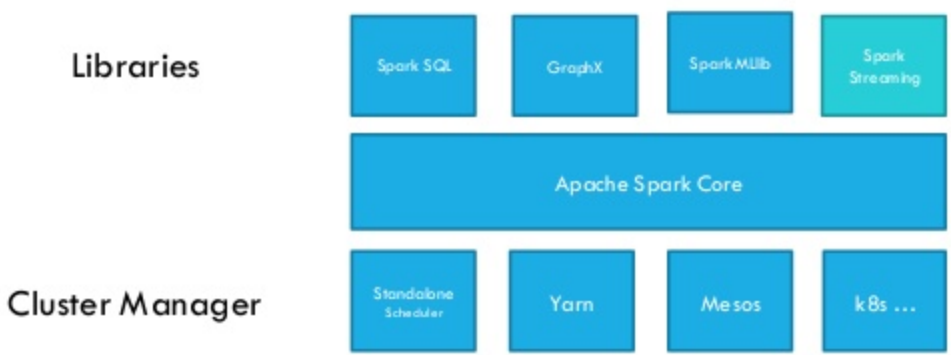
And the monitoring is, as always, a must



BRIEF PRESENTATION OF SPARK & KAFKA

APACHE SPARK

Distributed, data processing, in-memory engine for batch & streaming mode.



SPARK STREAMING

End-to-end latency ~ 100 ms

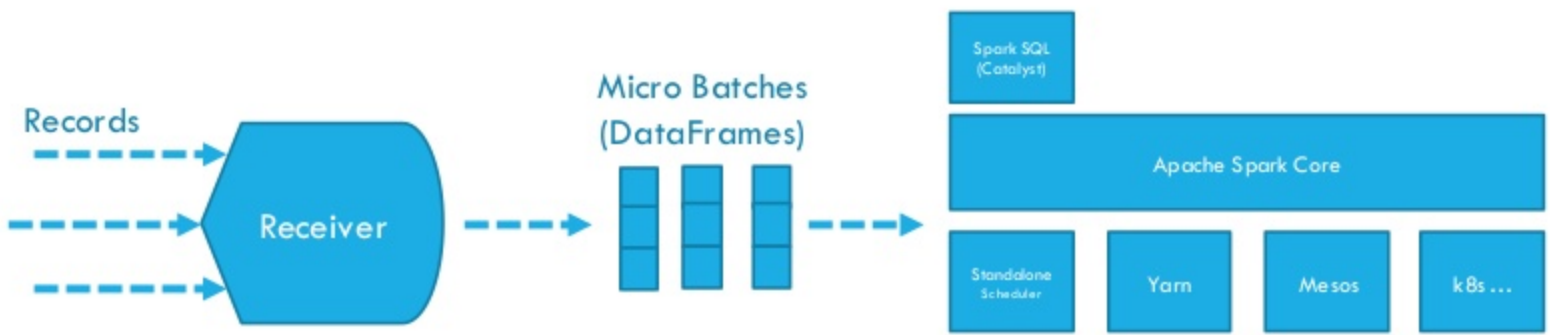
Exactly once guarantee

SPARK STRUCTURED STREAMING

Whenever you can, use Dataframe instead of Spark's RDD primitive

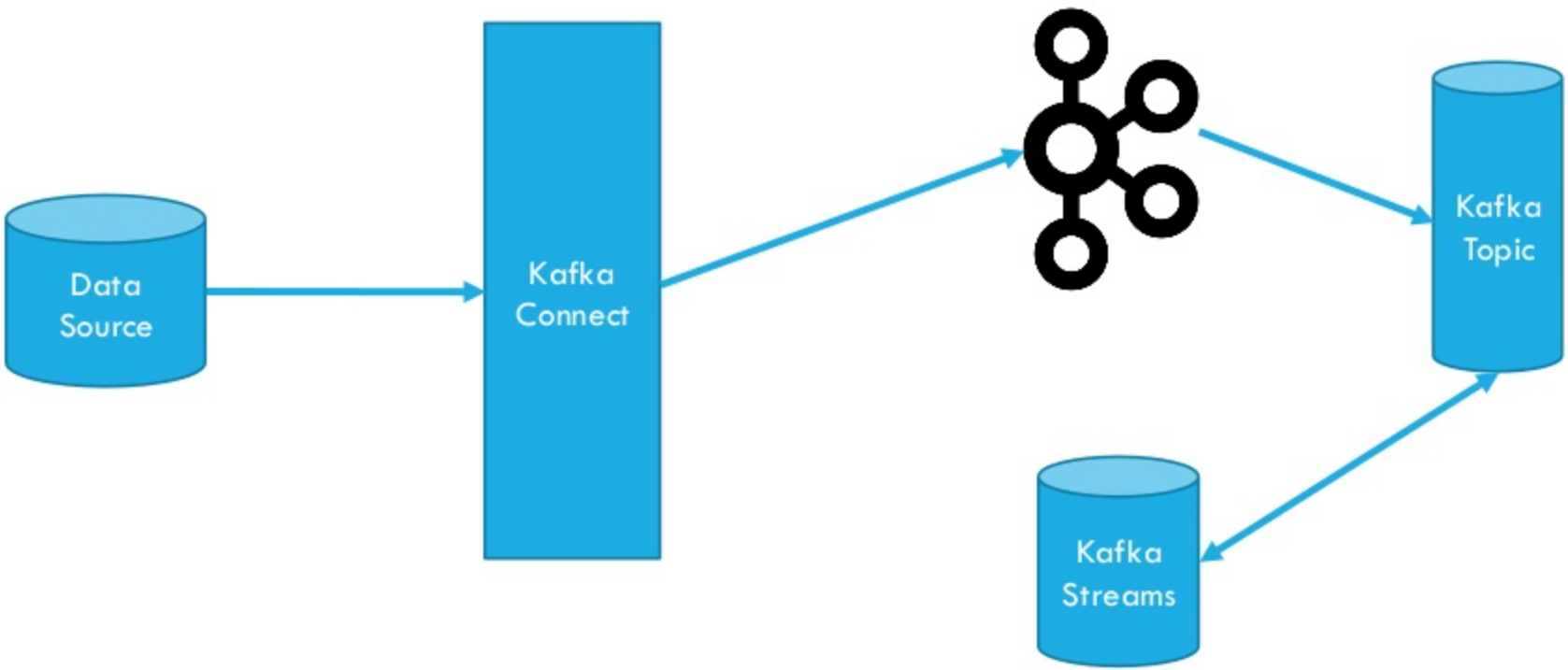
DataFrames get the benefit of the Catalyst query optimizer.

Multiple types of Triggers



(Experimental in Spark 2.3) End to end latency ~ 1 ms with at least once guarantee

KAFKA/ KAFKA STREAMS/ KAFKA CONNECT

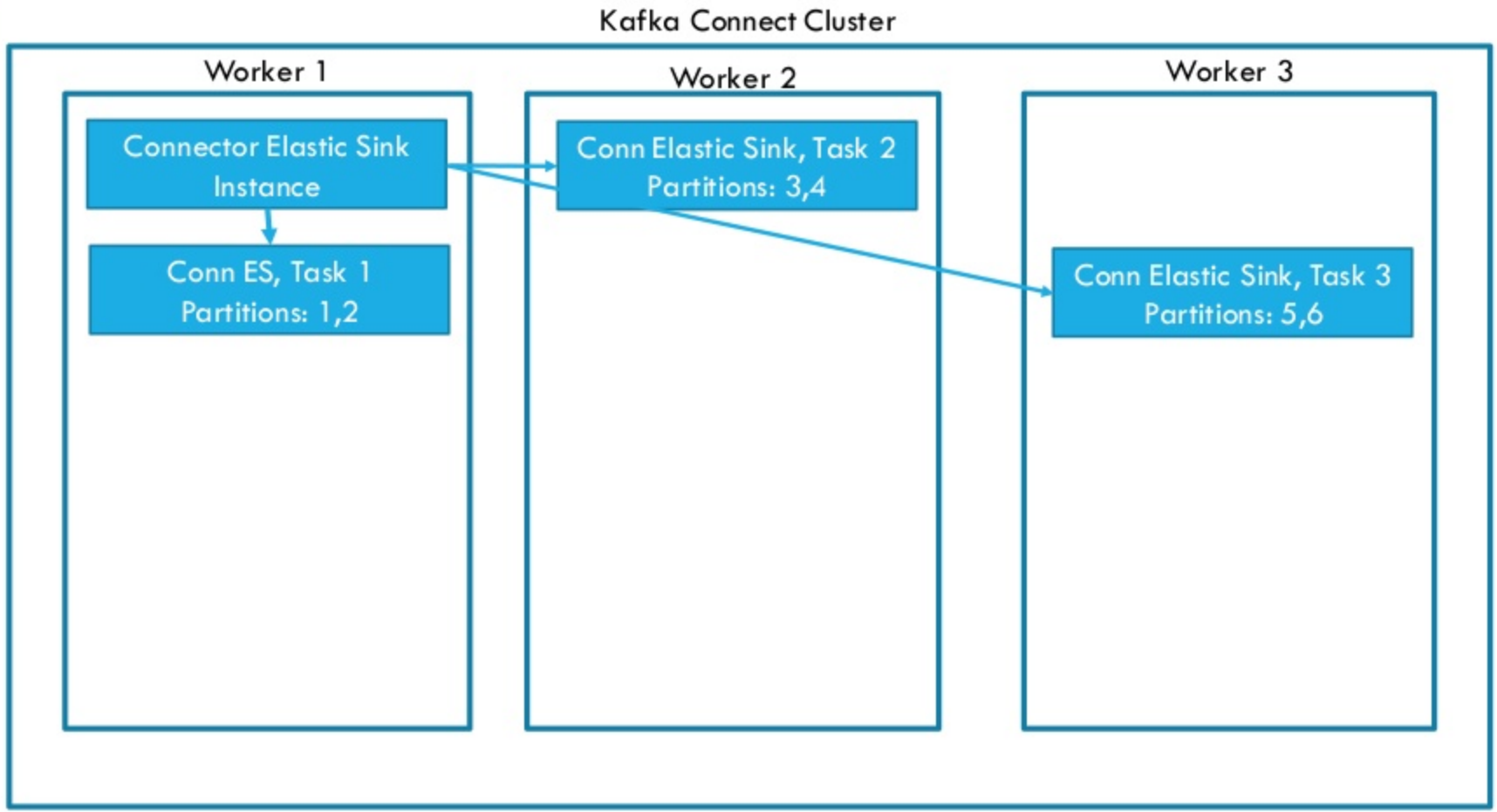


KAFKA CONNECT

Kafka Connect is a framework to stream data into & out of Kafka using connectors.

Two types of connectors: Sinks (Export) & Sources (Import)

poll.interval.ms (default 5ms)


Kafka Connect Cluster
Worker 1
Worker 2
Worker 3
Connector Elastic Sink Instance
Conn ES, Task 1 Partitions: 1,2
Conn Elastic Sink, Task 2 Partitions: 3,4
Conn Elastic Sink, Task 3 Partitions: 5,6

1. NEW FILE DETECTION

1. NEW FILE DETECTION - SPARK

Can detect new files within a dir

Should have the same schema

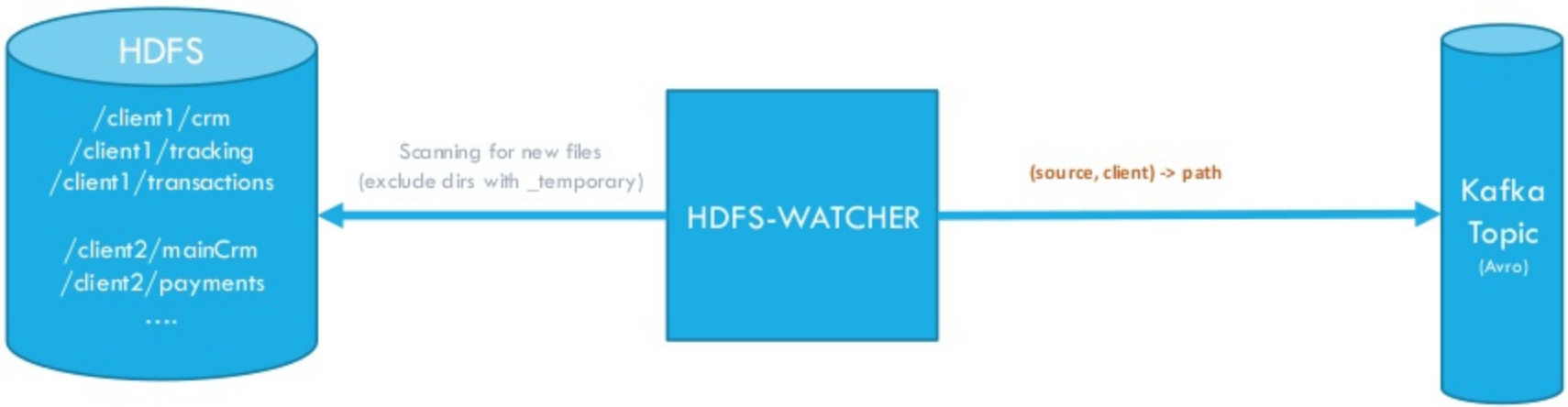
Can't create one stream for all source of the client X

N clients with M Source => N*M streams.

1 executor at least for each stream

1. NEW FILE DETECTION – KAFKA

1. NEW FILE DETECTION - HDFS-WATCHER (INOTIFY)



2. PROCESSING

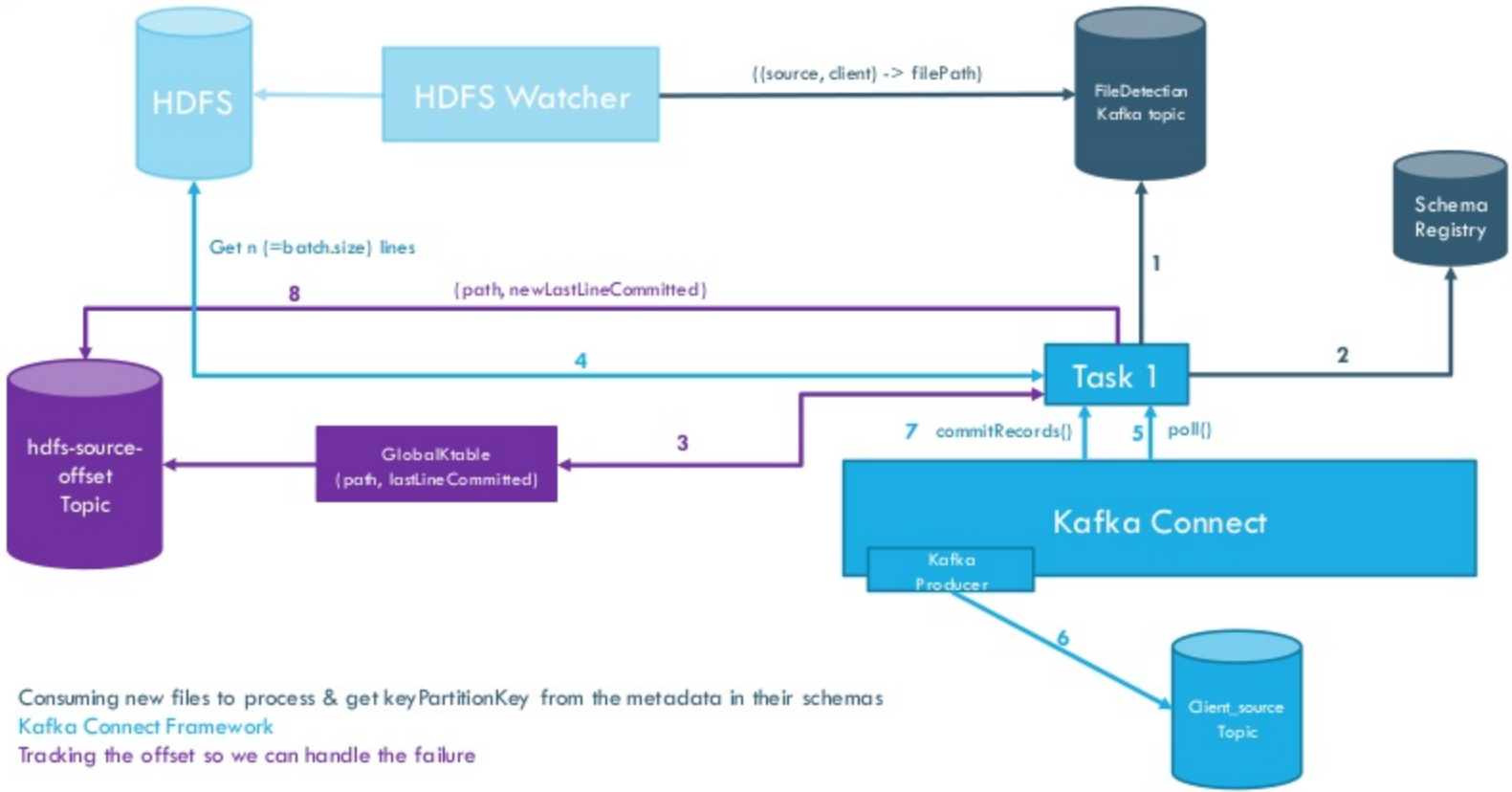
2. PROCESSING - SPARK

Read file as stream

Read partition by partition

At least once when writing to Kafka

2. PROCESSING - KAFKA



Consuming new files to process & get keyPartitionKey from the metadata in their schemas

Kafka Connect Framework

Tracking the offset so we can handle the failure

3. HANDLING FAILURE

3. HANDLING FAILURE - SPARK

The HDFS receiver is not reliable receiver

.option("checkpointLocation", "hdfs://checkpointPath") (Expensive)

spark.streaming.receiver.writeAheadLog.enable

spark.task.maxFailures = 0

```
trait StreamingListener {
  ...

  /** Called when a receiver has reported an error */
  def onReceiverError(receiverError: StreamingListenerReceiverError) { }
...
}
```

3. HANDLING FAILURE - KAFKA

GlobalKtable that track the last line commited.

KIP-298: Error Handling in Connect

- errors.retry.timeout
- errors.retry.delay.max.ms
- errors.tolerance (in a task) = none

Kafka producer config:

- acks = 1 (ou all)
- Retries > 0
- max.in.flight.requests.per.connection > 0(no need to preserve order)
- Batch.size (producer config) < number of lines * size of a single line
- linger.ms = 0 (Send the record as they arrive)

4. DEPLOYMENT

4. DEPLOYMENT - SPARK

Standalone

Cluster mode: Using spark-submit, the driver is in the cluster

Client mode : Start the app. The driver is in the Client machine.

4. DEPLOYMENT - KAFKA

Packager the connector (Classpath issues warning)

Make it in /plugins

Start Kafka connect cluster

Run via the The rest API

5. SCALING UP/DOWN

5. SCALING UP/DOWN- SPARK

Dynamic Allocation

Min*, max & init resources

Can't scale down manually the resources for executor without stopping the driver.

Can't scale up/down the number of executors manually.

5. SCALING UP/DOWN- KAFKA

Scale up/down easily the number of task.

Only Static allocation for the cluster resources.

Can't update the cluster/workers resources without stopping it.

6. MONITORING

6. MONITORING - SPARK

Spark ui

Spark ui rest API : http://localhost:4040/api/v1

JMX with DropWizard

6. MONITORING - KAFKA

Kafka connect REST API : to monitor Tasks

Jmx metrics

Connector Metrics

Task Metrics Worker

Metrics Worker

Rebalance Metrics

See KIP-196

10 POINTS TO NOT FORGET (FROM OUR EXPERIENCE)

AS ALWAYS IT REALLY DEPENDS ON YOUR NEEDS ☺

1. It depends on what kind of datasource & datasink you're using.
2. Always check the Receiver (Spark Streaming) & The connector (Kafka Connect)
3. The monitoring is pretty good for both
4. Integration Test is challenging with Kafka Connect
5. You want to add more tasks, go for Kafka Connect
6. You want dynamic allocation, go for Spark Streaming
7. Deploying Spark Streaming app is simple
8. Naming conventions in Kafka Connect Framework may be confusing with the naming in Kafka core
9. You prefer conf over code, go for Kafka Connect
10. You want a distributed, copy from or to Kafka, without using a cluster Manager : Go for Kafka Connect

THANK YOU