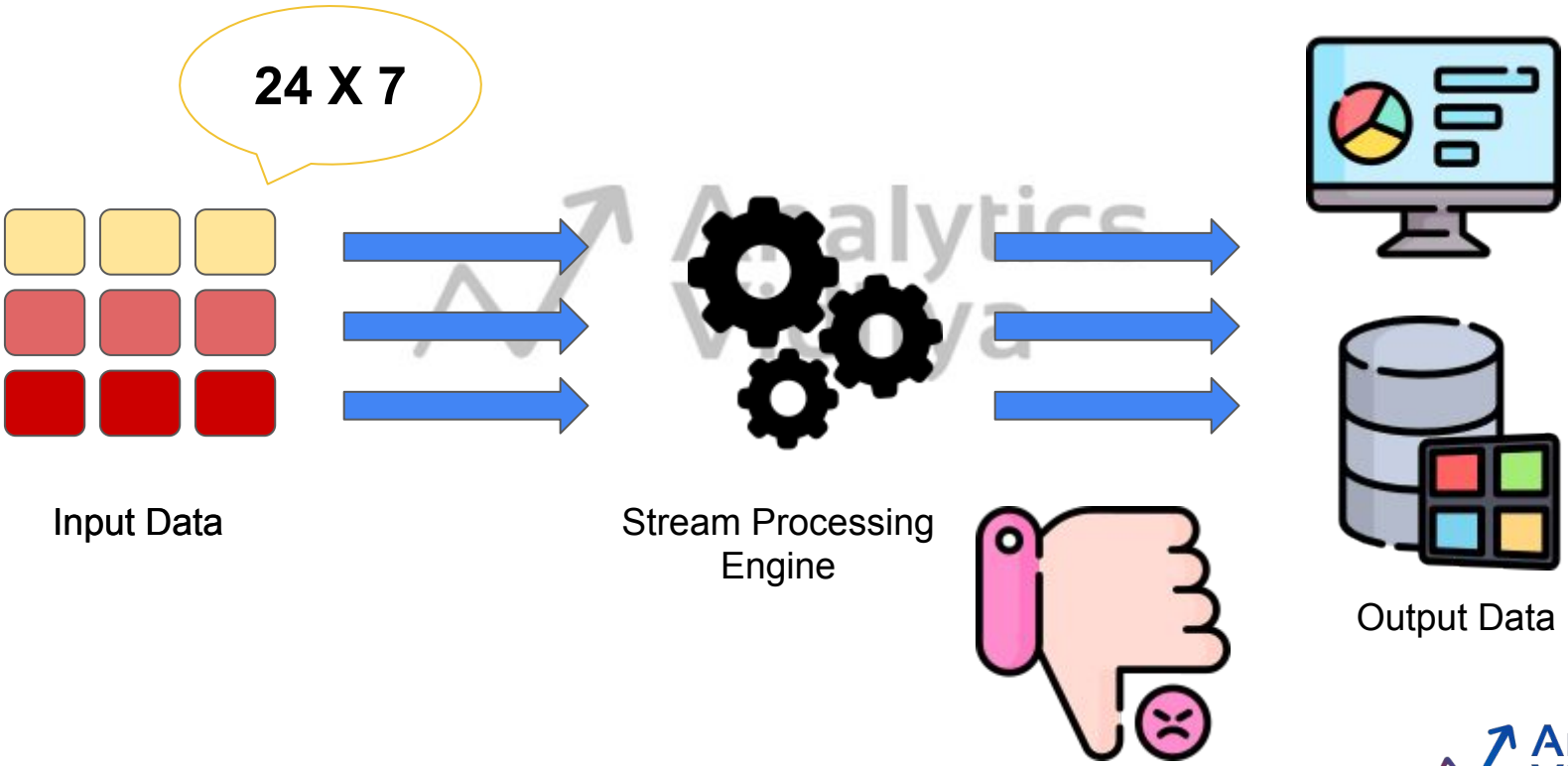
Checkpointing

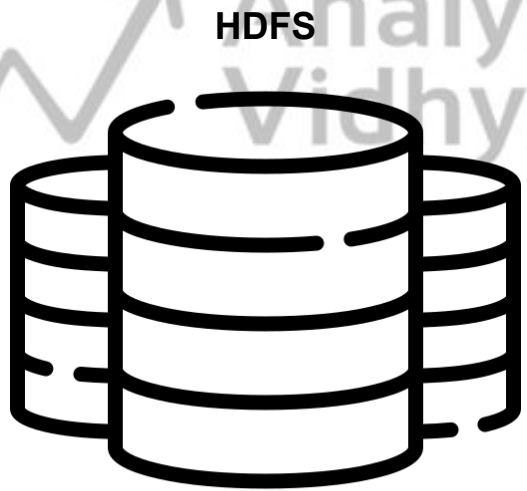
Analytics Vidhya

Fault Tolerance

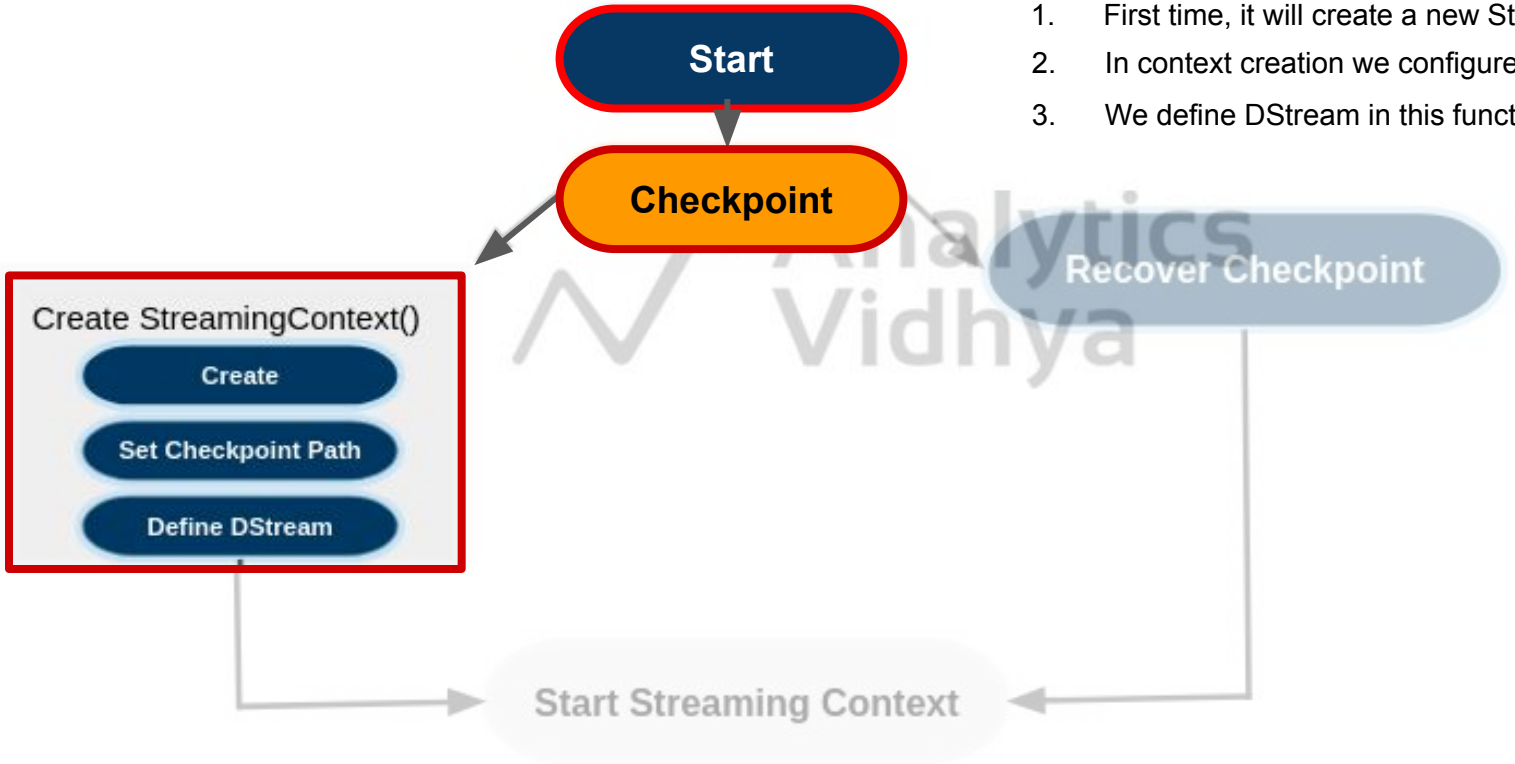


Checkpointing

Checkpointing is the process of writing received records at checkpoint intervals to HDFS.

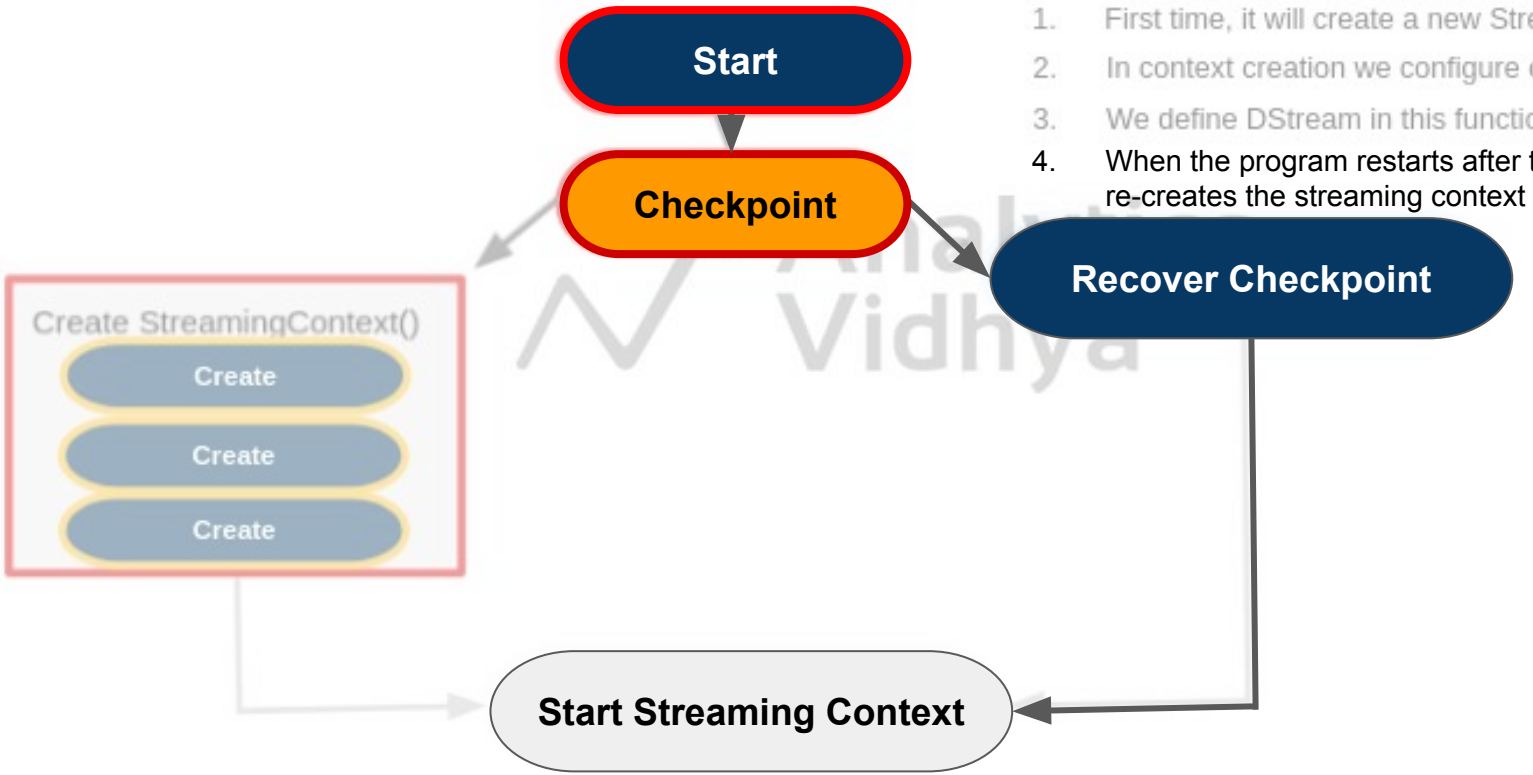


Checkpointing



1. First time, it will create a new Streaming Context
2. In context creation we configure checkpointing
3. We define DStream in this function

Checkpointing



1. First time, it will create a new Streaming Context
2. In context creation we configure checkpointing
3. We define DStream in this function
4. When the program restarts after the failure, it re-creates the streaming context from the checkpoint

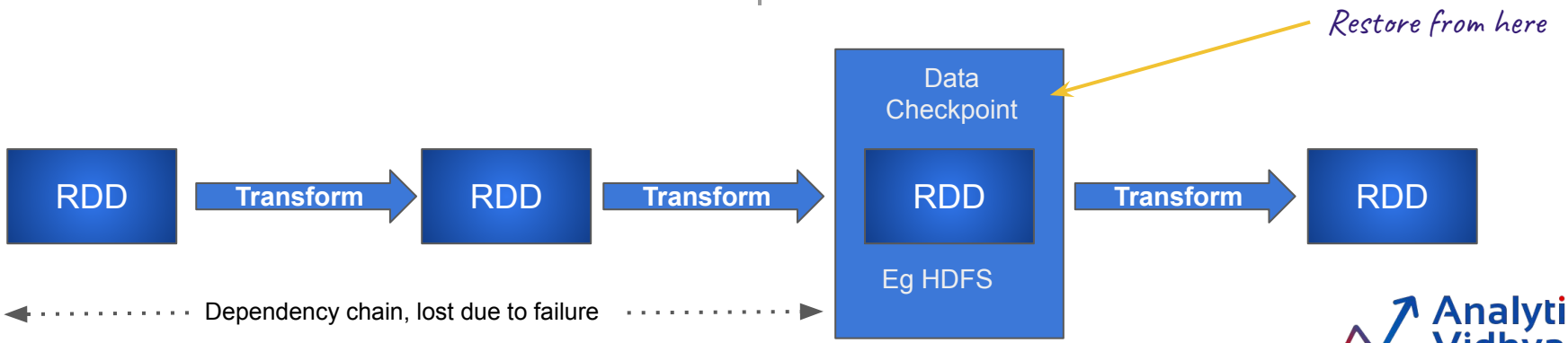
Types of Data to Checkpoint

Metadata Checkpointing

1. Configurations
2. DStream Operations
3. Incomplete batches

Data Checkpointing

1. When upcoming RDD's depend on RDD's of previous batches



Types of Checkpointing

- **Reliable Checkpointing**
  - Actual RDD is stored in a reliable distributed system
  - Call *SparkContext.setCheckpointDir(directory: String)*

- **Local Checkpointing**
  - RDD is persisted to local storage in the executor

Caching/Persisting

persist()

- Cache computed RDD and its **lineage** in **memory**
- **Next operation** on the same dataset will be faster
- DStreams from **window-based operations** are automatically persisted

cache()

- Shorthand for **persist** at the default storage level: **MEMORY_ONLY**

Storage Levels in Apache Spark

MEMORY_ONLY

Allows storage of RDD as deserialized Java objects

Recomputes any RDDs not fitted in memory

MEMORY_AND_DISK

Allows storage of RDD as deserialized objects

Also stores RDDs on disk

MEMORY_ONLY_SER

Stores RDD as serialized Java objects

Enables better space efficiency

MEMORY_AND_DISK_SER

Similar to **MEMORY_ONLY_SER**, but spills partitions not fitted in memory to disk

Storage Levels in Apache Spark

DISK_ONLY

Store the RDD partitions only on disk

MEMORY_ONLY_2, MEMORY_AND_DISK2

Replicate every partition on 2 cluster nodes

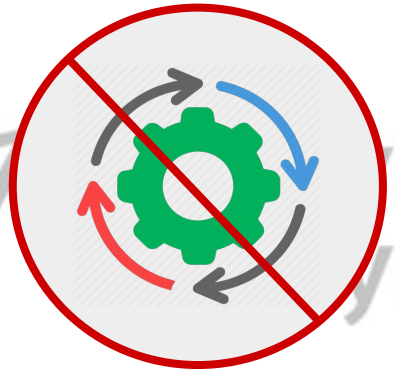
OFF_HEAP

Allows storage of RDD in serialized format in Tachyon

Reduces garbage collection overhead and avoids losing in-memory cache

Benefits of Caching/Persistence

Saves the data by default in memory



Avoids re-computation of the whole lineage

Improves Performance

When to Enable Checkpointing

Checkpointing must be enabled for applications with any of the following requirements:

- Usage of **stateful transformations**
- Recovering from **failures of the driver** running the application

How to Configure Checkpointing

```
# Function to create and setup a new StreamingContext
def functionToCreateContext():
    sc = SparkContext(...)   # new context
    ssc = StreamingContext(...)
    lines = ssc.socketTextStream(...)
    ...
    ssc.checkpoint(checkpointDirectory)   # set checkpoint directory
    return ssc

# Get StreamingContext from checkpoint data or create a new one
context = StreamingContext.getOrCreate(checkpointDirectory,functionToCreateContext)

# Do additional setup on context that needs to be done,
# irrespective of whether it is being started or restarted context. ...

# Start the context
context.start()
context.awaitTermination()
```

Checkpointing can be enabled by setting a directory in a fault-tolerant, reliable file system (e.g., HDFS, S3, etc.) to which the checkpoint information will be saved

Thank You

Analytics
Vidhya