Getting Started with Stream Processing with Spark Streaming

GETTING STARTED WITH DISCRETIZED STREAMS

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Understand the need for stream processing

Understand Spark DStreams and their relationship with RDDs

Perform basic transformations on DStreams

Distributed Computing and Its Limitations

How Much Data Do Organizations Deal With?

Google

Current storage = 15 exabytes

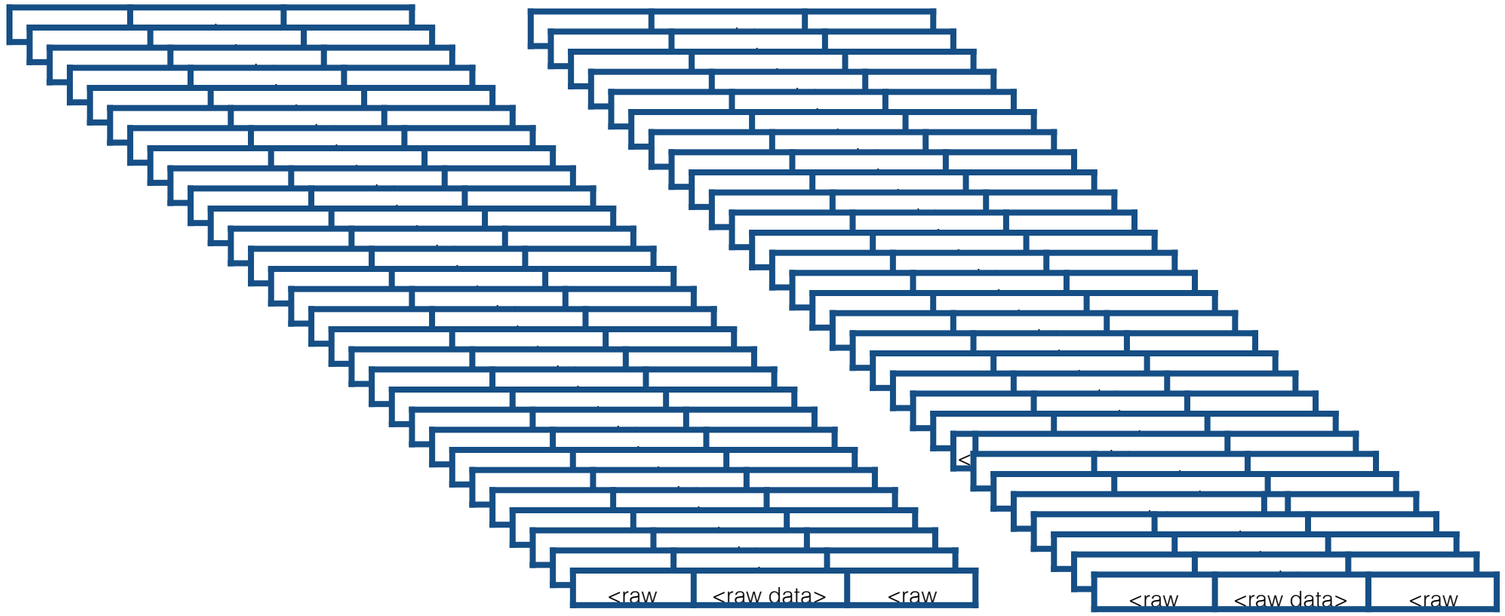
Processed per day = 100 petabytes

Number of pages indexed = 60 trillion

Unique search users per month > 1 billion

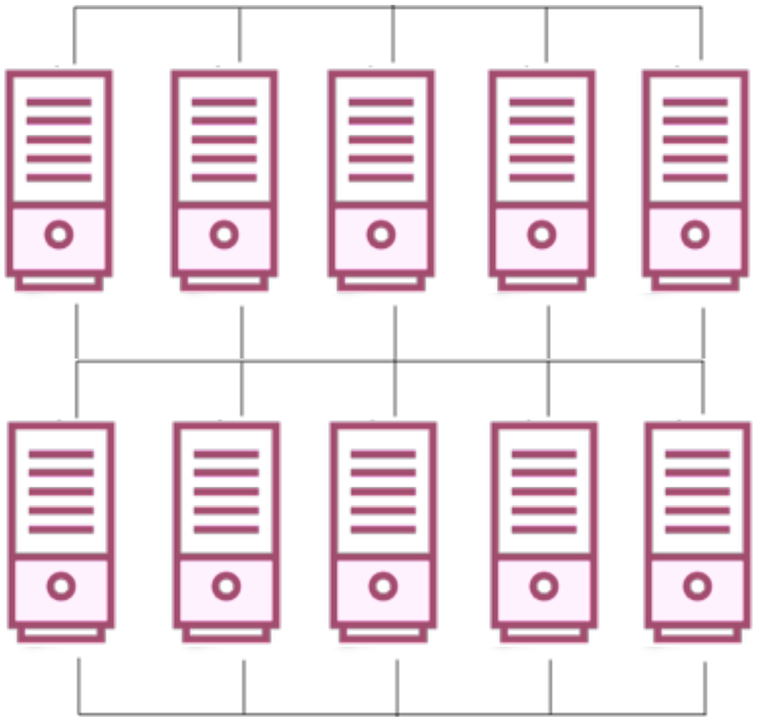
Searches per second = 2.3 million

Distributed Computing Solution



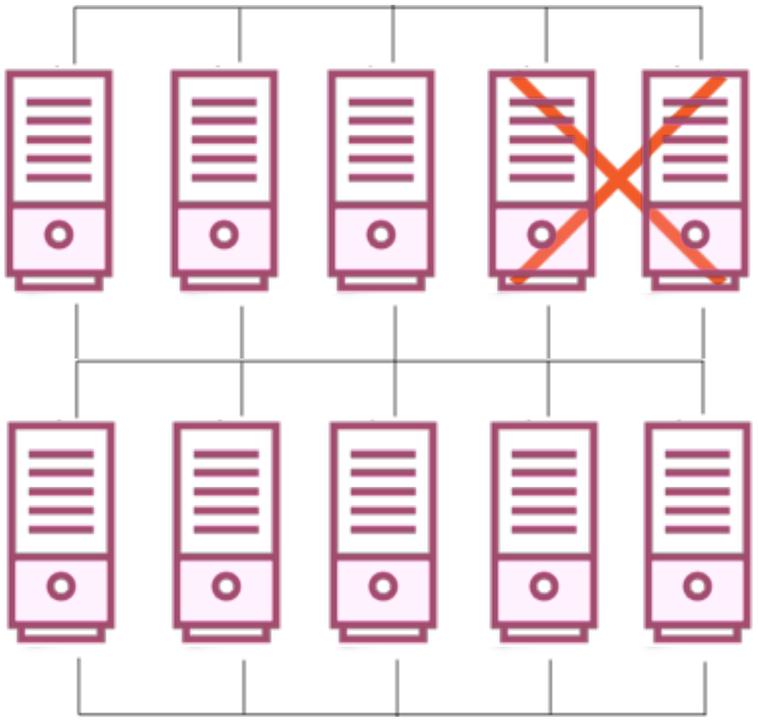
1: Store millions of records on multiple machines

Distributed Computing Solution



2: Run processes on all these machines to crunch data

Distributed Computing Solution

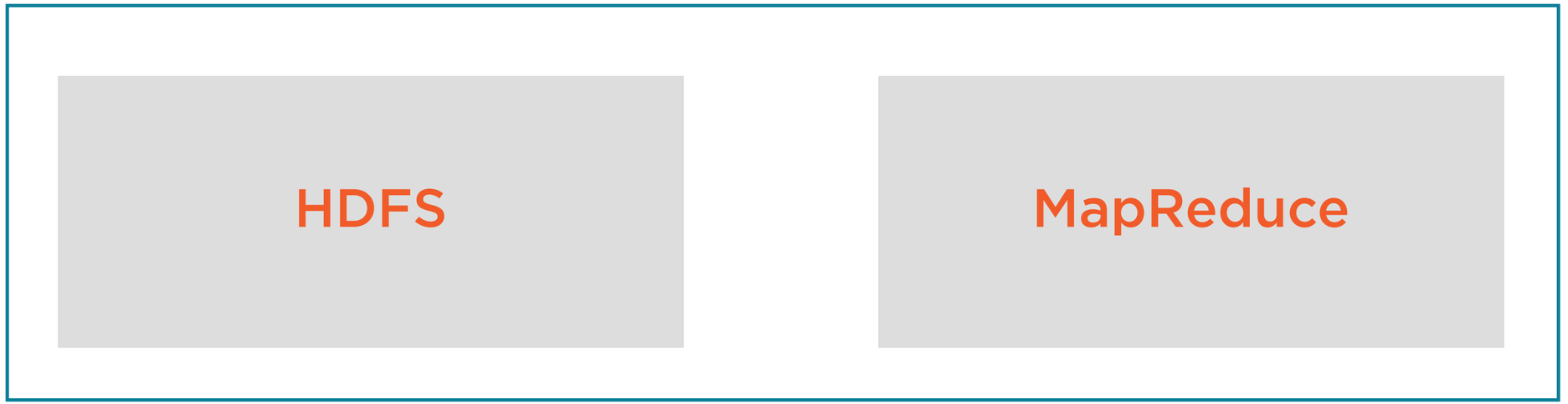


3: Handle fault tolerance and recovery when nodes crash

Hadoop

A distributed computing framework to process millions of records

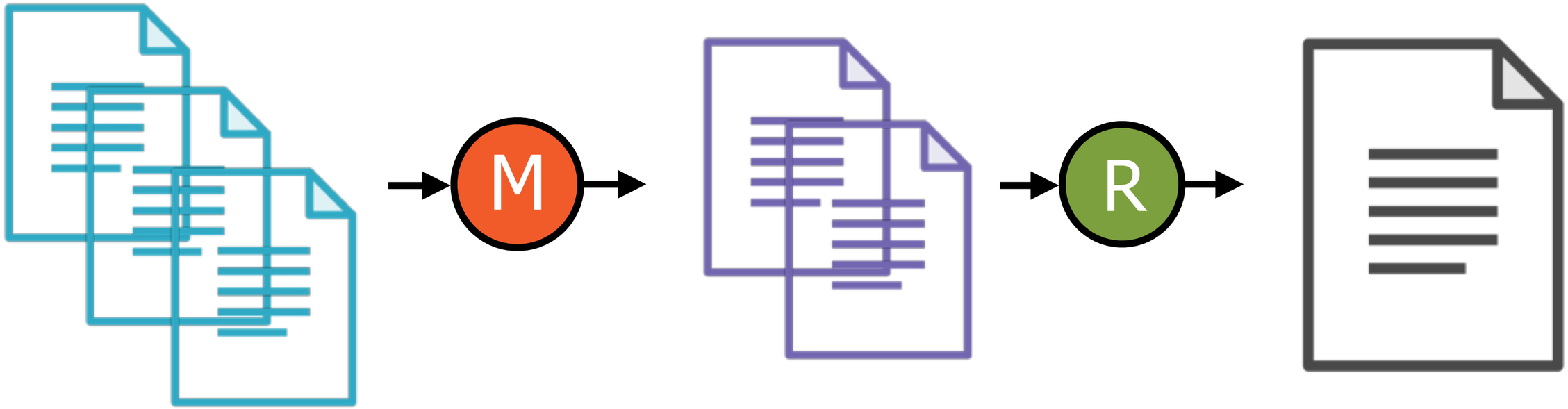
Hadoop



A file system to manage the storage of data

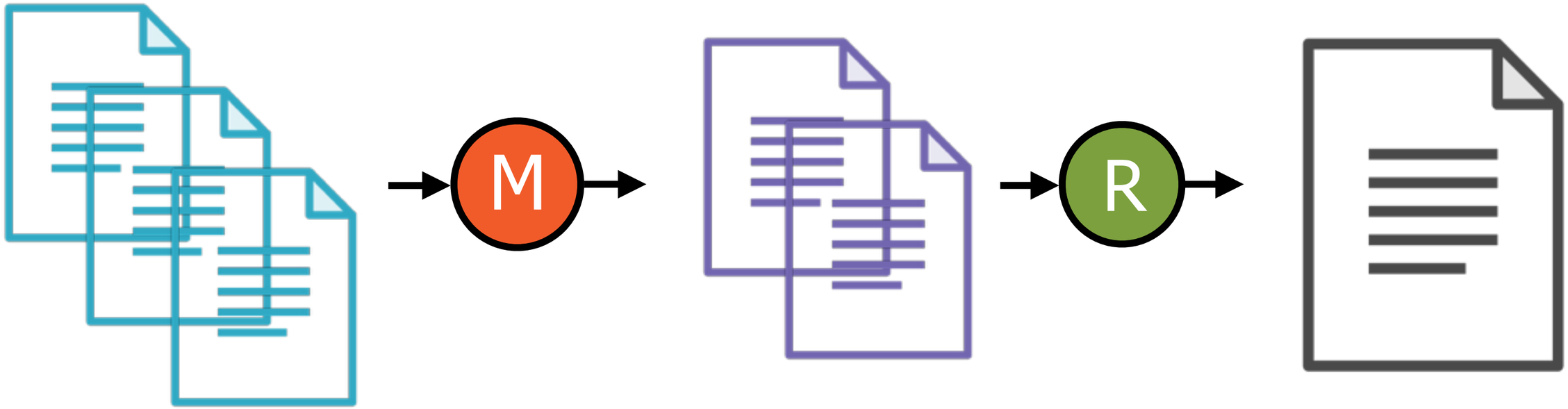
A framework to process data across multiple servers

MapReduce



MapReduce is fundamentally a batch processing operation

MapReduce



Jobs run for hours, even days to extract information

The MapReduce framework does **not allow real-time** processing of data

The Importance of Real-time Processing

Real-time Processing

Website for a bank

Reports, Tracking, Monitoring

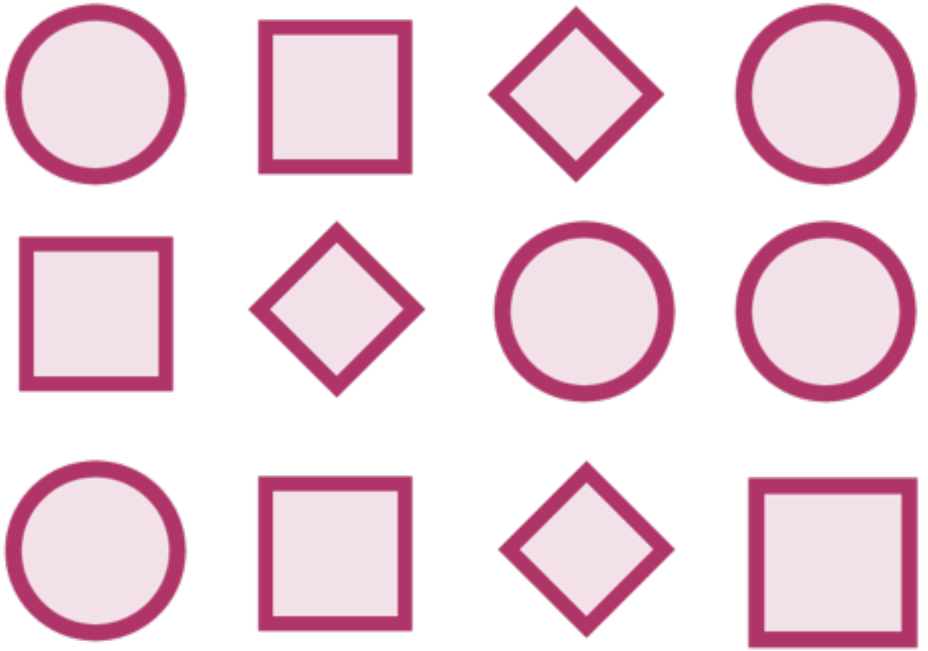
Yearly: New accounts, popular investment products

Monthly/Weekly: Account summaries, trends in spending

Daily/Hourly: Suspicious, fraudulent transactions

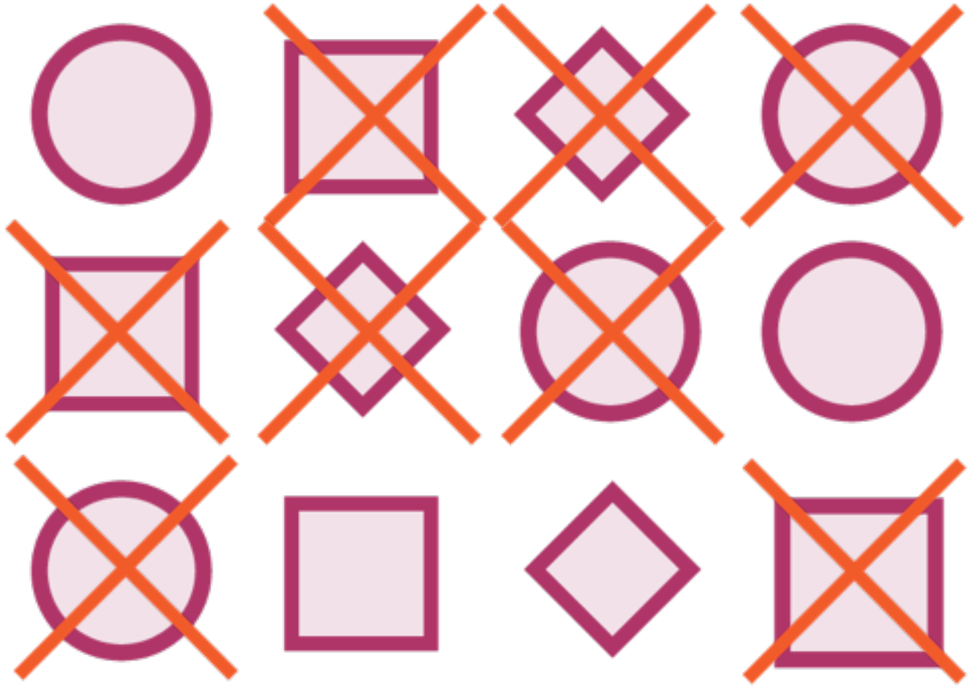
Long-running MapReduce jobs

Failures, Timeouts, Errors



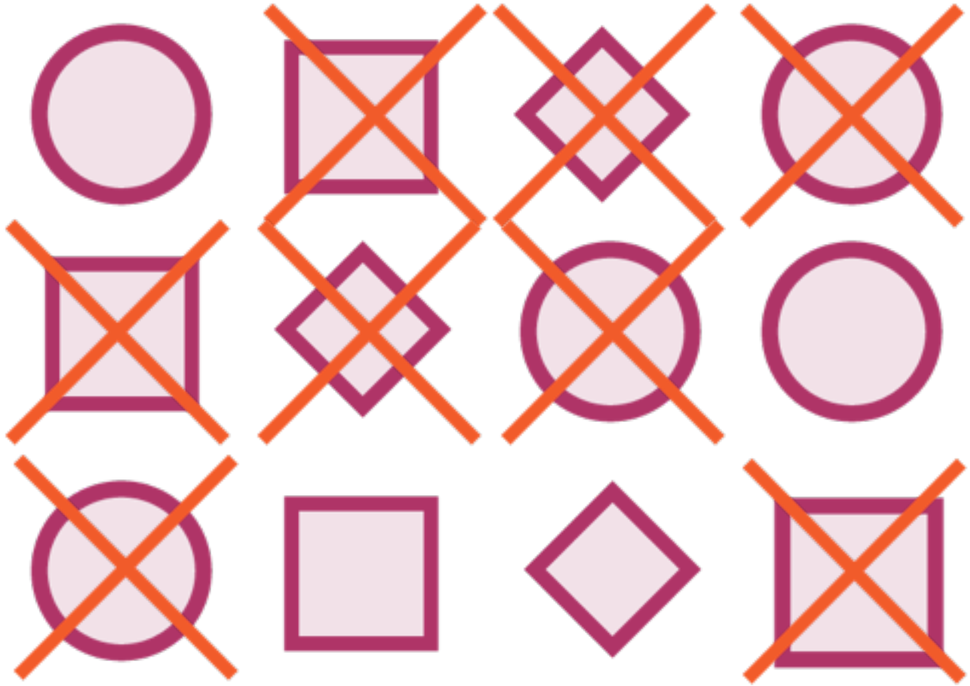
A bunch of requests are made to the server

Failures, Timeouts, Errors



Something is wrong with the site - many requests fail

Failures, Timeouts, Errors



Such failures are critical

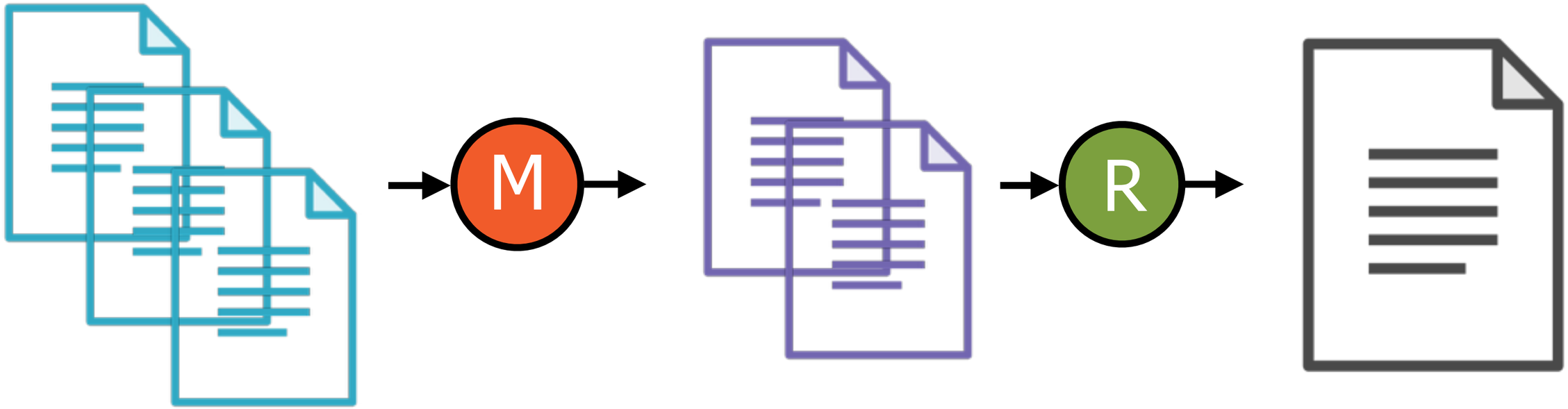
Failures, Timeouts, Errors

They require real-time monitoring

Failures, Timeouts, Errors

Errors can’t be stored in files to be processed later

MapReduce



This model does not work!

Apache Spark

Is a general purpose engine to solve a variety of data processing problems

Apache Spark

Can be used instead of Hadoop for traditional batch processing operations

Apache Spark

With additional support for real-time processing

Apache Spark

A fast, in-memory data processing engine

Part of the Hadoop eco-system

Runs on a distributed computing environment

Processes data on a cluster of machines

Apache Spark

REPL environment in Python and Scala

Stable production system

Specific modules for specific cases

- Spark SQL
- MLLib
- GraphX
- Spark Streaming

Spark offers the **Spark Streaming** module for real-time processing

This makes it a better alternative to Hadoop when manipulating **data streams**

Spark Streaming Module

Data is received as a stream

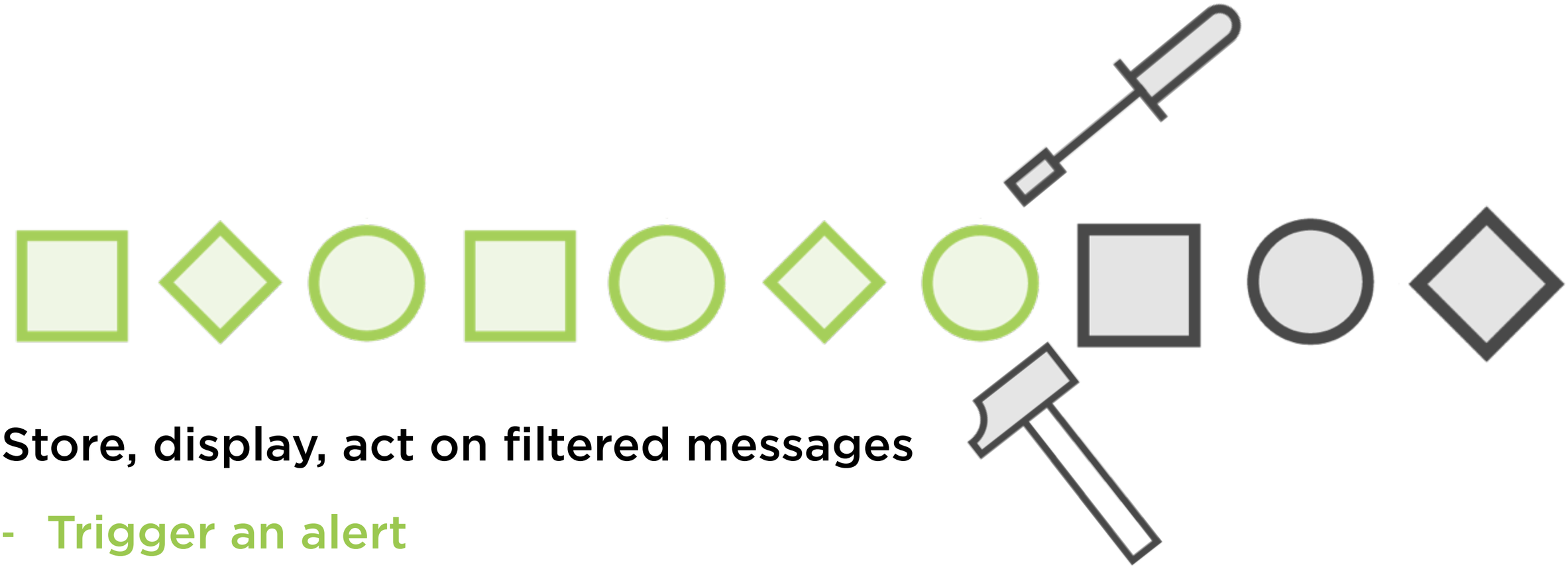
- Log messages
- Tweets
- GPS location information

Spark Streaming Module

Process the data one entity at a time

- Filter error messages
- Find references to the latest movies
- Map route of a car

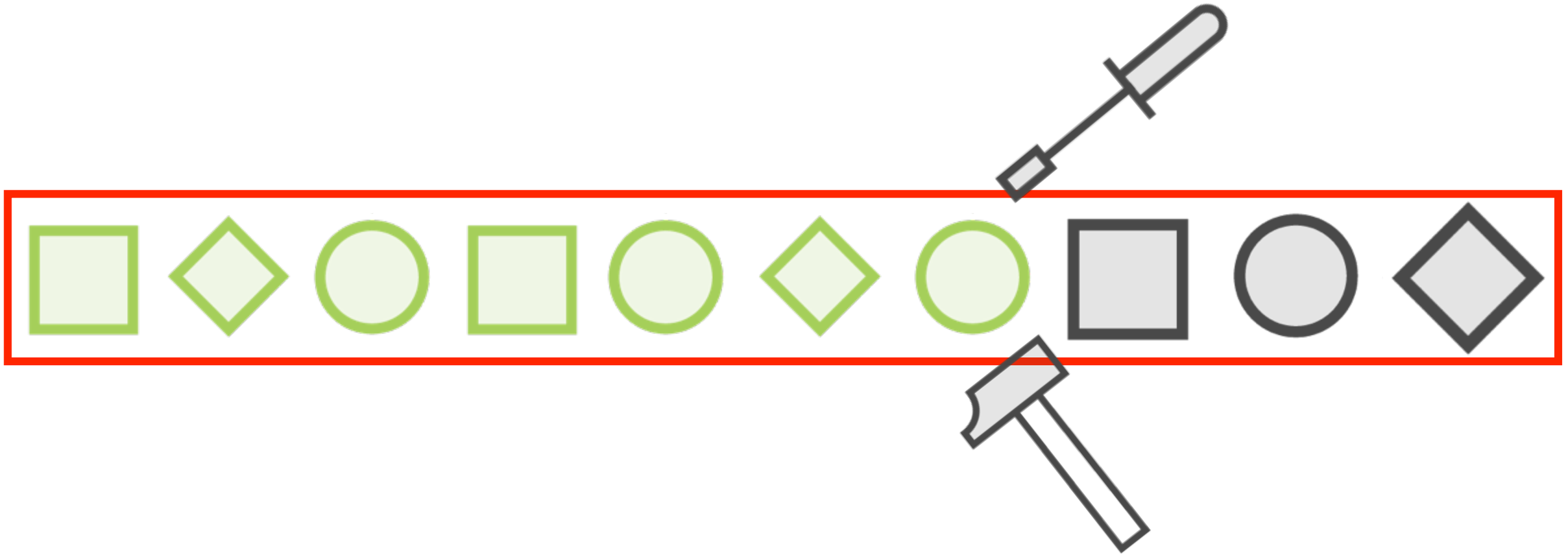
Spark Streaming Module



Store, display, act on filtered messages

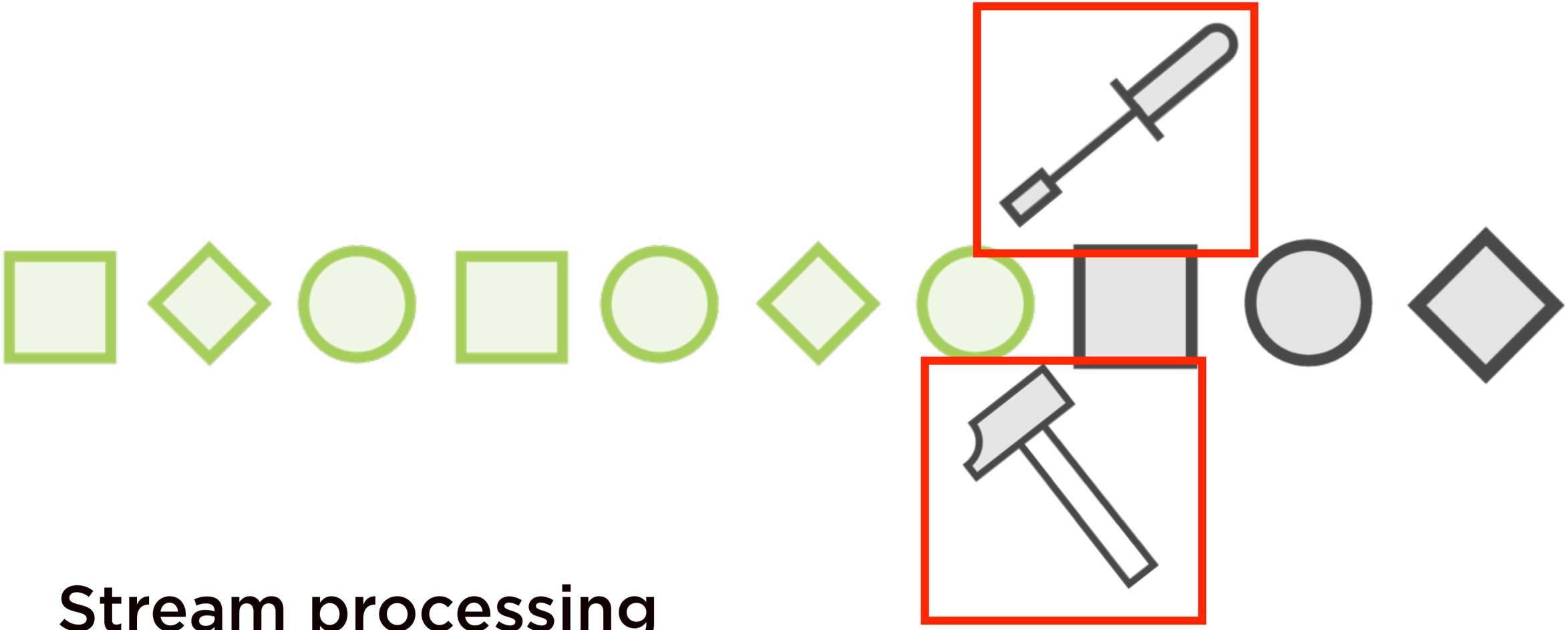
- Trigger an alert
- Show trending graphs
- Display route on the map

Spark Streaming Module



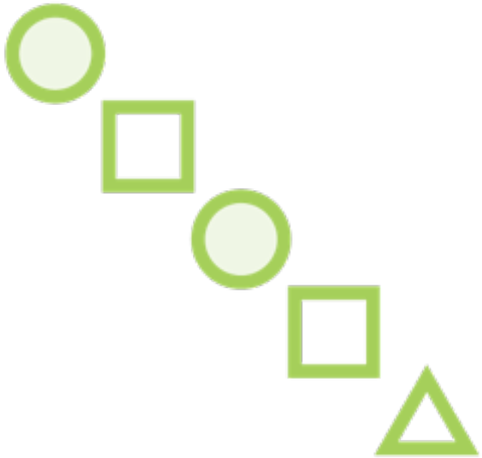
Streaming data

Spark Streaming Module



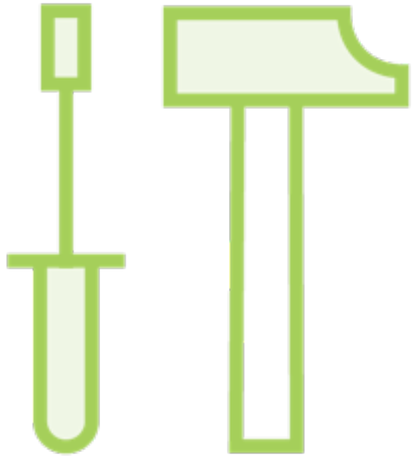
Stream processing

Spark Streaming Module



Streaming Data

Entities are available over a period of time i.e. bank transaction logs



Stream Processing

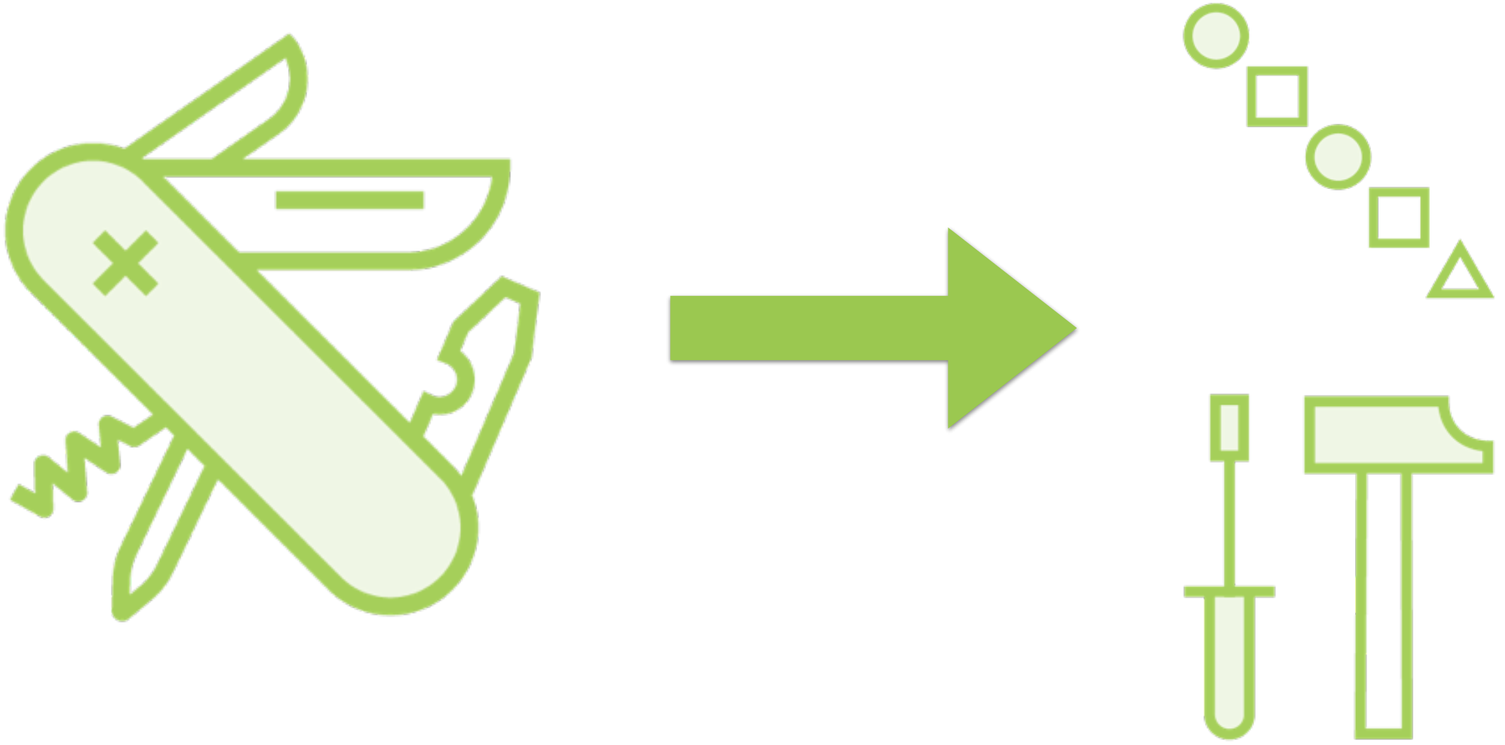
Operations performed on streaming data to extract useful information

Spark Streaming Module

Tracks statistics, trains ML models, detects anomalies in real-time

Runs on the same concepts and APIs used by Spark's batch processing system

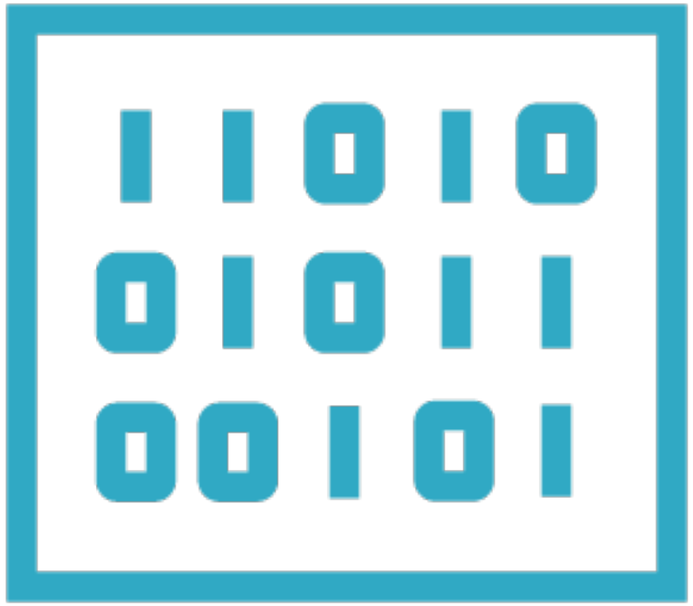
Apache Spark



This course focuses on the Spark Streaming module

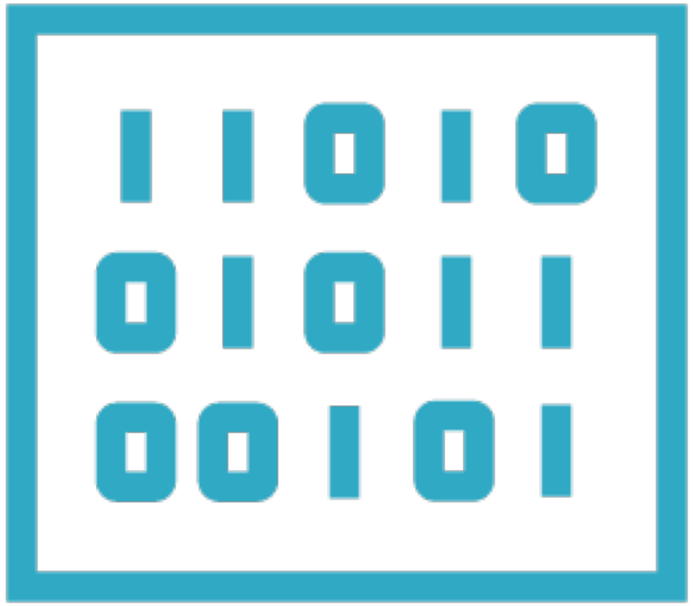
The Spark Programming Abstraction

Resilient Distributed Datasets



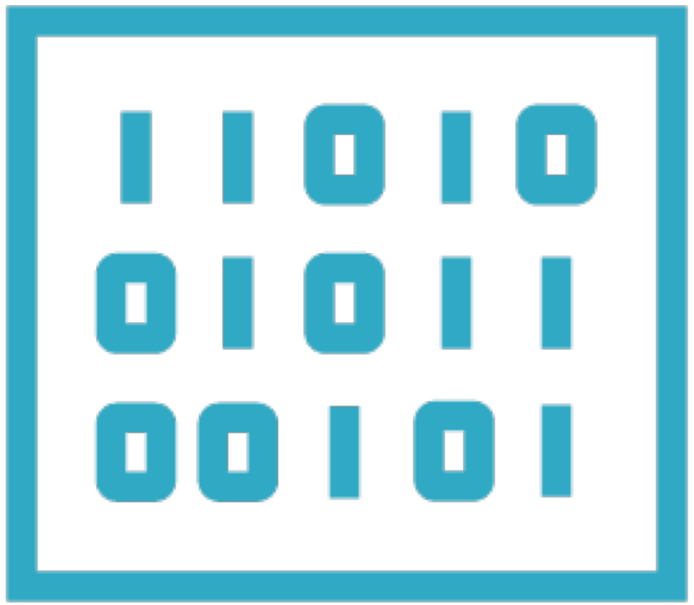
All operations in Spark are performed on in-memory objects

Resilient Distributed Datasets



An RDD is a collection of entities
- rows, records

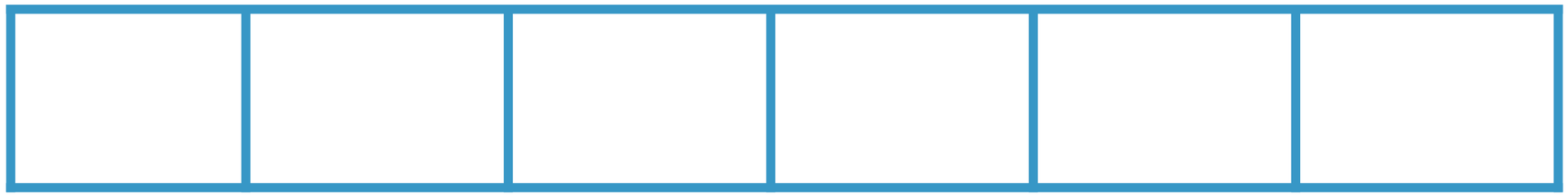
Resilient Distributed Datasets



An RDD in Spark is analogous to a Collection object in Java

It can be assigned to a variable and methods can be invoked on it

Methods return values or apply transformations on the RDDs



```
airlines = sc.textFile(airlinesDataPath)
```

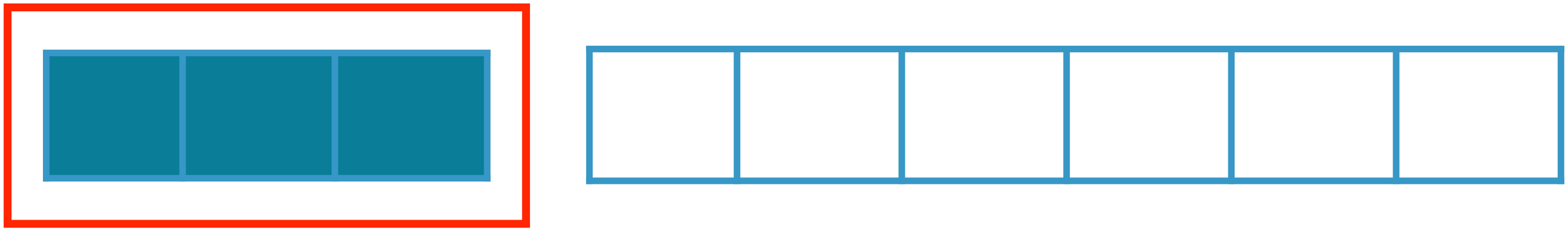
Create an RDD with Flight Data

Create an RDD from raw data in an input file

```
airlinesFiltered = airlines.filter(lambda x:
'Lufthansa' not in x)
```

Create a New RDD with Filtered Flight Data

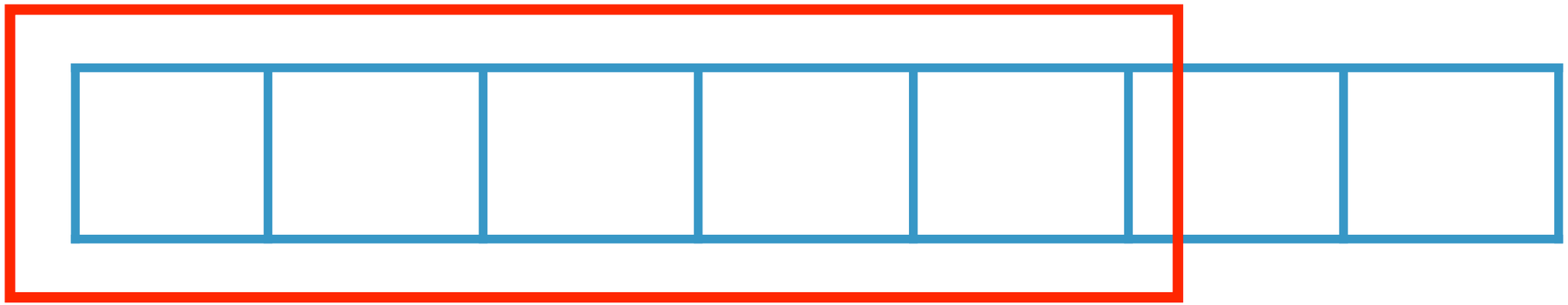
Leave out the rows referring to Lufthansa from the original RDD



```
airlinesFiltered = airlines.filter(lambda x:
'Lufthansa' not in x)
```

Create a New RDD with Filtered Flight Data

Leave out the rows referring to Lufthansa from the original RDD



```
airlinesFiltered.take(5)
```

View the First 5 Rows in the RDD

The rows will be displayed on screen

Characteristics of RDDs

Partitioned

Split across data nodes in a cluster

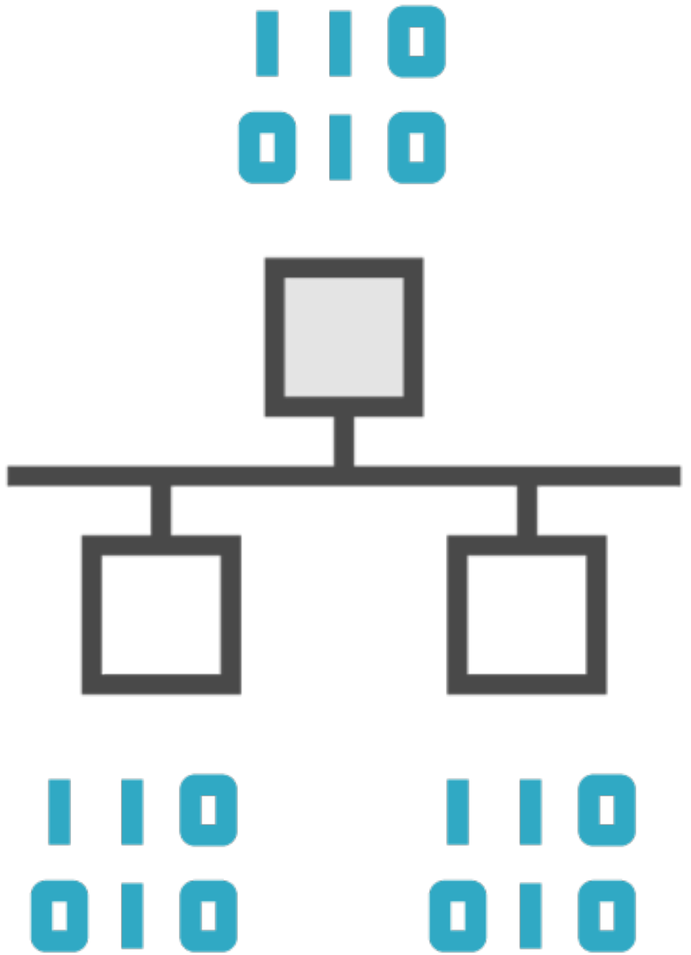
Immutable

RDDs, once created, cannot be changed

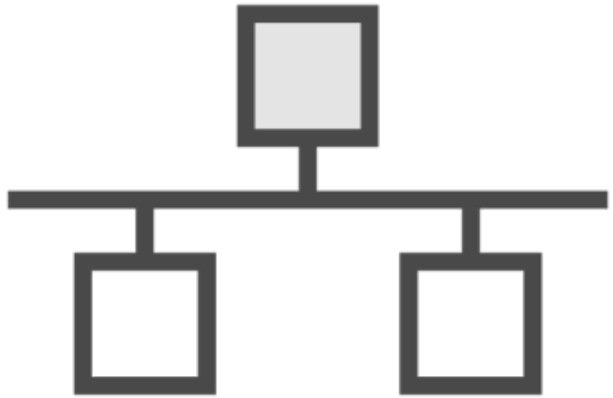
Resilient

Can be reconstructed even if a node crashes

Partitioned



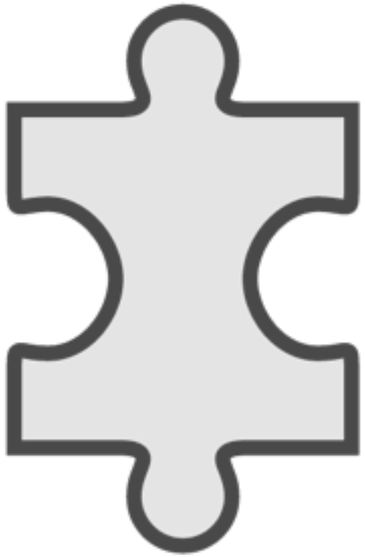
Partitioned



Processing occurs on nodes in parallel

Data is stored in memory for each node in the cluster

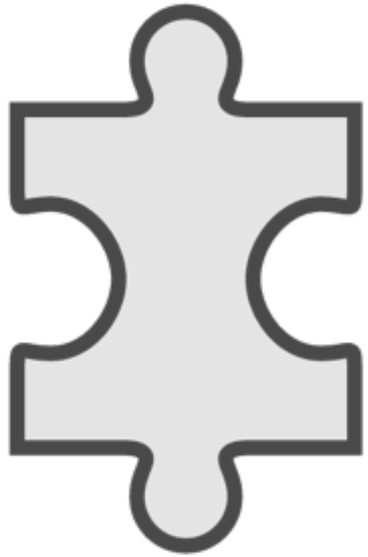
Immutable



An RDD cannot be mutated

Only two operations are permitted on an RDD

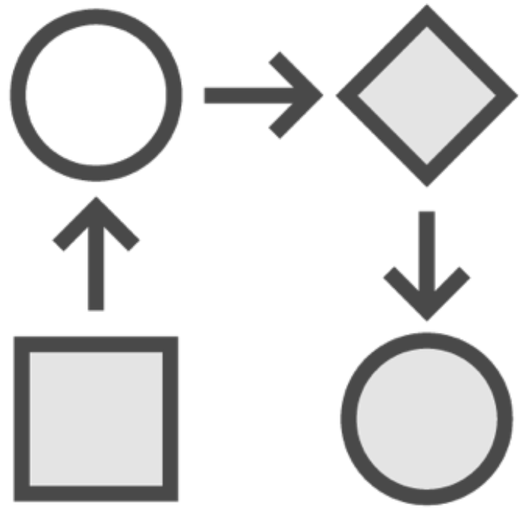
Immutable



Action: Read data from an RDD

Transformation: Transform the RDD to create another RDD

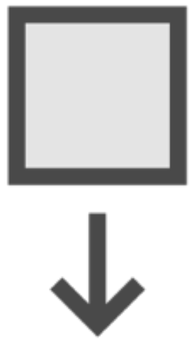
Resilient



RDDs can be reconstructed even if the node it lives on crashes

RDDs Are Resilient

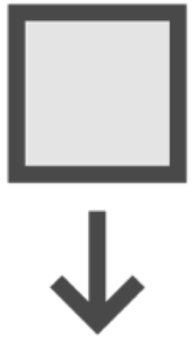
RDDs can be created in 2 ways



Reading a file

Transforming another RDD

RDDs Are Resilient



Reading a file

Transforming another RDD

Every RDD keeps track of where it came from

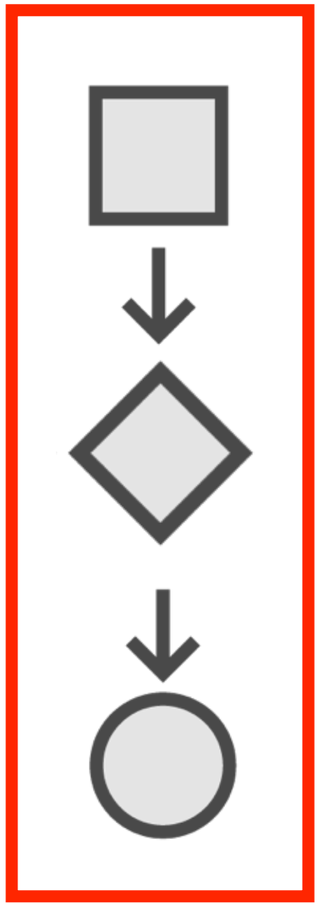
RDDs Are Resilient

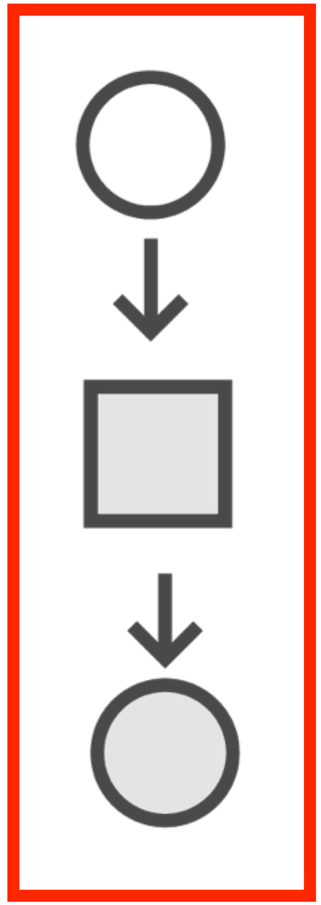
It tracks every transformation which led to the current RDD

RDDs Are Resilient

However many transformations it takes

RDDs Are Resilient



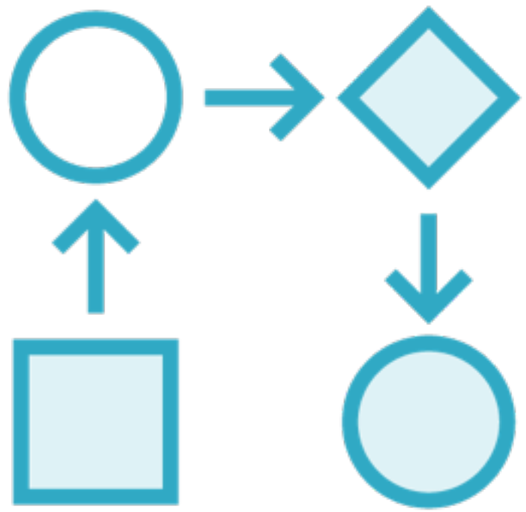


This is the RDD's lineage

RDDs Are Resilient

None of the transformations are applied till we access the results

Lineage



Allows RDDs to be reconstructed when nodes crash

Allows RDDs to be lazily instantiated (materialized) when accessing the results

Characteristics of RDDs

Partitioned

Split across data nodes in a cluster

Immutable

RDD once created cannot be changed

Resilient

Can be reconstructed even if a node crashers

Demo

Spark operations on RDDs in the Pyspark interactive shell

Discretized Streams

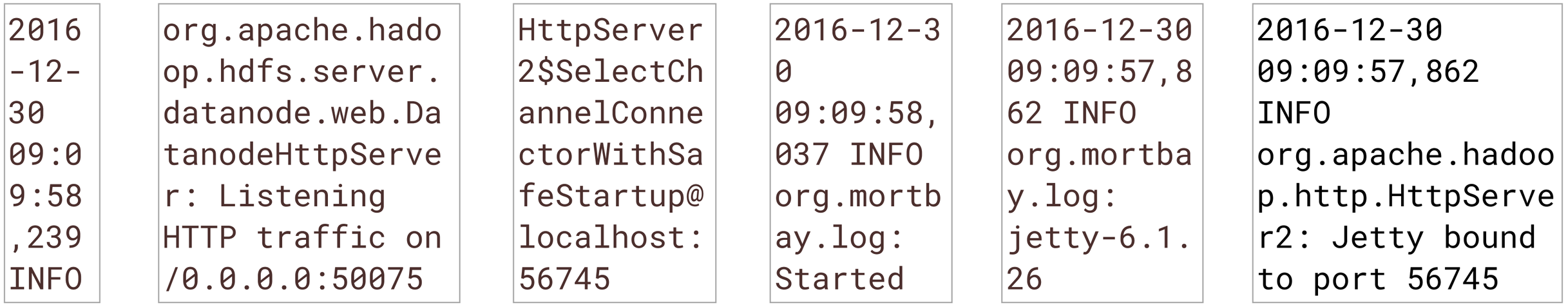
Streaming Data

Log messages received from a server can be thought of as a stream

Streaming Data

```
2016-12-30 09:09:57,862 INFO
org.apache.hadoop.http.HttpServer2: Jetty bound to port
56745
2016-12-30 09:09:57,862 INFO org.mortbay.log: jetty-6.1.26
2016-12-30 09:09:58,037 INFO org.mortbay.log: Started
HttpServer2$SelectChannelConnectorWithSafeStartup@localhost:
56745
2016-12-30 09:09:58,124 INFO
org.apache.hadoop.hdfs.server.datanode.web.DatanodeHttpServe
r: Listening HTTP traffic on /0.0.0.0:50075
2016-12-30 09:09:58,239 INFO
```

Streaming Data



Each message is one entity in this stream

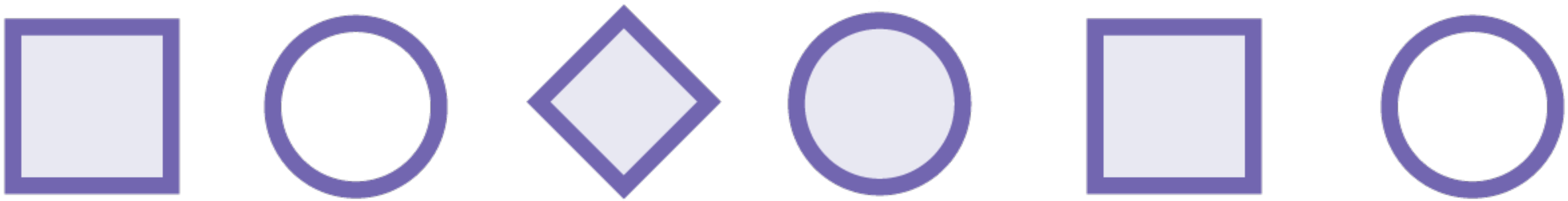
Streaming Data

2016-12-30 09:09:58,239 INFO	org.apache.hadoop.hdfs.server.datanode.web.DatanodeHttpServer: Listening HTTP traffic on /0.0.0.0:50075	HttpServer2$SelectChannelConnectorWithSafeStartup@localhost:56745	2016-12-30 09:09:58,037 INFO org.mortbay.log: Started	2016-12-30 09:09:57,862 INFO org.mortbay.log: jetty-6.1.26	2016-12-30 09:09:57,862 INFO org.apache.hadoop.http.HttpServer2: Jetty bound to port 56745

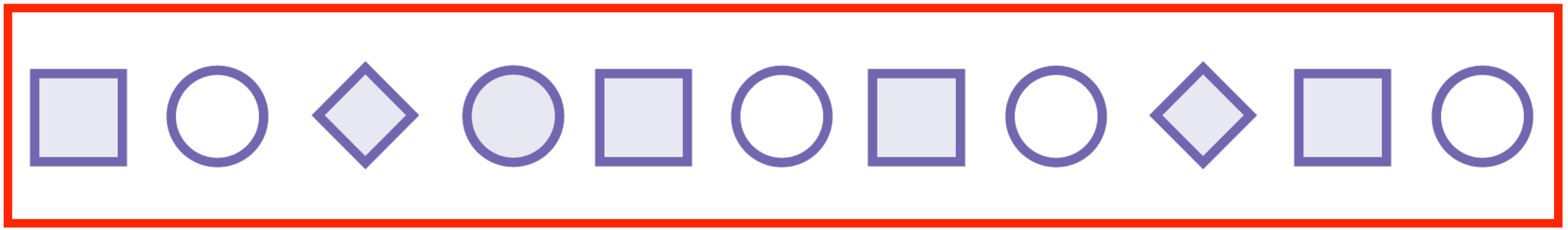
Spark works with stream data using the same batch RDD abstraction

Streaming Data

2016-12-30 09:09:58,239 INFO	org.apache.hadoop.hdfs.server.datanode.web.DatanodeHttpServer: Listening HTTP traffic on /0.0.0.0:50075	HttpServer2$SelectChannelConnectorWithSafeStartup@localhost:56745	2016-12-30 09:09:58,037 INFO org.mortbay.log: Started	2016-12-30 09:09:57,862 INFO org.mortbay.log: jetty-6.1.26	2016-12-30 09:09:57,862 INFO org.apache.hadoop.http.HttpServer2: Jetty bound to port 56745



Streaming Data



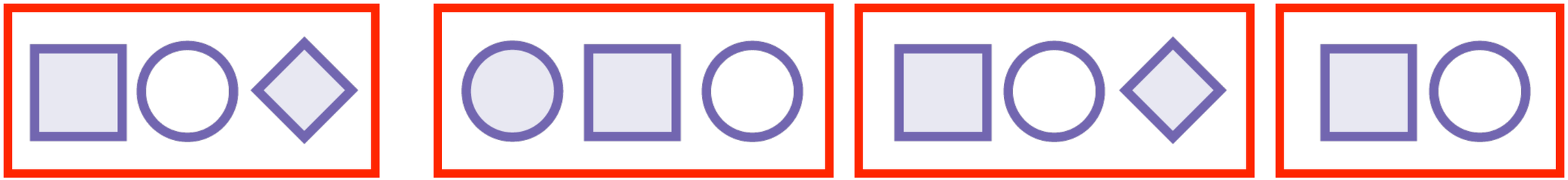
This stream of entities is represented as a discretized stream or DStream

DStream = Sequence of RDDs

Streaming Data

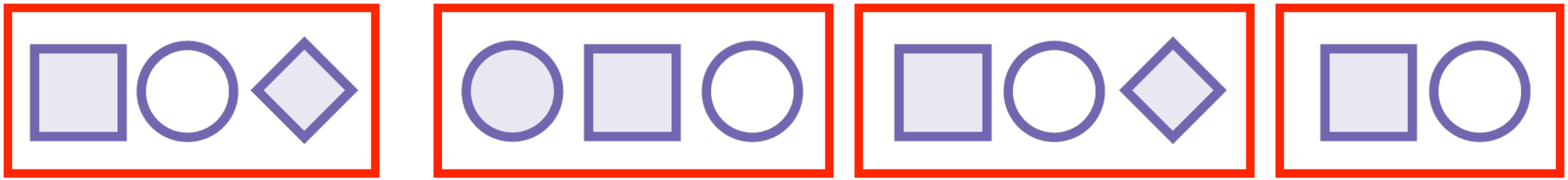
Entities => RDDs => DStream ?

Streaming Data



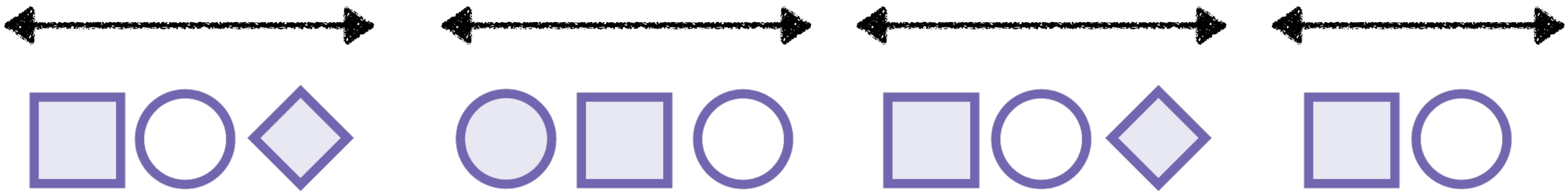
Entities in a Stream are grouped into batches

Streaming Data



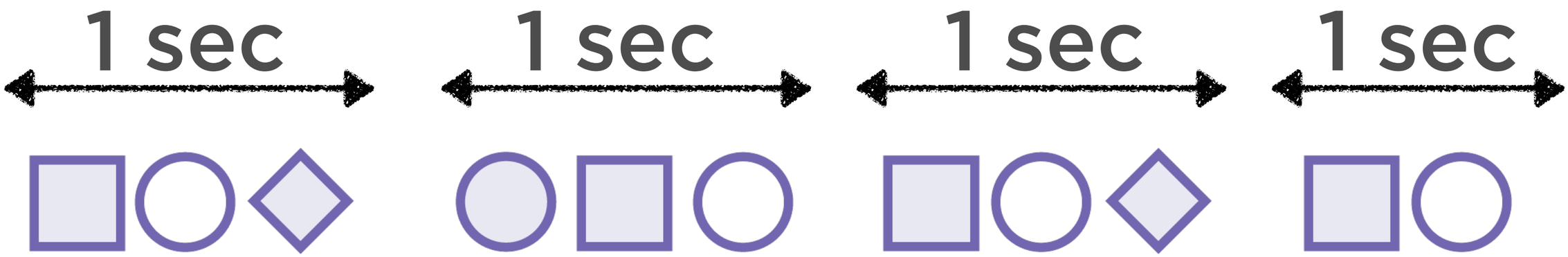
Each batch = 1 RDD

Streaming Data



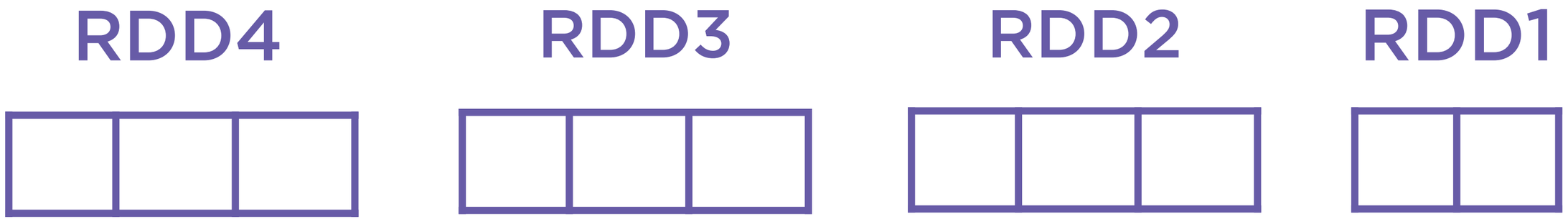
Batches are formed based on a batch interval

Streaming Data



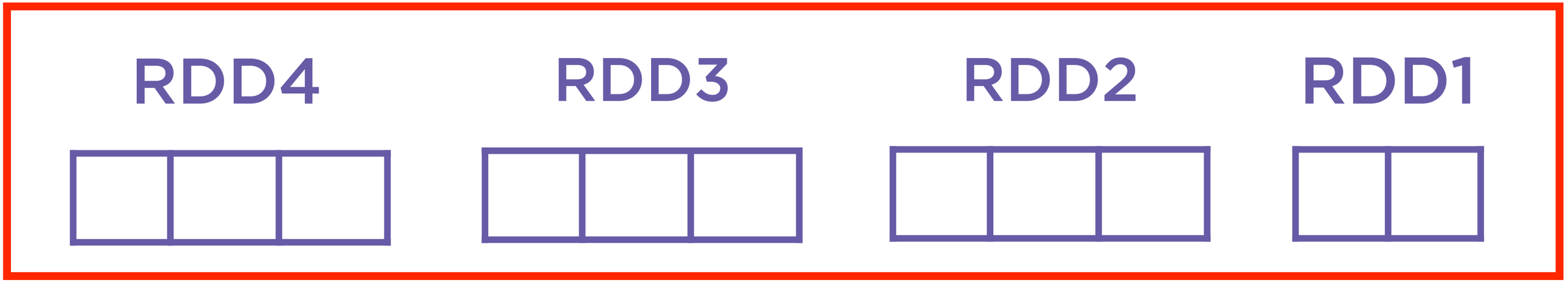
All logs received within the batch interval make one RDD

Streaming Data



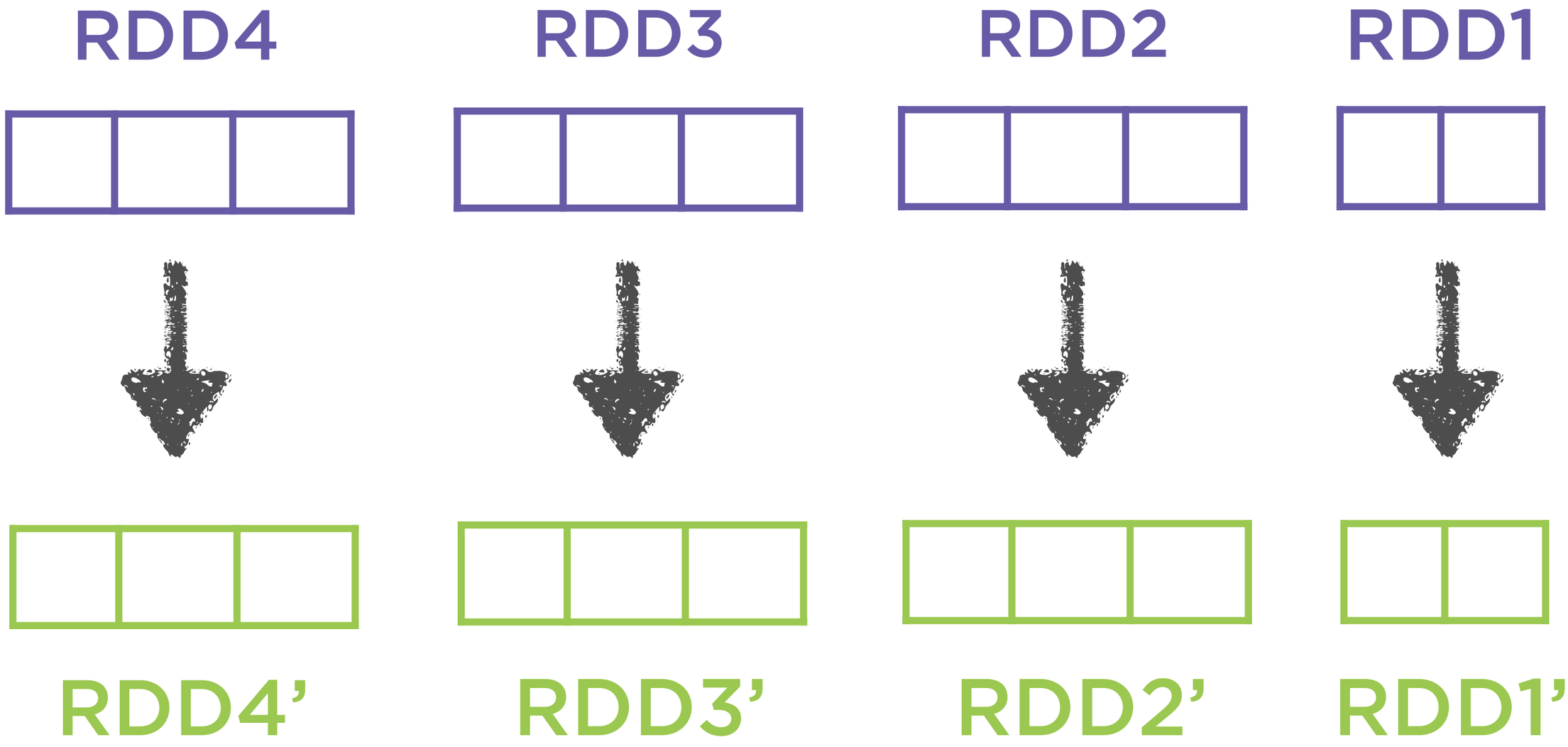
All logs received within the batch interval make one RDD

Streaming Data

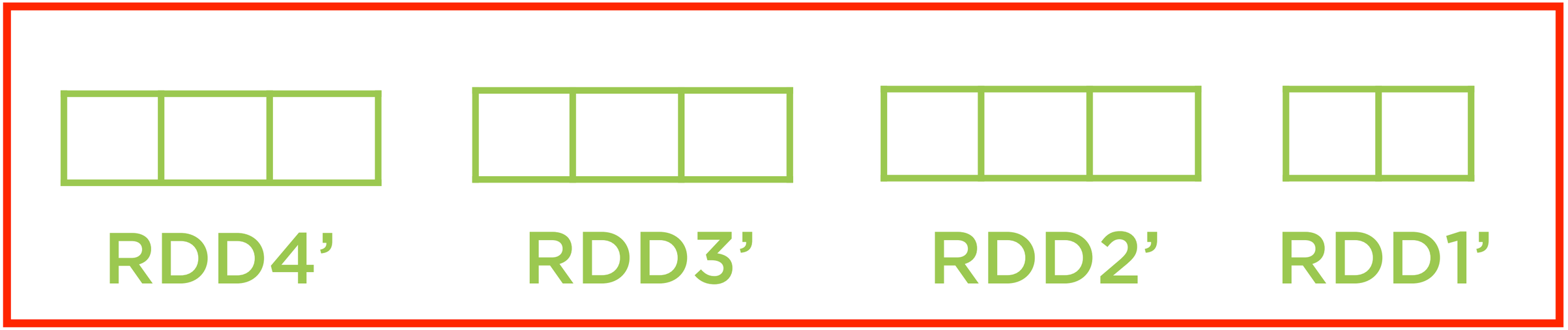


Within a DStream, Spark still performs operations on individual RDDs

Streaming Data



Streaming Data



Transformed DStream

Within DStreams, Spark does **batch processing on individual RDDs**

Demo

Listen for streaming text data on a host and port

Track error messages received using DStream transformations

Overview

Understood the role of the Spark streaming module in stream processing

Understood the structure of Spark DStreams as a sequence of RDDs

Implemented a basic streaming application in Python