Big Data Applications with Java

Who I am ?

Team Lead at THE STARS GROUP

Stefan Angelov Angelov

Master degree in Technical University Sofia

ТЕХНИЧЕСКИ УНИВЕРСИТЕТ СОФИЯ

Agenda

- What is Big Data?
- Apache Hadoop
- Apache Spark
- Apache Kafka
- Apache Cassandra
- Hadoop VS Apache Spark

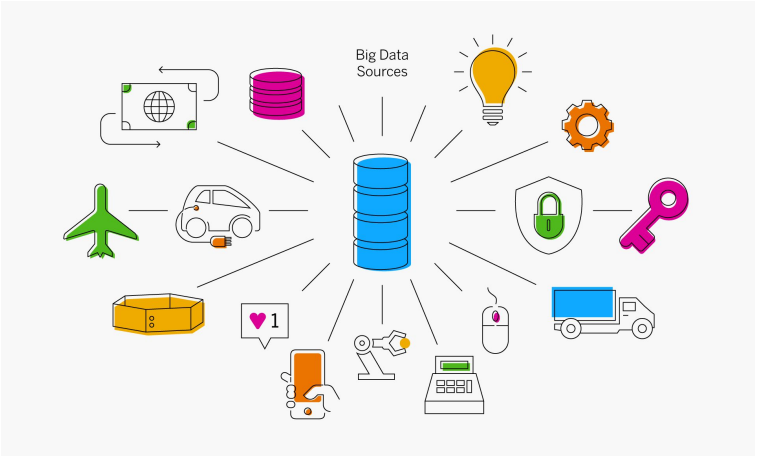
“Information is the oil of the 21st century, and analytics is the combustion engine”

— Peter Sondergaard, Senior Vice President, Gartner

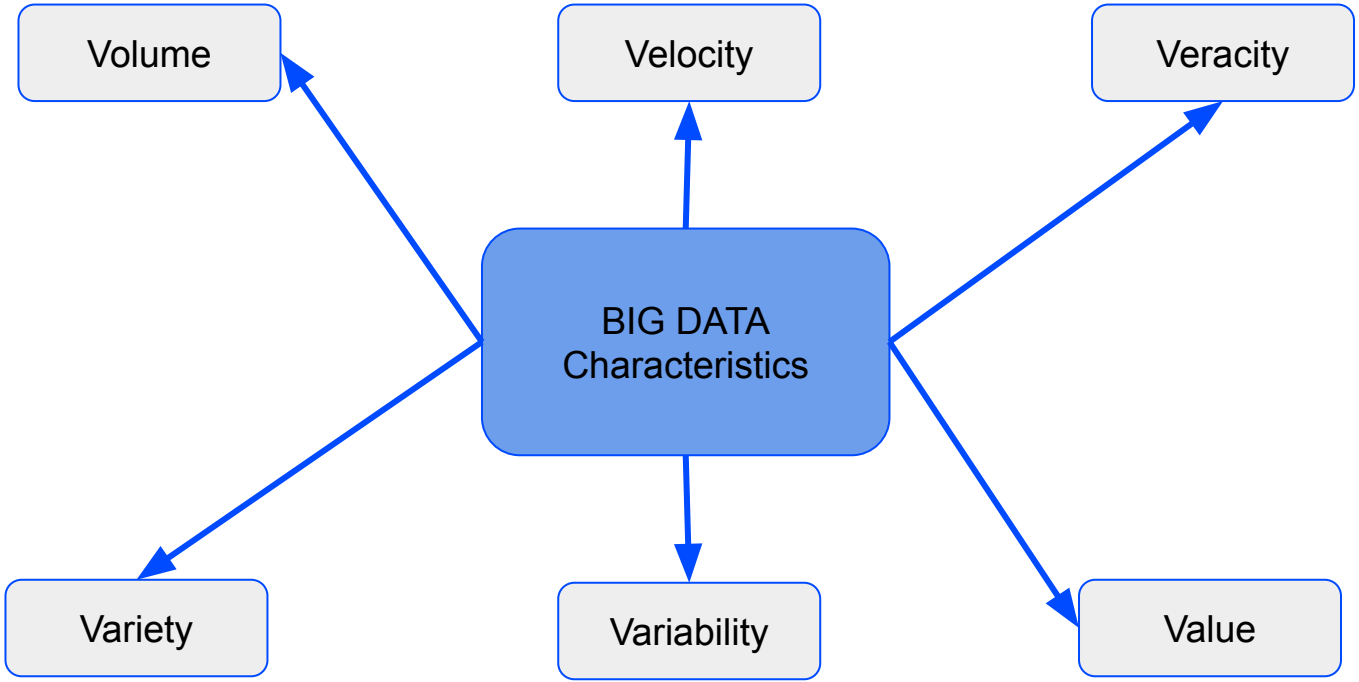
What is Big Data?

What is Big Data?

Huge volume of data that cannot be stored and processed using the traditional approach. As this data may contain valuable information, it needs to be processed in a short span of time. This valuable information can be used to make predictive analyses, as well as for marketing and many other purposes. If we use the traditional approach, we will not be able to accomplish this task within the given time frame, as the storage and processing capacity would not be sufficient for these types of tasks.

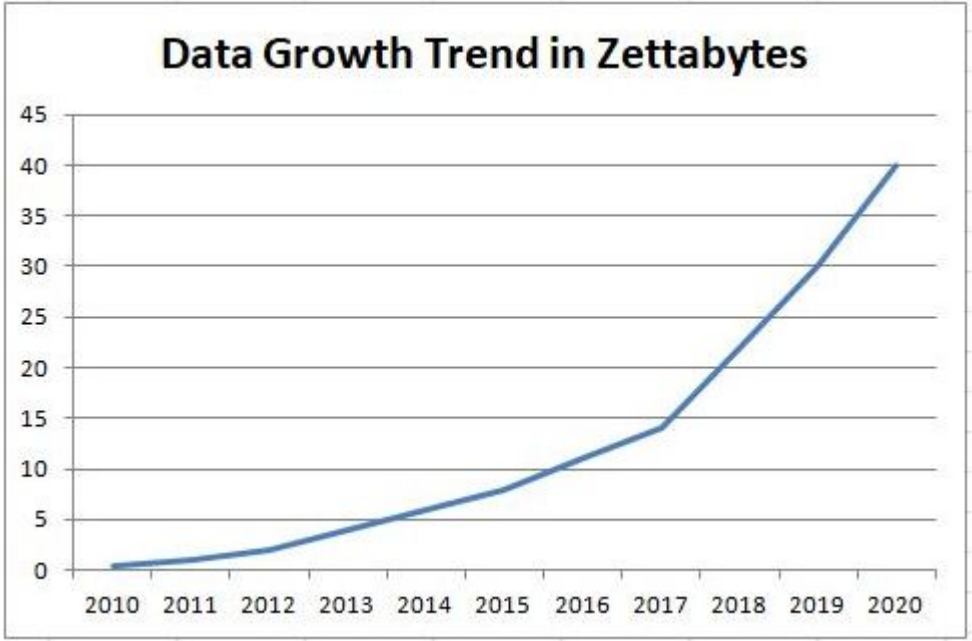


Characteristics of big data



Volume

When we talk about volume in a big data context, it is an amount of data that is massive with respect to the processing system that cannot be gathered, stored, and processed using traditional approaches. It is data at rest that is already collected and streaming data that is continuously being generated.



Velocity

Velocity is the rate at which the data is being generated, or how fast the data is coming in.

- The New York stock exchange captures 1 TB of data during each trading session.
- 120 hours of videos are being uploaded to YouTube every minute.
- Data generated by modern cars; they have almost 100 sensors to monitor each item from fuel and tire pressure to surrounding obstacles.
- 200 million emails are sent every minute.

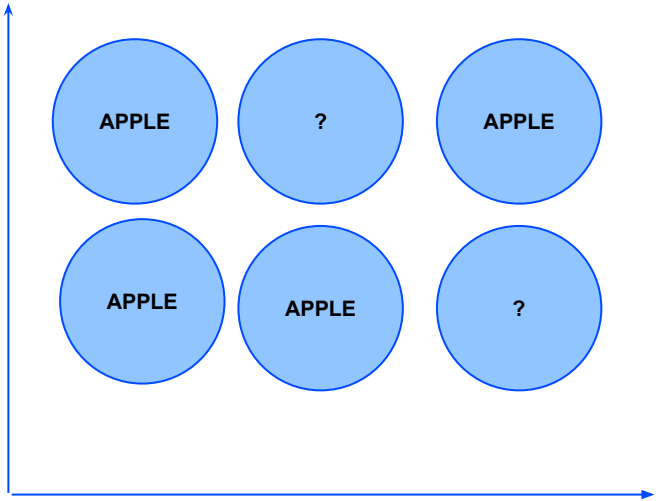
Variety

Velocity is the rate at which the data is being generated, or how fast the data is coming in.

- Structured data
- Unstructured data
- Semi structured data

Veracity

This vector deals with the uncertainty of data. It may be because of poor data quality or because of the noise in data. It's human behavior that we don't trust the information provided. This is one of the reasons that one in three business leaders don't trust the information they use for making decisions.

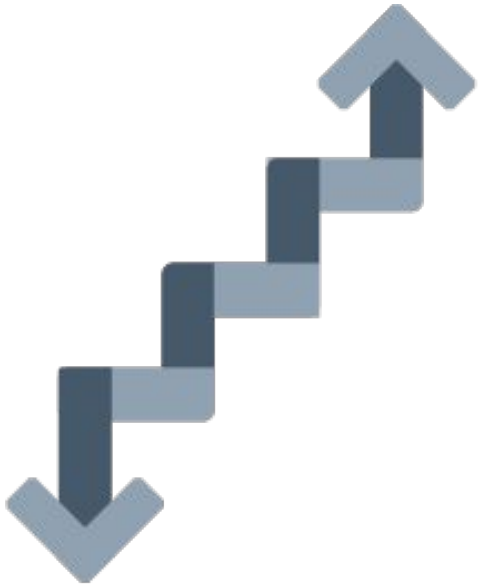


Variability

“ lack of consistency can bring on a lack of interest”

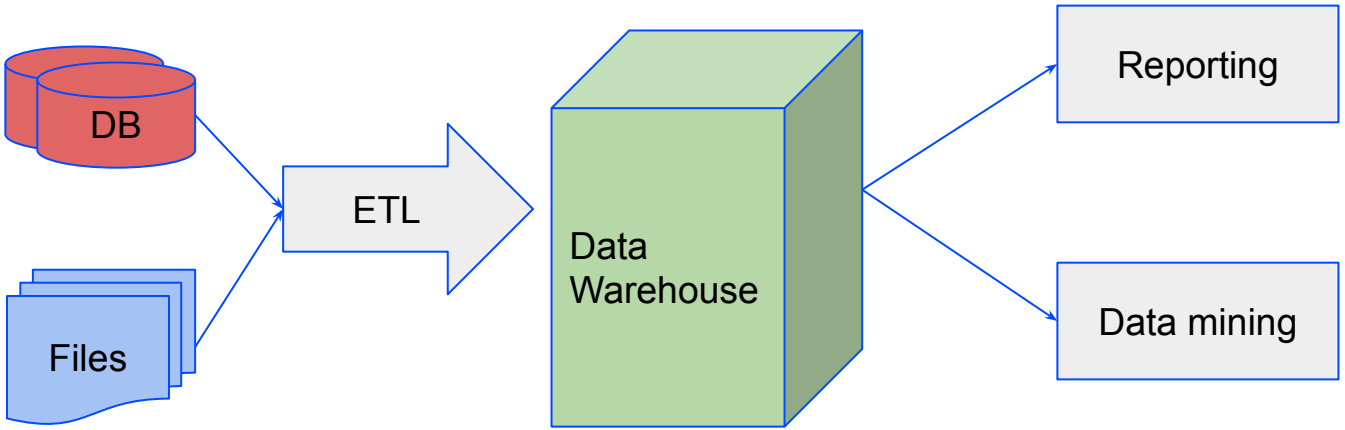
Value

10%
lower costs
and...



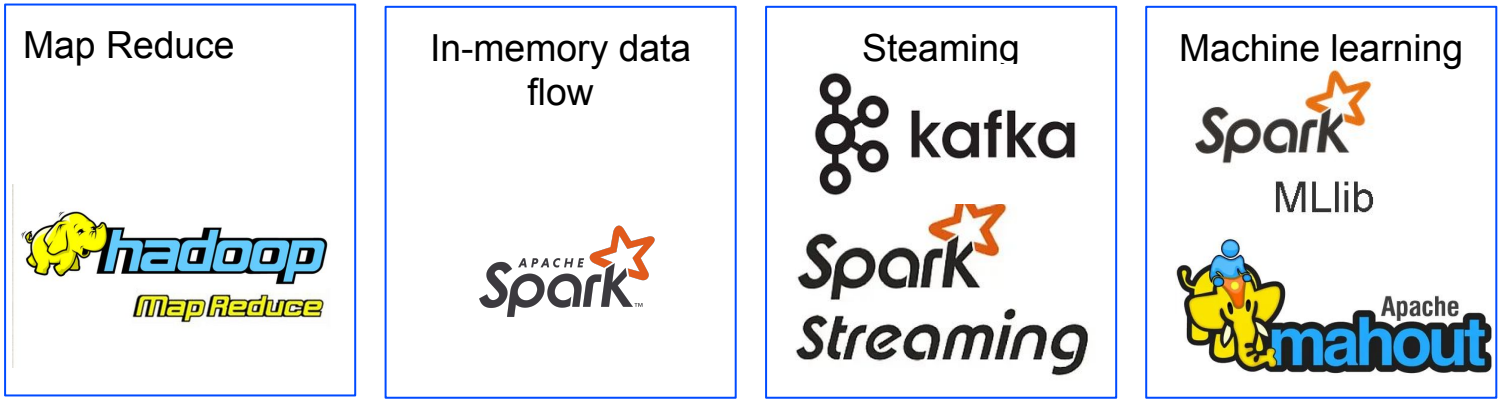
8%
higher revenues
due to
big data.

Traditional approaches to data storage



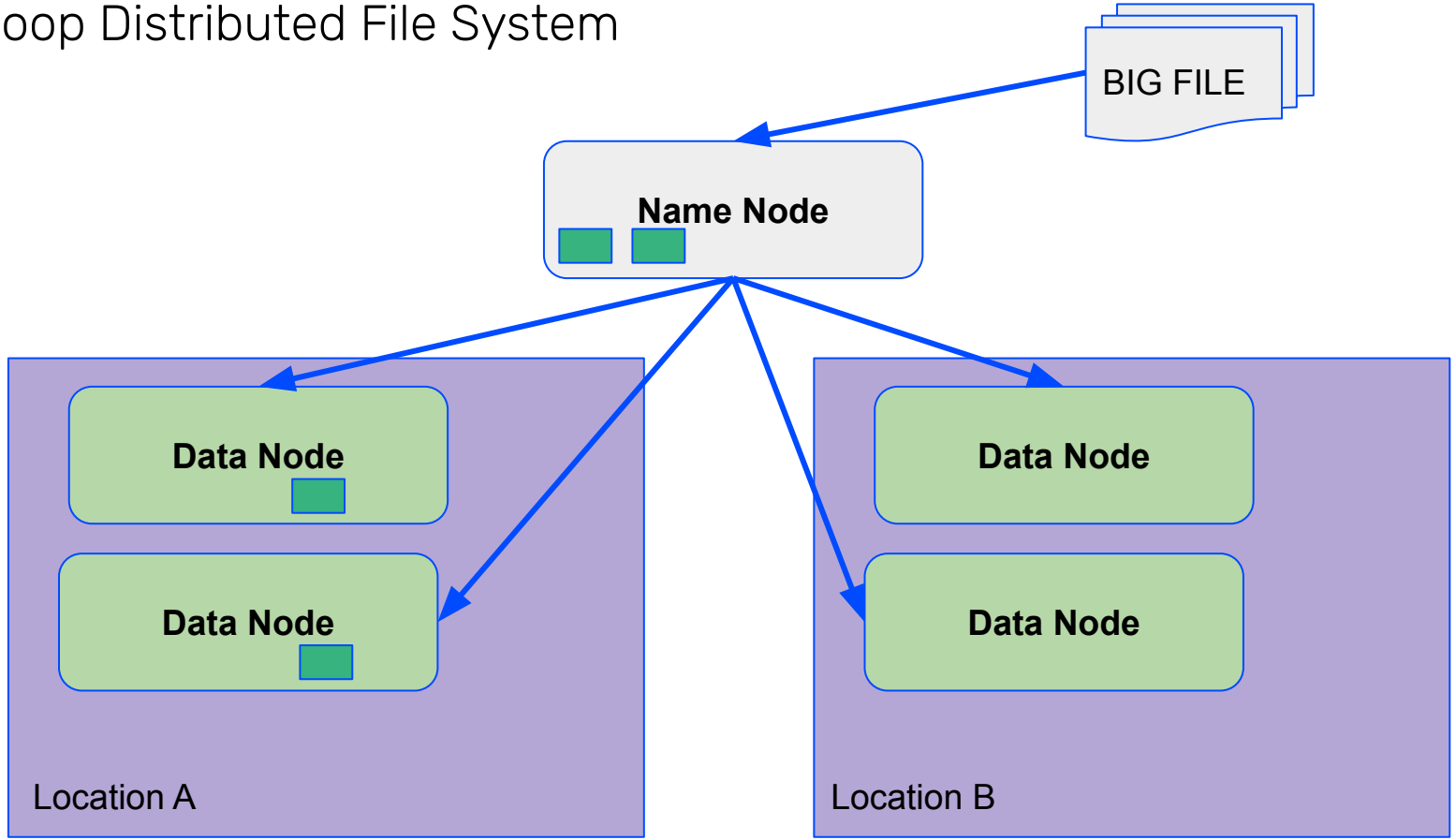
Apache Hadoop

Hadoop Ecosystem



Apache Hadoop

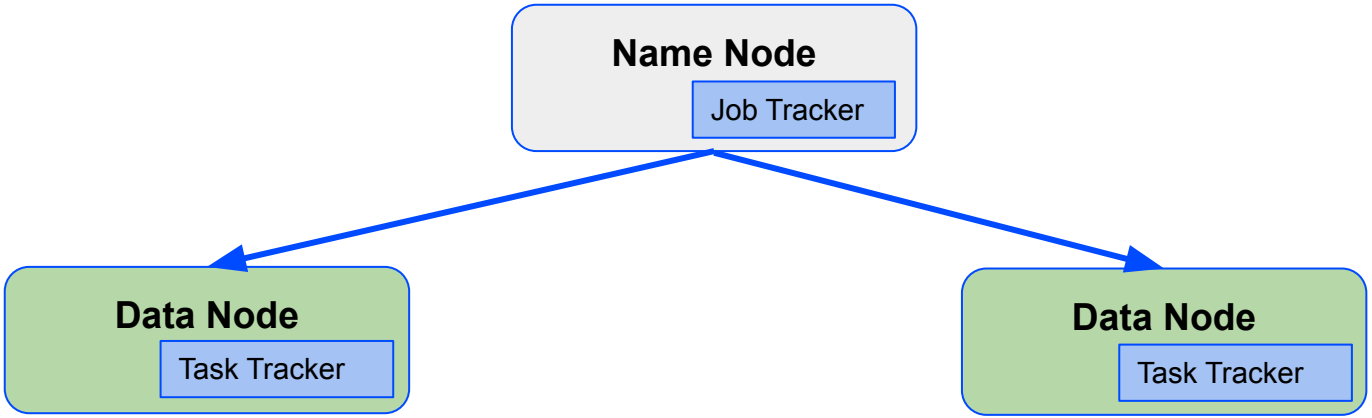
Hadoop Distributed File System



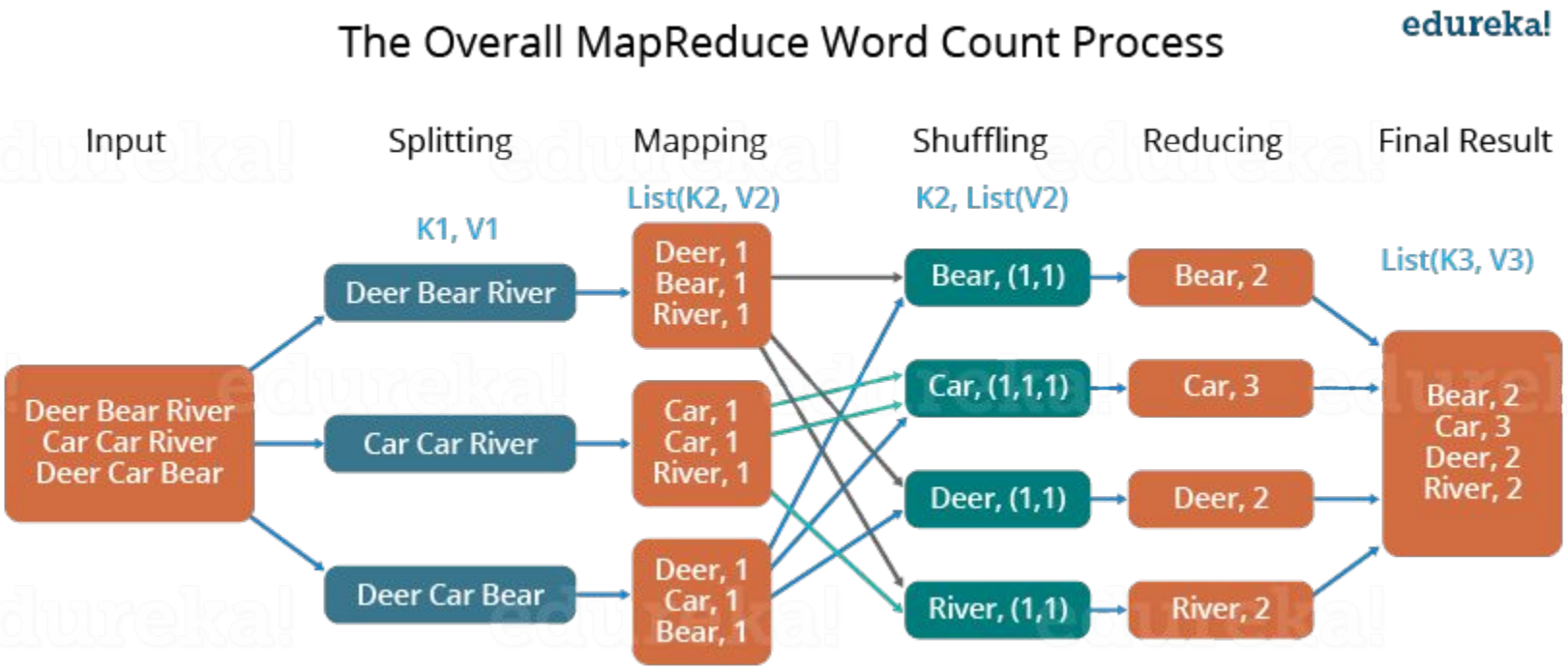
Hadoop Distributed File System


```
docker exec -it namenode /bin/bash
hdfs dfs -ls /
hdfs dfs -mkdir -p /user/stefan/softuni
hdfs dfs -put <file_name> <path>
hdfs dfs -cat <input_file
docker cp <file_name> namenode:/<path>
```

Hadoop MapReduce



Hadoop MapReduce – WordCounterExample

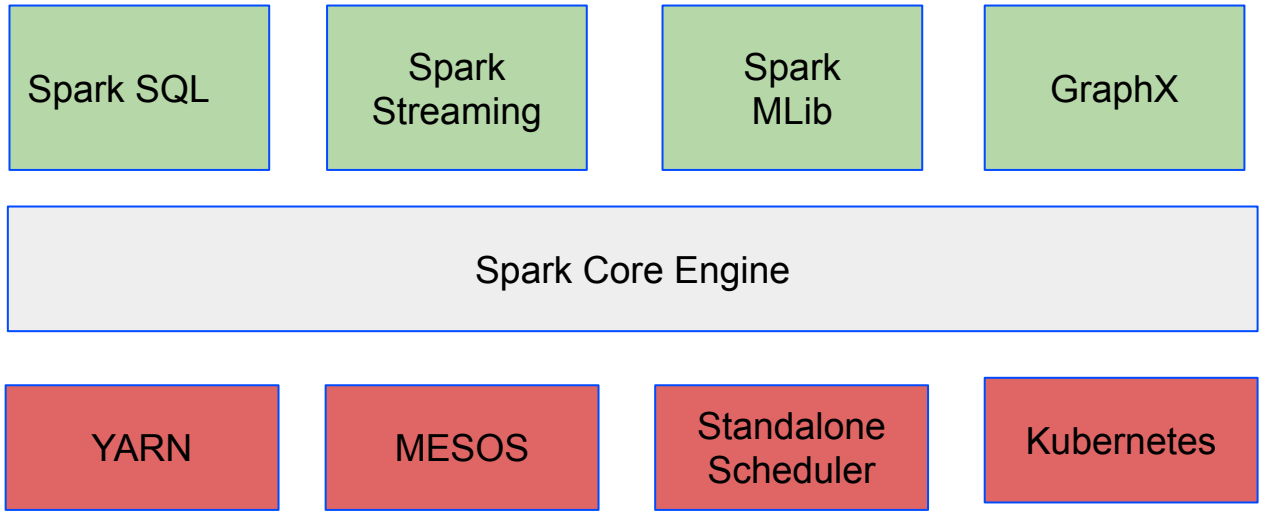


Apache Spark

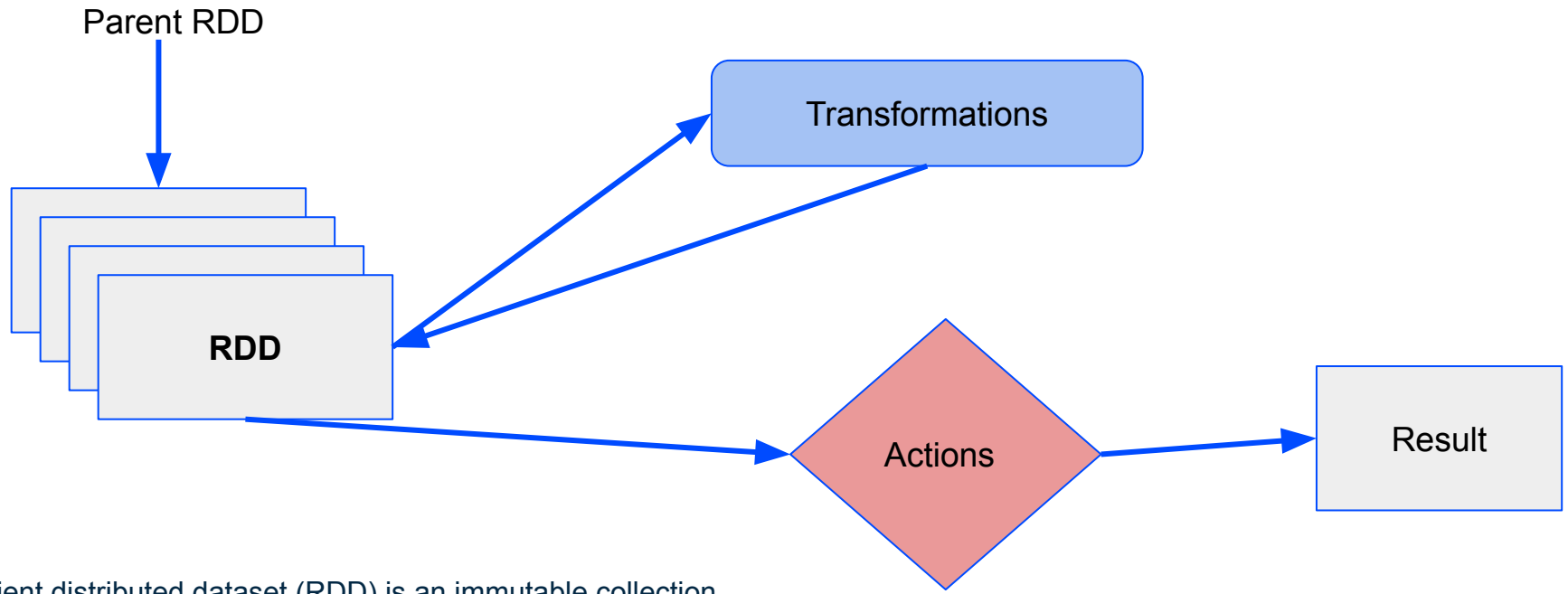
Apache Spark

Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size. It provides development APIs in Java, Scala, Python and R, and supports code reuse across multiple workloads—batch processing, interactive queries, real-time analytics, machine learning, and graph processing. You'll find it used by organizations from any industry, including at FINRA, Yelp, Zillow, DataXu, Urban Institute, and CrowdStrike. Apache Spark has become one of the most popular big data distributed processing framework with 365,000 meetup members in 2017.

Apache Spark

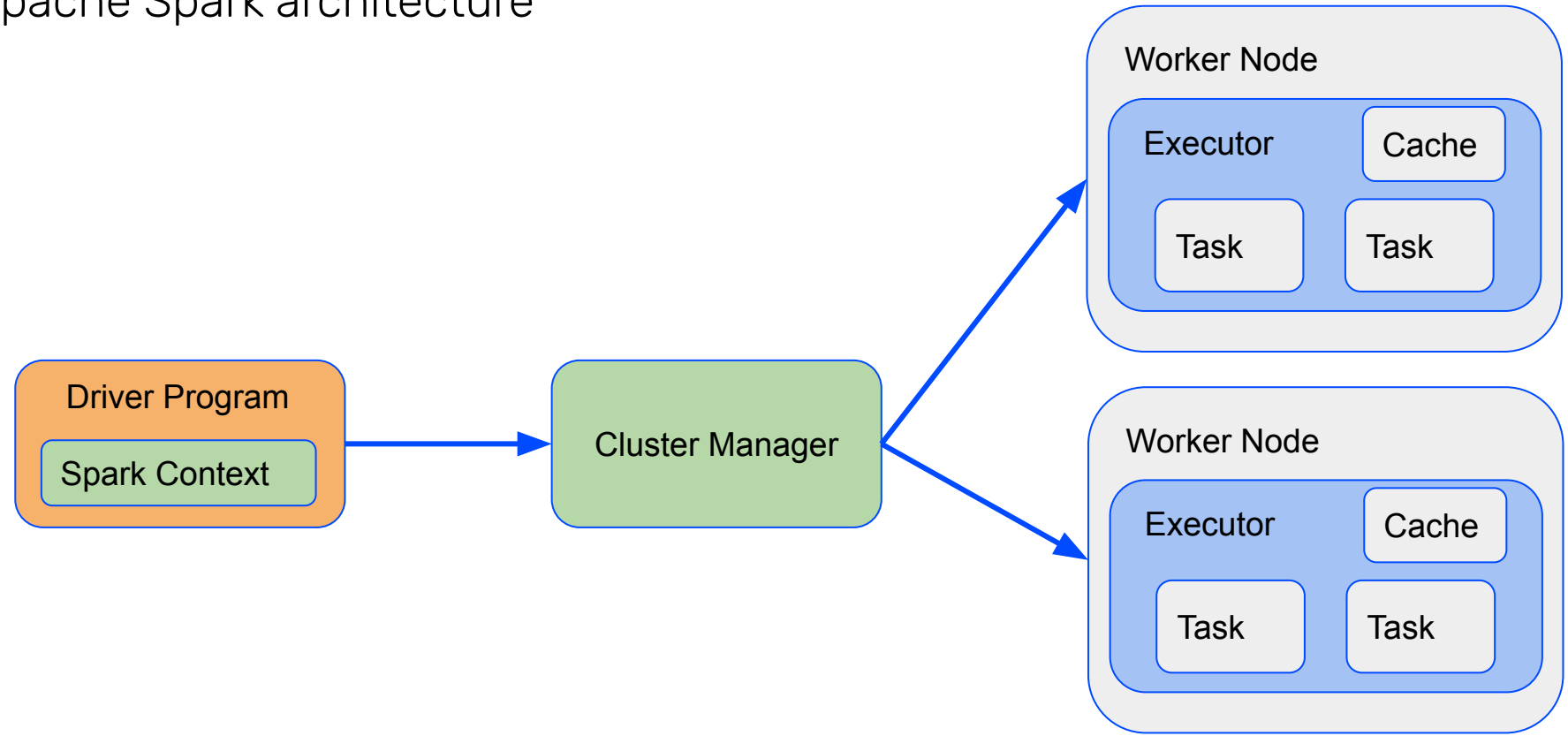


Apache Spark - Concepts

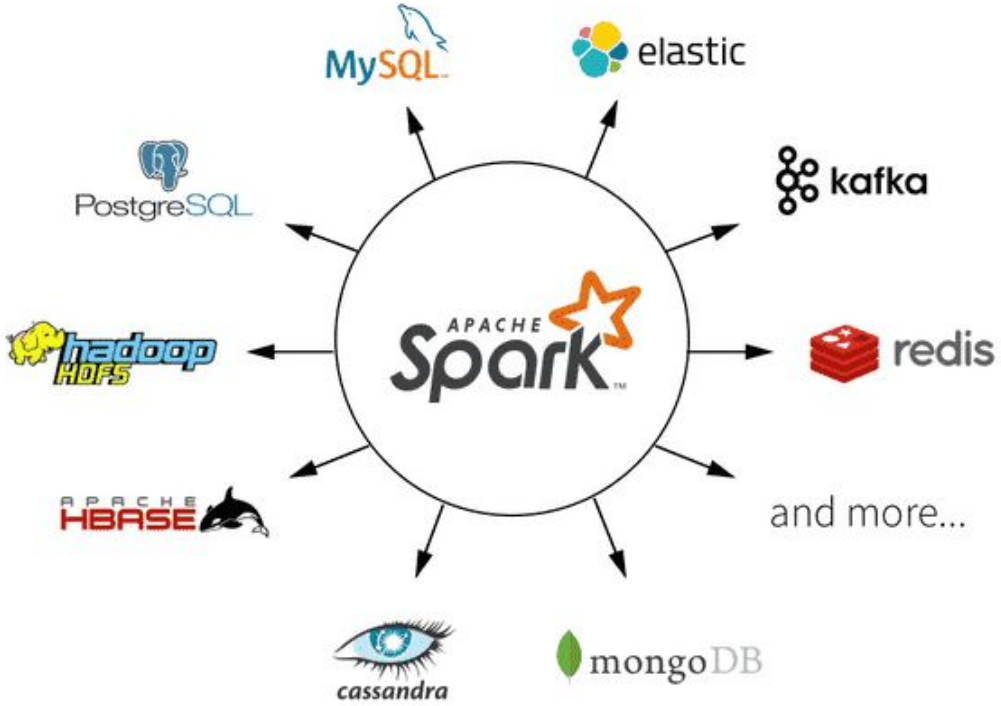


A resilient distributed dataset (RDD) is an immutable collection of objects. These objects are distributed across the different machines available in a cluster.

Apache Spark architecture



Sources

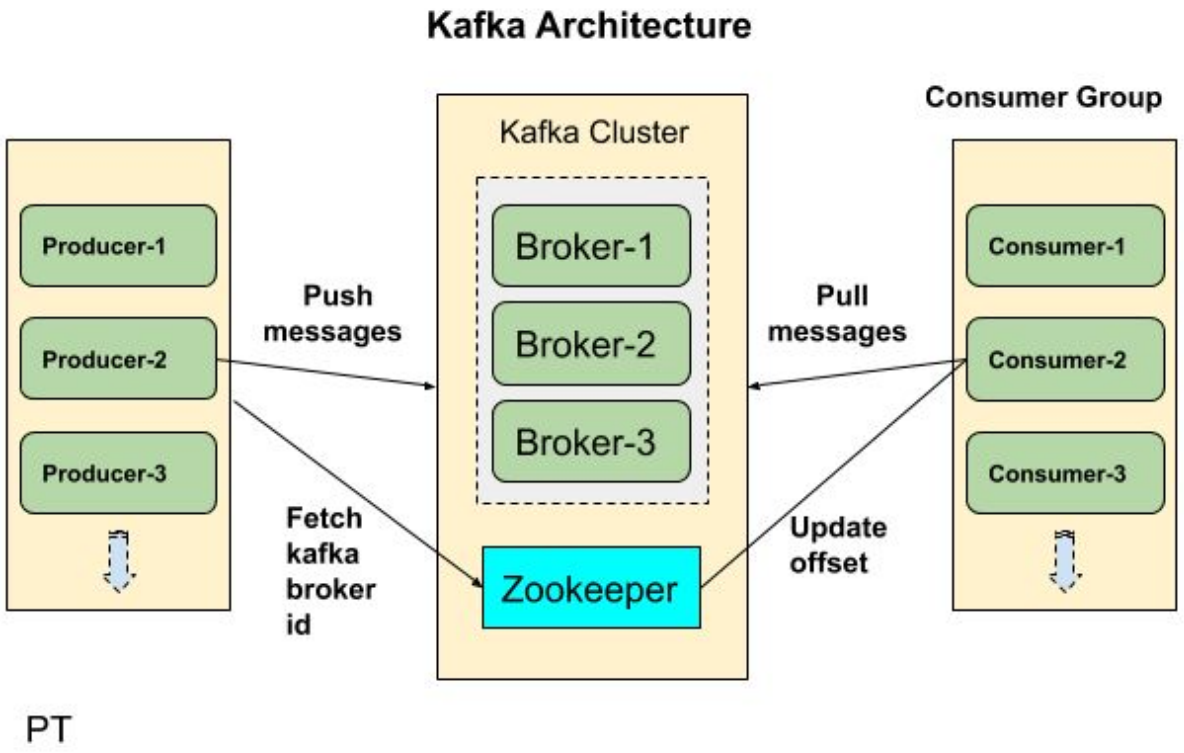

MySQL
elastic
PostgreSQL
kafka
hadoop
HDFS
APACHE
Spark
redis
APACHE
HBASE
and more...
cassandra
mongoDB

Apache Kafka

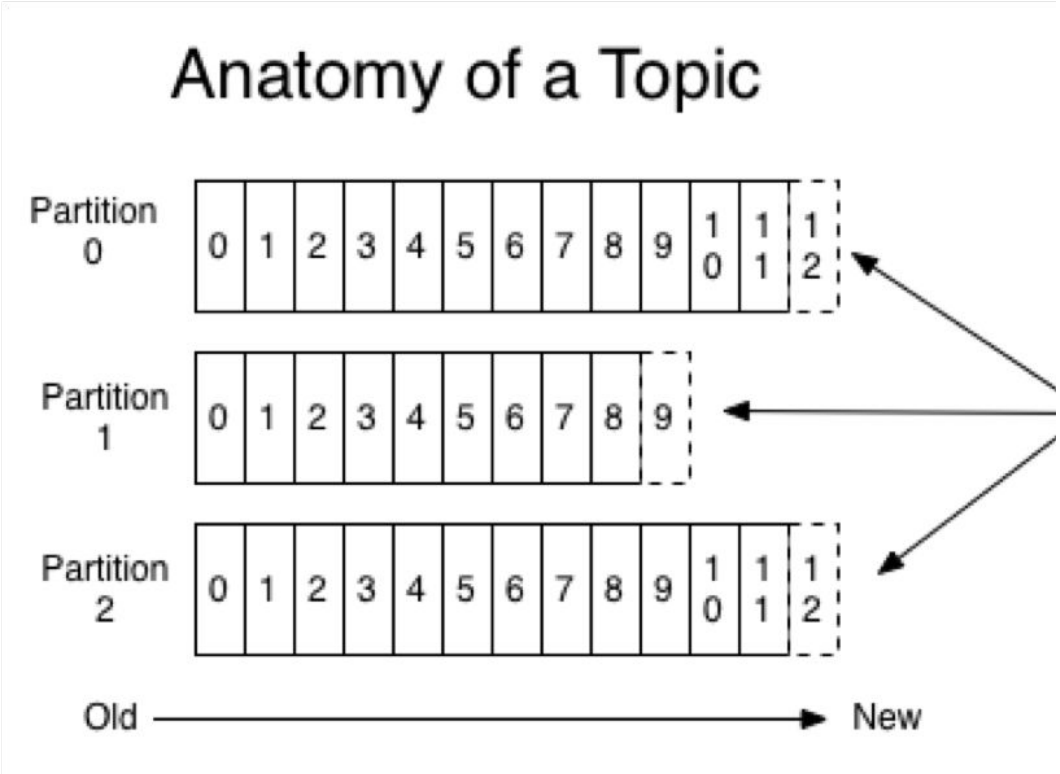
Apache Kafka

Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.

Apache Kafka Architecture



Apache Kafka Architecture



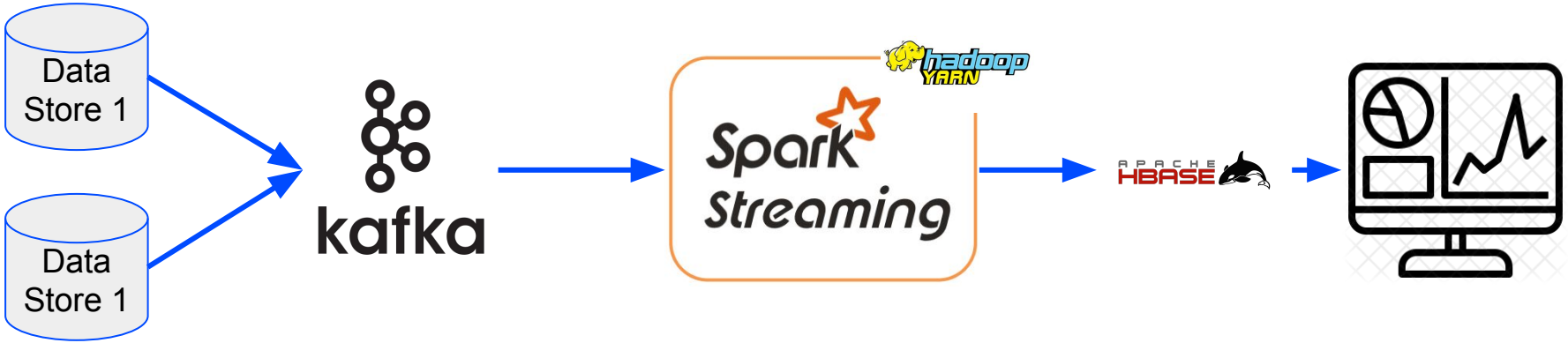
Kafka Big Data Function Applications

- Using Kafka Big Data Function as a Data Source
- Using Kafka Big Data Function as a Data Processor
- Using Kafka Big Data Function to Re-sync Nodes

Using Kafka Big Data Function as a Data Source

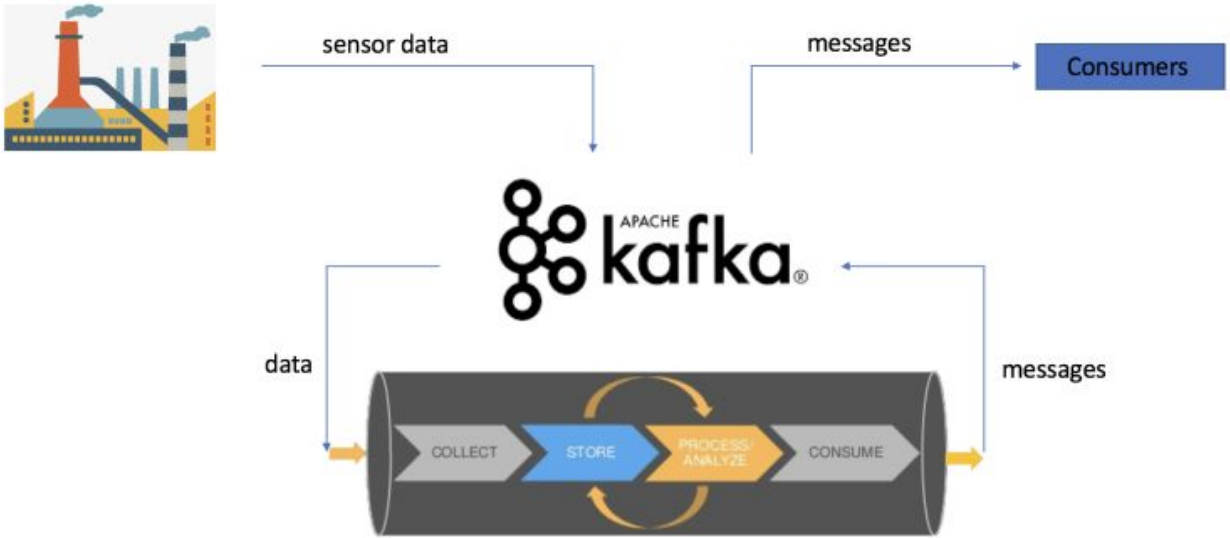
The very first use Kafka is put into is as a Data Broker or Data source. Kafka is used here as a multi-subscription system. The same published data set can be consumed multiple times, by different consumers. Kafka's built-in redundancy offers reliability and availability, at all times.

One popular combination is to use Kafka with Hadoop and Spark, where Hadoop stores the data in HDFS and performs Data Analytics, and Kafka acts as the data aggregator and distributor. You will need to know Kafka internals and configure its parameters for the approach to work.



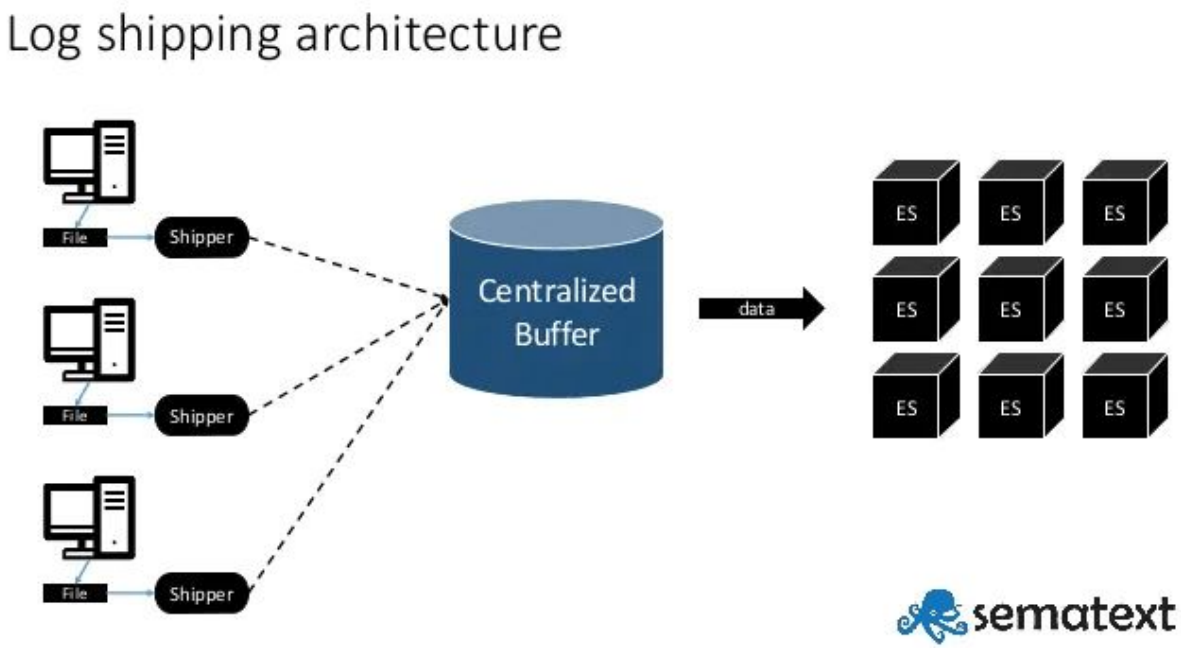
Using Kafka Big Data Function as a Data Processor

The second role that Kafka can play is that of a Data processor and Analyzer. Kafka provides a client library for analyzing data called Kafka streams. It can be used to process and analyze data and send the results to an external application or back to Kafka core.



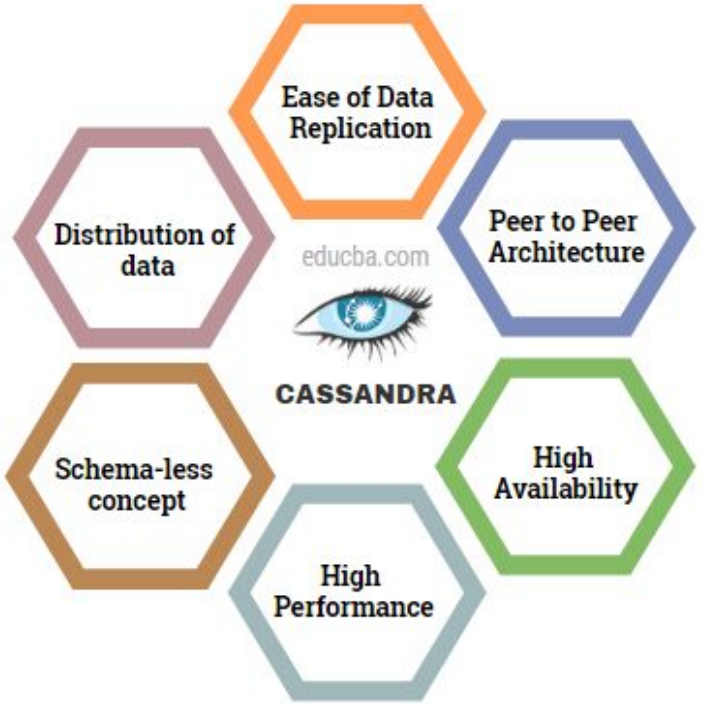
Using Kafka Big Data Function to Re-sync Nodes

Another important use Kafka can be put to is to re-sync your nodes(data stores) and restore the state. You can also use it for Log Aggregation, Messaging, Click-stream Tracking, Audit Trails, and much more.



Apache Cassandra

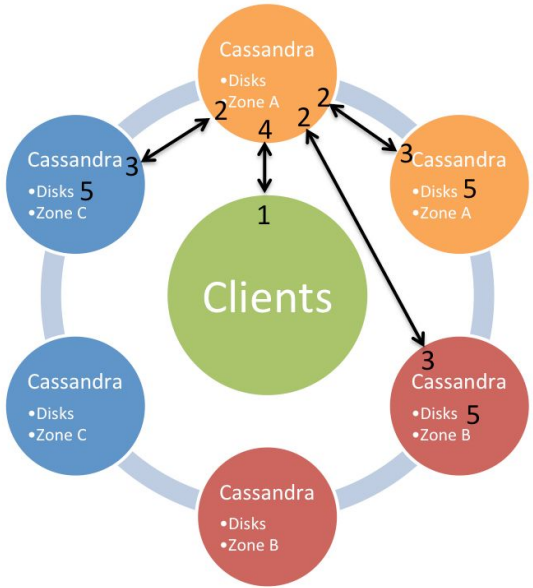
What is Cassandra



Cassandra Write Data Flows

Single Region, Multiple Availability Zone

1. Client Writes to any Cassandra Node
2. Coordinator Node replicates to nodes and Zones
3. Nodes return ack to coordinator
4. Coordinator returns ack to client
5. Data written to internal commit log disk



If a node goes offline, hinted handoff completes the write when the node comes back up.

Requests can choose to wait for one node, a quorum, or all nodes to ack the write

SSTable disk writes and compactions occur asynchronously

NETFLIX

Hadoop VS Apache Spark

Comparing Hadoop and Spark

Spark is a Hadoop enhancement to MapReduce. The primary difference between Spark and MapReduce is that Spark processes and retains data in memory for subsequent steps, whereas MapReduce processes data on disk.

- **Performance**: Spark is faster because it uses random access memory (RAM) instead of reading and writing intermediate data to disks. Hadoop stores data on multiple sources and processes it in batches via MapReduce.

- **Cost**: Hadoop runs at a lower cost since it relies on any disk storage type for data processing. Spark runs at a higher cost because it relies on in-memory computations for real-time data processing, which requires it to use high quantities of RAM to spin up nodes.

- **Processing**: Though both platforms process data in a distributed environment, Hadoop is ideal for batch processing and linear data processing. Spark is ideal for real-time processing and processing live unstructured data streams.

Comparing Hadoop and Spark

Spark is a Hadoop enhancement to MapReduce. The primary difference between Spark and MapReduce is that Spark processes and retains data in memory for subsequent steps, whereas MapReduce processes data on disk.

- **Scalability**: When data volume rapidly grows, Hadoop quickly scales to accommodate the demand via Hadoop Distributed File System (HDFS). In turn, Spark relies on the fault tolerant HDFS for large volumes of data.

- **Security**: Spark enhances security with authentication via shared secret or event logging, whereas Hadoop uses multiple authentication and access control methods. Though, overall, Hadoop is more secure, Spark can integrate with Hadoop to reach a higher security level.

- **Machine learning (ML)**: Spark is the superior platform in this category because it includes MLlib, which performs iterative in-memory ML computations. It also includes tools that perform regression, classification, persistence, pipeline construction, evaluation, etc.

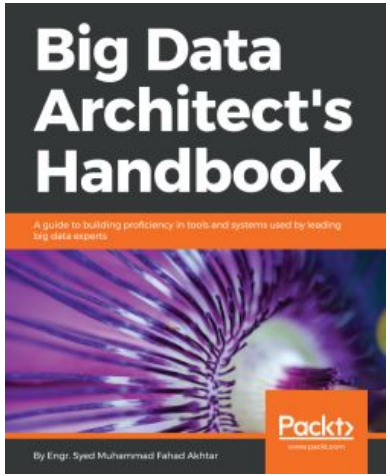
Hadoop use cases

- Processing big data sets in environments where data size exceeds available memory
- Batch processing with tasks that exploit disk read and write operations
- Building data analysis infrastructure with a limited budget
- Completing jobs that are not time-sensitive
- Historical and archive data analysis

Spark use cases

- Dealing with chains of parallel operations by using iterative algorithms
- Achieving quick results with in-memory computations
- Analyzing stream data analysis in real time
- Graph-parallel processing to model data
- All ML applications

Resources



1. https://hevodata.com/learn/kafka-big-data/
2. https://towardsdatascience.com/hdfs-simple-docker-installation-guide-for-data-science-workflow-b3ca764fc94b
3. https://github.com/bitnami/bitnami-docker-spark
4. https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
5. https://www.ibm.com/cloud/blog/hadoop-vs-spark

Questions ??