Spark vs MapReduce: The Epic Battle of Big Data!

Dr Amin Karami (PhD, FHEA, EE)

PG Academic Lead and Course Leader for MSc Big Data Tech UK

amin.karami@ymail.com

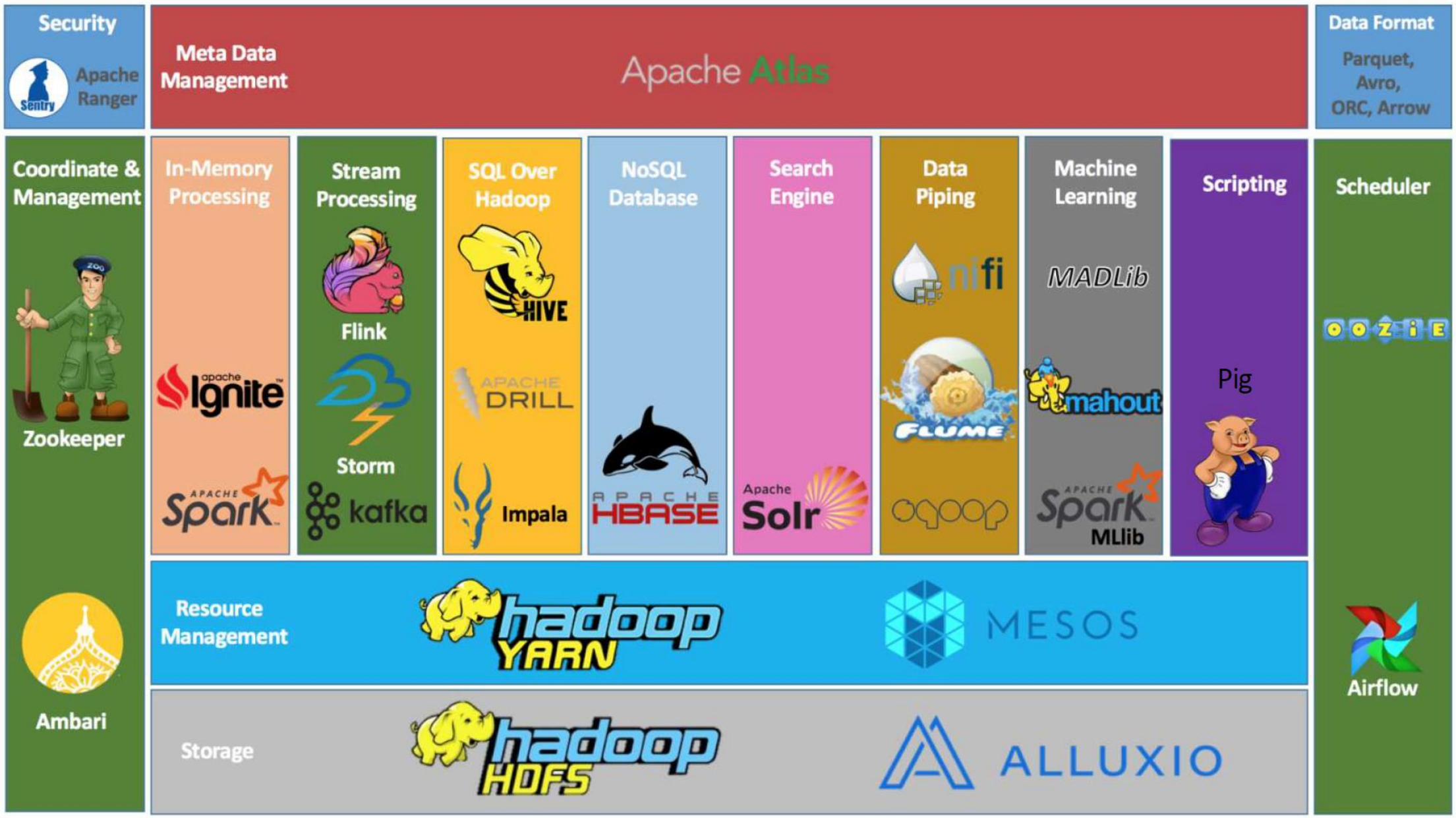
a.karami@uel.ac.uk

www.aminkarami.com

Learning Outcomes

- Gain knowledge on the distinctions between MapReduce and Spark
- Recognize the significance of Apache Spark and Hadoop MapReduce in Big Data Analytics

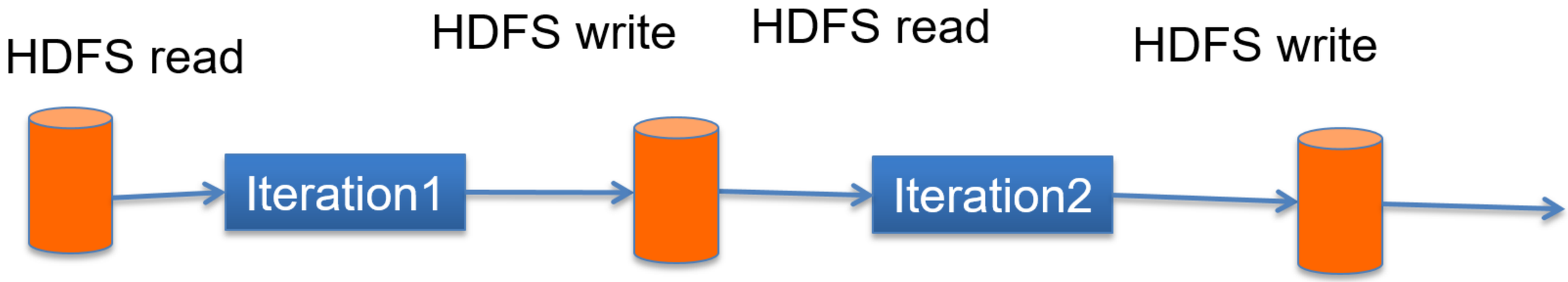
Hadoop Ecosystem



Source: https://www.oreilly.com/library/view/apache-hive-essentials/9781788995092/

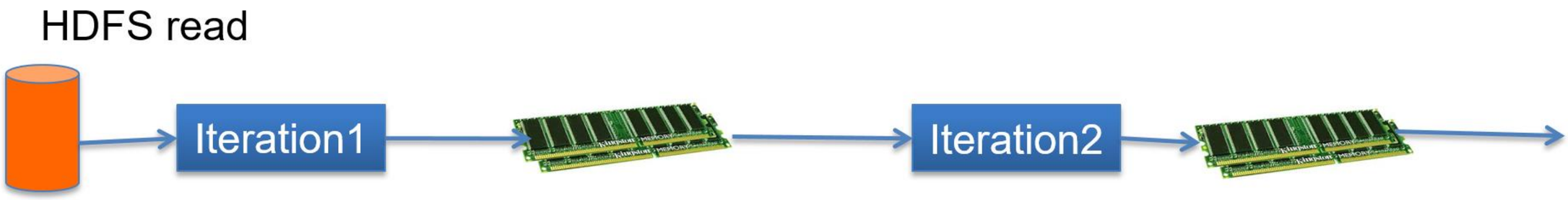
MapReduce Problems

- ✓ Many problems aren't easily described as map-reduce
- ✓ Accessing to disk is typically slower than memory



Solution: Apache Spark

✓ Apache Spark is a cluster-computing framework that is open-source and designed for distributed general-purpose computing. It features implicit data parallelism and fault tolerance, making it a reliable and efficient tool for a wide range of applications.



Spark vs Hadoop MapReduce

Spark and MapReduce are both big data processing frameworks, but they have some key differences:

Processing speed: Spark is generally faster than MapReduce due to its in-memory processing capabilities. Spark can cache data in memory, which reduces the need to read from disk, resulting in faster processing times.

Ease of use: Spark has a more user-friendly API compared to MapReduce. Spark provides APIs in Java, Scala, Python, and R, while MapReduce is limited to Java.

Real-time processing: Spark supports real-time processing through its streaming API, while MapReduce is designed for batch processing.

Spark vs Hadoop MapReduce

Fault-tolerance: Both frameworks provide fault-tolerance, but Spark's mechanism for fault-tolerance is more efficient and requires less disk I/O compared to MapReduce.

Ecosystem and Integration: Hadoop MapReduce has been around for longer and has a mature ecosystem with various tools and libraries built around it. If you are already using other Hadoop components like HDFS or Hive, MapReduce might be a natural choice for integration purposes. Spark, on the other hand, has a growing ecosystem and offers seamless integration with other big data tools like HBase, Kafka, redis, mongoDB, elastic, Cassandra, ...

Spark vs Hadoop MapReduce

Wrap up 1: Spark is faster, more user-friendly, and better suited for real-time processing (*e.g., continuously read and process tweets in real-time as they are being generated, or ingest and process the sensor data in real-time, enabling us to monitor and respond to changes in the data as they occur*), while MapReduce is designed for batch processing and has a proven track record for handling large-scale data processing.

Wrap up 2: both Spark and MapReduce are capable of handling large-scale data processing with **scalability and stability**. While MapReduce has a proven track record for batch processing, Spark is optimized for real-time processing and provides faster performance. The choice between Spark and MapReduce ultimately depends on your specific requirements and use case.

Scalability and Stability

Scalability: Batch processing often involves processing massive amounts of data, sometimes in the terabytes or petabytes range. Scalability refers to the ability to handle increasing data volumes without sacrificing performance.

- **Hadoop MapReduce:** Splits data into smaller chunks, distributes them across nodes in a cluster, and scales horizontally to handle larger datasets.
- **Spark:** Distributes workload across nodes, caches data in memory, and scales horizontally to handle increasing data volumes.

Scalability and Stability

Stability: Stability refers to the robustness and fault-tolerance of the system when processing large batches of data. In batch processing, it is crucial to ensure that jobs can handle failures gracefully and continue processing without data loss. MapReduce provides built-in fault tolerance mechanisms, such as automatic task recovery and data replication, which make it a reliable choice for handling failures during batch processing.

- **Hadoop MapReduce:** Automatically detects node failures, re-runs failed tasks on other nodes, and ensures job completion without data loss.
- **Spark:** Uses RDD lineage to recover lost data, re-computes lost partitions on other nodes, and maintains real-time processing without data loss.

The cost of running MapReduce & Spark

The cost of running MapReduce or Spark for processing data at the scale of terabytes (TB), petabytes (PB), or even yottabytes (YB) depends on several factors:

Infrastructure Costs: Both MapReduce and Spark require a distributed computing infrastructure with a cluster of machines to process large-scale data. The cost of setting up and maintaining the infrastructure, including servers, storage, networking, and power, can be significant.

Storage Costs: As the size of the data increases, the cost of storing and managing the data also increases. Whether using Hadoop Distributed File System (HDFS) for MapReduce or a distributed storage system like Hadoop-compatible object stores or cloud storage for Spark, there will be costs associated with storing and accessing the data.

The cost of running MapReduce & Spark

Computational Costs: The cost of running compute-intensive tasks on a large-scale data processing platform like MapReduce or Spark can be substantial. This includes the cost of CPU usage, memory, and other resources required to process the data efficiently.

Data Transfer Costs: If the data needs to be transferred from external sources or between different stages of processing, there may be costs associated with data transfer, especially if it involves transferring data over networks or across different cloud regions.

Licensing Costs: Depending on the specific distribution or vendor you choose for MapReduce or Spark, there may be licensing costs associated with using their software or specific features.

The winner of this Epic Battle?

Spark?

MapReduce?