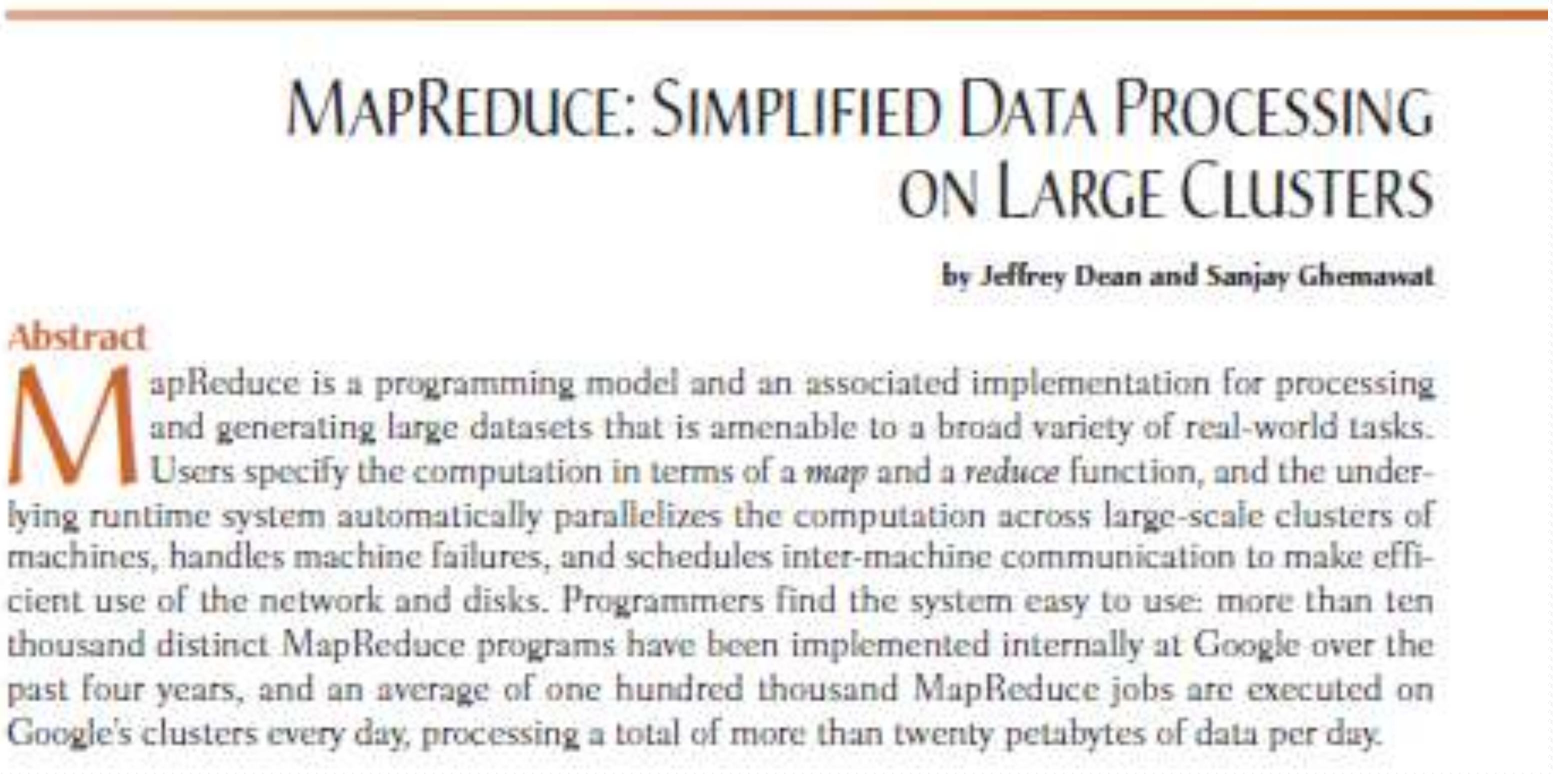
Classic Map Reduce



MAPREDUCE: SIMPLIFIED DATA PROCESSING ON LARGE CLUSTERS

by Jeffrey Dean and Sanjay Ghemawat

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real-world tasks. Users specify the computation in terms of a *map* and a *reduce* function, and the underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, handles machine failures, and schedules inter-machine communication to make efficient use of the network and disks. Programmers find the system easy to use: more than ten thousand distinct MapReduce programs have been implemented internally at Google over the past four years, and an average of one hundred thousand MapReduce jobs are executed on Google's clusters every day, processing a total of more than twenty petabytes of data per day.

✓Published in 2004 by **Jeffrey Dean and Sanjay Ghemawat** of Google.
✓The paper describes the MapReduce programming model and implementation, which was developed at Google to process and generate large data sets using a cluster of computers.

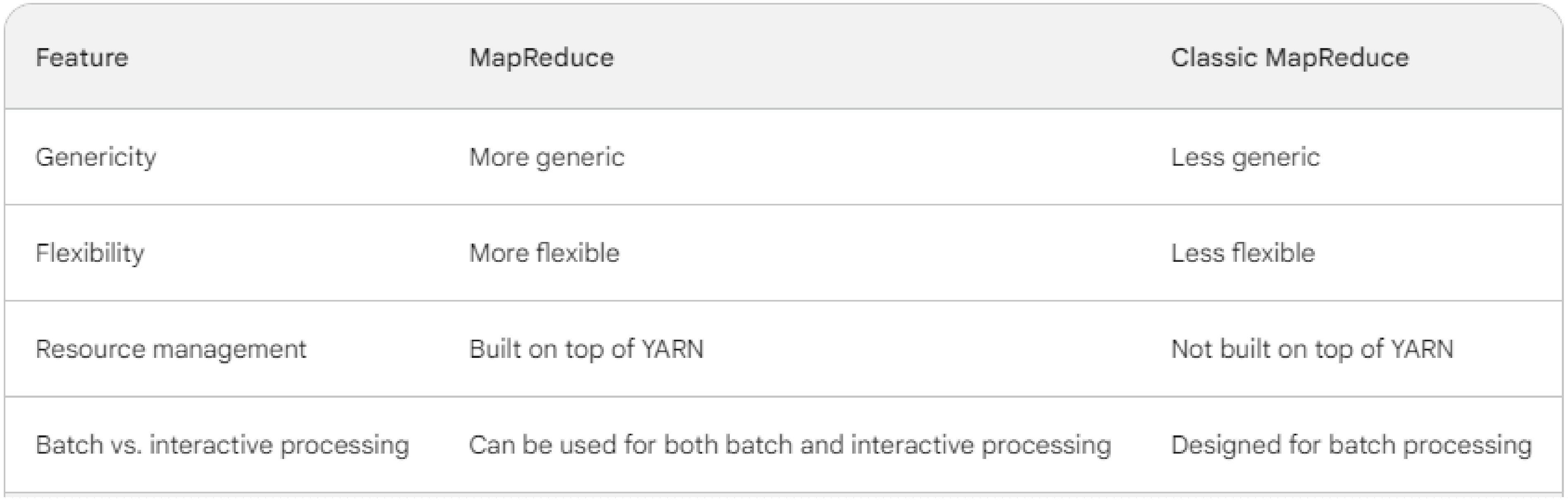
MapReduce Is A Two-Stage Process

Map

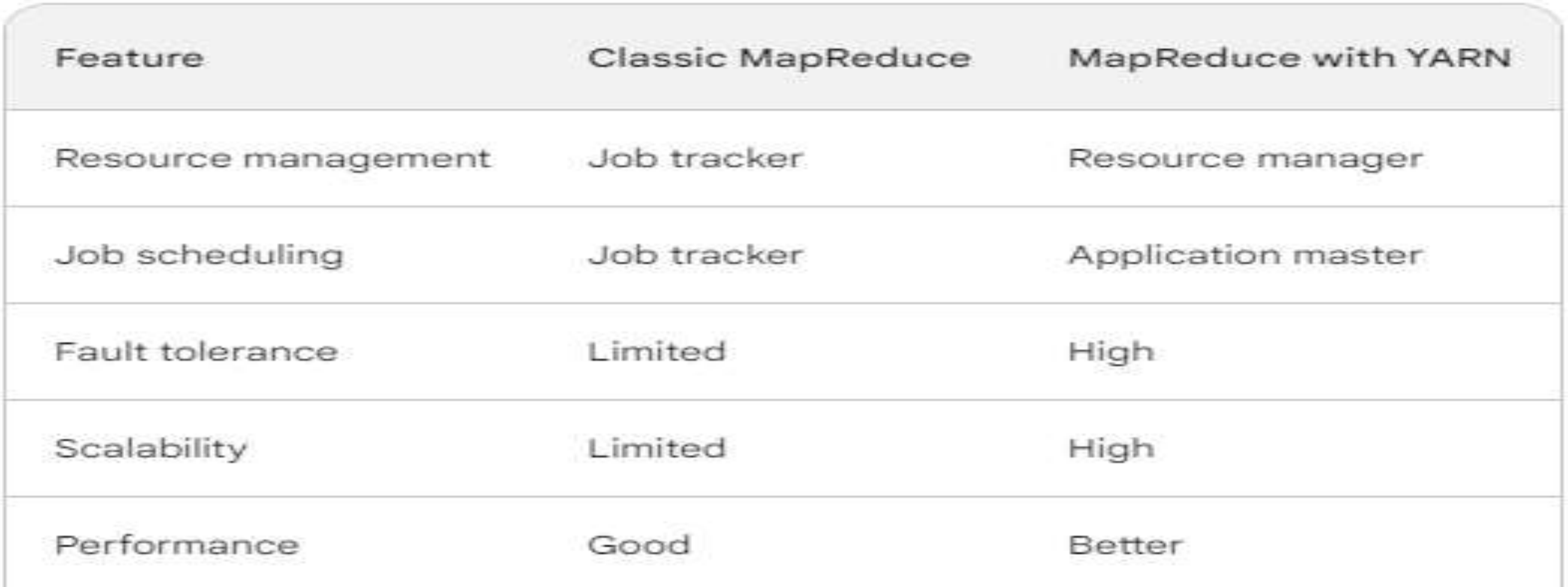
- ✓ In the map phase, input data is divided into smaller chunks and processed by mapper functions.
- ✓ Mapper functions take a key-value pair as input and output one or more key-value pairs.

Reduce:

- ✓ In the reduce phase, the output of the map phase is shuffled and grouped by key. Reduce functions then process each group of key-value pairs and output a single key-value pair.
- ✓ MapReduce is a very efficient way to process large data sets because it can be easily parallelized.
- ✓ Each mapper and reducer function can be executed on a different node in the cluster, which allows for significant speedups.



Feature	MapReduce	Classic MapReduce
Genericity	More generic	Less generic
Flexibility	More flexible	Less flexible
Resource management	Built on top of YARN	Not built on top of YARN
Batch vs. interactive processing	Can be used for both batch and interactive processing	Designed for batch processing



Feature	Classic MapReduce	MapReduce with YARN
Resource management	Job tracker	Resource manager
Job scheduling	Job tracker	Application master
Fault tolerance	Limited	High
Scalability	Limited	High
Performance	Good	Better

MapReduce with YARN

- MapReduce with YARN (Yet Another Resource Negotiator) is a **Newer Version Of Mapreduce** that was introduced in Hadoop 2.0.

- YARN provides a more efficient way to manage resources in a Hadoop cluster, which can lead to significant performance improvements for MapReduce jobs.

- In classic MapReduce, the **Job Tracker** was responsible for both managing resources and scheduling jobs.

- This could lead to performance bottlenecks, especially for large clusters.

- ✓ In MapReduce with YARN, the **Resource Manager** is responsible for managing resources and
- ✓ The **Application Master** is responsible for scheduling jobs.
- ✓ This separation of duties helps to improve performance and scalability.

It provides a number of advantages over classic MapReduce, including

- ✓ **Improved Performance**
- ✓ **Scalability**
- ✓ **Fault Tolerance**
- ✓ **Resource Management**

Failures in YARN

ResourceManager failures:

- ✓ manages all of the **Resources In The Cluster.**
- ✓ If the ResourceManager fails, all of the jobs running in the **Cluster Will Be Interrupted**.
- ✓ To avoid this, YARN can be configured to run **Two ResourceManagers** in an active-standby configuration.
- ✓ If the active ResourceManager fails, the standby ResourceManager will take over with minimal disruption.

ApplicationMaster Failures:-

- ✓ managing the **Execution Of A Single Job**.
- ✓ If the ApplicationMaster fails, the job will fail.
- ✓ However, the YARN framework will **Automatically Restart The ApplicationMaster** on a different node manager.
- ✓ By default, YARN will try to restart the ApplicationMaster twice. If it fails to restart the ApplicationMaster successfully, the job will be killed.

NodeManager failures

- ✓ Running the containers that **Execute The Tasks Of A Job**.
- ✓ If a NodeManager fails, all of the containers running on that node manager will fail.
- ✓ However, the YARN framework will automatically reschedule the failed containers to other NodeManagers in the cluster.

Container Failures

- ✓ Containers are the basic execution units of YARN.
- ✓ If a container fails, the YARN framework will automatically reschedule the failed container to another NodeManager in the cluster

Other Failures are Hardware and Software.

Here Are Some Tips For Troubleshooting Failures In YARN

- ✓ **Check the YARN logs**: The YARN logs can provide valuable information about the cause of a failure. The logs can be found on the YARN ResourceManager and NodeManagers.

- ✓ **Use the YARN web UI**: The YARN web UI provides a real-time view of the YARN cluster. It can be used to monitor the status of jobs, containers, and resources.

- ✓ **Use the YARN command-line tools**: The YARN command-line tools can be used to manage and troubleshoot YARN clusters. For example, the yarn application -list command can be used to list all of the running jobs in the cluster.

- ✓ **Use the YARN support forums**: The YARN support forums are a great place to get help with troubleshooting YARN failures.