BDA5-6

1

Hello everyone, I am Haiying Che, from Institute of Data Science and knowledge Engineering

School of Computer Science, in Beijing Institute of Technology,

in this session, we discuss in memory computing represented by Spark

2

The data processing system provides big data computing and processing capabilities and an application development platform.

From the perspective of computing architecture, the data processing system is divided into data algorithm layer, computing model layer, computing platform layer, computing engine layer, etc.

Computing models are the way that different kinds of big data is processed in different scenarios,

which include batch processing, stream computing, **Large-scale concurrent processing (MPP) model** for structured data, In-memory Computing model, and Data Flow Graph models.

Let’s look at the 4th computing model **In-memory Computing model**

3

In-memory computation (or in-memory computing) is the technique of running computer calculations entirely in computer memory (e.g., in RAM). This term typically implies large-scale, complex calculations which require specialized systems software to run the calculations on computers working together in a cluster. As a cluster, the computers pool together their RAM so the calculation is essentially run across computers and leverages the collective RAM space of all the computers together.

This term is mostly synonymous with in-memory computing and [**in-memory processing**](https://hazelcast.com/glossary/in-memory-processing/), with perhaps a slightly nuanced difference where in-memory computation is a specific type of task dealing with calculations, that falls under the umbrella of in-memory computing. In-memory processing can be considered a different type of task that also falls under in-memory computing.

**How Does In-Memory Computation Work?**

In-memory computation works by eliminating all slow data accesses and relying exclusively on data stored in RAM. Overall computation performance is greatly improved by removing the latency commonly seen when accessing hard disk drives or SSDs. Software running on one or more computers manages the computation as well as the data in memory, and in the case of multiple computers, the software divides the computation into smaller tasks which are distributed out to each computer to run in parallel.

4

Speak of **In-memory Computing model, we must talk about spark.**

* Spark was initially started by Matei Zaharia at UC Berkeley‘s AMP Lab in 2009, and open sourced in 2010.
* In 2013, donated to the Apache Software Foundation.
* one of the most active open source big data projects, Top-Level Apache Project

Parallel processing framework based on the memory computing model.

It can be built on the Hadoop platform and use the HDFS file system to store data, but a Resilient **Distributed dataset architecture (RDD)** is built on top of the file system for Supports efficient **Distributed Memory Computing.**

5

In 2013 Hadoop finish the task of sorting 100TB data within 72 minutes using 2100 machines.

Buit In November 2014, Spark founder M. Zaharia's company Databricks set a new world record in large scale sorting using Spark.

Spark sorting the 100TB data within 21minutes using 207 machine, which is one tenth of the number of the machines that Hadoop used and reduced the time from 72 minute to 23 minutes.

Far more faster and more efficient than Hadoop.

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 queries against data of any size. Simply put, Spark is a **fast and general engine for large-scale data processing**.

The **fast** part means that it’s faster than previous approaches to work with Big Data like classical [MapReduce](https://www.ibm.com/analytics/hadoop/mapreduce). The secret for being faster is that Spark runs on memory (RAM), and that makes the processing much faster than on disk drives.

The **general** part means that it can be used for multiple things like running distributed SQL, creating data pipelines, ingesting data into a database, running Machine Learning algorithms, working with graphs or data streams, and much more.

6

*resilient distributed dataset* (RDD), which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel.

RDDs are created by starting with a file in the Hadoop file system (or any other Hadoop-supported file system), or an existing Scala collection in the driver program, and transforming it.

There are only 2 kinds of operation of RDD, transformations and action. In transformation, data can be filter, joined map, reduced but no calculation is executed, only in action the calculation can be done, and the value result can be generated.

Users may also ask Spark to *persist* an RDD in memory, allowing it to be reused efficiently across parallel operations. Finally, RDDs automatically recover from node failures.

7

The Transformation return value is still an RDD. It uses the chain call design pattern, after calculating one RDD, transform it into another RDD, and then this RDD can be converted again.

This process is distributed.

Action return value is not an RDD. It is either a normal collection of Scala, or a value, or it is empty, and eventually it is returned to the Driver program, or the RDD is written to the file system.

Action is the return value returned to the driver or stored in a file, it is the transformation from RDD to result, and Transformation is the transformation from RDD to RDD.

Only when the action is executed, the rdd will be calculated and generated, which is the root of the lazy execution of rdd.

This diagram shows the topology of the RDD transformations and the linage of RDD transformation, which the executions should follow.

8

Spark running process is like this: The Spark architecture uses the Master-Slave model in distributed computing.

**Master** is the node containing the Master process in the corresponding cluster, and Slave is the node containing the Worker process in the cluster.

**Master** is the controller of the entire cluster and is responsible for the normal operation of the entire cluster; Worker is equivalent to a computing node, receiving commands from the master node and reporting status; **Executor** is responsible for task execution; Client as the user's client is responsible for submitting applications, and Driver is responsible for controlling one Application execution.

After the Spark cluster is deployed, it is necessary to start the **Master process and the Worker process** on the master node and the slave node respectively to control the entire cluster. During the execution of a Spark application, Driver and Worker are two important roles. The Driver program is the starting point of application logic execution, responsible for job scheduling, that is, the distribution of Task tasks, and multiple Workers are used to manage computing nodes and create Executor parallel processing tasks.

In the execution phase, the Driver serializes the files and jars that the Task and Task depend on and transfers them to the corresponding Worker machine. At the same time, the Executor processes the tasks of the corresponding data partition.Excecutor /Task each program has its own, different programs are isolated from each other, task is multi-threaded in parallel, The cluster is transparent to Spark, as long as Spark can obtain related nodes and processes。Driver keeps communication with Executor and cooperates in processing.

9

**Features of Spark include**

**In-memory Computation**

PySpark provides provision of in-memory computation. Computed results are stored in distributed memory (RAM) instead of stable storage (disk). It provides very fast computation

**Lazy Evolution**

Transformation in PySpark RDDs is lazy. It doesn't compute the result immediately means that execution does not start until an action is triggered. When we call some operation in RDD for transformation, it does not execute immediately. Lazy Evolution plays an important role in saving calculation overhead. It provides the optimization by reducing the number of queries.

**Fault Tolerant**

RDDs track data lineage information to reconstruct lost data automatically. If failure occurs in any partition of RDDs, then that partition can be re-computed from the original fault tolerant input dataset to create it.

**Immutability**

The created data can be retrieved anytime but its value can't be changed. RDDs can only be created through deterministic operations.

**Partitioning**

RDDs are the collection of various data items that are so huge in size. Because of its size they cannot fit into a single node and must be partitioned across various nodes.

**Persistence**

It is an optimization technique where we can save the result of RDD evaluation. It stores the intermediate result so that we can use it further if required. It reduces the computation complexity.

**Coarse-Grained Operation**

The coarse-grained operation means that we can transform the whole dataset but not individual element on the dataset. On the other hand, fine grained mean we can transform individual element on the dataset.

10

**Apache Spark Core** – Spark Core is the underlying general execution engine for the Spark platform that all other functionality is built upon. It provides in-memory computing and referencing datasets in external storage systems.

**Spark SQL** – Spark SQL is Apache Spark’s module for working with structured data. The interfaces offered by Spark SQL provides Spark with more information about the structure of both the data and the computation being performed.

**Spark Streaming** – This component allows Spark to process real-time streaming data. Data can be ingested from many sources like Kafka, Flume, and HDFS (Hadoop Distributed File System). Then the data can be processed using complex algorithms and pushed out to file systems, databases, and live dashboards.

**MLlib (Machine Learning Library)** – Apache Spark is equipped with a rich library known as MLlib. This library contains a wide array of machine learning algorithms- classification, regression, clustering, and collaborative filtering. It also includes other tools for constructing, evaluating, and tuning ML Pipelines. All these functionalities help Spark scale out across a cluster.

**GraphX** – Spark also comes with a library to manipulate graph databases and perform computations called GraphX. GraphX unifies ETL (Extract, Transform, and Load) process, exploratory analysis, and iterative graph computation within a single system.

Spark makes it easy to start working with distributed computing systems.  
  
Through its core API support for multiple languages, native libraries that enable easy streaming, machine learning, graph computing, and SQL - **the Spark ecosystem offers some of the most extensive capabilities and features of any technology out there.**Other [third-party contributions](https://spark.apache.org/third-party-projects.html) also make using Spark much easier and versatile.

11

**Spark Advantages include**

**Fast processing** – The most important feature of Apache Spark that has made the big data world choose this technology over others is its speed. Big data is characterized by volume, variety, velocity, and veracity which needs to be processed at a higher speed. Spark contains [Resilient Distributed Dataset (RDD)](https://intellipaat.com/blog/tutorial/spark-tutorial/programming-with-rdds/) which saves time in reading and writing operations, allowing it to run almost **ten to one hundred times faster than Hadoop**.

**Flexibility** – Apache Spark supports multiple languages and allows the developers to write applications in Java, Scala, R, or Python.

**In-memory computing** – Spark stores the data in the RAM of servers which allows quick access and in turn accelerates the speed of analytics.

**Real-time processing** – Spark is able to process real-time streaming data. Unlike MapReduce which processes only stored data, Spark is able to process real-time data and is, therefore, able to produce instant outcomes.

**Better analytics** – In contrast to [MapReduce](https://www.ibm.com/analytics/hadoop/mapreduce) that I ncludes Map and Reduce functions, Spark includes much more than that. Apache Spark consists of a rich set of SQL queries, machine learning algorithms, complex analytics, etc. With all these functionalities, analytics can be performed in a better fashion with the help of Spark.

12

**Spark is the most popular open-source distributed computing engine for big data analysis.**

Used by data engineers and data scientists alike in thousands of organizations worldwide, **Spark is the industry standard analytics engine for big data processing and machine learning.**

Spark enables you to process data at lightning speed for both batch and streaming workloads.

Spark can run on Hadoop, mongo DB, Cassandra Kubernetes, YARN, or standalone - and it works with a wide range of data inputs and outputs.

13

Apache Spark has seen immense growth over the past several years, becoming the most effective data processing and AI engine in enterprises today due to its speed, ease of use, and sophisticated analytics. However, the cost of Spark is high as it requires lots of RAM to run in-memory.

Spark unifies data and AI by simplifying data preparation at a massive scale across various sources. Moreover, it provides a consistent set of APIs for both data engineering and data science workloads, along with seamless integration of popular libraries such as TensorFlow, PyTorch, R and SciKit-Learn.

In this session we learned in memory computing represented by Spark.

Thank you for your attention, if you have any question, feel free to contact me.