Sengamala Thayaar Educational Trust Women's College

(Affiliated to Bharathidasan University)

(Accredited with 'A' Grade {3.45/4.00} By NAAC) (An ISO 9001: 2015 Certified Institution)

Sundarakkottai, Mannargudi-614 016.
Thiruvarur (Dt.), Tamil Nadu, India.

BIG DATA AND ANALYTICS

R.AKILANDESWARI

ASSISTANT PROFESSOR

PG & RESEARCH DEPARTMENT OF COMPUTER SCIENCE

UNIT –V

Hadoop Map Reduce and YARN framework:

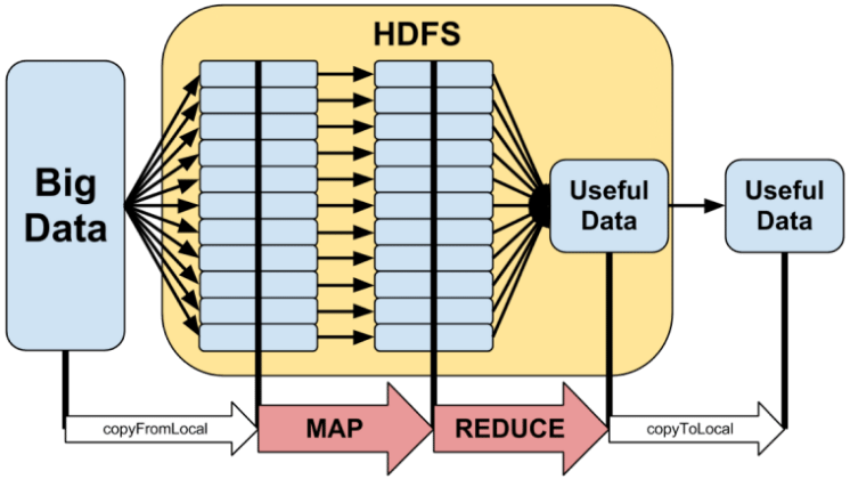
Introduction to MapReduce:

Big Data

Big Data is a collection of large datasets that cannot be processed using traditional computing techniques. For example, the volume of data Facebook or Youtube need require it to collect and manage on a daily basis, can fall under the category of Big Data. However, Big Data is not only about scale and volume, it also involves one or more of the following aspects − Velocity, Variety, Volume, and Complexity.

MapReduce

MapReduce is a programming model for writing applications that can process Big Data in parallel on multiple nodes. MapReduce provides analytical capabilities for analyzing huge volumes of complex data



AN INTRODUCTION TO YARN

Get a top-down introduction to YARN. YARN allows integration of frameworks, such as Spark and HAMA, into Hadoop to expand the popular Data tool beyond MapReduce.

YARN/Hadoop 2.x has a completely different architecture with compared to Hadoop 1.x.

In Hadoop 1.x JobTracker serves **two major functions**:

1. Resource management

2. Job scheduling/Job monitoring

Recall that in Hadoop 1.x, there was a single JobTracker per Hadoop cluster serving these functions in which scaling can overwhelm the JobTracker. Also, having a single JobTracker makes it a single point of failure; if the JobTracker goes down, the entire cluster goes down with all the current jobs.

YARN tries to separate above mentioned functionalities into two daemons:

1. Global Resource Manager

2. Per-application Application Master

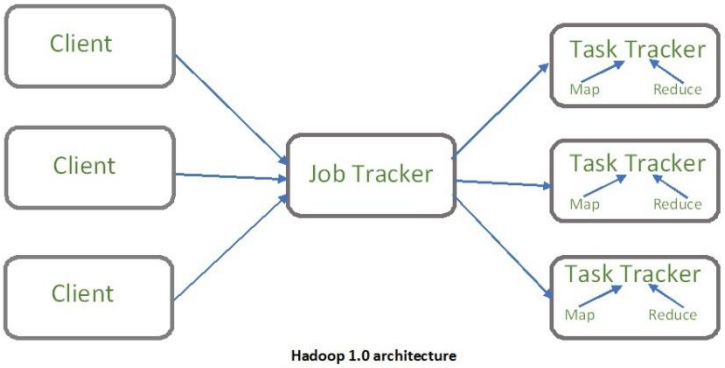
Before YARN, Hadoop was designed to support MapReduce jobs only. As time went by, people encountered Big Data computation problems that cannot be addressed by MapReduce and thus came up with different frameworks which work on top of HDFS to address their problems. Some of these were:

- Apache Spark
- Apache HAMA
- Apache Giraph.

YARN provides a way for these new frameworks to be integrated into Hadoop framework, sharing the same underlying HDFS. YARN enables Hadoop to handle jobs beyond MapReduce.

Hadoop YARN Architecture

YARN stands for "***Yet Another Resource Negotiator***". It was introduced in Hadoop 2.0 to remove the bottleneck on Job Tracker which was present in Hadoop 1.0. YARN was described as a "*Redesigned Resource Manager*" at the time of its launching, but it has now evolved to be known as large-scale distributed operating system used for Big Data processing.



Hadoop 1.0 architecture

YARN architecture basically separates resource management layer from the processing layer. In Hadoop 1.0 version, the responsibility of Job tracker is split between the resource manager and application manager.

YARN also allows different data processing engines like graph processing, interactive processing, stream processing as well as batch processing to run and process data stored in HDFS (Hadoop Distributed File System) thus making the system much more efficient. Through its various components, it can dynamically allocate various resources and schedule the application processing. For large volume data processing, it is quite necessary to manage the available resources properly so that every application can leverage them.

YARN Features: YARN gained popularity because of the following features-

- **Scalability:** The scheduler in Resource manager of YARN architecture allows Hadoop to extend and manage thousands of nodes and clusters.
- **Compatability:** YARN supports the existing map-reduce applications without disruptions thus making it compatible with Hadoop 1.0 as well.
- **Cluster Utilization:**Since YARN supports Dynamic utilization of cluster in Hadoop, which enables optimized Cluster Utilization.
- **Multi-tenancy:** It allows multiple engine access thus giving organizations a benefit of multi-tenancy.

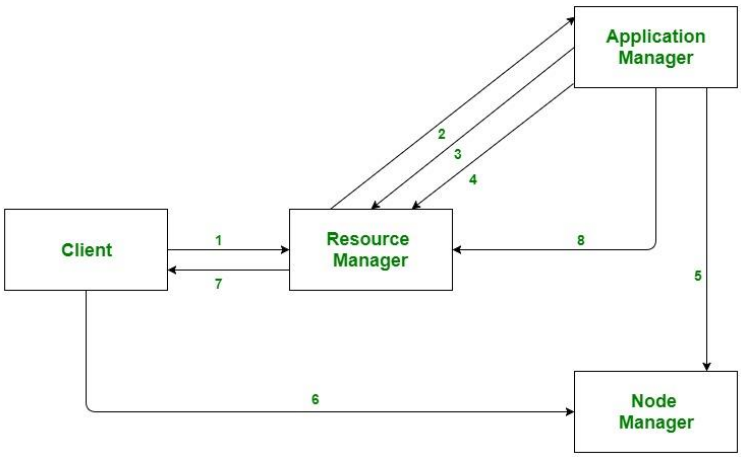
COMPONENTS OF YARN ARCHITECTURE

- **Client:** It submits map-reduce jobs.
- **Resource Manager:** It is the master daemon of YARN and is responsible for resource assignment and management among all the applications. Whenever it receives a processing request, it forwards it to the corresponding node manager and allocates resources for the completion of the request accordingly. It has two major components:

- **Scheduler:** It performs scheduling based on the allocated application and available resources. It is a pure scheduler, means it does not perform other tasks such as monitoring or tracking and does not guarantee a restart if a task fails. The YARN scheduler supports plugins such as Capacity Scheduler and Fair Scheduler to partition the cluster resources.
- **Application manager:** It is responsible for accepting the application and negotiating the first container from the resource manager. It also restarts the Application Manager container if a task fails

- **Node Manager:** It take care of individual node on Hadoop cluster and manages application and workflow and that particular node. Its primary job is to keep-up with the Node Manager. It monitors resource usage, performs log management and also kills a container based on directions from the resource manager. It is also responsible for creating the container process and start it on the request of Application master.
- **Application Master:** An application is a single job submitted to a framework. The application manager is responsible for negotiating resources with the resource manager, tracking the status and monitoring progress of a single application. The application master requests the container from the node manager by sending a Container Launch Context(CLC) which includes everything an application needs to run. Once the application is started, it sends the health report to the resource manager from time-to-time.
- **Container:** It is a collection of physical resources such as RAM, CPU cores and disk on a single node. The containers are invoked by Container Launch Context(CLC) which is a record that contains information such as environment variables, security tokens, dependencies etc.

Application workflow in Hadoop YARN:



1. Client submits an application
2. The Resource Manager allocates a container to start the Application Manager

3. The Application Manager registers itself with the Resource Manager
4. The Application Manager negotiates containers from the Resource Manager
5. The Application Manager notifies the Node Manager to launch containers
6. Application code is executed in the container
7. Client contacts Resource Manager/Application Manager to monitor application's status
8. Once the processing is complete, the Application Manager un-registers with the Resource Manager

MapReduce Applications

1. Given a repository of text files, find the frequency of each word. This is called the WordCount problem.

2. Given a repository of text files, find the number of words of each word length.

3. Given two matrices in sparse matrix format, compute their product.

4. Factor a matrix given in sparse matrix format.

5. Given a symmetric graph whose nodes represent people and edges represent friendship, compile a list of common friends.

6. Given a symmetric graph whose nodes represent people and edges represent friendship, compute the average number of friends by age.

7. Given a repository of weather records, find the annual global minima and maxima by year.

8. Sort a large list. Note that in most implementations of the MapReduce framework, this problem is trivial, because the framework automatically sorts the output from the map() function.

9. Reverse a graph.

10. Find a minimal spanning tree (MST) of a...

DATA SERIALIZATION

Data serialization is the process of converting data objects present in complex data structures into a byte stream for storage, transfer and distribution purposes on physical devices

Text-based Data Serialization formats

Without being exhaustive, here are some common ones:

- XML (Extensible Markup Language) - Nested textual format. Human-readable and editable. Schema based validation. Used in metadata applications, web services data transfer, web publishing.
- CSV (Comma-Separated Values) - Table structure with delimiters. Human-readable textual data. Opens as spreadsheet or plaintext. Used as plaintext Database.
- JSON (JavaScript Object Notation) - Short syntax textual format with limited data types. Humanreadable. Derived from JavaScript data formats. No need of a separate parser (like XML) since they map to JavaScript objects. Can be fetched with an XMLHttpRequest call. No direct support for DATE data type. All data is dynamically processed. Popular format for web API parameter passing. Mobile apps use this extensively for user interaction and database services.
- YAML (YAML Ain't Markup Language) - Lightweight text format. Human-readable. Supports comments and thus easily editable. Superset of JSON. Supports complex data types. Maps easily to native data structures. Used in configuration settings, document headers, Apps with need for MySQL style selfreferences in relational data.

Binary Data Serialization formats and their key features

Without being exhaustive, here are some common ones:

- BSON (Binary JSON) - Created and internally used by MongoDB. Binary format, not human-readable. Deals with attribute-value pairs like JSON. Includes datetime, bytearray and other data types not present in JSON. Used in web apps with rich media data types such as live video. Primary use is storage, not network communication.
- MessagePack - Designed for data to be transparently converted from/to JSON. Compressed binary format, not human-readable. Supports static typing. Supports RPC. Better JSON compatibility than BSON. Primary use is network communication, not storage. Used in apps with distributed file systems.
- protobuf (Protocol Buffers) - Created by Google. Binary message format that allows programmers to specify a schema for the data. Also includes a set of rules and tools to define and exchange these messages. Transparent data compression. Used in multi-platform applications due to easy interoperability between languages. Universal RPC framework. Used in performance-critical distributed applications.

Data Serialization Storage format

Storage formats are a way to define how information is stored in the file. Most of the time, this information can be assumed from the extension of the data. Both structured and unstructured data can be stored on HADOOP enabled systems. Common Hdfs file formats are –

- Plain text storage
- Sequence files
- RC files

- AVRO
- Parquet

Why Storage Formats?

- File format must be handy to serve complex data structures
- HDFS enabled applications to take time to find relevant data in a particular location and write back data to another location.
- Dataset is large
- Having schemas
- Having storage constraints

Why choose different File Formats?

Proper selection of file format leads to –

- Faster read time
- Faster write time
- Splittable files (for partial data read)
- Dchema evolution support (modifying dataset fields)
- Advance compression support
- Snappy compression leads to high speed and reasonable compression/decompression.
- File formats help to manage Diverse data.

Guide to Data Serialization in Hadoop

Data serialization is a process to format structured data in such a way that it can be reconverted back to the original form.

- Serialization is done to translate data structures into a stream of data. This stream of data can be transmitted over the network or stored in DB regardless of the system architecture.
- Isn't storing information in binary form or stream of bytes is a right approach.
- Serialization does the same but isn't dependent on architecture.

Consider CSV files contains a comma (,) in between data, so while Deserialization wrong outputs may occur. Now, if metadata is stored in XML form, which is a self architected form of data storage, data can be easily deserialized in the future.

Why Data Serialization for Storage Formats?

- To process records faster (Time-bound).

- When Proper format of data need to be maintained and to be transmitted over data without schema support on another end.
- Now when in future, data without structure or format needs to be processed, complex Errors may occur.
- Serialization offers data validation over transmission.

Areas of Serialization for Storage Formats

To maintain the proper format of a data serialization system must have the following four properties –

- Compact – helps in the best use of network bandwidth
- Fast – reduces the performance overhead
- Extensible – can match new requirements
- Inter-operable – not language-specific

Serialization in Hadoop has two areas –

Inter process communication

When a client calls a function or subroutine from one pc to the pc in-network or server, that Procedure of calling is known as a remote procedure call.

Persistent storage

It is better than java ‘s inbuilt serialization as java serialization isn’ t compact Serialization and Deserialization of data helps in maintaining and managing corporate decisions for effective use of resources and data available in Data warehouse or any other database -writable – language specific to java