UNIT – II

Apache Hadoop

Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume.

Hadoop is written in Java and is not OLAP (online analytical processing). It is used for batch/offline processing.

It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more. Moreover it can be scaled up just by adding nodes in the cluster.

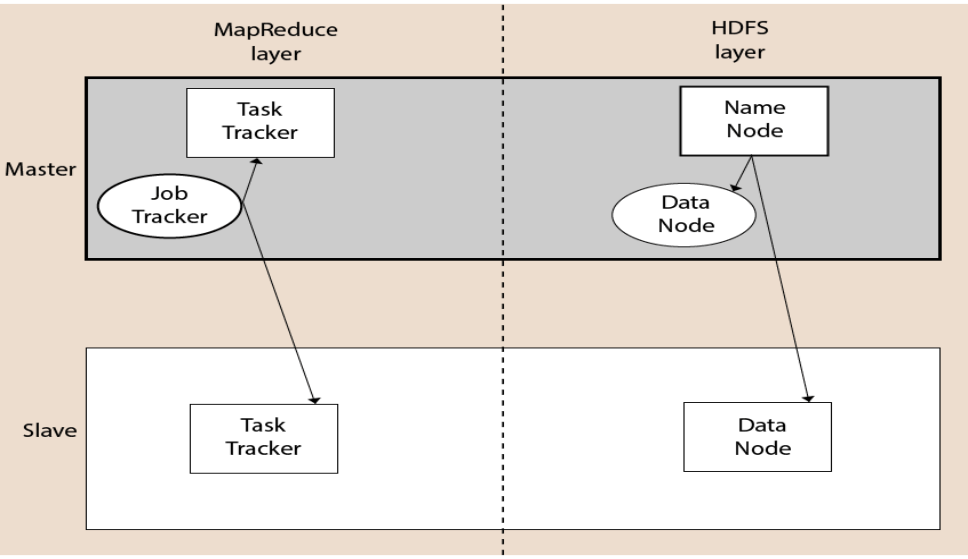
Modules of Hadoop

1. **HDFS:** Hadoop Distributed File System. Google published its paper GFS and on the basis of that HDFS was developed. It states that the files will be broken into blocks and stored in nodes over the distributed architecture.
2. **Yarn:** Yet another Resource Negotiator is used for job scheduling and manage the cluster.
3. **Map Reduce:** This is a framework which helps Java programs to do the parallel computation on data using key value pair. The Map task takes input data and converts it into a data set which can be computed in Key value pair. The output of Map task is consumed by reduce task and then the out of reducer gives the desired result.
4. **Hadoop Common:** These Java libraries are used to start Hadoop and are used by other Hadoop modules.

Hadoop Architecture

The Hadoop architecture is a package of the file system, MapReduce engine and the HDFS (Hadoop Distributed File System). The MapReduce engine can be MapReduce/MR1 or YARN/MR2.

A Hadoop cluster consists of a single master and multiple slave nodes. The master node includes Job Tracker, Task Tracker, NameNode, and DataNode whereas the slave node includes DataNode and TaskTracker.



Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is a distributed file system for Hadoop. It contains a master/slave architecture. This architecture consist of a single NameNode performs the role of master, and multiple DataNodes performs the role of a slave.

Both NameNode and DataNode are capable enough to run on commodity machines. The Java language is used to develop HDFS. So any machine that supports Java language can easily run the NameNode and DataNode software.

NameNode

- It is a single master server exist in the HDFS cluster.
- As it is a single node, it may become the reason of single point failure.
- It manages the file system namespace by executing an operation like the opening, renaming and closing the files.
- It simplifies the architecture of the system.

DataNode

- The HDFS cluster contains multiple DataNodes.
- Each DataNode contains multiple data blocks.
- These data blocks are used to store data.
- It is the responsibility of DataNode to read and write requests from the file system's clients.
- It performs block creation, deletion, and replication upon instruction from the NameNode.

Job Tracker

- The role of Job Tracker is to accept the MapReduce jobs from client and process the data by using NameNode.
- In response, NameNode provides metadata to Job Tracker.

Task Tracker

- It works as a slave node for Job Tracker.
- It receives task and code from Job Tracker and applies that code on the file. This process can also be called as a Mapper.

MapReduce Layer

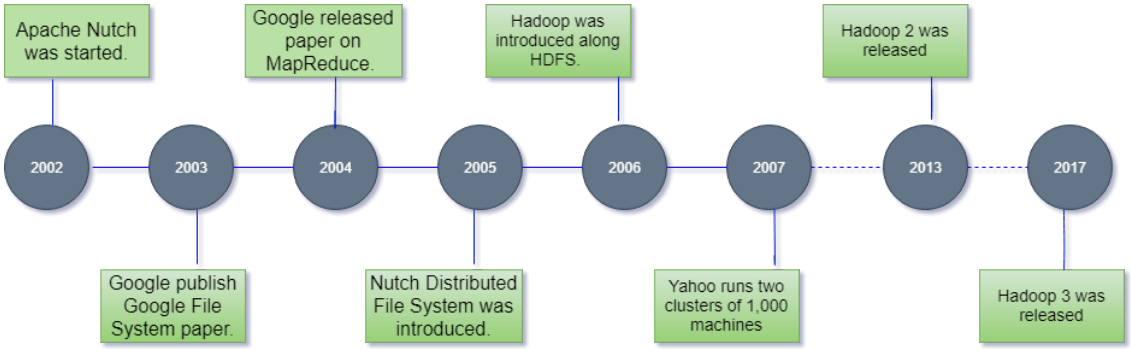
The MapReduce comes into existence when the client application submits the MapReduce job to Job Tracker. In response, the Job Tracker sends the request to the appropriate Task Trackers. Sometimes, the TaskTracker fails or time out. In such a case, that part of the job is rescheduled.

Advantages of Hadoop

- **Fast:** In HDFS the data distributed over the cluster and are mapped which helps in faster retrieval. Even the tools to process the data are often on the same servers, thus reducing the processing time. It is able to process terabytes of data in minutes and Peta bytes in hours.
- **Scalable:** Hadoop cluster can be extended by just adding nodes in the cluster.
- **Cost Effective:** Hadoop is open source and uses commodity hardware to store data so it really cost effective as compared to traditional relational database management system.
- **Resilient to failure:** HDFS has the property with which it can replicate data over the network, so if one node is down or some other network failure happens, then Hadoop takes the other copy of data and use it. Normally, data are replicated thrice but the replication factor is configurable.

History of Hadoop

The Hadoop was started by Doug Cutting and Mike Cafarella in 2002. Its origin was the Google File System paper, published by Google.



Let's focus on the history of Hadoop in the following steps: -

- In 2002, Doug Cutting and Mike Cafarella started to work on a project, **Apache Nutch.** It is an open source web crawler software project.

- While working on Apache Nutch, they were dealing with big data. To store that data they have to spend a lot of costs which becomes the consequence of that project. This problem becomes one of the important reason for the emergence of Hadoop.
- In 2003, Google introduced a file system known as GFS (Google file system). It is a proprietary distributed file system developed to provide efficient access to data.
- In 2004, Google released a white paper on Map Reduce. This technique simplifies the data processing on large clusters.
- In 2005, Doug Cutting and Mike Cafarella introduced a new file system known as NDFS (Nutch Distributed File System). This file system also includes Map reduce.
- In 2006, Doug Cutting quit Google and joined Yahoo. On the basis of the Nutch project, Dough Cutting introduces a new project Hadoop with a file system known as HDFS (Hadoop Distributed File System). Hadoop first version 0.1.0 released in this year.
- Doug Cutting gave named his project Hadoop after his son's toy elephant.
- In 2007, Yahoo runs two clusters of 1000 machines.
- In 2008, Hadoop became the fastest system to sort 1 terabyte of data on a 900 node cluster within 209 seconds.
- In 2013, Hadoop 2.2 was released.
- In 2017, Hadoop 3.0 was released.

Year	Event
2003	Google released the paper, Google File System (GFS).
2004	Google released a white paper on Map Reduce.
2006	o Hadoop introduced. o Hadoop 0.1.0 released. o Yahoo deploys 300 machines and within this year reaches 600 machines.
2007	o Yahoo runs 2 clusters of 1000 machines. o Hadoop includes HBase.

2008	o YARN JIRA opened o Hadoop becomes the fastest system to sort 1 terabyte of data on a 900 node cluster within 209 seconds. o Yahoo clusters loaded with 10 terabytes per day. o Cloudera was founded as a Hadoop distributor.
2009	o Yahoo runs 17 clusters of 24,000 machines. o Hadoop becomes capable enough to sort a petabyte. o MapReduce and HDFS become separate subproject.
2010	o Hadoop added the support for Kerberos. o Hadoop operates 4,000 nodes with 40 petabytes. o Apache Hive and Pig released.
2011	o Apache Zookeeper released. o Yahoo has 42,000 Hadoop nodes and hundreds of petabytes of storage.
2012	Apache Hadoop 1.0 version released.
2013	Apache Hadoop 2.2 version released.
2014	Apache Hadoop 2.6 version released.
2015	Apache Hadoop 2.7 version released.
2017	Apache Hadoop 3.0 version released.
2018	Apache Hadoop 3.1 version released.

MapReduce

Hadoop MapReduce is the data processing layer. It processes the huge amount of structured and unstructured data stored in HDFS. MapReduce handles data in parallel by splitting the job into the set of independent tasks. So, parallel processing increases speed and reliability.

Hadoop MapReduce data processing occurs in 2 phases- Map and Reduce phase.

- **Map phase:** It is the initial phase of data processing. In this phase, we state all the complex logic/business rules/costly code.
- **Reduce phase:** The second phase of processing is the Reduce Phase. In this phase, we state light-weight processing like aggregation/summation.

Steps of MapReduce Job Execution flow

MapReduce processes the data in various phases with the help of different components. Let us discuss the steps of job execution in Hadoop.

Input Files

The data for MapReduce job is stored in Input Files. Input files reside in HDFS. The input file format is random. Line-based log files and binary format can also be used.

InputFormat

After that InputFormat defines how to divide and read these input files. It selects the files or other objects for input. InputFormat creates InputSplit.

InputSplits

It represents the data that will be processed by an individual Mapper. For each split, one map task is created. Therefore the number of map tasks is equal to the number of InputSplits. Framework divide split into records, which mapper process.

RecordReader

It communicates with the inputSplit. And then transforms the data into key-value pairs suitable for reading by the Mapper. RecordReader by default uses TextInputFormat to transform data into a key-value pair. It interrelates to the InputSplit until the completion of file reading. It allocates a byte offset to each line present in the file. Then, these key-value pairs are further sent to the mapper for added processing.

Mapper

It processes input records produced by the RecordReader and generates intermediate key-value pairs. The intermediate output is entirely different from the input pair. The output of the mapper is a full group of key-value pairs. Hadoop framework does not store the output of the mapper on HDFS. Mapper doesn't store, as data is temporary and writing on HDFS will create unnecessary multiple copies. Then Mapper passes the output to the combiner for extra processing.

Combiner

Combiner is Mini-reducer that performs local aggregation on the mapper's output. It minimizes the data transfer between mapper and reducer. So, when the combiner functionality completes, the framework passes the output to the partitioner for further processing.

Partitioner

Partitioner comes into existence if we are working with more than one reducer. It grabs the output of the combiner and performs partitioning.

Partitioning of output occurs based on the key in MapReduce. By hash function, the key (or a subset of the key) derives the partition.

Based on key-value in MapReduce, partitioning of each combiner output occur. And then the record having a similar key-value goes into the same partition. After that, each partition is sent to a reducer.

Partitioning in MapReduce execution permits even distribution of the map output over the reducer.

Shuffling and Sorting

After partitioning, the output is shuffled to the reduced node. The shuffling is the physical transfer of the data which is done over the network. As all the mappers complete and shuffle the output on the reducer nodes. Then the framework joins this intermediate output and sort. This is then offered as input to reduce the phase.

Reducer

The reducer then takes a set of intermediate key-value pairs produced by the mappers as the input. After that runs a reducer function on each of them to generate the output. The output of the reducer is the decisive output. Then the framework stores the output on HDFS.

RecordWriter

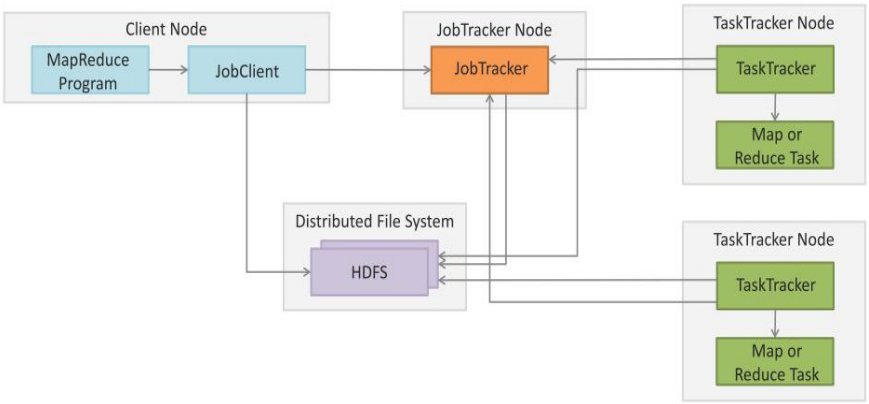
It writes these output key-value pairs from the Reducer phase to the output files.

OutputFormat

OutputFormat defines the way how RecordReader writes these output key-value pairs in output files. So, its instances offered by the Hadoop write files in HDFS. Thus OutputFormat instances write the decisive output of reducer on HDFS.

MapReduce Job Execution Workflow:

- MapReduce job execution starts when the client applications submit jobs to the Job tracker.
- The JobTracker returns a JobID to the client application. The JobTracker talks to the NameNode to determine the location of the data.
- The JobTracker locates TaskTracker nodes with available slots at/or near the data.
- The TaskTrackers send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the JobTracker that they are still alive. These messages also inform the JobTracker of the number of available slots, so the JobTracker can stay up to date with where in the cluster, new work can be delegated.
- The JobTracker submits the work to the TaskTracker nodes when they poll for tasks. To choose a task for a TaskTracker, the JobTracker uses various scheduling algorithms (default is FIFO).
- The TaskTracker nodes are monitored using the heartbeat signals that are sent by the TaskTrackers to JobTracker.
- The TaskTracker spawns a separate JVM process for each task so that any task failure does not bring down the TaskTracker.
- The TaskTracker monitors these spawned processes while capturing the output and exit codes. When the process finishes, successfully or not, the TaskTracker notifies the JobTracker. When the job is completed, the JobTracker updates its status.

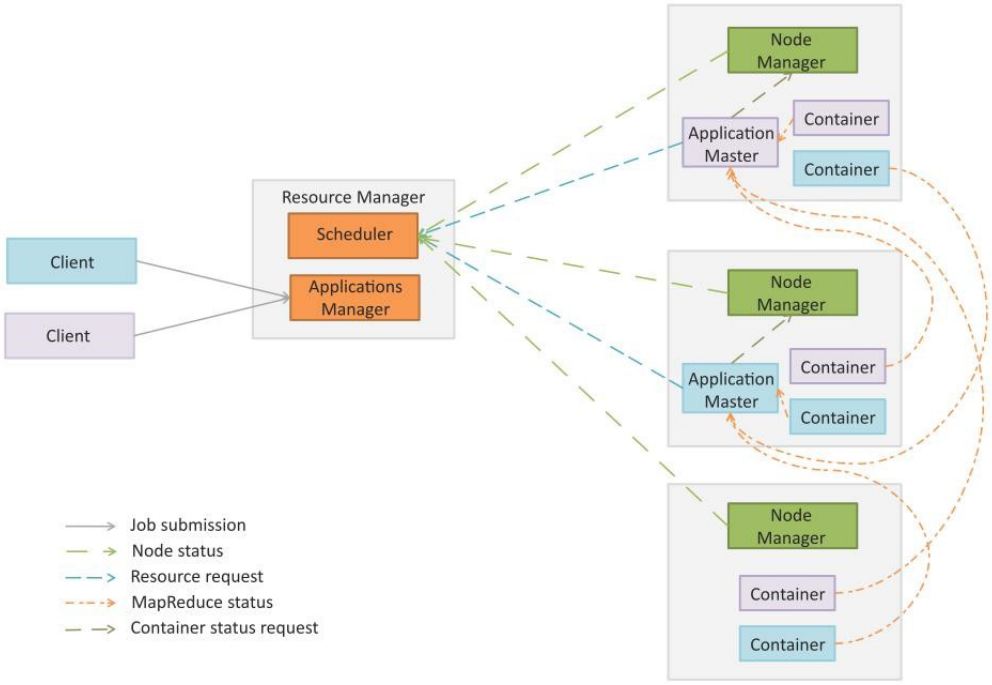


MapReduce 2.0 – YARN

- In Hadoop 2.0 the original processing engine of Hadoop (MapReduce) has been separated from the resource management (which is now part of YARN).
- This makes YARN effectively an operating system for Hadoop that supports different processing engines on a Hadoop cluster such as MapReduce for batch processing, Apache Tez for interactive queries, Apache Storm for stream processing, etc.
- YARN architecture divides architecture divides the two major functions of the JobTracker - resource management and job life-cycle management - into separate components:
    - ResourceManager
    - ApplicationMaster.

YARN Components

- **Resource Manager (RM)**: RM manages the global assignment of compute resources to applications. RM consists of two main services:
    - ***Scheduler***: Scheduler is a pluggable service that manages and enforces the resource scheduling policy in the cluster.
    - ***Applications Manager (AsM)***: AsM manages the running Application Masters in the cluster. AsM is responsible for starting application masters and for monitoring and restarting them on different nodes in case of failures.
- **Application Master (AM):** A per-application AM manages the application's life cycle. AM is responsible for negotiating resources from the RM and working with the NMs to execute and monitor the tasks.
- **Node Manager (NM):** A per-machine NM manages the user processes on that machine.
- **Containers:** Container is a bundle of resources allocated by RM (memory, CPU, network, etc.). A container is a conceptual entity that grants an application the privilege to use a certain amount of resources on a given machine to run a component task.



Hadoop Scheduler

Prior to Hadoop 2, **Hadoop MapReduce** is a software framework for writing applications that process huge amounts of data (terabytes to petabytes) in-parallel on the large Hadoop cluster. This framework is responsible for scheduling tasks, monitoring them, and re-executes the failed task.

In Hadoop 2, a **YARN** called Yet Another Resource Negotiator was introduced. The basic idea behind the YARN introduction is to split the functionalities of resource management and job scheduling or monitoring into separate daemons that are ResorceManager, ApplicationMaster, and NodeManager.

ResorceManager is the master daemon that arbitrates resources among all the applications in the system. NodeManager is the slave daemon responsible for containers, monitoring their resource usage, and reporting the same to ResourceManager or Schedulers. ApplicationMaster negotiates resources from the ResourceManager and works with NodeManager in order to execute and monitor the task.

The ResourceManager has two main components that are Schedulers and ApplicationsManager.

Schedulers in YARN ResourceManager is a pure scheduler which is responsible for allocating resources to the various running applications.

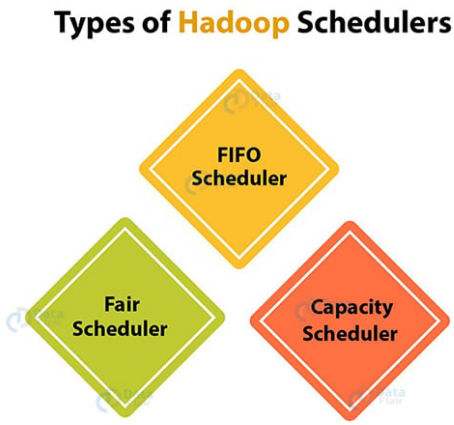
It is not responsible for monitoring or tracking the status of an application. Also, the scheduler does not guarantee about restarting the tasks that are failed either due to hardware failure or application failure.

It has some pluggable policies that are responsible for partitioning the cluster resources among the various queues, applications, etc.

The FIFO Scheduler, CapacityScheduler, and FairScheduler are such pluggable policies that are responsible for allocating resources to the applications.

Let us now study each of these Schedulers in detail.

TYPES OF HADOOP SCHEDULER



1. FIFO Scheduler

First In First Out is the default scheduling policy used in Hadoop. FIFO Scheduler gives more preferences to the application coming first than those coming later. It places the applications in a queue and executes them in the order of their submission (first in, first out).

Here, irrespective of the size and priority, the request for the first application in the queue are allocated first. Once the first application request is satisfied, then only the next application in the queue is served.

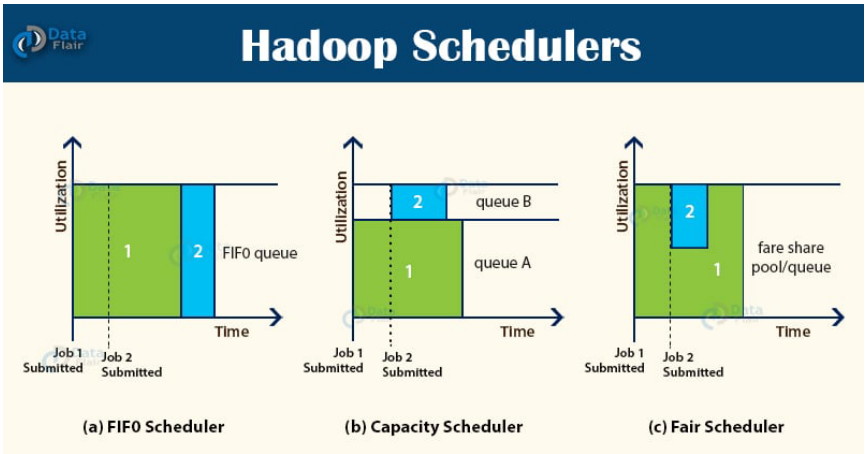
Advantage:

- It is simple to understand and doesn't need any configuration.
- Jobs are executed in the order of their submission.

Disadvantage:

- It is not suitable for shared clusters. If the large application comes before the shorter one, then the large application will use all the resources in the cluster, and the shorter application has to wait for its turn. This leads to starvation.

- It does not take into account the balance of resource allocation between the long applications and short applications.



2. Capacity Scheduler

The CapacityScheduler allows multiple-tenants to securely share a large Hadoop cluster. It is designed to run Hadoop applications in a shared, multi-tenant cluster while maximizing the throughput and the utilization of the cluster.

It supports hierarchical queues to reflect the structure of organizations or groups that utilizes the cluster resources. A queue hierarchy contains three types of queues that are root, parent, and leaf.

The root queue represents the cluster itself, parent queue represents organization/group or sub-organization/sub-group, and the leaf accepts application submission.

The Capacity Scheduler allows the sharing of the large cluster while giving capacity guarantees to each organization by allocating a fraction of cluster resources to each queue.

Also, when there is a demand for the free resources that are available on the queue who has completed its task, by the queues running below capacity, then these resources will be assigned to the applications on queues running below capacity. This provides elasticity for the organization in a cost-effective manner.

Apart from it, the CapacityScheduler provides a comprehensive set of limits to ensure that a single application/user/queue cannot use a disproportionate amount of resources in the cluster.

To ensure fairness and stability, it also provides limits on initialized and pending apps from a single user and queue.

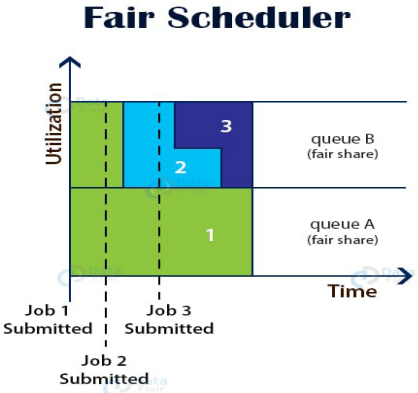
Advantages:

- It maximizes the utilization of resources and throughput in the Hadoop cluster.
- Provides elasticity for groups or organizations in a cost-effective manner.
- It also gives capacity guarantees and safeguards to the organization utilizing cluster.

Disadvantage:

- It is complex amongst the other scheduler.

3. Fair Scheduler



FairScheduler allows YARN applications to fairly share resources in large Hadoop clusters. With FairScheduler, there is no need for reserving a set amount of capacity because it will dynamically balance resources between all running applications.

It assigns resources to applications in such a way that all applications get, on average, an equal amount of resources over time.

The FairScheduler, by default, takes scheduling fairness decisions only on the basis of memory. We can configure it to schedule with both memory and CPU.

When the single application is running, then that app uses the entire cluster resources. When other applications are submitted, the free up resources are assigned to the new apps so that every app eventually gets roughly the same amount of resources. FairScheduler enables short apps to finish in a reasonable time without starving the long-lived apps.

Similar to CapacityScheduler, the FairScheduler supports hierarchical queue to reflect the structure of the long shared cluster.

Apart from fair scheduling, the FairScheduler allows for assigning minimum shares to queues for ensuring that certain users, production, or group applications always get sufficient resources. When an app is present in the queue, then the app

gets its minimum share, but when the queue doesn't need its full guaranteed share, then the excess share is split between other running applications.

Advantages:

- It provides a reasonable way to share the Hadoop Cluster between the number of users.
- Also, the FairScheduler can work with app priorities where the priorities are used as weights in determining the fraction of the total resources that each application should get.

Disadvantage: It requires configuration.

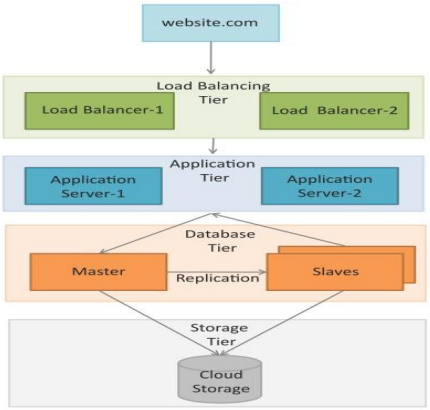
Cloud Application Design

Design Considerations for Cloud Applications

- Scalability
    - Scalability is an important factor that drives the application designers to move to cloud computing environments. Building applications that can serve millions of users without taking a hit on their performance has always been challenging. With the growth of cloud computing application designers can provision adequate resources to meet their workload levels.
- Reliability & Availability
    - Reliability of a system is defined as the probability that a system will perform the intended functions under stated conditions for a specified amount of time. Availability is the probability that a system will perform a specified function under given conditions at a prescribed time.
- Security
    - Security is an important design consideration for cloud applications given the outsourced nature of cloud computing environments.
- Maintenance & Upgradation
    - To achieve a rapid time-to-market, businesses typically launch their applications with a core set of features ready and then incrementally add new features as and when they are complete. In such scenarios, it is important to design applications with low maintenance and upgradation costs.
- Performance
    - Applications should be designed while keeping the performance requirements in mind.

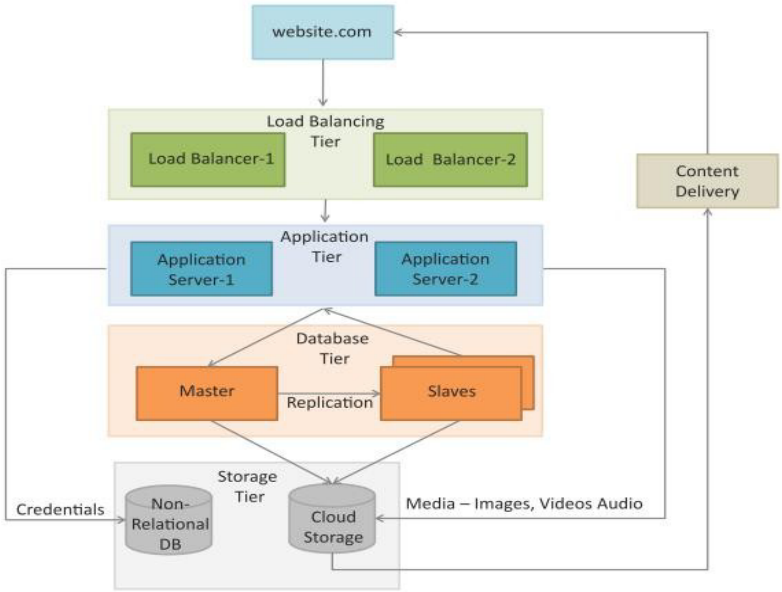
Reference Architectures –e-Commerce, Business-to-Business, Banking and Financial apps

- Load Balancing Tier
    - Load balancing tier consists of one or more load balancers.
- Application Tier
    - For this tier, it is recommended to configure auto scaling.
    - Auto scaling can be triggered when the recorded values for any of the specified metrics such as CPU usage, memory usage, etc. goes above defined thresholds.
- Database Tier
    - The database tier includes a master database instance and multiple slave instances.
    - The master node serves all the write requests and the read requests are served from the slave nodes.
    - This improves the throughput for the database tier since most applications have a higher number of read requests than write requests.



Reference Architectures –Content delivery apps

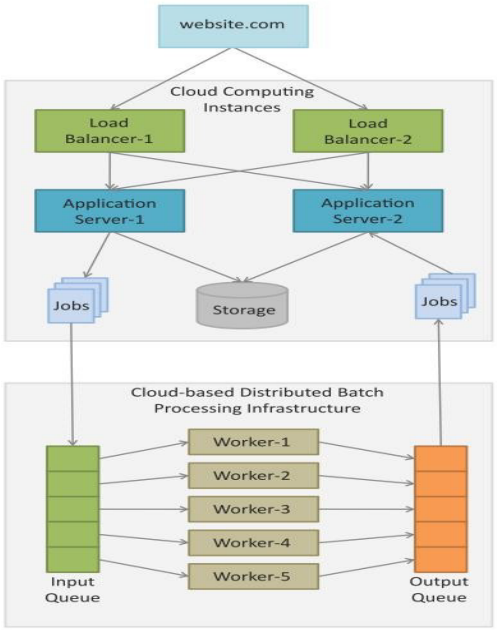
- Figure shows a typical deployment architecture for content delivery applications such as online photo albums, video webcasting, etc.
- Both relational and non-relational data stores are shown in this deployment.
- A content delivery network (CDN) which consists of a global network of edge locations is used for media delivery.
- CDN is used to speed up the delivery of static content such as images and videos.



Reference Architectures –Analytics apps

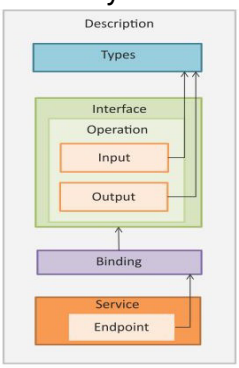
- Figure shows a typical deployment architecture for compute intensive applications such as Data Analytics, Media Transcoding, etc.
- Comprises of web, application, storage, computing/analytics and database tiers.
- The analytics tier consists of cloud-based distributed batch processing frameworks such as Hadoop which are suitable for analyzing big data.
- Data analysis jobs (such as MapReduce) jobs are submitted to the analytics tier from the application servers.

- The jobs are queued for execution and upon completion the analyzed data is presented from the application servers.



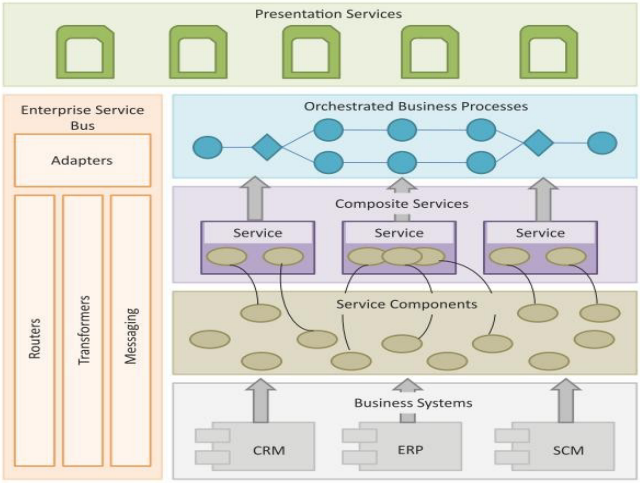
Service Oriented Architecture:

- Service Oriented Architecture (SOA) is a well established architectural approach for designing and developing applications in the form services that can be shared and reused.
- SOA is a collection of discrete software modules or services that form a part of an application and collectively provide the functionality of an application.
- SOA services are developed as loosely coupled modules with no hard- wired calls embedded in the services.
- The services communicate with each other by passing messages.
- Services are described using the Web Services Description Language (WSDL).
- WSDL is an XML-based web services description language that is used to create service descriptions containing information on the functions performed by a service and the inputs and outputs of the service.



SOA Layers:

1. **Business Systems:** This layer consists of custom built applications and legacy systems such as Enterprise Resource Planning (ERP), Customer Relationship Management (CRM), Supply Chain Management (SCM), etc.
2. **Service Components:** The service components allow the layers above to interact with the business systems. The service components are responsible for realizing the functionality of the services exposed.
3. **Composite Services:**These are coarse-grained services which are composed of two or more service components. Composite services can be used to create enterprise scale components or business-unit specific components.
4. **Orchestrated Business Processes:** Composite services can be orchestrated to create higher level business processes. In this layers the compositions and orchestrations of the composite services are defined to create business processes.
5. **Presentation Services:** This is the topmost layer that includes user interfaces that exposes the services and the orchestrated business processes to the users.
6. **Enterprise Service Bus:** This layer integrates the services through adapters, routing, transformation and messaging mechanisms.



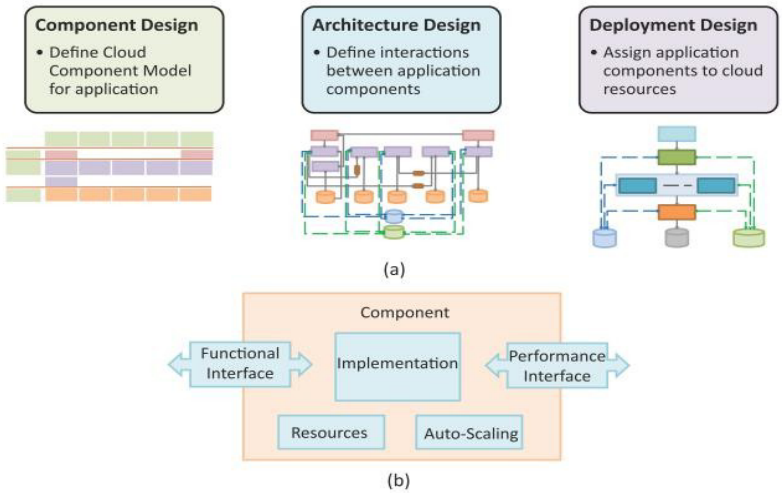
Cloud Component Model:

- Cloud Component Model is an application design methodology that provides a flexible way of creating cloud applications in a rapid, convenient and platform independent manner.
- CCM is an architectural approach for cloud applications that is not tied to any specific programming language or cloud platform.
- Cloud applications designed with CCM approach can have innovative hybrid deployments in which different components of an application can be deployed on cloud infrastructure and platforms of different cloud vendors.
- Applications designed using CCM have better portability and interoperability.
- CCM based applications have better scalability by decoupling application components and providing asynchronous communication mechanisms.

CCM Application Design Methodology:

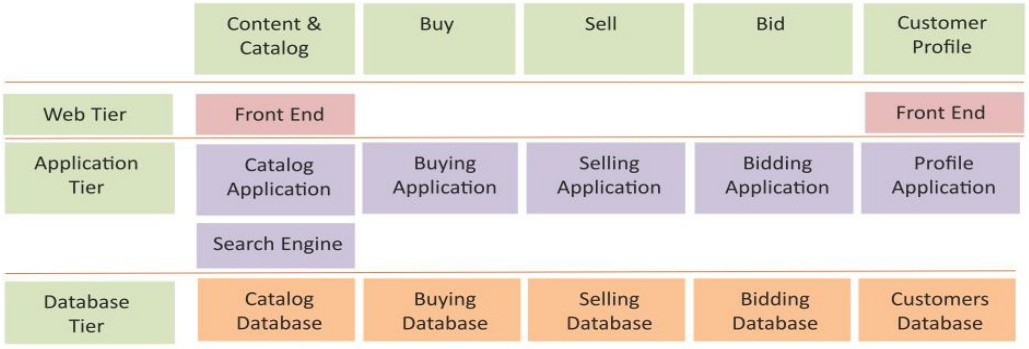
- CCM approach for application design involves:

- Component Design
- Architecture Design
- Deployment Design



CCM Component Design:

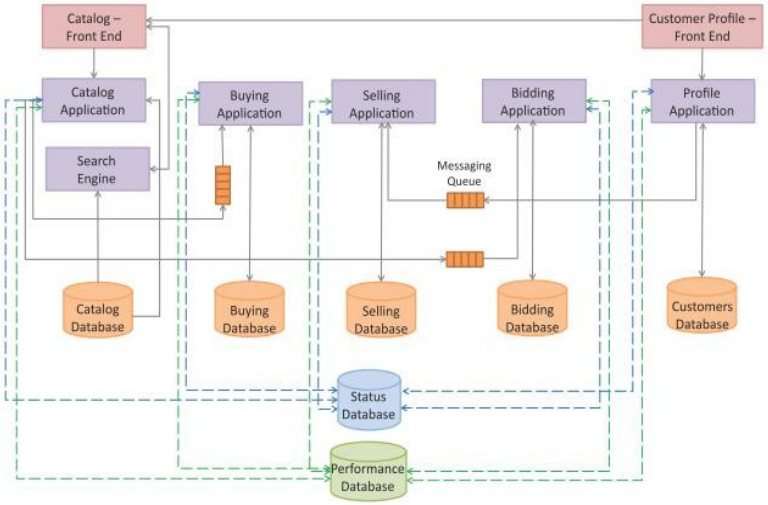
- Cloud Component Model is created for the application based on comprehensive analysis of the application's functions and building blocks.
- Cloud component model allows identifying the building blocks of a cloud application which are classified based on the functions performed and type of cloud resources required.
- Each building block performs a set of actions to produce the desired outputs for other components.
- Each component takes specific inputs, performs a pre- defined set of actions and produces the desired outputs.
- Components offer their functions as services through a functional interface which can be used by other components.
- Components report their performance to a performance database through a performance interface.



CCM Architecture Design:

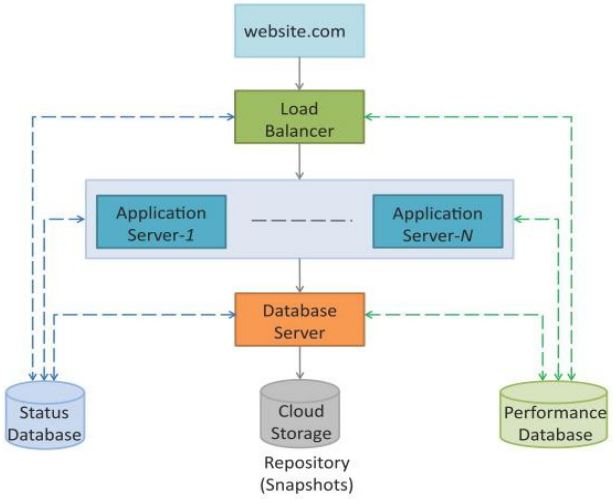
- In Architecture Design step, interactions between the application components are defined.
- CCM components have the following characteristics:
  - Loose Coupling

- Components in the Cloud Component Model are loosely coupled.
- Asynchronous Communication
  - By allowing asynchronous communication between components, it is possible to add capacity by adding additional servers when the application load increases. Asynchronous communication is made possible by using messaging queues.
- Stateless Design
  - Components in the Cloud Component Model are stateless. By storing session state outside of the component (e.g. in a database), stateless component design enables distribution and horizontal scaling.



CCM Deployment Design:

- In Deployment Design step, application components are mapped to specific cloud resources such as web servers, application servers, database servers, etc.
- Since the application components are designed to be loosely coupled and stateless with asynchronous communication, components can be deployed independently of each other.
- This approach makes it easy to migrate application components from one cloud to the other.
- With this fiexibility in application design and deployment, the application developers can ensure that the applications meet the performance and cost requirements with changing contexts.



SOA vs CCM:

Similarities

	SOA	CCM
Standardization & Re-use	SOA advocates principles of reuse and well defined relationship between service provider and service consumer.	CCM is based on reusable components which can be used by multiple cloud applications.
Loose coupling	SOA is based on loosely coupled services that minimize dependencies.	CCM is based on loosely coupled components that communicate asynchronously
Statelessness	SOA services minimize resource consumption by deferring the management of state information.	CCM components are stateless. State is stored outside of the components.

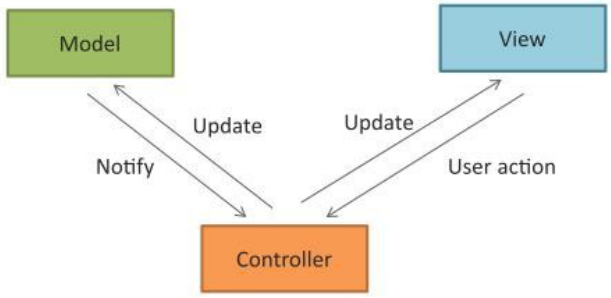
Difference

	SOA	CCM
End points	SOA services have small and well-defined set of endpoints through which many types of data can pass.	CCM components have very large number of endpoints. There is an endpoint for each resource in a component, identified by a URI.
Messaging	SOA uses a messaging layer above HTTP by using SOAP which provide prohibitive constraints to developers.	CCM components use HTTP and REST for messaging.

Security	Uses WS-Security , SAML and other standards for security	CCM components use HTTPS for security.
Interfacing	SOA uses XML for interfacing.	CCM allows resources in components represent different formats for interfacing (HTML, XML, JSON, etc.).
Consumption	Consuming traditional SOA services in a browser is cumbersome.	CCM components and the underlying component resources are exposed as XML, JSON (and other formats) over HTTP or REST, thus easy to consume in the browser.

Model View Controller:

- Model View Controller (MVC) is a popular software design pattern for web applications.
- **Model**
  - Model manages the data and the behavior of the applications. Model processes events sent by the controller. Model has no information about the views and controllers. Model responds to the requests for information about its state (from the view) and responds to the instructions to change state (from controller).
- **View**
  - View prepares the interface which is shown to the user. Users interact with the application through views. Views present the information that the model or controller tell the view to present to the user and also handle user requests and sends them to the controller.
- **Controller**
  - Controller glues the model to the view. Controller processes user requests and updates the model when the user manipulates the view. Controller also updates the view when the model changes.



RESTful Web Services:

- Representational State Transfer (REST) is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred.

- The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system.
- A RESTful web service is a web API implemented using HTTP and REST principles.
- The REST architectural constraints are as follows:
    - Client-Server
    - Stateless
    - Cacheable
    - Layered System
    - Uniform Interface
    - Code on demand

Relational Databases:

- A relational database is database that conforms to the relational model that was popularized by Edgar Codd in 1970.
- The 12 rules that Codd introduced for relational databases include:
    - Information rule
    - Guaranteed access rule
    - Systematic treatment of null values
    - Dynamic online catalog based on relational model
    - Comprehensive sub-language rule
    - View updating rule
    - High level insert, update, delete
    - Physical data independence
    - Logical data independence
    - Integrity independence
    - Distribution independence
    - Non-subversion rule
- **Relations:**A relational database has a collection of relations (or tables). A relation is a set of tuples (or rows).
- **Schema:**Each relation has a fixed schema that defines the set of attributes (or columns in a table) and the constraints on the attributes.
- **Tuples:**Each tuple in a relation has the same attributes (columns). The tuples in a relation can have any order and the relation is not sensitive to the ordering of the tuples.
- **Attributes:**Each attribute has a domain, which is the set of possible values for the attribute.
- **Insert/Update/Delete:**Relations can be modified using insert, update and delete operations. Every relation has a primary key that uniquely identifies each tuple in the relation.
- **Primary Key:**An attribute can be made a primary key if it does not have repeated values in different tuples.

ACID Guarantees:

Relational databases provide ACID guarantees.

- **Atomicity:**Atomicity property ensures that each transaction is either "all or nothing". An atomic transaction ensures that all parts of the transaction complete or the database state is left unchanged.
- **Consistency:**Consistency property ensures that each transaction brings the database from one valid state to another. In other words, the data in a database always conforms to the defined schema and constraints.
- **Isolation:**Isolation property ensures that the database state obtained after a set of concurrent transactions is the same as would have been if the transactions were executed serially. This provides concurrency control, i.e. the results of incomplete transactions are not visible to other transactions. The transactions are isolated from each other until they finish.
- **Durability:**Durability property ensures that once a transaction is committed, the data remains as it is, i.e. it is not affected by system outages such as power loss. Durability guarantees that the database can keep track of changes and can recover from abnormal terminations**.**

Non-Relational Databases

- Non-relational databases (or popularly called No-SQL databases) are becoming popular with the growth of cloud computing.
- Non-relational databases have better horizontal scaling capability and improved performance for big data at the cost of less rigorous consistency models.
- Unlike relational databases, non-relational databases do not provide ACID guarantees.
- Most non-relational databases offer "eventual" consistency, which means that given a sufficiently long period of time over which no updates are made, all updates can be expected to propagate eventually through the system and the replicas will be consistent.
- The driving force behind the non-relational databases is the need for databases that can achieve high scalability, fault tolerance and availability.

- These databases can be distributed on a large cluster of machines. Fault tolerance is provided by storing multiple replicas of data on different machines.

Non-Relational Databases – Types

- Key-value store: Key-value store databases are suited for applications that require storing unstructured data without a fixed schema. Most key-value stores have support for native programming language data types.
- Document store: Document store databases store semi-structured data in the form of documents which are encoded in different standards such as JSON, XML, BSON, YAML, etc.
- Graph store: Graph stores are designed for storing data that has graph structure (nodes and edges). These solutions are suitable for applications that involve graph data such as social networks, transportation systems, etc.
- Object store: Object store solutions are designed for storing data in the form of objects defined in an object-oriented programming language.

Python Basics:

Pythonisageneral-purposehighlevelprogramminglanguageandsuitableforprovidingasolidfoundationto thereaderinthe area ofcloud computing.

ThemaincharacteristicsofPythonare:

Multi-paradigmprogramminglanguage

Pythonsupportsmorethanoneprogrammingparadigmsincludingobject-orientedprogrammingandstructuredprogramming

InterpretedLanguage

Python is an interpreted language and does not require an explicit compilation step. The Python interpreterexecutestheprogramsourcecodedirectly,statementbystatement,asaprocessororscriptingenginedoes.

InteractiveLanguage

Pythonprovidesaninteractivemodeinwhichtheusercansubmitcommandsatthe Pythonpromptandinteractwiththeinterpreterdirectly.

Python – Benefits

- **Easy-to-learn, read and maintain**
  - Python is a minimalistic language with relatively few keywords, uses English keywords and has fewer syntactical constructions as compared to other languages. Reading Python programs feels like English with pseudo-code like constructs. Python is easy to learn yet an extremely powerful language for a wide range of applications.
- **Object and Procedure Oriented**
  - Python supports both procedure-oriented programming and object-oriented programming. Procedure oriented paradigm allows programs to be written around procedures or functions that allow reuse

of code. Procedure oriented paradigm allows programs to be written around objects that include both data and functionality.

- **Extendable**
  - Python is an extendable language and allows integration of low-level modules written in languages such as C/C++. This is useful when you want to speed up a critical portion of a program.
- **Scalable**
  - Due to the minimalistic nature of Python, it provides a manageable structure for large programs.
- **Portable**
  - Since Python is an interpreted language, programmers do not have to worry about compilation, linking and loading of programs. Python programs can be directly executed from source
- **Broad Library Support**
  - Python has a broad library support and works on various platforms such as Windows, Linux, Mac, etc.

Python – Setup

- Windows
  - Python binaries for Windows can be downloaded from http://www.python.org/getit .
  - For the examples and exercise in this book, you would require Python 2.7 which can be directly downloaded from: http://www.python.org/ftp/python/2.7.5/python-2.7.5.msi
  - Once the python binary is installed you can run the python shell at the command prompt using

    > python
  - Linux

```
#Install Dependencies
sudo apt-get install build-essential
sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev
libsqlite3-devtk-dev libgdbm-dev libc6-dev libbz2-dev
#Download Python
wget http://python.org/ftp/python/2.7.5/Python-2.7.5.tgz tar -xvf
Python-2.7.5.tgz
cd Python-2.7.5
#Install Python
./configure make
sudo make install
```

Numbers

- Numbers
  - Number data type is used to store numeric values. Numbers are immutable data types, therefore changing the value of a number data type results in a newly allocated object.