# What makes querying so fast in a Data Warehouse?

## Distribution Styles and Keys

When you create a table, you can designate one of four distribution styles; AUTO, EVEN, KEY, or ALL.

If you don't specify a distribution style, Amazon Redshift uses AUTO distribution.

### AUTO distribution

- With AUTO distribution, Amazon Redshift assigns an optimal distribution style based on the size of the table data.
- For example, Amazon Redshift initially assigns ALL distribution to a small table, then changes to EVEN distribution when the table grows larger.
- When a table is changed from ALL to EVEN distribution, storage utilization might change slightly. The change in distribution occurs in the background, in a few seconds.

### EVEN distribution

- The leader node distributes the rows across the slices in a round-robin fashion, regardless of the values in any particular column.
- EVEN distribution is appropriate when a table doesn't participate in joins. It's also appropriate when there isn't a clear choice between KEY distribution and ALL distribution.

### KEY distribution

- The rows are distributed according to the values in one column. The leader node places matching values on the same node slice.
- If you distribute a pair of tables on the joining keys, the leader node collocates the rows on the slices according to the values in the joining columns. This way, matching values from the common columns are physically stored together.

### ALL distribution

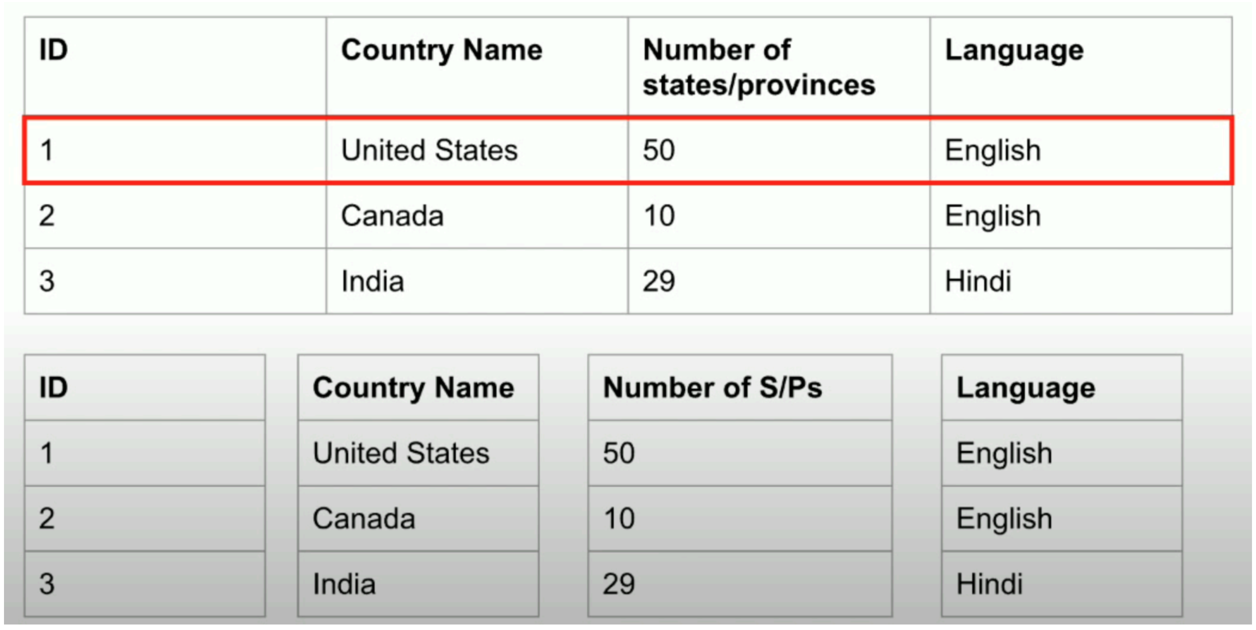
- A copy of the entire table is distributed to every node. Where EVEN distribution or KEY distribution place only a portion of a table's rows on each node, ALL distribution ensures that every row is collocated for every join that the table participates in.
- ALL distribution multiplies the storage required by the number of nodes in the cluster, and so it takes much longer to load, update, or insert data into multiple tables.
- ALL distribution is appropriate only for relatively slow moving tables; that is, tables that are not updated frequently or extensively. Because the cost of redistributing small tables during a query is low, there isn't a significant benefit to define small dimension tables as DISTSTYLE ALL.

## 6 essential features of Redshift

Here are the six features of that architecture that help Redshift stand out from other data warehouses.

### 1. Column-oriented databases

Data can be organized either into rows or columns. What determines the type of method is the nature of the workload.



| ID | Country Name | Number of states/provinces | Language |
|---|---|---|---|
| 1 | United States | 50 | English |
| 2 | Canada | 10 | English |
| 3 | India | 29 | Hindi |

| ID |
|---|
| 1 |
| 2 |
| 3 |

| Country Name |
|---|
| United States |
| Canada |
| India |

| Number of S/Ps |
|---|
| 50 |
| 10 |
| 29 |

| Language |
|---|
| English |
| English |
| Hindi |

- The most common system of organizing data is by row. That's because row-oriented systems are designed to quickly process a large number of small operations. This is known as online transaction processing, or OLTP, and is used by most operational databases.
- In contrast, column-oriented databases allow for increased speed when it comes to accessing large amounts of data.
- For example, in an online analytical processing or OLAP environment such as Redshift, users generally apply a smaller number of queries to much larger datasets.
- In this scenario, being a column-oriented database allows Redshift to complete massive data processing jobs quickly.

### 2. Massively parallel processing (MPP)

- MPP is a distributed design approach in which several processors apply a "divide and conquer" strategy to large data jobs.
- A large processing job is organized into smaller jobs which are then distributed among a cluster of processors (compute nodes).

- The processors complete their computations simultaneously rather than sequentially. The result is a large reduction in the amount of time Redshift needs to complete a single, massive job.

**3. End-to-end data encryption**

- No business or organization is exempt from data privacy and security regulations, and encryption is one of the pillars of data protection.
- Encryption options in Redshift are robust and highly customizable. This flexibility allows users to configure an encryption standard that best fits their needs. Redshift security encryption features include:
- Migrating data between encrypted and unencrypted clusters

**4. Network isolation**

- For businesses that want additional security, administrators can choose to isolate their network within Redshift.
- In this scenario, network access to an organization's cluster(s) is restricted by enabling the Amazon VPC.
- The user's data warehouse remains connected to the existing IT infrastructure with IPsec VPN.

**5. Fault tolerance**

- Fault tolerance refers to the ability of a system to continue functioning even when some components fail. When it comes to data warehousing, fault tolerance determines the capacity for a job to continue being run when some processors or clusters are offline.
- Data accessibility and data warehouse reliability are paramount for any user.
- AWS monitors its clusters around the clock. When drives, nodes, or clusters fail, Redshift automatically re-replicates data and shifts data to healthy nodes.

**6. Concurrency limits**

- Concurrency limits determine the maximum number of nodes or clusters that a user can provision at any given time.
- These limits ensure that adequate compute resources are available to all users. In this sense, concurrency limits democratize the data warehouse.
- Redshift maintains concurrency limits that are similar to other data warehouses, but with a degree of flexibility. For example, the number of nodes that are available per cluster is determined by the cluster's node type.
- Redshift also configures limits based on regions, rather than applying a single limit to all users. Finally, in some situations, users may submit a limit increase request.

## How do we ensure we get the best out of Redshift?

Some suggestions for the best approach follow:

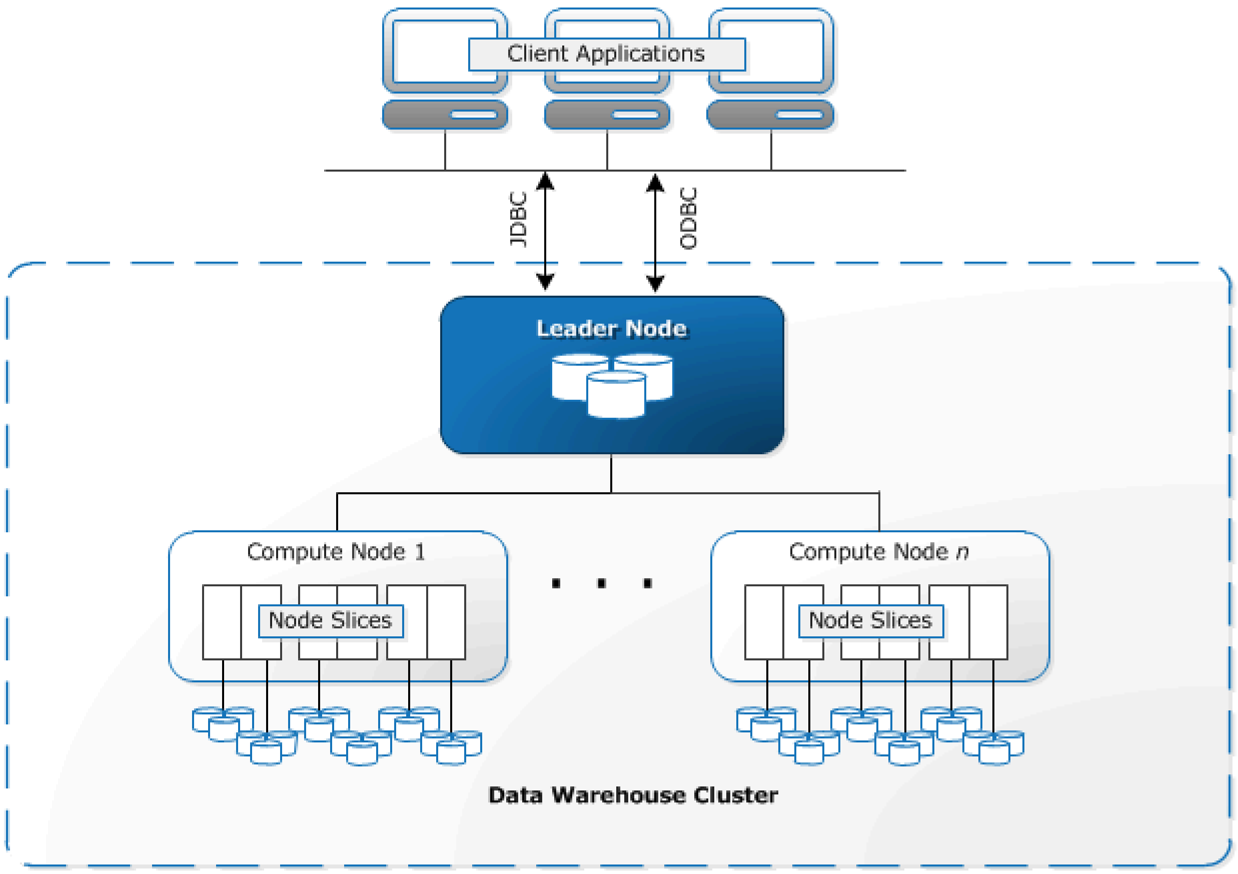
- To have Amazon Redshift choose the appropriate sort order, specify AUTO for the sort key.
- If recent data is queried most frequently, specify the timestamp column as the leading column for the sort key.
  Queries are more efficient because they can skip entire blocks that fall outside the time range.
- If you do frequent range filtering or equality filtering on one column, specify that column as the sort key.
  Amazon Redshift can skip reading entire blocks of data for that column.
- It can do so because it tracks the minimum and maximum column values stored on each block and can skip blocks that don't apply to the predicate range.
- If you frequently join a table, specify the join column as both the sort key and the distribution key.
- Doing this enables the query optimizer to choose a sort-merge join instead of a slower hash join. Because the data is already sorted on the join key, the query optimizer can bypass the sort phase of the sort-merge join.

## How do we optimize data storage in Redshift?

- A *compression encoding* specifies the type of compression that is applied to a column of data values as rows are added to a table.
- ENCODE AUTO is the default for tables. Amazon Redshift automatically manages compression encoding for all columns in the table.
- You can specify the ENCODE AUTO option for the table to enable Amazon Redshift to automatically manage compression encoding for all columns in the table. For more information, see CREATE TABLE and ALTER TABLE.
- However, if you specify compression encoding for any column in the table, the table is no longer set to ENCODE AUTO. Amazon Redshift no longer automatically manages compression encoding for all columns in the table.
- If you specify compression encoding for any column in the table or if you don't specify the ENCODE AUTO option for the table, Amazon Redshift automatically assigns compression encoding to columns for which you don't specify compression encoding as follows:
    - Columns that are defined as sort keys are assigned RAW compression.
    - Columns that are defined as BOOLEAN, REAL, or DOUBLE PRECISION data types are assigned RAW compression.
    - Columns that are defined as SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIMESTAMP, or TIMESTAMPTZ data types are assigned AZ64 compression.

    ◦ Columns that are defined as CHAR or VARCHAR data types are assigned LZO compression.

## Redshift Architecture



### Cluster

- The core infrastructure component of an Amazon Redshift data warehouse is a *cluster*.
- A distributed systems cluster is a group of machines that are virtually or geographically separated and that work together to provide the same service or application to clients.
- It is possible that many of the services you run in your network today are part of a distributed systems cluster.

### Leader node

- The leader node manages communications with client programs and all communication with compute nodes. It parses and develops execution plans to carry out database operations, in particular, the series of steps necessary to obtain results for complex queries.
- Based on the execution plan, the leader node compiles code, distributes the compiled code to the compute nodes, and assigns a portion of the data to each compute node.
- The leader node distributes SQL statements to the compute nodes only when a query references tables that are stored on the compute nodes. All other queries run exclusively on the leader node.
- Amazon Redshift is designed to implement certain SQL functions only on the leader node.

### Compute nodes

- The leader node compiles code for individual elements of the execution plan and assigns the code to individual compute nodes. The compute nodes run the compiled code and send intermediate results back to the leader node for final aggregation.
- Each compute node has its own dedicated CPU, memory, and attached disk storage, which are determined by the node type. As your workload grows, you can increase the compute capacity and storage capacity of a cluster by increasing the number of nodes, upgrading the node type, or both.
- Amazon Redshift provides several node types for your compute and storage needs.

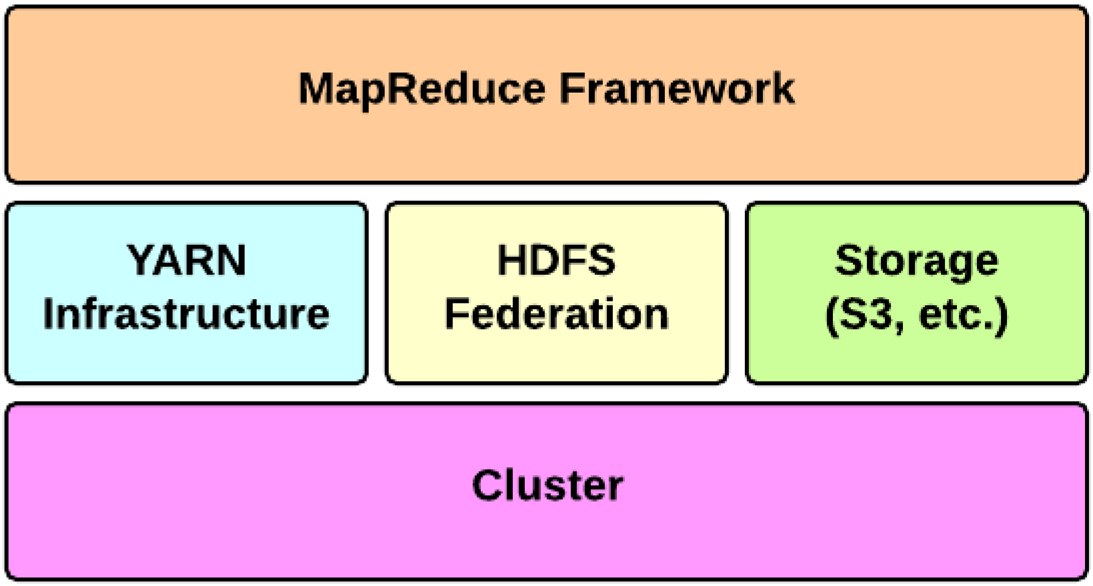
### Node slices

- A compute node is partitioned into slices. Each slice is allocated a portion of the node's memory and disk space, where it processes a portion of the workload assigned to the node.
- The leader node manages distributing data to the slices and apportions the workload for any queries or other database operations to the slices. The slices then work in parallel to complete the operation.
- The number of slices per node is determined by the node size of the cluster.
- When you create a table, you can optionally specify one column as the distribution key. When the table is loaded with data, the rows are distributed to the node slices according to the distribution key that is defined for a table.
- Choosing a good distribution key enables Amazon Redshift to use parallel processing to load data and run queries efficiently

## Map Reduce Yarn

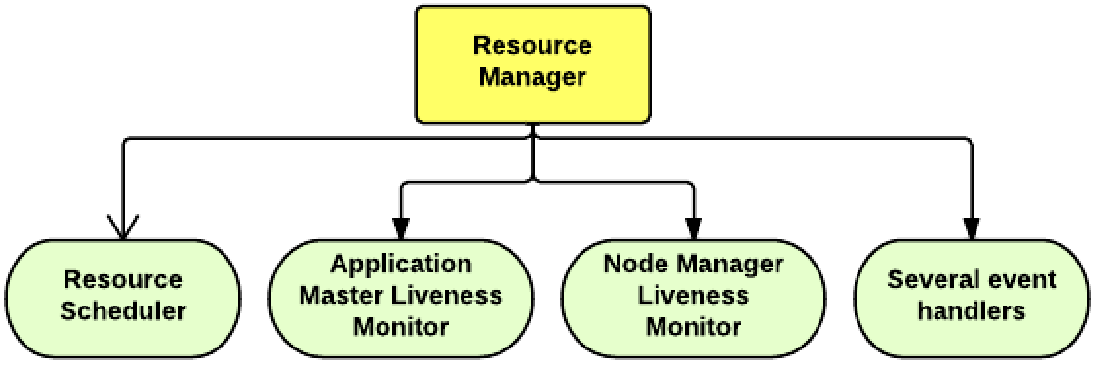
**Let's try to recall hadoop ecosystem what we discussed in Intro to Data Engineering**

Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware. There are mainly five building blocks inside this runtime environment (from bottom to top):

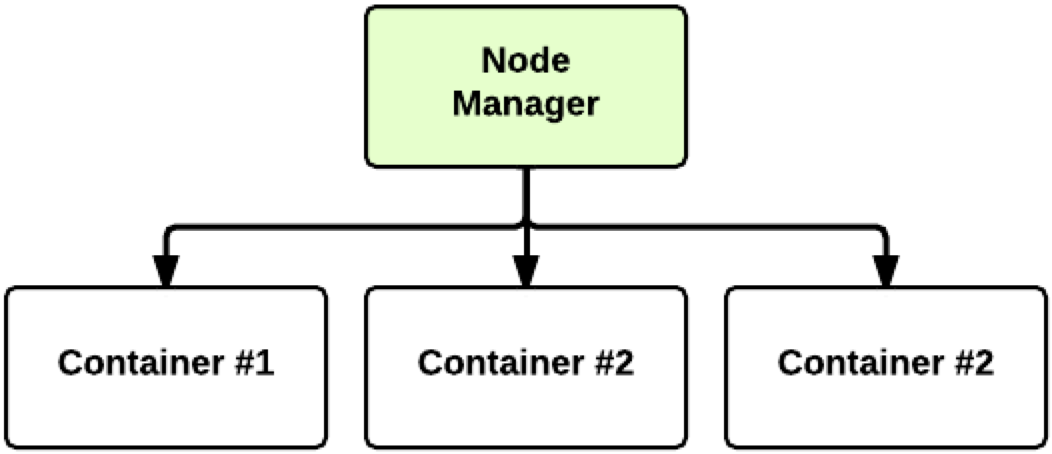


- The cluster is the set of host machines (nodes). Nodes may be partitioned in racks. This is the hardware part of the infrastructure.
- The YARN Infrastructure (Yet Another Resource Negotiator) is the framework responsible for providing the computational resources (e.g., CPUs, memory, etc.) needed for application executions. Two important elements are:
    - The Resource Manager (one per cluster) is the master. It knows where the slaves are located (Rack Awareness) and how many resources they have. It runs several services, the most important is

the Resource Scheduler which decides how to assign the resources.


Resource Manager
Resource Scheduler
Application Master Liveness Monitor
Node Manager Liveness Monitor
Several event handlers

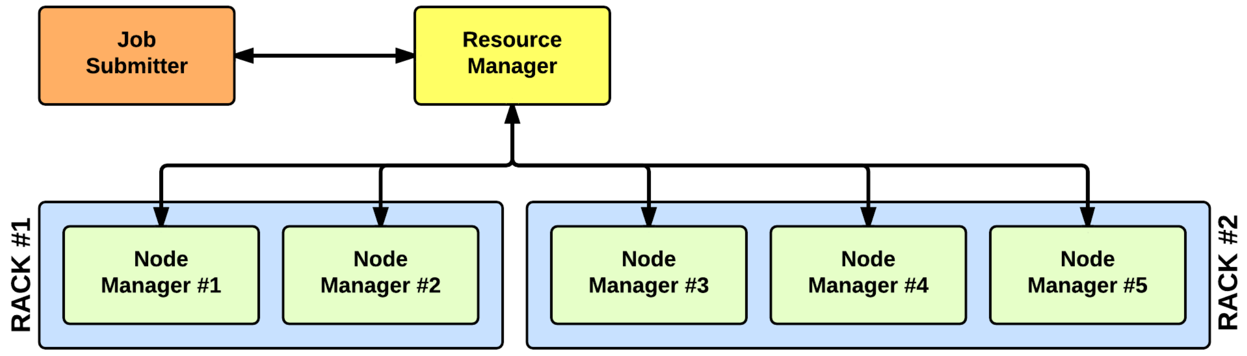
- The Node Manager (many per cluster) is the slave of the infrastructure. When it starts, it announces itself to the Resource Manager. Periodically, it sends a heartbeat to the Resource Manager. Each Node Manager offers some resources to the cluster. Its resource capacity is the amount of memory and the number of vcores. At run-time, the Resource Scheduler will decide how to use this capacity: a Container is a fraction of the NM capacity and it is used by the client
  for running a programme.



- The HDFS Federation is the framework responsible for providing permanent, reliable and distributed storage. This is typically used for storing inputs and output (but not intermediate ones).
- Other alternative storage solutions. For instance, Amazon uses the Simple Storage Service (S3).
- The MapReduce Framework is the software layer implementing the MapReduce paradigm.

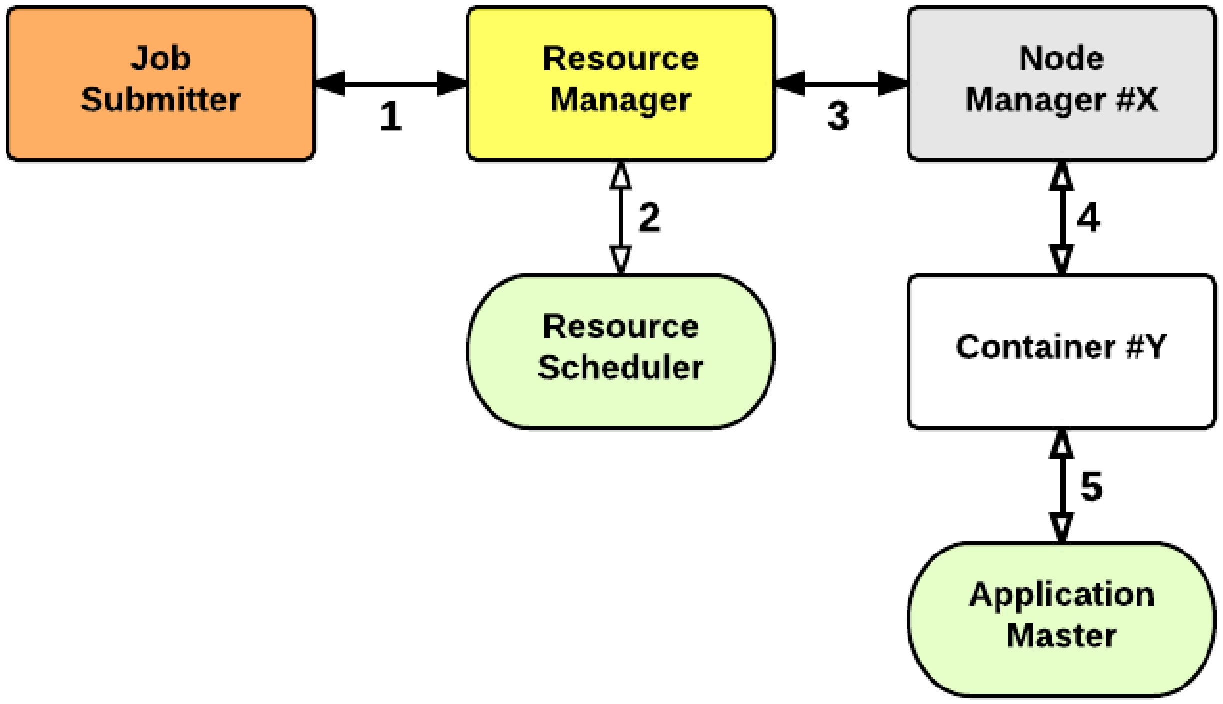
The YARN infrastructure and the HDFS federation are completely decoupled and independent: the first one provides resources for running an application while the second one provides storage. The MapReduce framework runs on top of YARN.

YARN: Application Startup



In YARN, there are at least three actors:

- the Job Submitter (the client)
- the Resource Manager (the master)
- the Node Manager (the slave)



The application startup process is the following:

1. A client submits an application to the Resource Manager
2. The Resource Manager allocates a container
3. The Resource Manager contacts the related Node Manager
4. The Node Manager launches the container
5. The Container executes the Application Master