**Apache Spark**

* Apache spark project is aims at providing a distributed computing engine to work with massive data
* It doesn’t worry about the storage options, resource allocation etc. Apache spark can be integrated with Cluster managers like
  + Apache Mesos
  + Yarn
  + Standalone cluster manager ( in built by Apache Spark)

***Core Principle***

* In memory computing which are resilient in nature. (i.e) immutable and if there is any fault , the data can be retrieved from previous step rather than starting from first
* Each stage is called a transformation. Each stage data is all linked via DAG – Direct acyclic graph.
* Action step – This step is where the actual output/convergence takes place

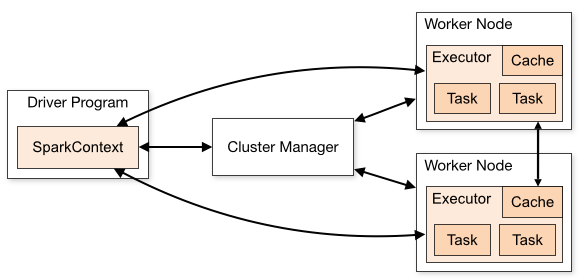
**Example** : 10000 lines of data is analyzed

1. Read the file 🡪 Outputs an RDD (Transformation)
2. Filter records which contain hello 🡪 outputs another RDD (Transformation)
3. Save as Text File / Table / Hive store etc 🡪 Action

If there is a failure in step 3 , the job need not start from Step 1 – instead it will start from step 2

**Apache Spark architecture**

* **Driver program** is the place where spark application is submitted and where spark context is created
* Drive program communicates with **Cluster manager ( Yarn, Mesos)** for resources etc. and works with worker nodes to get the work done.
* All the tasks happens in worker nodes in form of executors
* The final result like saving to tables, file etc happens in drive nodes



**Worker Node/ Slave Node/ Data Node**

* Daemon process is running in each node
* Different executors would be created based on the number of cores available

<< When reading the HDFS data, the data is split into chunks or partitions. These partitions are allocated to each tasks>>

* Each partition will be created as a separate **task** to work with

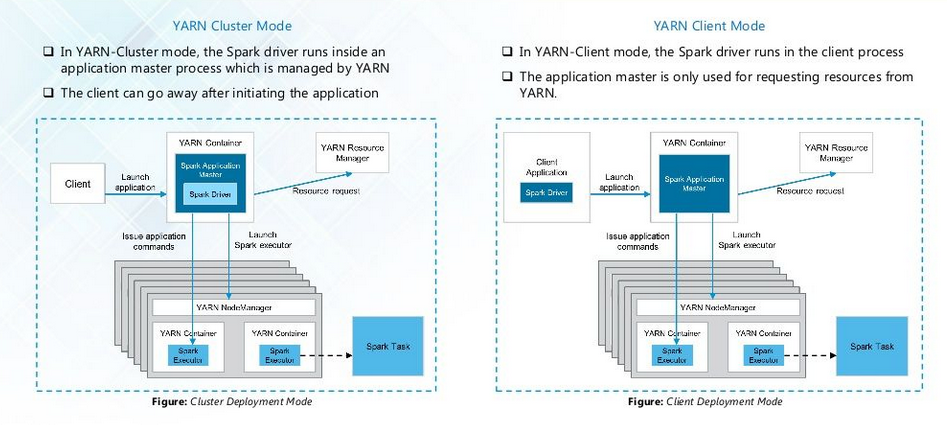
<< By default, number of RDD partitions is chosen automatically but there is an option to provide with custom numbers>>

**Spark Context is the handle of Spark application**

**Building Spark Applications**

* Build a Fat jar in case of java and invoke spark-submit

**Deploying Spark Applications**

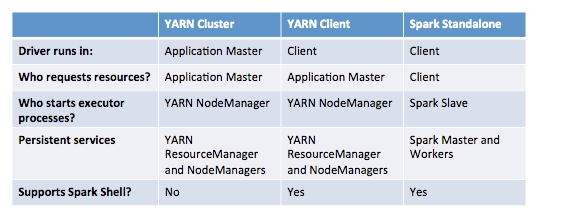


**Client mode**

* spark-shell,spark-sql etc . Especially anything which needs interaction can use client mode
* Driver and Resource negotiation with Yarn happens as different entities

**Cluster mode**

* Client can initiate the application and go away basically
* Driver and Resource negotiation with Yarn happens at same place



**RDD**