* The complete structure and the working of “Oozie Workflow scheduler “.

**Apache Oozie is a Java Web application used to schedule Apache Hadoop jobs. It supports Apache MapReduce, Apache Pig, Apache Hive, and Apache Sqoop. Oozie can also schedule jobs specific to a system, like Java programs or shell scripts.**

**Oozie accomplishes the jobs in two parts**

**Oozie Workflow**

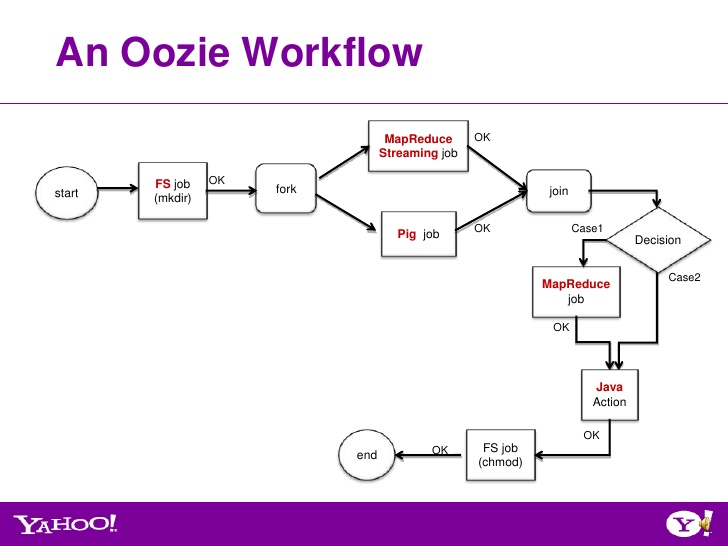
**Oozie Coordinator**

**Oozie Workflow**

**Oozie workflow are the set of actions that are arranged in directed acyclic graph (DAG). This graph can contain two types of nodes: control nodes and action nodes.control nodes are used to define how the jobs will progress in the course of execution. Also the order of the job execution is also decide by the control node, provide the rules for beginning and ending a workflow. The action node triggers the execution of the job.**

**Workflow of oozie uses the feedback method in order to get the status of the task.**

**The job sends the completion status to oozie if the job is completed and if the feedback is not received then oozie assumes that the job is not complete and still needs time to complete and polls it for the job execution. This feedback received id by a unique callback HTTP URL which is provided to the task at the time of start. Oozie workflows can be also parameterized. When submitting a workflow job, values for the parameters must be provided. If the appropriate parameters are used, several identical workflow jobs can occur concurrently.**



**Working of the workflow –**

**Here once task starts it make a file for the job and checks the number jobs in the task and checks interdependency of the job on each other. Then it will fork the job according the flow mentioned by control node. Again the results will be joined. If the further processing of the result is required then the again the map reduce job will be done else directly will lead to end of the task.**

**Oozie Coordinator – As the name suggests that it is used to co-ordinate amoung different jobs that are running parallelly. Sometimes it may happen that start of certain job may be dependent on the end of certain job. All this is managed by oozie co-ordinator. The Oozie Coordinator can also manage multiple workflows that are dependent on the outcome of subsequent workflows. The outputs of subsequent workflows become the input to the next workflow. This chain is called a *data application pipeline*.**

**Oozie working is defined using the xml file and is called Hadoop process definition language. Oozie comes with a command-line program for submitting jobs.This command-line program interacts with the Oozie server using REST .**

**Example of the Hadoop process definition language.**

<workflow-app xmlns='uri:oozie:workflow:0.1' name='SimpleWorkflow'>

    <start to='ingestor'/>

    <action name='ingestor'>

        </java>

            <job-tracker>${jobTracker}</job-tracker>

            <name-node>${nameNode}</name-node>

            <configuration>

                <property>

                    <name>mapred.job.queue.name</name>

                    <value>default</value>

                </property>

            </configuration>

            <arg>${driveID}</arg>

        </java>

        <ok to='merging'/>

        <error to='fail'/>

    </action>

    <fork name='merging'>

        <path start='mergeT1'/>

        <path start='mergeT2'/>

    </fork>

    <action name='mergeT1'>

        <java>

            <job-tracker>${jobTracker}</job-tracker>

            <name-node>${nameNode}</name-node>

            <configuration>

                <property>

                    <name>mapred.job.queue.name</name>

                    <value>default</value>

                </property>

            </configuration>

            <arg>-drive</arg>

            <arg>${driveID}</arg>

            <arg>-type</arg>

            <arg>T1</arg>

        </java>

        <ok to='completed'/>

        <error to='fail'/>

    </action>

    <action name='mergeT2'>

        <java>

            <job-tracker>${jobTracker}</job-tracker>

            <name-node>${nameNode}</name-node>

            <configuration>

                <property>

                    <name>mapred.job.queue.name</name>

                    <value>default</value>

                </property>

            </configuration>

            <main-class>com.navteq.assetmgmt.hdfs.merge.MergerLoader</main-class>

            <arg>-drive</arg>

            <arg>${driveID}</arg>

            <arg>-type</arg>

            <arg>T2</arg>

        </java>

        <ok to='completed'/>

        <error to='fail'/>

    </action>

    <join name='completed' to='end'/>

    <kill name='fail'>

        <message>Java failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>

    </kill>

    <end name='end'/>

</workflow-app>