**ASSIGNMENT 34.1**

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

**Apache oozie:**

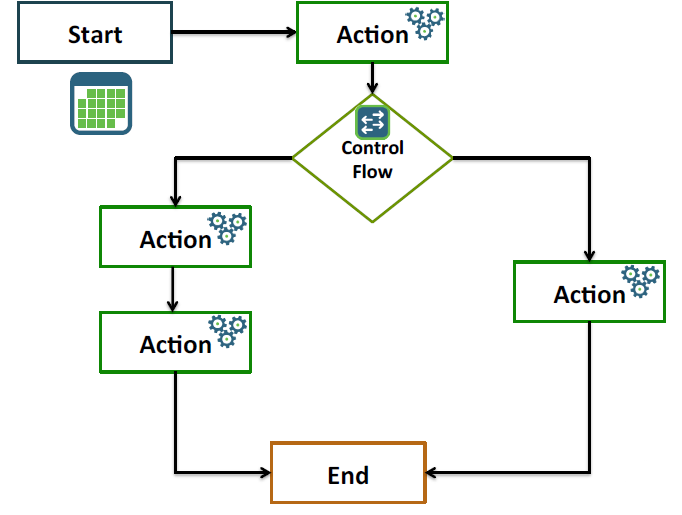
* Apache Oozie is an open source project based on Java technology that simplifies the process of creating workflows and managing coordination among jobs.
* In principle, Oozie offers the ability to combine multiple jobs sequentially into one logical unit of work.
* One advantage of the Oozie framework is that it is fully integrated with the Apache Hadoop stack and supports Hadoop jobs for Apache MapReduce, Pig, Hive, and Sqoop.
* In addition, it can be used to schedule jobs specific to a system, such as Java programs.
* Therefore, using Oozie, Hadoop administrators are able to build complex data transformations that can combine the processing of different individual tasks and even sub-workflows.
* This ability allows for greater control over complex jobs and makes it easier to repeat those jobs at predetermined periods.

**In practice, there are different types of Oozie jobs:**

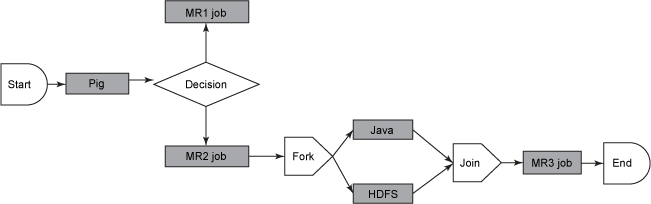
* **Oozie Workflow jobs** — Represented as directed acyclical graphs to specify a sequence of actions to be executed.
* **Oozie Coordinator jobs** — Represent Oozie workflow jobs triggered by time and data availability.
* **Oozie Bundle**— Facilitates packaging multiple coordinator and workflow jobs, and makes it easier to manage the life cycle of those jobs.

**Structure and Working of oozie scheduler:**

* An Oozie workflow is a collection of actions arranged in a directed acyclic graph (DAG). This graph can contain two types of nodes:
* control nodes
* action nodes.
* *Control nodes*, which are used to define job chronology, provide the rules for beginning and ending a workflow and control the workflow execution path with possible decision points known as fork and join nodes.
* *Action nodes* are used to trigger the execution of tasks. In particular, an action node can be a MapReduce job, a Pig application, a file system task, or a Java application. (The shell and ssh actions have been deprecated).



* Oozie is a native Hadoop stack integration that supports all types of Hadoop jobs and is integrated with the Hadoop stack.
* In particular, Oozie is responsible for triggering the workflow actions, while the actual execution of the tasks is done using Hadoop MapReduce.
* Therefore, Oozie becomes able to leverage existing Hadoop machinery for load balancing, fail-over, etc. Oozie detects completion of tasks through callback and polling. When Oozie starts a task, it provides a unique callback HTTP URL to the task, and notifies that URL when it is complete.
* If the task fails to invoke the callback URL, Oozie can poll the task for completion. Figure 1 illustrates a sample Oozie workflow that combines six action nodes (Pig scrip, MapReduce jobs, Java code, and HDFS task) and five control nodes (Start, Decision control, Fork, Join, and End).
* 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.



**BENEFITS OF OOZIE:**

1. Oozie is designed to scale in a Hadoop cluster. Each job will be launched from a different datanode. This means that the workflow load will be balanced and no single machine will become overburdened by launching workflows. This also means that the capacity to launch workflows will grow as the cluster grows.
2. Oozie is well integrated with Hadoop security. This is especially important in a kerberized cluster. Oozie knows which user submitted the job and will launch all actions as that user, with the proper privileges. It will handle all the authentication details for the user as well.
3. Oozie is the only workflow manager with built-in Hadoop actions, making workflow development, maintenance and troubleshooting easier.
4. Oozie UI makes it easier to drill down to specific errors in the data nodes. Other systems would require significantly more work to correlate jobtracker jobs with the workflow actions.

**SAMPLE WORKFLOW OF WORDCOUNT PROGRAM:**

<workflow-app xmlns='uri:oozie:workflow:0.1' name='java-main-wf'>  
<start to='mapreduce-wordcount-example' />  
<action name='mapreduce-wordcount-example'>  
<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>org.apache.hadoop.examples.ExampleDriver</main-class>  
<arg>wordcount</arg>  
<arg>${inputDir}</arg>  
<arg>${outputDir}</arg>  
</java>  
<ok to="end" />  
<error to="fail" />  
</action>  
<kill name="fail">  
<message>Java failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>  
</kill>  
<end name='end' />  
</workflow-app>