ASSIGNMENT 10.1

Problem Statement:

Explain in Brief:

1. The workflow of Oozie and its Benefits
2. The workflow of Sqoop and its Benefits

Solution:

1. The workflow of Oozie and its Benefits:

Apache Oozie is a server based workflow engine that simplifies the mechanism of creating workflows, scheduling jobs and managing coordination among jobs. Let’s focus on the workflow part as stated in the given problem.

Oozie workflow jobs are represented as Directed Acyclic Graphs (DAG’s) that define a set of actions to be executed. This graph has two kinds of nodes: control nodes and action nodes.

1. Control nodes: these nodes are used to specify the beginning and end of a workflow and they also control the execution path of the workflow with possible decision points and with operations such as fork and join nodes.
2. Action nodes: these nodes are used to trigger the execution of a task or computation. An action node can be a MapReduce job, a Pig application, an HDFS task or a custom Java application.

Since Oozie is a native Hadoop ecosystem component it supports all kinds of Hadoop tasks, especially it is used for triggering actions on workflow and execution of those actions will be carried out using Hadoop MapReduce. The completion of tasks are identified via callback and polling methods. Whenever a task is started on Oozie, a unique HTTP callback URL is provided to the client and it also indicates on the same URL when the task gets completed. In case a task couldn’t invoke the callback URL, the task will be polled for completion by Oozie.

The figure below depicts a sample Oozie workflow which has six action nodes (MapReduce jobs, Pig script, Java application and HDFS task) and five control nodes (Start, Decision point, Fork, Join as well as End).

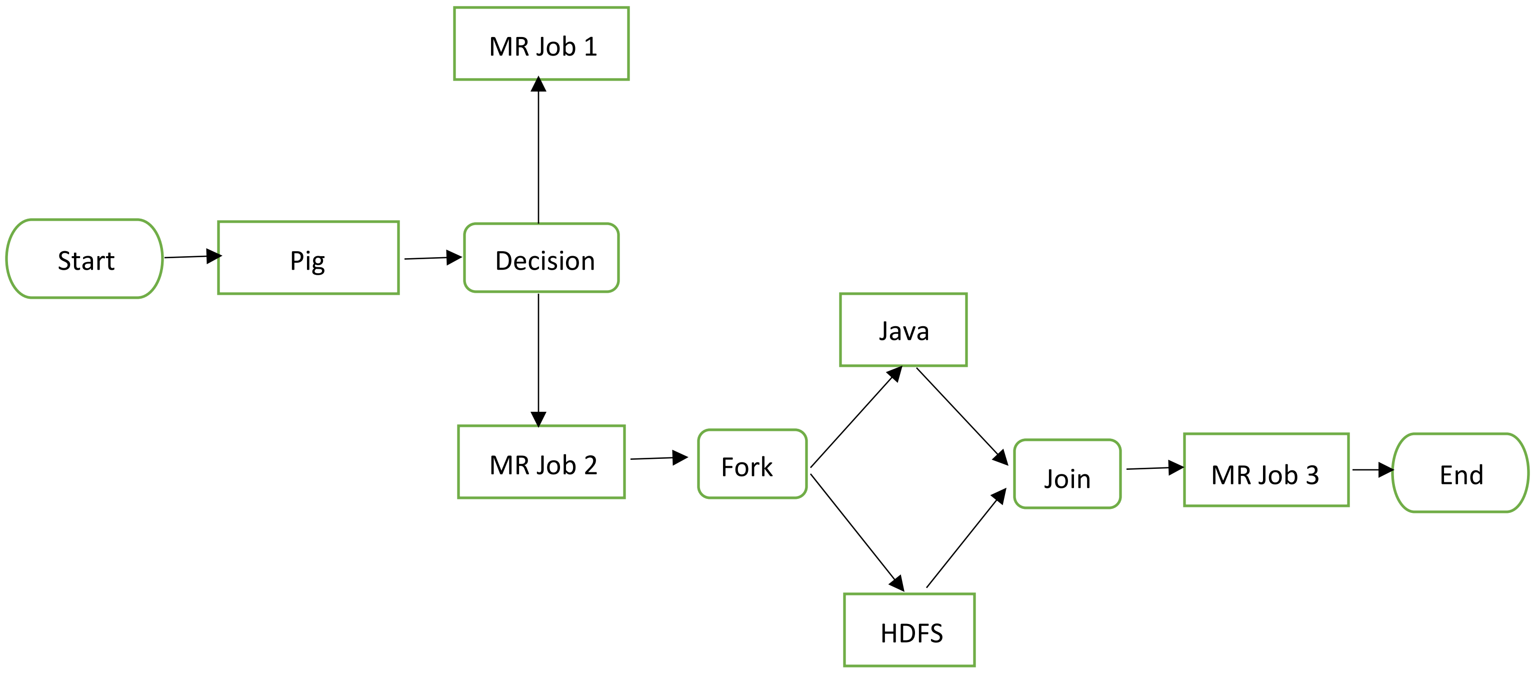

Figure: Sample Oozie workflow

Oozie workflows can be parameterized. We must provide values for parameters while submitting a workflow job. Once we provide relevant parameters, many similar workflow jobs can run concurrently.

Oozie uses XML as workflow definition language. Two files are required to run Oozie workflows:

1. workflow.xml: it is stored in HDFS. It consists of structure of workflow.
2. job.properties: it is stored local. It consists of the configuration properties.

Oozie server: It is designed to work with either MR V1 or YARN. Please note that it cannot work with both simultaneously.

- It can be configured with CATALINA_BASE variable in /etc/oozie/conf/oozie-env.sh

Hadoop 1

- CATALINA_BASE = /usr/lib/oozie/oozie-server-0.20

Hadoop 2

- CATALINA_BASE=/usr/lib/oozie/oozie-server

Oozie workflows are written as an XML file that represents a directed acyclic graph. Let’s have a look at the following example that defines a simple workflow:

```
<workflow-app name="DemoOozie" xmlns="uri:oozie:workflow:0.1">
	<start to="demo-hive"/>

	<action name="demo-hive">
		<hive xmlns="uri:oozie:hive-action:0.2">
			<job-tracker>${jobTracker}</job-tracker>
			<name-node>${nameNode}</name-node>
			<job-xml>${appPath}/hive-site.xml</job-xml>
			<configuration>
				<property>
					<name>oozie.hive.defaults</name>
					<value>${appPath}/hive-site.xml</value>
				</property>
				<property>
					<name>hadoop.proxyuser.oozie.hosts</name>
					<value>*</value>
				</property>
				<property>
					<name>hadoop.proxyuser.oozie.groups</name>
					<value>*</value>
				</property>
			</configuration>
			<script>create_table.hql</script>
		</hive>
		<ok to="end"/>
		<error to="end"/>
	</action>

	<end name="end"/>
</workflow-app>
```

We have named the workflow application as ‘DemoOozie’. It has two control nodes (start and end) and an action node named ‘demo-hive’. We need to provide required details for this action to take place such as path of job tracker, name node and core XML configuration file of requested service within this action. For example, in the above workflow we are providing configuration details of Hive and instructing Oozie to execute a Hive script. Oozie controls the flow of operations in a workflow by using <ok to = > and <error to = > in workflow.xml.

Oozie Coordinator: It is a collection of predicates (conditional statements based on time-frequency and data availability) and actions (i.e. Hadoop Map/Reduce jobs, Hadoop file system, Hadoop Streaming, Pig, Java and Oozie sub-workflow). Actions are recurrent workflow jobs invoked each time predicate returns true.

Here is a sample coordinator XML file:

```
<coordinator-app name="Demo_Scheduler" frequency="10" start="2017-03-01T22:08Z"
end="2018-01-20T22:12Z" timezone="UTC" xmlns="uri:oozie:coordinator:0.1">
  <action>
    <workflow>
      <app-path>${nameNode}/user/${user.name}/workflows</app-path>
    </workflow>
  </action>
</coordinator-app>
```

Benefits of Oozie:

- Oozie is a native Hadoop ecosystem component that has inbuilt Hadoop actions. It makes the development of workflow, troubleshooting and maintenance easier.
- Oozie is proved to be a scalable solution to execute workflows over a large cluster.
- In Oozie, every task will be launched from a separate data node thereby the work load will be balanced between different nodes so that there is no overburden on one specific machine.
- Oozie has been designed to be secure as well. It knows which user has submitted the job and it will launch jobs as that user by getting privileges from system files.
- Oozie monitors the execution of MapReduce tasks. It determines whether a job execution got over or it’s hanging in the middle of the same. It is a unique feature offered by Oozie.
- Oozie callback URL makes it easy for clients to check status of a workflow execution. A client can explore any errors that occur in the execution through the UI.
- Oozie is supported by vendors of Hadoop i.e. Apache organization. They provide support in case you are facing any issues with it.
- The coordinator in Oozie allows for triggering of actions as and when data arrives at HDFS. Hence it works for real time data use cases as well. Scheduling of jobs makes it more useful.

2. The workflow of Sqoop and its Benefits:

Apache Sqoop is a Hadoop tool that is used to import and export data between Hadoop cluster and relational databases such as MySQL, Oracle, etc. Sqoop uses JDBC to interact with those external databases.

With Oozie's Sqoop action facility, users can run Sqoop jobs as part of Oozie workflow. The following elements are part of every workflow:

- job-tracker (mandatory field)
- name-node (mandatory field)
- prepare
- job-xml
- configuration
- command (required only if arg is not used)
- arg (required only if command is not used)
- file
- archive

The action requires to know a couple of important components of Hadoop cluster where Oozie needs to execute the Sqoop action – JobTracker and NameNode. The <prepare> tag is optional and is generally used to remove output directories or partitions of tables in HCatalog or to create directories as the action requires. Up next, the <job-xml> and <configuration> elements are used to get all the properties of Hadoop job configuration.

Oozie also allows users to specify any data files the job require, libraries, scripts and archives using the <file> and <archive> elements. The arguments for a Sqoop job can either sent through <command> in a single line or be split into many through <arg> element.

```
<action name = “sampleSqoopAction”>
        <sqoop>
                <command>import –connect jdbc:hsqldb:file:db.hsqldb –table sample_table
                                --target-dir hdfs://localhost:7078/acadgild/sqoop_tbl –m 1
                </command>
        </sqoop>
</action>
```

Oozie workflow.xml:

```
<workflow-app name="HDFSToOracleWorkflow" xmlns="uri:oozie:workflow:0.4">
        <global>
                <job-tracker>${jobTracker}</job-tracker>
                <name-node>${nameNode}</name-node>
                <configuration>
                        <property>
                                <name>mapred.job.queue.name</name>
                                <value>${queueName}</value>
                        </property>
                </configuration>
        </global>
        <start to="SqoopExportToOracle" />
                <action name=" SqoopExportToOracle">
                        <sqoop xmlns="uri:oozie:sqoop-action:0.2">
                                <job-tracker>${jobTracker}</job-tracker>
                                <name-node>${nameNode}</name-node>
                                <arg>export</arg>
                                <arg>-Dmapred.child.java.opts=-Xmx4096m</arg>
                                <arg>--connect</arg>
                                <arg>${oracle_database}</arg>
                                <arg>--table</arg>
                                <arg>${oracle_table}</arg>
                                <arg>--columns</arg>
                                <arg>${oracle_columns}</arg>
                                <arg>--export-dir</arg>
                                <arg>${RESULTS_FOLDER}</arg>
                                <arg>--input-fields-terminated-by</arg>
                                <arg>,</arg>
                                <arg>--input-lines-terminated-by</arg>
                                <arg>'\n'</arg>
                                <arg>--username</arg>
                                <arg>${oracle_username}</arg>
                                <arg>--password</arg>
                                <arg>${oracle_password}</arg>
                                <arg>--m</arg>
                                <arg>${num_mappers}</arg>
                        </sqoop>
                        <ok to="Email_success" />
                        <error to="Email_failure" />
                </action>
                <action name="Email_success">
                        <email xmlns="uri:oozie:email-action:0.1">
                                <to>${success_emails}</to>
                                <subject>
                                        SUCCESS:update:${wf:id()}
```

```
                    </subject>

                    <body>
                            Hi,
                            Database update is success for workflow ID : ${wf:id()}
                            This is auto-generated email. Please do not reply to this
                            email.
                            Thanks,
                            Lingaraj A J
                    </body>
            </email>
            <ok to="end" />
            <error to="kill" />
        </action>
        <action name="Email_failure">
            <email xmlns="uri:oozie:email-action:0.1">
                    <to>${failure_emails}</to>
                    <subject>
                            FAILURE:Database update: ${wf:id()}
                    </subject>
                    <body>
                            Hi,
                            Database update is failed for workflow ID : ${wf:id()}
                            This is auto-generated email. Please do not reply to this
                            email.
                            Thanks,
                            Lingaraj A J
                    </body>
            </email>
            <ok to="end" />
            <error to="kill" />
        </action>
        <kill name="kill">
            <message>
            Action failed, error message [${wf:errorMessage(wf:lastErrorNode())}]
            </message>
        </kill>
    <end name="end" />
</workflow-app>
```

The above workflow defines a Sqoop job that exports data from HDFS to Oracle database. Initially, the global properties are provided in it such as JobTracker, NameNode and the export operation added to the MapReduce job queue. Inside action part, all the parameters required to connect to Oracle database and to fetch HDFS data files are provided to perform the data transfer operation. An email will be sent post completion of the MapReduce job in both the cases: success as well as failure.

Benefits of Sqoop:

- Sqoop has the ability to perform data transfer operation in parallel. Hence, the job execution will be faster as well as cost effective.
- Sqoop supports wide variety of file formats like SequenceFile, Text and Avro.
- Sqoop is used to transfer data from external structured data stores into Hadoop (which is mainly for unstructured data stores). In this way, Sqoop allows us to combine both kinds of data and can be used to perform various analytical operations.
- Sqoop supports transfer of data from different structured databases such as MySQL, Oracle, PostgreSQL, Teradata, etc.
- It supports import of sequential data from the mainframe. This will not only reduce the usage of mainframe, but also reduces the cost of job execution and the usage of mainframe hardware resource required to perform such operations.
- Apart from providing support for JDBC based connectors for relational database systems, it also offers direct connectors which uses native tools that results in better performance.
- Sqoop provides an extension mechanism which is used to build variety of connectors. Many companies have written and offered their own extensions that are standardized and well supported across Sqoop platform. For example, VoltDB connector is built by VoltDB itself and Oracle connector is built by Quest Software.
- Sqoop allows users to offload the ETL (Extract, Transform and Load) process into an effective, low cost and fast Hadoop processes, since the data gets transferred and stored in Hadoop.