Introduction

Oozie is an extensible, scalable, and data-aware service to orchestrate Hadoop jobs, manage job workflow, coordination, dependencies, and execute jobs based on event triggers such as time and data availability. It is a system which runs workflow of dependent jobs. Here, users are permitted to create Directed Acyclic Graphs of workflows, which can be run in parallel and sequentially in Hadoop. Apache Oozie Workflow Scheduler for Hadoop is a workflow and coordination service for managing Apache Hadoop jobs.

- Oozie Workflow jobs are Directed Acyclical Graphs (DAGs) of actions; actions are typically Hadoop jobs (MapReduce, Streaming, Pipes, Pig, Hive, Sqoop, etc).
- Oozie Coordinator jobs trigger recurrent Workflow jobs based on time (frequency) and data availability.
- Oozie Bundle jobs are sets of Coordinator jobs managed as a single job.

It consists of two parts:

- Workflow engine : Responsibility of a workflow engine is to store and run workflows composed of Hadoop jobs e.g., MapReduce, Pig, Hive.
- Coordinator engine: It runs workflow jobs based on predefined schedules and availability of data.

History

Year 2008 – InYahoo! Bangalore a person called Alejandra noticed that other teams were taking a variety of manual, ad hoc approaches (whether using shell scripts, JobControl, cron etc) to managing multiple Hadoop jobs. Project Pac-Man was created, then renamed as Oozie, a general-purpose workflow system for Hadoop. Oozie is a Burma name for Elephant keeper where as name Mahout was taken for machine learning project already, hence Oozie is named.

Year 2010 - Yahoo! open-sourced Oozie's source code.

Year 2011 - Oozie became an Apache Incubator project.

Year 2012 - Graduating into top-level project in 2012.

Why Oozie

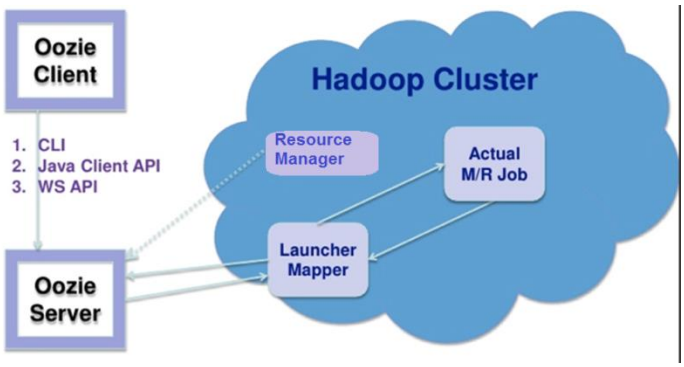
Main purpose of using Oozie is to manage different type of jobs being processed in Hadoop system.

Dependencies between jobs are specified by user in the form of Directed Acyclic Graphs. Oozie is very much flexible, as well. One can easily start, stop, suspend and rerun jobs. Oozie makes it very easy to rerun failed workflows. One can easily understand how difficult it can be to catch up missed or failed jobs due to downtime or

failure. It is even possible to skip a specific failed node. Oozie consumes this information and takes care of their execution in correct order as specified in a workflow. That way user's time to manage complete workflow is saved. In addition, Oozie has a provision to specify frequency of execution of a particular job with the following features.

- Retry Failure
- Timely or data available execution
- Dependency order
- Common framework for communication and execution

Architecture



Multiple ways to interact with Oozie:

• CLI
• Java client
• Web Service Endpoints

The Java client / CLI are just an abstraction for the web service endpoints and it is easy to extend this functionality in your own apps.

How does OOZIE work?

Oozie runs as a service in the cluster and clients submit workflow definitions for immediate or later processing.

Oozie workflow consists of **action nodes** and **control-flow nodes**.

An **action node** represents a workflow task, e.g., moving files into HDFS, running a MapReduce, Pig or Hive jobs, importing data using Sqoop or running a shell script of a program written in Java.

A **control-flow node** controls the workflow execution between actions by allowing constructs like conditional logic wherein different branches may be followed depending on the result of earlier action node.

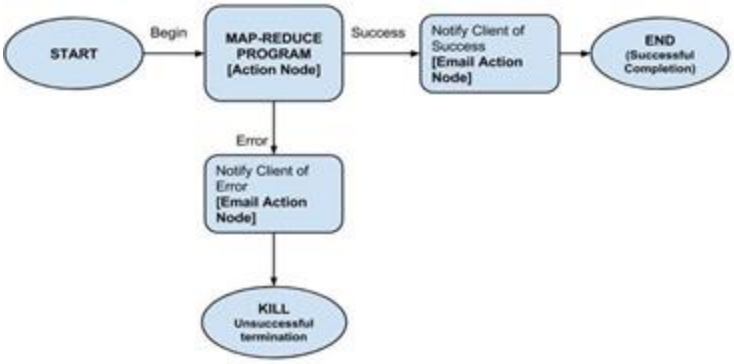
Start Node, End Node and Error Node fall under this category of nodes.

Start Node, designates start of the workflow job.

End Node, signals end of the job.

Error Node, designates an occurrence of error and corresponding error message to be printed.

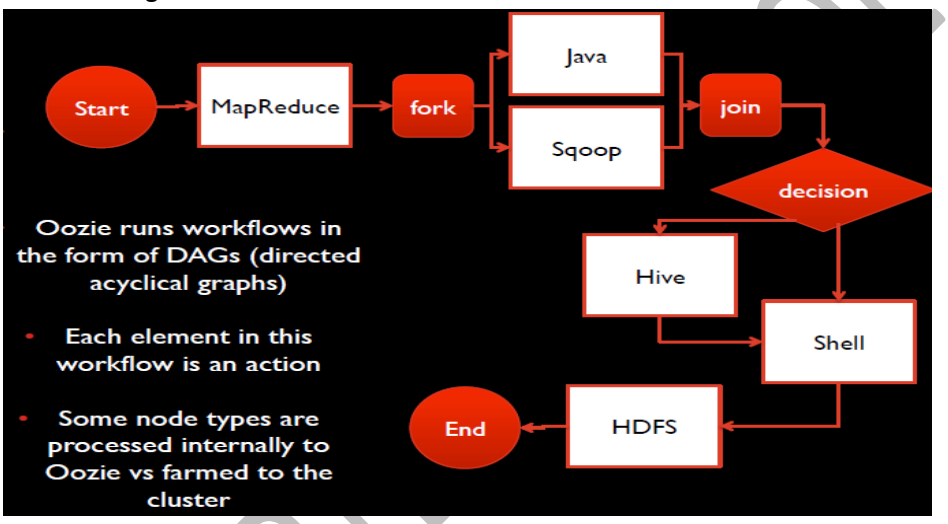
At the end of execution of workflow, HTTP callback is used by Oozie to update client with the workflow status. Entry-to or exit-from an action node may also trigger callback.



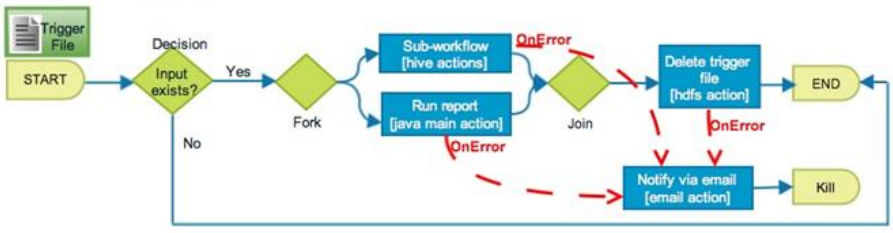
Components

- Actions
- Workflow
- Coordinator
- Bundle

Workflow Engine



Control Nodes



Action Types

Action	Type
MapReduce	Asynchronous
Java	Asynchronous
Pig	Asynchronous
Filesystem	Synchronous
Sub-Workflow	Synchronous
Hive	Asynchronous
DistCp	Asynchronous
Email	Synchronous
Shell	Asynchronous
SSH	Synchronous
Sqoop	Asynchronous

,.

Action Nodes

```
<start to="pig-node"/>
<action name="pig-node">
<pig>

<job-tracker>${jobTracker}</job-tracker>
<name-node>${nameNode}</name-node>

<prepare>
    <delete path="${nameNode}/user/hduser/output-data/pig"/>
</prepare>

<configuration>
<property>
<name>mapred.job.queue.name</name>
<value>${queueName}</value>
</property>
<property>
<name>mapred.compress.map.output</name>
<value>true</value>
</property>
</configuration>

<script>id.pig</script>
<param>INPUT=/user/hduser/input-data/text</param>
<param>OUTPUT=/user/hduser/output-data/pig</param>
</pig>

<ok to="end"/>
<error to="fail"/>
</action>

<kill name="fail">
<message>Pig failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
</kill>
<end name="end"/>
</workflow-app>
```

Properties File

```
nameNode=hdfs://Master:8020
jobTracker=Master:8021
baseDir=${nameNode}/user/${user.name}/WC-Coordinator
oozie.coord.application.path = ${baseDir}/job
oozie.libpath=${oozie.coord.application.path}/lib
inputDir=${baseDir}/input
outputDir=${baseDir}/output
```

```
]$ oozie job -config coordinator.properties -run
```

EL Functions

Oozie, besides allowing the use of workflow job properties to parameterize workflow jobs, it provides a set of build in EL functions that enable a more complex parameterization of workflow action nodes as well as the predicates in decision nodes.

String firstNotNull(String value1, String value2)

It returns the first not null value, or null if both are null .

String concat(String s1, String s2)

It returns the concatenation of 2 strings. A string with null value is considered as an empty string.

String trim(String s)

It returns the trimmed value of the given string. A string with null value is considered as an empty string.

String timestamp()

It returns the UTC current date and time in W3C format down to the second (YYYY-MM-DDThh:mm:ss.sZ). I.e.: 1997-07-16T19:20:30.45Z

Workflow EL Functions

String wf:id()

It returns the workflow job ID for the current workflow job.

String wf:name()

It returns the workflow application name for the current workflow job.

String wf:appPath()

It returns the workflow application path for the current workflow job.

String wf:user()

It returns the user name that started the current workflow job.

String wf:group()

It returns the group/ACL for the current workflow job.

String wf:errorCode(String node)

It returns the error code for the specified action node, or an empty string if the action node has not exited with ERROR state.

Each type of action node must define its complete error code list.

String wf:errorMessage(String message)

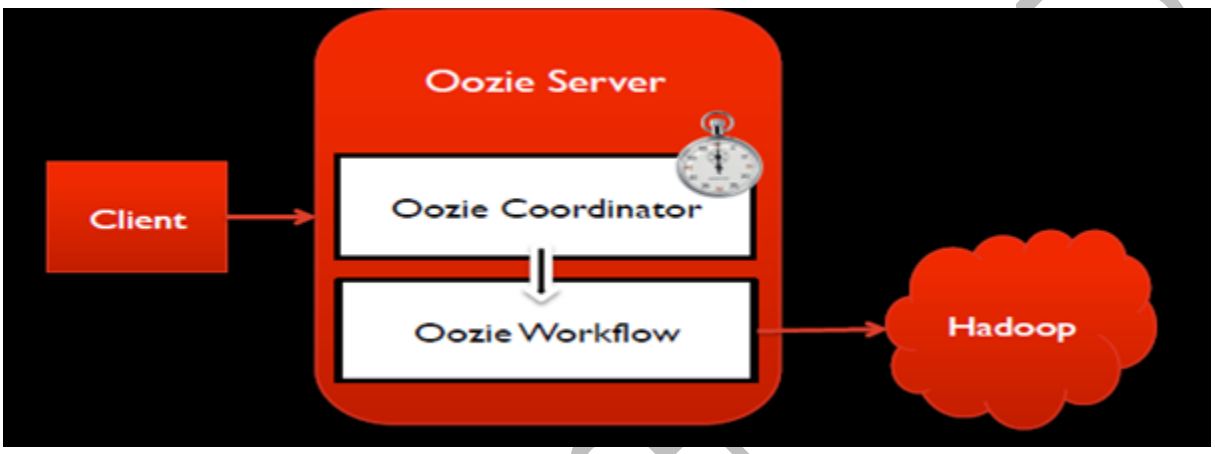
It returns the error message for the specified action node, or an empty string if no action node has not exited with ERROR state.

The error message can be useful for debugging and notification purposes.

int wf:run()

It returns the run number for the current workflow job, normally 0 unless the workflow job is re-run, in which case indicates the current run.

Coordinator



Oozie coordinators can execute workflows based on time and data dependencies.
Each coordinator is specified a workflow to execute upon meeting its trigger criteria.
Coordinators can pass variables to the workflow layer allowing for dynamic resolution.

- Start time
- Frequency
- End time

```
<dataset name="ds_input1" frequency="${coord:hours(6)}"
initial-instance="2014-12-29T02:00Z">
<uri-template>
${baseDataDir}/revenue_feed/${YEAR}-${MONTH}-${DAY}-${HOUR}
</uri-template>
<done-flag>_trigger</done-flag>
</dataset>
```

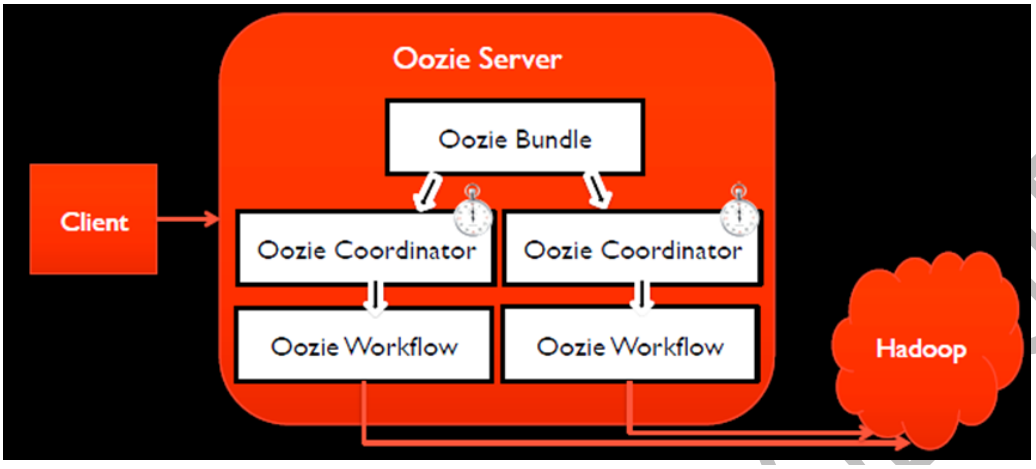
This coordinator will run every hour and invoke the workflow found in the /test_job folder

```
<?xml version="1.0" ?><coordinator-app end="${COORD_END}"
frequency="${coord:hours(1)}" name="test_job_coord" start="${COORD_START}" timezone="UTC" xmlns="uri:oozie:coordinator:0.1">
<action>
<workflow>
<app-path>hdfs://bar:9000/user/hadoop/oozie/app/test_job</
app-path>
</workflow>
</action>
</coordinator-app>
```

Bundle

Bundles are higher level abstractions that will batch a set of coordinators together.

There is no explicit dependency between coordinators within a bundle but it can be used to more formally define a data pipeline.



Command line examples

```
oozie job <OPTIONS> : job operations
      -action <arg>       coordinator rerun on action ids (requires -rerun); coordinator log retrieval on action ids (requires -log)
       -change <arg>      change a coordinator/bundle job
       -config <arg>      job configuration file '.xml' or '.properties'
       -info <arg>        info of a job
       -kill <arg>        kill a job
       -log <arg>         job log
       -run               run a job
       -start <arg>       start a job
       -submit            submit a job
       -suspend <arg>     suspend a job
       -value <arg>       new endtime/concurrency/pausetime value for changing a coordinator job; new pausetime value for changing a
                          bundle job

oozie job -oozie http://localhost:8080/oozie -config job.properties –run
oozie job -oozie http://localhost:8080/oozie -suspend 14-20090525161321-oozie
oozie job -oozie http://localhost:8080/oozie -resume 14-20090525161321-oozie
oozie job -oozie http://localhost:8080/oozie -kill 14-20090525161321-oozie
```

Submitting a Workflow, Coordinator or Bundle Job

$ oozie job -oozie http://localhost:8080/oozie -config job.properties -submit

.

job: 14-20090525161321-oozie-joe

The parameters for the job must be provided in a file, either a Java Properties file (.properties) or a Hadoop XML Configuration file (.xml). This file must be specified with the -configoption.

The workflow application path must be specified in the file with the oozie.wf.application.path property. The coordinator application path must be specified in the file with theoozie.coord.application.path property.The bundle application path must be specified in the file with the oozie.bundle.application.path property. Specified path must be an HDFS path.

The job will be created, but it will not be started, it will be in PREP status.

Starting a Workflow, Coordinator or Bundle Job

$ oozie job -oozie http://localhost:8080/oozie -start 14-20090525161321-oozie-joe

The start option starts a previously submitted workflow job, coordinator job or bundle job that is in PREP status. After the command is executed the workflow job will be in RUNNING status, coordinator job will be in RUNNING status and bundle job will be in RUNNING status.

Running a Workflow, Coordinator or Bundle Job

$ oozie job -oozie http://localhost:8080/oozie -config job.properties -run

.

job: 15-20090525161321-oozie-joe

The workflow application path must be specified in the file with the oozie.wf.application.path property. The coordinator application path must be specified in the file with the oozie.coord.application.path property. The bundle application path must be specified in the file with the oozie.bundle.application.path property.The specified path must be an HDFS path.

The job will be created and it will started, the job will be in RUNNING status.

Suspending a Workflow, Coordinator or Bundle Job

$ oozie job -oozie http://localhost:8080/oozie -suspend 14-20090525161321-oozie-joe

The suspend option suspends a workflow job in RUNNING status. After the command is executed the workflow job will be in SUSPENDED status.

The suspend option suspends a coordinator/bundle job in RUNNING , RUNNIINGWITHERROR or PREP status. When the coordinator job is suspended, running coordinator actions will stay in running and the workflows will be suspended. If the coordinator job is in RUNNING status, it will transit to SUSPENDED status; if it is in RUNNINGWITHERROR status, it will transit toSUSPENDEDWITHERROR ; if it is in PREP status, it will transit to PREPSUSPENDED status.

When the bundle job is suspended, running coordinators will be suspended. If the bundle job is in RUNNING status, it will transit to SUSPENDED status; if it is in RUNNINGWITHERROR status, it will transit to SUSPENDEDWITHERROR ; if it is in PREP status, it will transit to PREPSUSPENDED status.

Resuming a Workflow, Coordinator or Bundle Job

$ oozie job -oozie http://localhost:8080/oozie -resume 14-20090525161321-oozie-joe

The resume option resumes a workflow job in SUSPENDED status.

After the command is executed the workflow job will be in RUNNING status.

The suspend option suspends a coordinator/bundle job in SUSPENDED , SUSPENDEDWITHERROR or PREPSUSPENDED status. If the coordinator job is in SUSPENDED status, it will transit to RUNNINGstatus; if it is in SUSPENDEDWITHERROR status, it will transit to RUNNINGWITHERROR ; if it is in PREPSUSPENDED status, it will transit to PREP status.

When the coordinator job is resumed it will create all the coordinator actions that should have been created during the time it was suspended, actions will not be lost, they will delayed.

When the bundle job is resumed, suspended coordinators will resume running. If the bundle job is in SUSPENDED status, it will transit to RUNNING status; if it is in SUSPENDEDWITHERRORstatus, it will transit to RUNNINGWITHERROR ; if it is in PREPSUSPENDED status, it will transit to PREP status.

Killing a Workflow, Coordinator or Bundle Job

$ oozie job -oozie http://localhost:8080/oozie -kill 14-20090525161321-oozie-joe

The kill option kills a workflow job in PREP , SUSPENDED or RUNNING status and a coordinator/bundle job in =PREP=, RUNNING , PREPSUSPENDED , SUSPENDED , PREPPAUSED , or PAUSED status.
After the command is executed the job will be in KILLED status.

Rerunning a Workflow Job

$ oozie job -oozie http://localhost:8080/oozie -config job.properties -rerun 14-20090525161321-oozie-joe

The rerun option reruns a completed (SUCCCEDED , FAILED or KILLED) job skipping the specified nodes.
The parameters for the job must be provided in a file, either a Java Properties file (.properties) or a Hadoop XML Configuration file (.xml). This file must be specified with the -configoption.
The workflow application path must be specified in the file with the oozie.wf.application.path property. The specified path must be an HDFS path.
The list of nodes to skipped must be provided in the oozie.wf.rerun.skip.nodes property separated by commas.
After the command is executed the job will be in RUNNING status.

Checking the Status of a Workflow, Coordinator or Bundle Job or a Coordinator Action

$ oozie job -oozie http://localhost:8080/oozie -info 14-20090525161321-oozie-joe

Checking the server logs of a Workflow, Coordinator or Bundle Job

$ oozie job -oozie http://localhost:8080/oozie -log 14-20090525161321-oozie-joe

Checking the server logs for particular actions of a Coordinator Job

$ oozie job -log <coord_job_id> [-action 1, 3-4, 7-40] (-action is optional.)