platform
By Per Scholas

Oozie Fundamentals

Oozie is the tool in which all sort of programs can be pipelined in a desired order to work in Hadoop's distributed environment

Week 11 - Day 3

Overview

- What is Oozie?
- How Oozie works?
- Oozie Workflow
- Oozie Property File
- Oozie Coordinator
- Oozie Bundle

What is Oozie?

- Apache Oozie is a scheduler system to run and **manage Hadoop jobs** in a distributed environment. It allows to combine multiple complex jobs to be run in a sequential order to achieve a bigger task. Within a sequence of task, two or more jobs can also be programmed to run parallel to each other.

- One of the main advantages of Oozie is that it is tightly integrated with Hadoop stack supporting various Hadoop jobs like **Hive, Pig, Sqoop** as well as system-specific jobs like **Java and Shell**.

- Oozie is an **Open Source Java Web-Application** available under Apache license 2.0. It is responsible for triggering the workflow actions, which in turn uses the Hadoop execution engine to actually execute the task. Hence, Oozie is able to leverage the existing Hadoop machinery for load balancing, fail-over, etc.

How Oozie Works?

- An Oozie Workflow is a collection of actions arranged in a Directed Acyclic Graph (DAG). Control nodes define job chronology, setting rules for beginning and ending a workflow. In this way, Oozie controls the workflow execution path with decision, fork and join nodes. Action nodes trigger the execution of tasks.

- Oozie triggers workflow actions, but Hadoop MapReduce executes them. This allows Oozie to leverage other capabilities within the Hadoop stack to balance loads and handle failures.

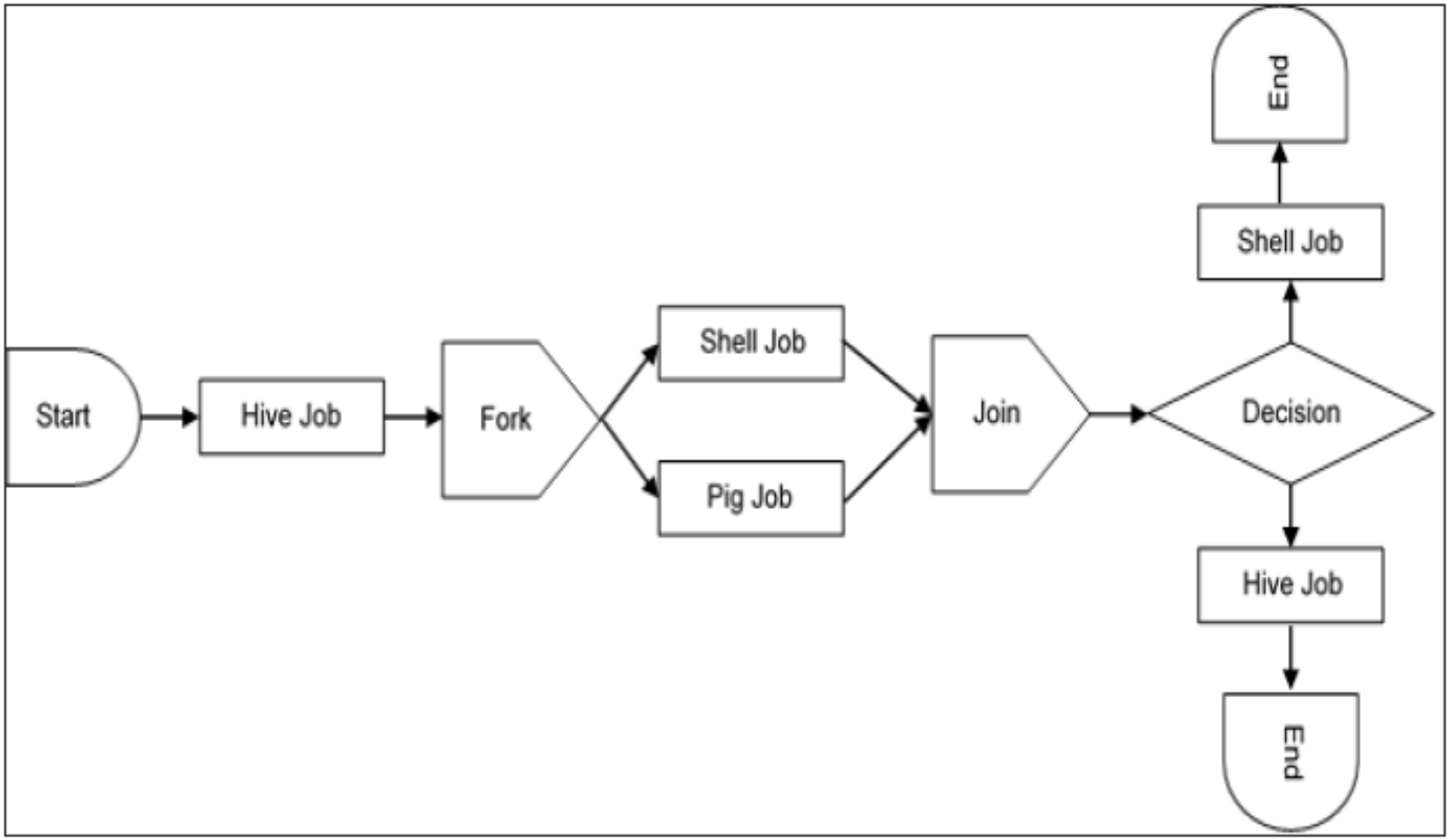
- Oozie detects completion of tasks through callback and polling. When Oozie starts a task, it provides a unique callback HTTP URL to the task, thereby notifying that URL when it's complete. If the task fails to invoke the callback URL, Oozie can poll the task for completion.

How Oozie Works? --- Continued

- Often it is necessary to run Oozie workflows on regular time intervals, but in coordination with unpredictable levels of data availability or events. In these circumstances, Oozie Coordinator allows you to model workflow execution triggers in the form of the data, time or event predicates. The workflow job is started after those predicates are satisfied.
- 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".

- Following three types of jobs are common in Oozie:
  - **Oozie Workflow Jobs** — These are represented as Directed Acyclic Graphs (DAGs) to specify a sequence of actions to be executed.
  - **Oozie Coordinator Jobs** — These consist of workflow jobs triggered by time and data availability.
  - **Oozie Bundle** — These can be referred to as a package of multiple coordinator and workflow jobs.

How Oozie Works? --- Sample Workflow



Oozie Workflow

- Workflow in Oozie is a sequence of actions arranged in a control dependency **DAG (Direct Acyclic Graph)**. The actions are in controlled dependency as the next action can only run as per the output of current action. Subsequent actions are dependent on its previous action. A workflow action can be a **Hive action**, **Pig action**, **Java action**, **Shell action**, etc. There can be decision trees to decide how and on which condition a job should run.
- A fork is used to run multiple jobs in parallel. Oozie workflows can be parameterized (variables like **${nameNode}** can be passed within the workflow definition). These parameters come from a configuration file called as property file.

Oozie Workflow --- Continued

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

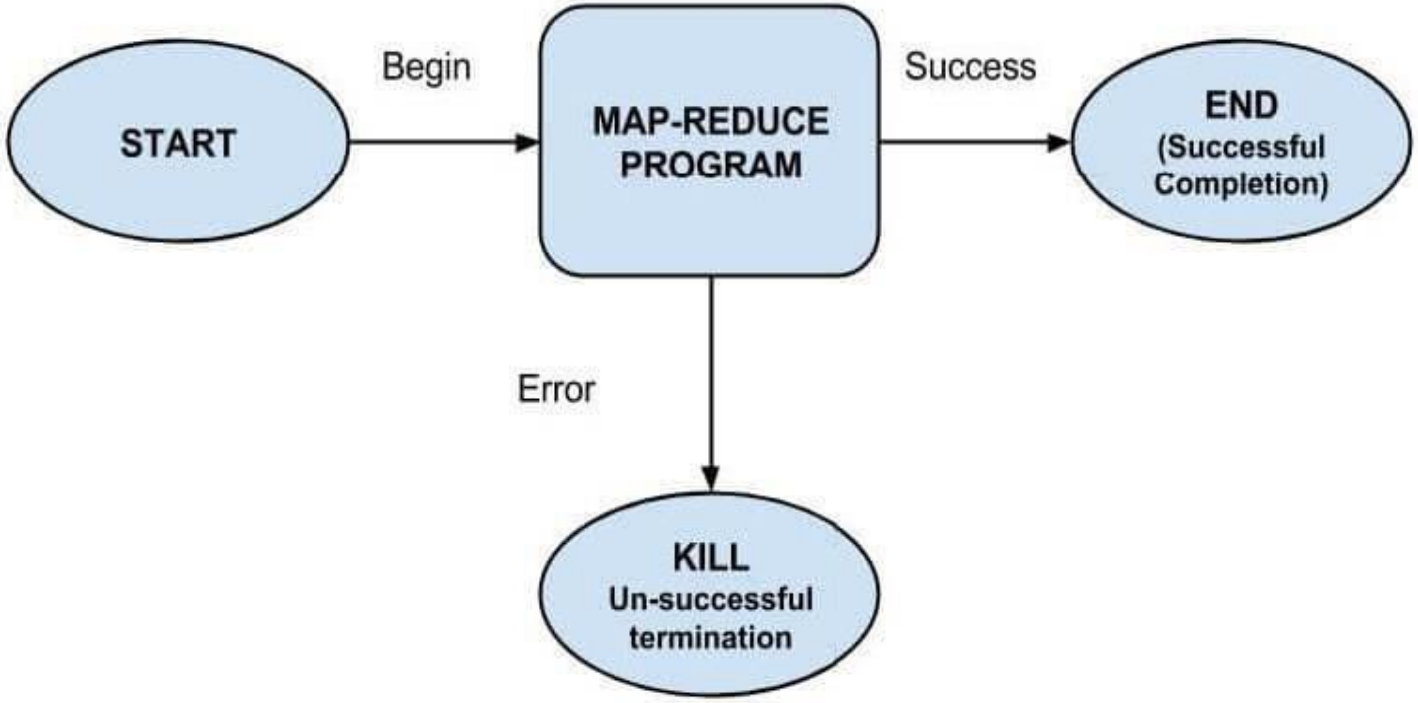
- **Action Nodes** are the mechanisms which triggers the execution of a computation/processing task. Oozie provides support for different types of Hadoop actions out of the box - Hadoop MapReduce, Hadoop file system, Pig etc. In addition Oozie also provides support for system specific jobs - SSH, HTTP, email etc.
- 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.

Oozie Workflow --- Continued

- **Control Flow Nodes** - Control flow nodes are the mechanisms that define the beginning and end of the workflow (start, end, fail). In addition, control flow nodes also provide mechanism to control the execution path of the workflow (decision, fork and join)
- 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.

Oozie Workflow --- Continued

Example Workflow Diagram



An Apache Oozie Workflow job can have the following states - PREP , RUNNING , SUSPENDED , SUCCEEDED , KILLED and FAILED.

Oozie Property File

- Oozie workflows can be parameterized. The parameters come from a configuration file called as property file. We can run multiple jobs using same workflow by using multiple **.property** files (one property for each job).

- Suppose we want to change the **jobtracker url** or change the script name or value of a param.
- We can specify a **config file (.property)** and pass it while running the workflow.
- Variables like **${nameNode}** can be passed within the workflow definition. The value of this variable will be replaced at the run time with the value defined in the '.properties' file.

- A single property file can have more parameters than required in a single workflow and no error will be thrown. This makes it possible to run more than one workflow by using the same properties file. But if the property file does not have a parameter required by a workflow then an error will occur.

Oozie Coordinator

- Coordinator applications allow users to schedule complex workflows, including workflows that are scheduled regularly. Oozie Coordinator models the workflow execution triggers in the form of time, data or event predicates. The workflow job mentioned inside the **Coordinator** is started only after the given conditions are satisfied.

- The following parameters have to be given for the coordinator to work:
  - **start:** It means the start datetime for the job. Starting at this time the actions will be materialized.
  - **end:** The end datetime for the job. When actions will stop being materialized.
  - **timezone:** The timezone of the coordinator application.
  - **frequency:** The frequency, in minutes, to materialize actions.

Oozie Coordinator --- Continued

- There are optional Control options for coordinator:
  - **timeout:** The maximum time, in minutes, that a materialized action will be waiting for the additional conditions to be satisfied before being discarded. A timeout of 0 indicates that at the time of materialization all the other conditions must be satisfied, else the action will be discarded. A timeout of 0 indicates that if all the input events are not satisfied at the time of action materialization, the action should timeout immediately. A timeout of -1 indicates no timeout, the materialized action will wait forever for the other conditions to be satisfied. The default value is -1.
  - **concurrency:** The maximum number of actions for this job that can be running at the same time. This value allows to materialize and submit multiple instances of the coordinator app, and allows operations to catchup on delayed processing. The default value is 1.
  - **execution:** Specifies the execution order if multiple instances of the coordinator job have satisfied their execution criteria. Valid values are:
    - FIFO (oldest first) default.
    - LIFO (newest first).
    - LAST_ONLY (discards all older materializations).

Configuring Frequency

- Coordinator Cron Scheduling: Various Cron syntaxes exist
  - Oozie Cron has 5 fields since oozie operates on per minute base.
  - Complicated Overflowing ranges are discouraged to use

Field Name	Allowed Values	Allowed Special Chars
Minute	0-59	, - * /
Hour	0-23	, - * /
Day-of-Month	1-31	, - * ? / L W
Month	1-12 or JAN-DEC	, - * /
Day-of-Week	1-7 or SUN-SAT	, - * ? / L #

Configuring Frequency --- Continued

- The Allowed Values for each field are fairly self-explanatory (but note that while in many Cron implementations, Day-of-Week accepts 0-6, here we accept 1-7, instead)

- Allowed Special Characters are allowed in all fields:
  - "*" (asterisk), which matches all values
  - "," (comma), which lets you specify multiple values
  - "-" (dash), which lets you specify ranges
  - "/", which lets you specify increments.

- Oozie's processing time zone is UTC, so if you are in a different time zone, you'll have to add/subtract the appropriate offset from the time.

Oozie Bundle

- The **Oozie Bundle system** allows the user to define and execute a bunch of coordinator applications often called a data pipeline. There is no explicit dependency among the coordinator applications in a bundle. However, a user could use the data dependency of coordinator applications to create an implicit data application pipeline.
- The user will be able to start/stop/suspend/resume/rerun in the bundle level resulting in a better and easy operational control.

Summary Review

- What is Oozie?
  - Why use Oozie?
- How Oozie works?
- Oozie Workflow
  - Action nodes for Hadoop Jobs
- Oozie Property File
- Oozie Coordinator
  - Configuring Frequency
- Oozie Bundle

Reference

- https://www.tutorialspoint.com/apache_oozie/index.htm
- https://oozie.apache.org/docs/3.3.1/WorkflowFunctionalSpec.html
- https://blog.cloudera.com/blog/2014/04/how-to-use-cron-like-scheduling-in-apache-oozie/

Questions?

End of Module