Oozie

WL

Topics

Oozie Workflow

Oozie Coordinator

What is Apache 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.

What is Apache Oozie?

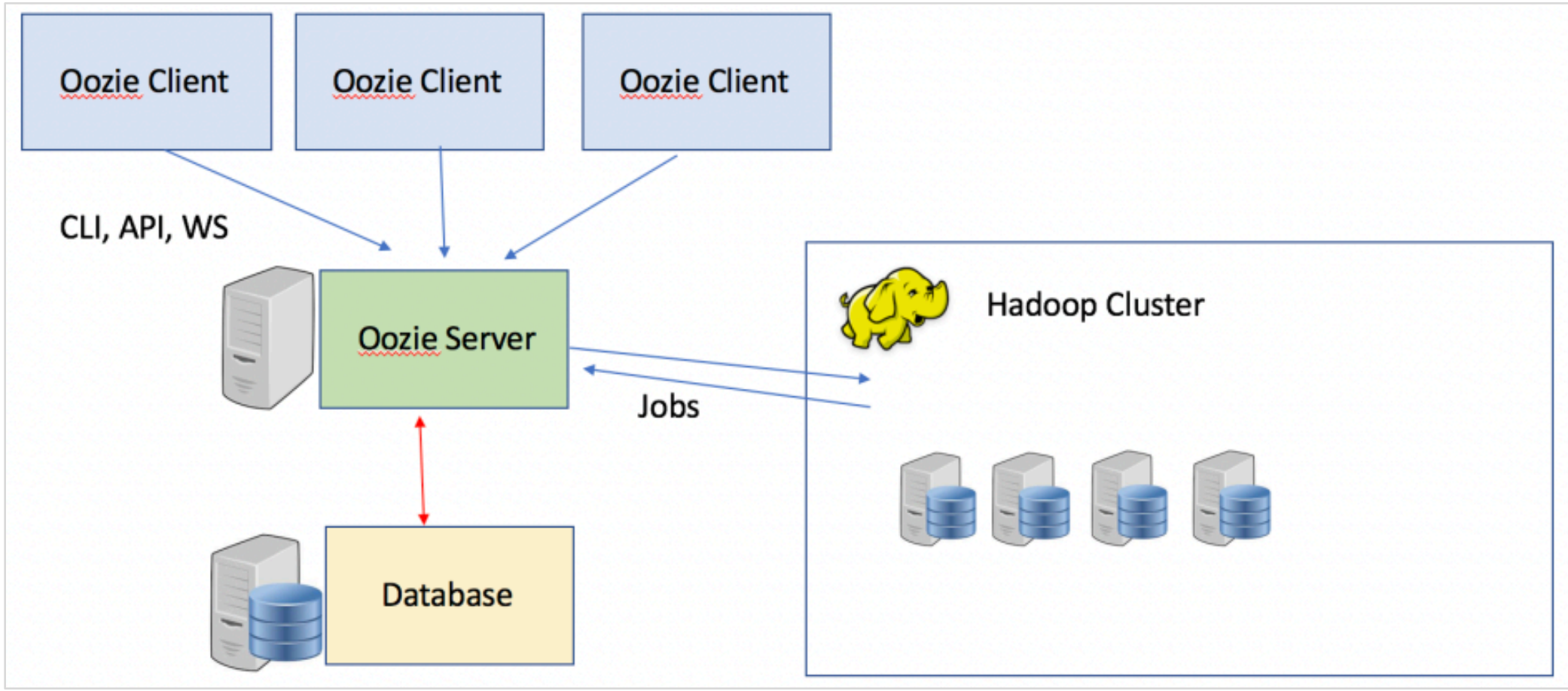
- Oozie is an Open Source Java Web-Application available under Apache license 2.0. It is responsible for triggering the workflow actions, which in turn uses the Hadoop execution engine to actually execute the task. Hence, Oozie is able to leverage the existing Hadoop machinery for load balancing, fail-over,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

the shortcomings of Cron

- A large number of crontab tasks need to be managed
- The task did not execute on time, failed for various reasons, and needs to be retried
- In a multi-server environment, crontab is spread across many clusters

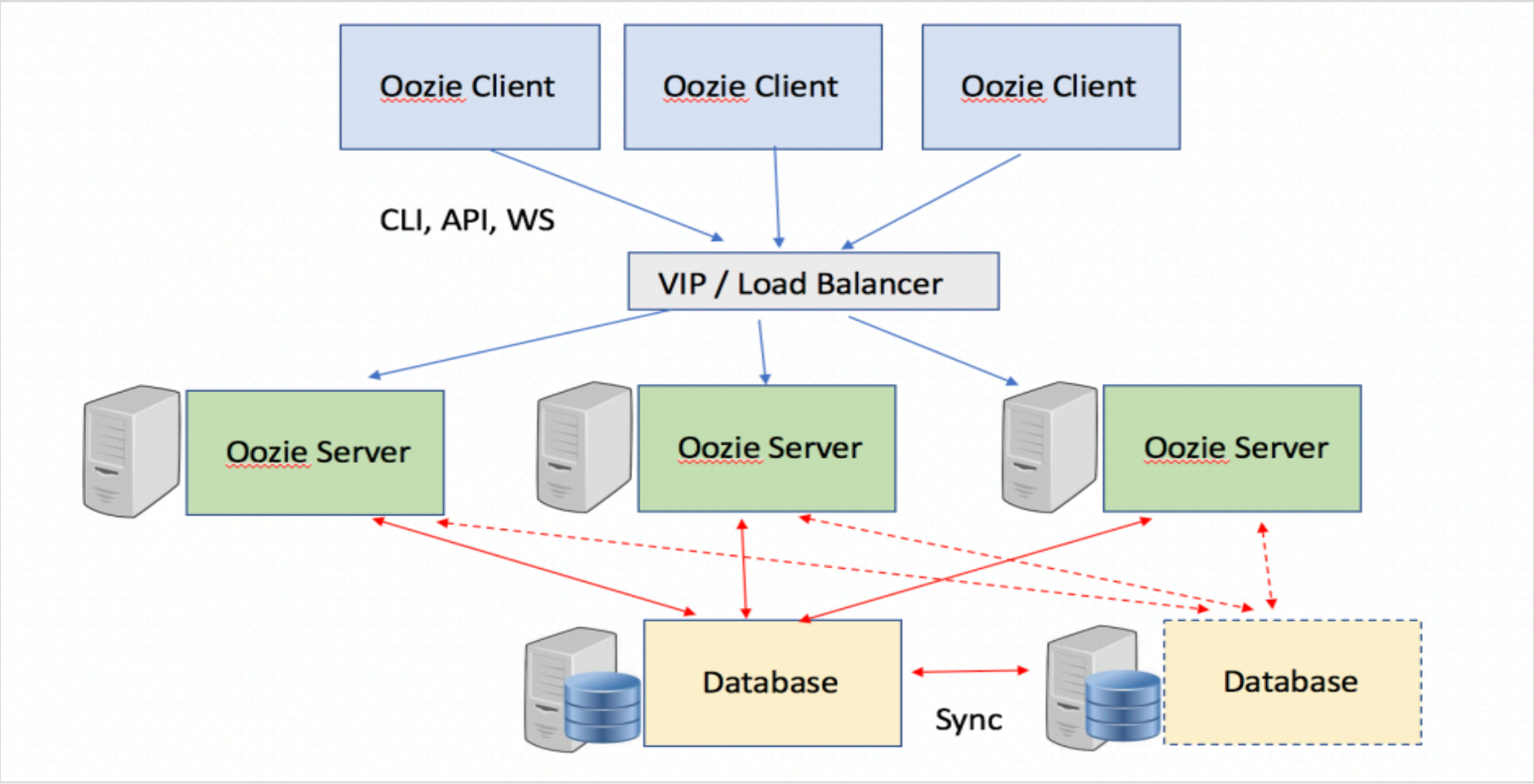
The architecture

ATHEMASTER



The architecture in HA

ATHEMASTER



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.

the application.path

- When we start the Oozie job, we must specify the corresponding application path to the Oozie server. Different job types need to be defined with different application types.
  - oozie.wf.application.path (Path to a workflow application directory/file)
  - oozie.coord.application.path (Path to a coordinator application directory/file)
  - oozie.bundle.application.path (Path to a bundle application directory/file)

ATHEMASTER

Workflow

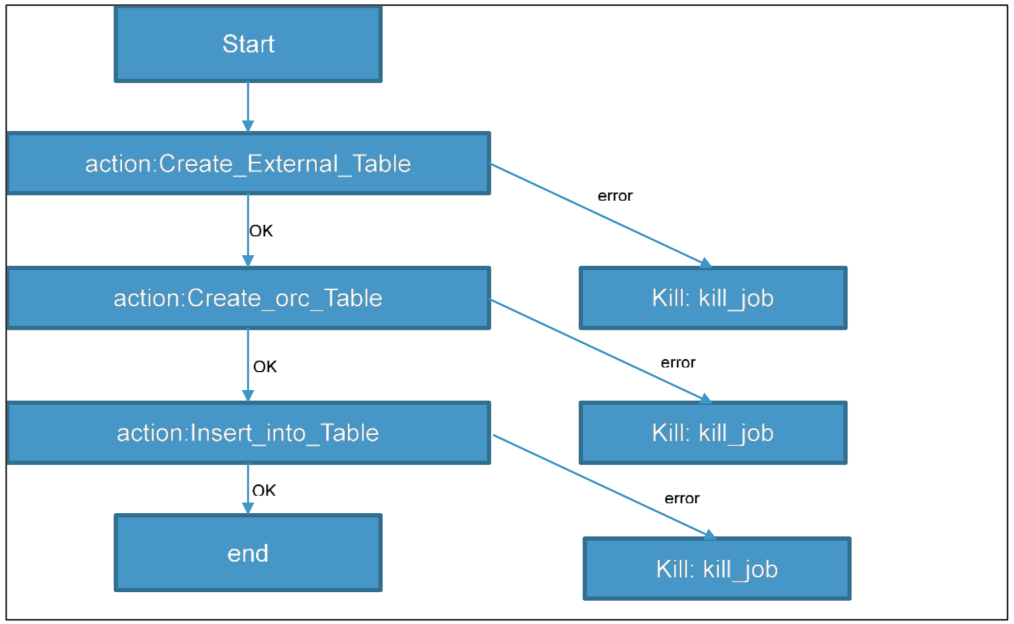
workflow

- An Oozie workflow is a multistage Hadoop job. A workflow is a collection of action and control nodes that captures control dependency where each action typically is a Hadoop job (e.g., a MapReduce, Pig, Hive, Sqoop, or Hadoop DistCp job). There can also be actions that are not Hadoop jobs (e.g., a Java application, a shell script, or an email notification).

workflow

ATHEMASTER

- 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.



action node & controller node

- Action nodes define the jobs, which are the individual units of work that are chained together to make up the Oozie workflow. Actions do the actual processing in the workflow. An action node can run a variety of jobs: ***MapReduce***, ***Pig***, ***Hive***, ***shell***, ***e-mail***, ***java*** and more.
- Workflow control nodes define the start and end of a workflow and they define any control changes in the execution flow. All nodes except for the <start> node have a name attribute. Node names must be a valid Java identifier with a maximum length of 40 characters. Node names can also use dashes.

hive action


```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="simple-Workflow">
   <start to="Create_External_Table" />
   <action name="Create_External_Table">
      <hive xmlns="uri:oozie:hive-action:0.4">
         <job-tracker>master:8032</job-tracker>
         <name-node>hdfs://master:8020</name-node>
         <job-xml>hive-site.xml</job-xml>
         <script>/user/root/oozie_example/external.hive</script>
      </hive>
      <ok to=”end" />
      <error to="kill_job" />
   </action>
   <kill name="kill_job">
      <message>Job failed</message>
   </kill>
   <end name="end" />
</workflow-app>
```

how to run the oozie job

- After successfully completing the workflow.xml setup, the first thing need to do is to place all defined files into CDH cluster (HDFS), including the defined workflow.xml, orc.hive, and external.hive file used in this example.
  - hdfs dfs -put oozie_example /user/root/
- Then use the utility provided by Oozie, such as api package, WS or oozie cli tools, we use this to operate in this example, and so on, to start our defined oozie job
  - oozie job –D oozie.wf.application.path=hdfs://master:8020/user/root/oozie_example/hive-workflow.xml

the job id

- If everything goes well, you will get Oozie job id returned by Oozie server
  - job: 000018-190703190900079-oozie-oozi-W

```
[root@worker3 oozie_example]# oozie job -D oozie.wf.application.path=hdfs://master:8020/user/roo
job: 0000018-190703190900079-oozie-oozi-W
```

- You can use it to get the status of the job execution, and when there is a problem with this job, we also use it to find further error messages.

the job id

- we use job info to get further execution information
  - oozie job -info 000018-190703190900079-oozie-oozi-W

```
[root@worker3 oozie_example]# oozie job -info 0000018-190703190900079-oozie-oozi-W
Job ID : 0000018-190703190900079-oozie-oozi-W
------------------------------------------------------------------------------------------------------------------------------------
Workflow Name : simple-Workflow
App Path      : hdfs://master:8020/user/root/oozie_example/hive-workflow.xml
Status        : SUCCEEDED
Run           : 0
User          : root
Group         : -
Created       : 2019-07-03 13:24 GMT
Started       : 2019-07-03 13:24 GMT
Last Modified : 2019-07-03 13:24 GMT
Ended         : 2019-07-03 13:24 GMT
CoordAction ID: -

Actions
------------------------------------------------------------------------------------------------------------------------------------
ID                                                                            Status    Ext ID                 Ext Status Err Code
------------------------------------------------------------------------------------------------------------------------------------
0000018-190703190900079-oozie-oozi-W@:start:                                  OK        -                      OK         -
------------------------------------------------------------------------------------------------------------------------------------
0000018-190703190900079-oozie-oozi-W@Create_External_Table                    OK        job_1561712280341_0046 SUCCEEDED  -
------------------------------------------------------------------------------------------------------------------------------------
0000018-190703190900079-oozie-oozi-W@Create_orc_Table                         OK        job_1561712280341_0047 SUCCEEDED  -
------------------------------------------------------------------------------------------------------------------------------------
0000018-190703190900079-oozie-oozi-W@end                                      OK        -                      OK         -
------------------------------------------------------------------------------------------------------------------------------------
```

shell action


```
<workflow-app name="ssh-oozie-application"   xmlns="uri:oozie:workflow:0.3">
   <start to="exec-shell-script"  />
   <action name="exec-shell-script">
      <shell xmlns="uri:oozie:shell-action:0.2">
         <job-tracker>master:8032</job-tracker>
         <name-node>hdfs://master:8020</name-node>
         <exec>echo.sh</exec>
         <argument>${wf:id()}</argument>
         <argument>${private_message}</argument>
         <file>hdfs://master:8020/user/root/oozie_example/echo.sh</file>
      </shell>
      <ok to="end" />
      <error to="kill_job" />
   </action>
   <kill name="kill_job">...</kill>
   <end name="end" />
</workflow-app>
```

the job.properties file

- this time, we use configure file to pass our need. Oozie workflows are typically invoked with the following command to set the values for the EL variables. Then we can use these variables by using EL variable or EL function.
  - oozie job -config ~/shell.properties –run
- The *job.properties* file must be stored on the local filesystem, not on HDFS

Sample job.properties file

- oozie.wf.application.path=hdfs://master:8020/user/root/oozie_example/shell-workflow.xml
- private_message=this is a messgae passed from shell.properties

e-mail action


```
<workflow-app name="email-workflow" xmlns="uri:oozie:workflow:0.1">
    <start to="an-email" />
    <action name="an-email">
        <email xmlns="uri:oozie:email-action:0.1">
            <to>c*******@gmail.com</to>
            <cc>c*******@yahoo.com.tw</cc>
            <subject>Email notifications for ${wf:id()}</subject>
            <body>The wf ${wf:id()} successfully completed.</body>
        </email>
        <ok to="end"/>
        <error to="kill_job"/>
    </action>
    <kill name="kill_job">
        <message>Job failed</message>
    </kill>
    <end name="end" />
</workflow-app>
```

Control Nodes

- <start> and <end>
- <fork> and <join>
- <decision>
- <kill>
- <OK> and <ERROR>

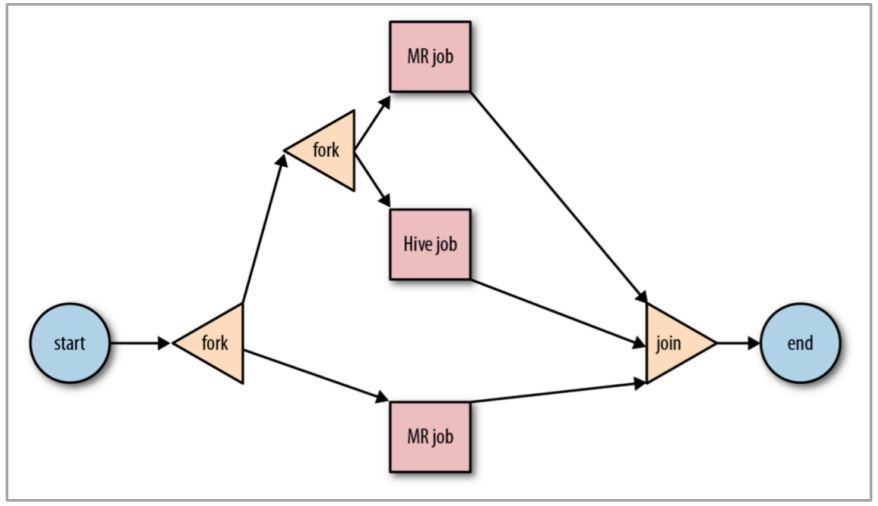
<fork> and <join>


```
<workflow-app xmlns="uri:oozie:workflow:0.5" name="forkJoinNodeWF">
    <start to="forkActions"/>
    <fork name="forkActions">
        <path name="act1"/>
        <path name="act2"/>
    </fork>
    <action name="act1">
        <shell xmlns="uri:oozie:shell-action:0.2">...</shell>
        <ok to="joinActions"/>
        <error to="joinActions"/>
    </action>
    <action name="act2">
        <shell xmlns="uri:oozie:shell-action:0.2">...</shell>
        <ok to="joinActions"/>
        <error to="joinActions"/>
    </action>
    <join name="joinActions" to="done"/>
    <end name="done"/>
</workflow-app>
```

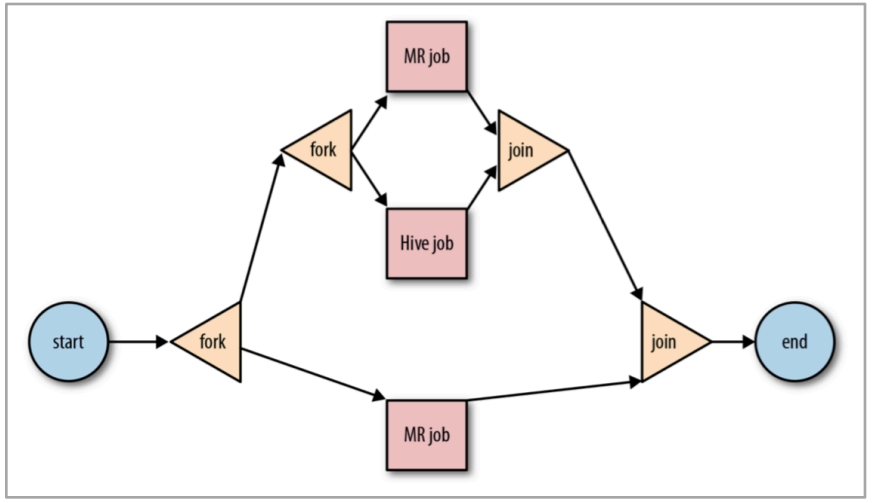
nested <fork> and <join> nodes

ATHEMASTER

- It is possible to have nested <fork> and <join> nodes. The only constraint is that <fork> and <join> nodes always go in pairs and all execution paths starting from a given <fork> must end in the same <join> node



invalid nesting of <fork> and <join> nodes

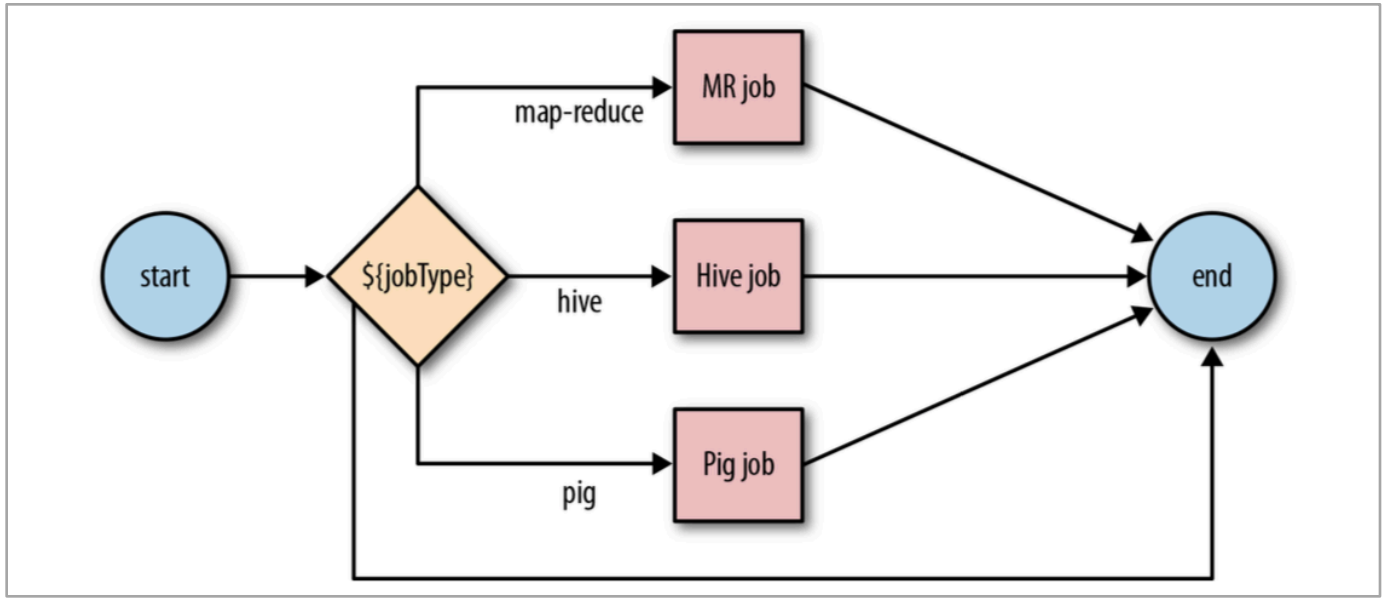


valid nesting of <fork> and <join> nodes

decision

ATHEMASTER

- A <decision> node behavior is best described as an if-then-else-if-then-else... sequence, where the first predicate that resolves to true will determine the execution path.



decision example


```
<workflow-app xmlns="uri:oozie:workflow:0.5"  name="decisionNodeWF">
    <start to="decision"/>
    <decision name="decision">
        <switch>
            <case to="mapReduce">${jobType  eq "mapReduce"}</case>
            <case to="hive">${jobType  eq "hive"}</case>
            <case to="pig">${jobType  eq "pig"}</case>
            <default to="mapReduce"/>
        </switch>
    </decision>
    <action name="mapReduce">...</action>
    <action name="hive">...</action>
    <action name="pig">...</action>
    <end name="done"/>
</workflow-app>
```

<kill>, <OK> and <ERROR>

- <kill>
  - The <kill> node allows a workflow to kill itself. If any execution path of a workflow reaches a <kill> node, Oozie will terminate the workflow immediately, failing all running actions and setting the completion status of the workflow to KILLED
- <OK> and <ERROR>
  - When an action completes, its status is typically in either OK or ERROR status depending on whether or not the execution was successful. If an action ends in OK status, the workflow execution path transitions to the node specified in the <ok> element. If the action ends in ERROR status, the workflow execution path transitions to the node specified in the <error> element

ATHEMASTER

Coordinator

Coordinator

- Oozie supports another abstraction called the coordinator that schedules and executes the workflow based on triggers.
- The Oozie coordinator supports two of the most common triggering mechanisms, namely time and data availability
  - Time Trigger
  - Data Availability Trigger

Time Trigger

- Time-based triggers are easy to explain and resembles the Unix cron utility. In a time- aware coordinator, a workflow is executed at fixed intervals or frequency. A user typically specifies a time trigger in the coordinator using three attributes:
- *Start time (ST)*
  - Determines when to execute the first instance of the workflow
- *Frequency (F)*
  - Specifies the interval for the subsequent executions
- *End time (ET)*
  - Bounds the last execution start time (i.e., no new execution is permitted on or after this time)

coordinator.xml


```
<coordinator-app xmlns="uri:oozie:coordinator:0.2"  name="coord_copydata_from_external_orc" frequency="0/5 * * * *"
start="2019-06-30T00:00Z"  end="2025-12-31T00:00Z""  timezone="Asai/Taipei">
   <controls>
      <timeout>1</timeout>
      <concurrency>1</concurrency>
      <execution>FIFO</execution>
      <throttle>1</throttle>
   </controls>
   <action>
      <workflow>
         <app-path>oozie_example/email-workflow.xml</app-path>
      </workflow>
   </action>
</coordinator-app>
```

frequency faster than 5 minutes ?

- oozie.service.coord.check.maximum.frequency:
  - default value: true
  - When true, Oozie will reject any coordinators with a frequency faster than 5 minutes. It is not recommended to disable this check or submit coordinators with frequencies faster than 5 minutes: doing so can cause unintended behavior and additional system stress.

Coordinator Job Status

- At any time, a coordinator job is in one of the following statuses: *PREP, RUNNING, PREPSUSPENDED, SUSPENDED, PREPPAUSED, PAUSED, SUCCEEDED, DONWITHERROR, KILLED, FAILED*.

Coordinator Job Status

- **PREP** → PREPSUSPENDED | PREPPAUSED | RUNNING | KILLED
- **RUNNING** → SUSPENDED | PAUSED | SUCCEEDED | DONWITHERROR | KILLED | FAILED
- **PREPSUSPENDED** → PREP | KILLED
- **SUSPENDED** → RUNNING | KILLED
- **PREPPAUSED** → PREP | KILLED
- **PAUSED** → SUSPENDED | RUNNING | KILLED

The execution policies

- Timeout
  - A coordinator job can specify the timeout for its coordinator actions, this is, how long the coordinator action will be in WAITING or READY status before giving up on its execution.
- Concurrency
  - A coordinator job can specify the concurrency for its coordinator actions, this is, how many coordinator actions are allowed to run concurrently (RUNNING status) before the coordinator engine starts throttling them.

The execution policies

- Execution order:
  - A coordinator job can specify the execution strategy of its coordinator actions when there is backlog of coordinator actions in the coordinator engine. The different execution strategies are 'oldest first', 'newest first', 'none' and 'last one only'. A backlog normally happens because of delayed input data, concurrency control or because manual re-runs of coordinator jobs.
- Throttle:
  - A coordinator job can specify the materialization or creation throttle value for its coordinator actions, this is, how many maximum coordinator actions are allowed to be in WAITING state concurrently.

Oozie CLI

The retry mechanism

- Sometimes, during the execution process, the operation will fail due to various uncertain factors. In most cases, we will need to login the system to further understand the cause of the error, but there are also some cases where the error is temporarily. In these case, we will need to have an automatic re-execution mechanism to help us solve such problems. We can use retry-max and retry-interval attributes.

```
<workflow-app xmlns="uri:oozie:workflow:0.5"  name="wf-name">
   <action name="a" retry-max="2" retry-interval="1"> .... </action>
</workflow-app>
```

Oozie CLI

- Oozie provides a command line utility to perform job and admin tasks. All operations are done via sub-commands of the oozie CLI. The oozie CLI interacts with Oozie via its WS API.

Oozie job

- oozie job -info <coord_id>
- oozie job -suspend <coord_id>
- oozie job -resume <coord_id>
- oozie job -kill <coord_id>
- oozie jobs -filter <arg>
- oozie jobs -jobtype <arg>

Oozie admin

- -servers list available Oozie servers (more than one only if HA is enabled)
- -status show the current system status
- -osenv show Oozie system OS environment
- -version show Oozie server build version, is equal to command oozie version

ATHEMASTER

Q & A

ATHEMASTER

THANK YOU

- THE PATHFINDER TO OPENSOURCE -

www.athemaster.com
info@athemaster.com
FB:athemaster