Ways to schedule Hadoop tasks

Background

As every project has business requirements which will undergo refinement process and eventually scope is locked then finally developed and deployed. The deployment tasks need coordination between deliverables along with their dependency. These tasks will be scheduled to continuously deliver the business needs. For the purpose job scheduler orchestration tools are needed. The selection can be very basic to more advanced like cron scheduler, Oozie, GUI based Zena. The selection of these tools can be varied upon the usage and can be discussed in the below approaches before a tool can be identified for final use. In Hadoop world the tasks are achieved by invoking commands hdfs/sqoop/pig/hive/beeline/spark-submit etc. The jobs need to be scheduled to run at fixed/varied intervals or a particular date/day/week/month/year etc.

Cron Approach

As Hadoop installations majorly deployed on Unix systems, they come with cron schedulers. These cron schedulers mainly accessible for development on Edge Nodes. Usually source systems drop the files onto Edge Nodes which can be uploaded on to Hadoop DataLake. As hadoop ecosystem have client utilities they can be launched from Edge Node and these activities can be Pig/Hive/Sqoop/Spark/Distcp. Using cron shell scripts can be launched with some frequency like once per day/hour/min or particular time/week/mon/yr/day. As developer can start developing shell script to configure in cron setup, there is situation where developers can write shell scripts in their own standards. This becomes challenging when it comes to debugging or enhance the running scripts as there is no particular standard followed. Overall it provides the flexibility without need of additional tools but lacks the hadoop implementation standards.

The usage of crontab schedule.

```
10 * * * * /home/mysuer/hive-run.sh
10 – 10th minute
*  -- every hour
*  -- every day
*  -- every month
*  -- every day of the week
```

Zena Approach

A multi-platform, workload automation tool to abstract business processes into application rules. This tool supports event-based and interval-based schedules. It provides exception-based management, highly scalable architecture, high availability and role-based security. The component-based architecture minimizes redundant definitions to increase the flexibility. These tasks can be reference to shell scripts in turn invokes Hadoop tasks like Pig/Hive/Sqoop/Spark/Distcp etc. Here the shell scripts development and standards depend entirely on developer. These non-standard scripts create challenge to debug or enhance as business requirements changes.

Image Source (https://content.asg.com)

Oozie Approach

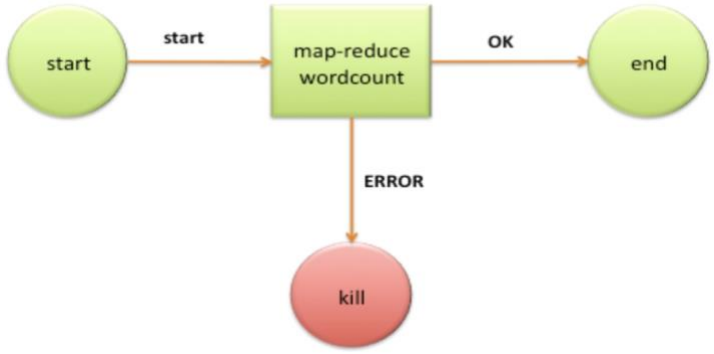
Oozie is workflow scheduler framework available within Hadoop and it has 3 major components Bundle, Coordinator & Workflow. The workflow jobs are defined as Direct Acyclic Graph (DAG) of actions like Pig/Sqoop/MapReduce/Streaming/Java//Hive/Hive2/Spark/Spark2/SSH/SH/Email etc. The Coordinator jobs are based on time frequency or data availability. The big advantage of Oozie which is part of Hadoop itself to support different actions. It is scalable, extensible and reliable system. Oozie is Java Web-Application server runs in a Java servlet-container. The control flow defines the begin and end of the flow. These actions are configured within the XML file with specific elements dedicated to Hadoop Component ex: map-reduce element. These action specific elements will help develop Hadoop actions which will help in debug and enhance tasks.

A workflow xml configuration file

```
<workflow-app name='wordcount-wf' xmlns="uri:oozie:workflow:0.1">
    <start to='wordcount'/>
    <action name='wordcount'>
        <map-reduce>
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>mapred.mapper.class</name>
                    <value>org.myorg.WordCount.Map</value>
                </property>
                <property>
                    <name>mapred.reducer.class</name>
                    <value>org.myorg.WordCount.Reduce</value>
                </property>
                <property>
                    <name>mapred.input.dir</name>
                    <value>${inputDir}</value>
                </property>
                <property>
                    <name>mapred.output.dir</name>
                    <value>${outputDir}</value>
                </property>
```

```
            </configuration>
        </map-reduce>
        <ok to='end'/>
        <error to='end'/>
    </action>
    <kill name='kill'>
        <message>Something went wrong:
${wf:errorCode('wordcount')}</message>
    </kill/>
    <end name='end'/>
</workflow-app>
```

Workflow Diagram



Conclusion

After going through above mentioned approaches, by considering the standard and availability Oozie framework fits ideally for hadoop tasks scheduling process where development-based standards are introduced and takes the advantage of Hadoop cluster resource as it is part of it.

References

https://content.asg.com

https://oozie.apache.org/