

Configure Spark Job Server on YARN

Deployment on YARN (Hortonworks 2.3.2)

Steps:

1. Download spark job server 0.6.0 that is compatible with Spark 1.4.1 installed in Hortonworks 2.3.3
2. Unzip and cd into spark-jobserver directory.
3. Copy config/local.sh.template to hdp.sh and edit as appropriate regarding DEPLOY_HOST, APP_USER, APP_GROUP, INSTALL_DIR and SPARK_HOME variables

Ex of hdp.sh:

```
# Environment and deploy file
# For use with bin/server_deploy, bin/server_package etc.
DEPLOY_HOSTS="sandbox.hortonworks.com"

APP_USER=root
APP_GROUP=hdfs
# optional SSH Key to login to deploy server
#SSH_KEY=/path/to/keyfile.pem
INSTALL_DIR=/usr/hdp/2.3.2.0-2950/spark-server
LOG_DIR=/var/log/job-server
PIDFILE=spark-jobserver.pid
JOBSERVER_MEMORY=1G
SPARK_VERSION=1.4.1
SPARK_HOME=/usr/hdp/2.3.2.0-2950/spark
SPARK_CONF_DIR=$SPARK_HOME/conf
# Only needed for Mesos deploys
SPARK_EXECUTOR_URI=/home/spark/spark-0.8.0.tar.gz
# Only needed for YARN running outside of the cluster
# You will need to COPY these files from your cluster to the remote machine
# Normally these are kept on the cluster in /etc/hadoop/conf
# YARN_CONF_DIR=/pathToRemoteConf/conf
# HADOOP_CONF_DIR=/pathToRemoteConf/conf
SCALA_VERSION=2.10.4
```

4. Copy config/local.conf.template to hdp.conf and edit as appropriate spark.master

Ex of hdp.conf

```
# Template for a Spark Job Server configuration file
# When deployed these settings are loaded when job server starts
#
# Spark Cluster / Job Server configuration
spark {
    # spark.master will be passed to each job's JobContext
    # master = "local[4]"
    # master = "mesos://vm28-hulk-pub:5050"
```

```

master = "yarn-client"

# Default # of CPUs for jobs to use for Spark standalone cluster
job-number-cpus = 4

jobserver {
  port = 8090
  jar-store-rootdir = /tmp/jobserver/jars

  jobdao = spark.jobserver.io.JobFileDAO

  filedao {
    rootdir = /tmp/spark-job-server/filedao/data
  }
}

# predefined Spark contexts
# contexts {
#   my-low-latency-context {
#     num-cpu-cores = 1           # Number of cores to allocate. Required.
#     memory-per-node = 512m      # Executor memory per node, -Xmx style eg 512m,
1G, etc.
#   }
#   # define additional contexts here
# }

# universal context configuration. These settings can be overridden, see README.md
context-settings {
  num-cpu-cores = 2             # Number of cores to allocate. Required.
  memory-per-node = 512m        # Executor memory per node, -Xmx style eg 512m, #1G,
etc.

  # in case spark distribution should be accessed from HDFS (as opposed to being
installed on every mesos slave)
  # spark.executor.uri = "hdfs://namenode:8020/apps/spark/spark.tgz"

  # uris of jars to be loaded into the classpath for this context. Uris is a string
list, or a string separated by commas ','
  # dependent-jar-uris =
["file:///some/path/present/in/each/mesos/slave/somepackage.jar"]

  # If you wish to pass any settings directly to the sparkConf as-is, add them here in
passthrough,
  # such as hadoop connection settings that don't use the "spark." prefix
  passthrough {
    #es.nodes = "192.1.1.1"
  }
}

# This needs to match SPARK_HOME for cluster SparkContexts to be created successfully
# home = "/home/spark/spark"
}

# Note that you can use this file to define settings not only for job server,
# but for your Spark jobs as well. Spark job configuration merges with this
configuration file as defaults.

```

```
# Timeout configuration
spray.can.server {
  idle-timeout = 210 s
  request-timeout = 200 s
}
```

5. Launch `bin/server_deploy.sh hdp --` this packages the job server along with config files and pushes it to the remotes you have configured in `hdp.sh`

6. On the remote server, start it in the deployed directory with `server_start.sh` and stop it with `server_stop.sh` (test is port configured, 8090 by default, is in LISTENING mode)

Observation:

- for logging configuration one should copy `log4j-server.properties` from the installation directory to the remote machine