# Spark DAOS Job Script on JLSE

## The Spark DAOS Job script is the tool for submitting Spark jobs in JLSE. The tool enables all Spark optimizations from Intel automatically including DAOS Hadoop Filesystem, Spark remote shuffle manager and OAP MLlib - optimized Spark ML algorithms with oneDAL as backend. The following is an example of running Spark KMeans that reads input data from DAOS, shuffles intermediate data through DAOS, and trains a model on an ATS cluster.

## Quick Start

### Prepare the conda environment and install necessary dependencies.

wget <https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh> -P /tmp/

bash /tmp/Miniconda3-latest-Linux-x86\_64.sh

conda create --name spark python==3.8

echo "conda activate spark" >> ~/.bashrc

conda activate spark

pip install numpy

### Create Spark Configuration Files

The Spark DAOS job script is available at “/soft/storage/daos/spark/spark-daos-job/”. The “conf-to-your-submit-dir” directory contains configuration files that you need to copy to your own directory where you submit Spark applications. For example:

mkdir ~/arcticus

cp /soft/storage/daos/spark/spark-daos-job/conf-to-your-submit-dir/\* ~/arcticus/

### Prepare Input Dataset in DAOS

The default DAOS is installed under “/soft/storage/daos/sles1/daos-2tb5/install”. The DAOS server is “daos01:10001”. Referring to [Configuration](#_Configuration) for using and connecting to a different DAOS server. For the KMeans example, there is an example dataset in DAOS path “daos://pool0/cont1/kmeans/input/libsvm”. You can also put your dataset by referring [Copy dataset to DAOS container via Hadoop](#_Copy_dataset_to) in [Appendix](#_Appendix).

### Submit Spark Job

Here are the steps to submit the Dense KMeans job to “arcticus” from JLSE login node. Please make sure you run the submit script from the directory where the configuration files locate.

$ cd ~/arcticus

$ /soft/storage/daos/spark/spark-daos-job/bin/submit-spark.sh -t 30 -n 2 -q arcticus /soft/storage/daos/spark/spark-daos-job/example/dense\_kmeans\_example.py

After submission, you will see job ID info as below.

# Submitting job: /soft/storage/daos/spark/spark-daos-job/example/dense\_kmeans\_example.py

# Submitted

SPARKJOB\_JOBID=352536

You can view Job output from “352536.cobaltlog”, “352536.output” and “352536.error”. You are supposed to get KMeans output from “352536.error”. If something is wrong, you can find error message from either “352536.output” or “352536.error”.

## 

## Script Introduction

The script only support “arcticus” and “yarrow” nodes for now since we have different DAOS and resources setups for different compute nodes. It can extend to other nodes easily if DAOS and resources are ready.

### Structure

Here is the script structure.

Text

Description automatically generated

### How to Run

#### Script Options

The script is self-explanatory. You can see provided options and examples with below commands.

$ /soft/storage/daos/spark/spark-daos-job/bin/submit-spark.sh -h

You’ll get below help messages. Some of options are for “qsub” command. You may be already familiar with them. Here is explanation of newly added options for Spark.

Text

Description automatically generated

1. -o OUTPUTDIR Directory for script output files (default: current dir)

If you don’t want to pollute your submission folder with log files, you can change COBALT and Spark job log output directory via this option.

1. -s Enable script mode

With this option, your script runs as it is instead of being submitted to Spark.

1. -m Master uses a separate node

As default, there will be a Spark worker daemon in each node, including the node where Spark master daemon locates. If you want a dedicated Spark master node without Spark worker running inside, you can give this option.

1. -I Start an interactive ssh session

If you want to interact with assigned nodes whilst your Job is running, you can add this option.

It will wait for time specified by “-w” before giving up. The node you logged in is randomly assigned.

1. -w WAITTIME Time to wait for prompt in minutes (default: 30)

Worked together with “-I” to wait for “WAITTIME” before giving up getting into shell prompt.

1. -b Prefer Spark built-in mllib to default OAP mllib

As default, it uses OAP optimized mllib to accelerate training phase.

#### Submit Script

From the help message, you can see several example command lines to either submit Spark job in python and Jar file or run standalone shell script if the “script mode” is enabled. You can follow these examples to submit your own job. Check [Submit KMeans](#_Submit_Spark_Job).

For kmeans example with dataset format in “csv”, you can run it like below command,

$ submit-spark.sh -t 60 -n 2 -q arcticus kmeans\_example.py daos://pool0/cont1/kmeans/csv/ukmeans.csv csv

#### Advanced Usage

For some advanced users, they may want to control how Spark job runs. Here are steps to run “spark-shell” or “pyspark” without “submit-spark.sh” after you logged in “arcticus” node.

## make sure all the config files are under the “~/arcticus” folder

$ cd ~/arcticus

$ /soft/storage/daos/spark/spark-daos-job/bin/start-spark.sh

$ source /soft/storage/daos/spark/spark-daos-job/bin/env\_spark\_daos.sh

$ cd $SPARK\_HOME/bin

$ ./pyspark # or ./spark-shell

From the steps, you can see we run “bin/start-spark.sh” instead of “bin/submit-spark” since we’ll run interactive statements without job being submitted in advance.

After Spark daemons being started, we can source necessary environments into current shell via the “source …” command. Until then, we can run some Spark shells, like “spark-shell” or “pyspark”, as well as any valid DAOS commands, like some commands in [Appendix](#_Appendix).

#### Switch between GPU and CPU

OAP mllib and GPU are enabled for Spark job by default. If user wants to run OAP mllib CPU, he can edit “env\_local.sh” and locate line “--conf spark.oap.mllib.useGPU=true”. Then replace “true” with “false”.

### “bin” Folder

The “bin” folder contains scripts to submit spark job. The entry point script is “submit-spark.sh” which calls “qsub” to run “start-spark.sh” in assigned nodes. “start-spark.sh” does some environment setup and preparation before calling “run-spark.sh”. “run-spark.sh” first starts up Spark master and worker daemons according to assigned nodes info from COBALT\_NODEFILE. Then, it calls Spark’s own script “spark-submit.sh” to finally submit job to Spark.

During the submission, the “setup.sh” may get called several times to set Spark and DAOS environment, as well as startup DAOS agent. It calls “env\_<node>.sh” (xxx means node name, like “arcticus”) and “env\_local.sh”. “env\_<node>.sh” sets env variables and starts up DAOS agent pertaining to the assigned nodes. Once set, it remains stable for different kinds of Spark jobs. Whilst the “env\_local.sh” may vary for different jobs since it sets Spark resources per workload or job.

“env\_gpu.sh” is also under the “bin” folder and get sourced in the “setup.sh”. It loads GPU and OneAPI related settings. If “-b” option is specified, it will not be sourced.

### “conf-to-your-submit-dir” Folder

In addition to the “env\_local.sh”, the directory contains other five files, “core-site.xml”, “gpuResourceFile\_arcticus.json”, “env\_arcticus.sh”, “env\_yarrow.sh” and “log4j.properties”.

1. “core-site.xml” sets Hadoop related configurations, like default URI and temp working dir.
2. “gpuResourceFile\_arcticus.json” sets GPU resources of compute node.
3. “env\_<node>.sh” sets DAOS and GPU resources for specific compute node. It also starts up DAOS agent.
4. “env\_local.sh” for setting executor cores and memories, number of GPUs per task, class paths and JVM parameters.
5. “log4j.property” file for controlling log output of Spark job.

### “example” Folder

It contains example KMeans python script and jar, as well as KMeans dataset. You can get the example commands with “bin/submit-spark.sh -h”.

## Configuration

With default configurations, you can run Spark job by following above instruction. For advanced user or potential later adjustment, user can change configurations in several places.

1. env\_<node>.sh

* Software Location

You can configure where software locates. DAOS, Spark, Hadoop and Java have their corresponding env variables, DAOS\_ROOT, SPARK\_HOME, HADOOP\_HOME and JAVA\_HOME.

* GPU

GPU resources are also put here for Spark worker, executor and task, like number of GPUs per executor.

* Spark Worker Resources

Usually, Spark Worker daemon takes up almost all available physical memory which may not be desired since some native memory should be reserved for application. We can configure

SPARK\_WORKER\_CORES and SPARK\_WORKER\_MEMORY.

* Software Environment Variables

There are many environment variables for different kinds of software, like JVM class path, DAOS NIC. Normal user should delegate them to professional administrator.

* DAOS Agent Startup

There is script to automatically startup DAOS agent daemon. It should remain untouched unless necessary. Of course, you can change which DAOS server to connect to by setting env var “DAOS\_AGENT\_CONF”.

1. env\_local.sh

Here you can change Spark driver and executor configuration, as well as shuffle manager. When you tune number of executors for specific job, you need to take GPU resource into consideration in addition to CPU and memory. If you want to have more virtual GPUs, you can edit “gpuResourceFile\_arcticus.json”.

1. core-site.xml, see [“conf-to-your-submit-dir Folder”](file:///C:\Users\jiafuzha\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\ZPD5HQGM\conf-to-your-submit-dir#_)section.

## Appendix

## Copy dataset to DAOS container via Hadoop

When user logged in arcticus node via “qsub -I …”, we can access DAOS POSIX container configured after “source bin/env\_spark\_daos.sh”. For example, here are commands to copy dataset to DAOS.

## make sure all the config files are under the “~/arcticus” folder

$ cd ~/arcticus

$ source /soft/storage/daos/spark/spark-daos-job/bin/env\_spark\_daos.sh

$ hadoop fs -copyFromLocal data/ukmeans.csv /kmeans/input/csv

### verify if data copied correctly

$ hadoop fs -ls /kmeans/input/csv

Then user can access the data in Spark with URI, “daos://pool0/cont1/kmeans/input/csv”.