# COBALT JLSE Script to Submit Spark Job Running with DAOS

## Quick Start (Run Spark KMeans in “presque\_debug”)

### Preparation

Login to JLSE login node and find location of COBALT script. The default location is under “/soft/storage/daos/spark/cobalt/”. You should see “bin”, “conf-to-your-submit-dir” and “example” folders. The submit scripts is under the “bin” folder. KMeans python script and ML Jar file are under the “example” folder. You’ll see details about them in [Script](#_Script) section.

As folder name, “conf-to-your-submit-dir”, suggests, please copy all config files under this folder to your job submission folder, like “/home/<user>/cobalt”. The submission script, “submit-spark.sh”, will load all of config files from the submission folder.

Since the example KMeans job uses Spark KMeans lib depending on python lib “numpy”, you should install numpy directly or via Conda under your home directory. Here are simple steps to install MiniConda and “numpy”.

$ wget https://repo.anaconda.com/miniconda/Miniconda3-4.5.4-Linux-x86\_64.sh

$ sh Miniconda3-4.5.4-Linux-x86\_64.sh

### re-login to load Miniconda3 and use default “base” env

$ pip3 install numpy

### verify if numpy is installed correctly

$ pip3 list | grep numpy

You can refer to [Configuration](#_Configuration) for creating Spark specific env instead of “base” env.

Before submitting Spark job, please make sure DAOS is installed and DAOS server is started. The default DAOS is installed under “/home/kalfizah/daos-2tb2/install”. And DAOS server is “daos18:10001”. Referring to [Configuration](#_Configuration) for using and connecting to a different DAOS server.

Besides, the KMeans job needs a dataset as input. The dataset for KMeans example is put to DAOS path “/jlse/kmeans/ukmeans.csv”. You can also extract it from “cobalt/example/data/ukmeans.csv.tgz” and put it into DAOS POSIX container via either dfuse or [Copy dataset to DAOS container via Hadoop](#_Copy_dataset_to) in [Appendix](#_Appendix).

### Submit Spark Job

Here are the steps to submit the KMeans job to “presque\_debug” from JLSE login node. Please make sure you run the submit script from the directory where the configuration files locate. For simplicity, we use “cobalt” directory to submit job.

$ cd cobalt

$ /path-to-submit-script/bin/submit-spark.sh -t 60 -n 2 -q presque\_debug /path-to-submit-script/example/kmeans\_example.py daos:///jlse/kmeans/ukmeans.csv csv 10

After submission, you should see some job ID info as below.

# Submitting job: example/kmeans\_example.py daos:///jlse/kmeans/ukmeans.csv csv 10

# 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.output” if there is no error from “352536.error”. For Spark effective configuration, master/worker and executor logs, you can get them from “352536” folder.

## Script Introduction

### 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.

$ 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. -e mark flag to end option parsing which may wrongly consume job parameters otherwise

When submit Spark job as Jar with “--class” option or any option prefixed with “--”, the script complains unrecognized “--” option due to built-in “getopts” command doesn’t support “--“ in some shell, like bash. There several ways to bypass this limitation. To make script concise and simple, we add “-e” option to tell “submit-spark.sh” to stop parsing more unexpected options which are for Spark job only.

1. -o OUTPUTDIR Directory for COBALT 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.

#### 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).

#### 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”.

## make sure all the config files are under the “cobalt” folder

$ cd cobalt

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

$ source /soft/storage/daos/spark/cobalt/bin/env\_skylake.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 “source bin/env\_skylake.sh”. 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).

### “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 skylake 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\_skylake.sh” under the same “bin” folder and “env\_local.sh” under the “cobalt” folder or any folder where you run “submit-spark.sh”. “env\_skylake.sh” sets env variables and starts up DAOS agent pertaining to skylake 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 executor cores and memories.

### “cobalt” Folder

In addition to the “env\_local.sh”, “cobalt” directory contains three xml files, “daos\_agent.xml”, “daos-site.xml” and “core-site.xml”.

1. “daos\_agent.xml” points to which DAOS server Spark job uses.
2. “daos-site.xml” sets DAOS URI, DAOS pool/container UUIDs, DAOS read/write buffer sizes, synchronous or asynchronous IO and block size. When DAOS UNS is fully supported in JLSE, we can eliminate this file.
3. “core-site.xml” sets Hadoop related configurations, like default URI and temp working dir.

There is a “log4j.property” file for controlling log output of Spark job.

If you have multiple DAOS servers or POSIX containers to work with, you can copy the above five files to a new directory, like “cobalt/daos2” and configure them properly. Then, call “../bin/submit-spark.sh” from “cobalt/daos2” folder.

### “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 if you setup the numpy lib and put dataset correctly in DAOS container. For advanced user or potential later adjustment, user can change configurations in several places.

1. bin/env\_skylake.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.

* 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.

1. cobalt/env\_local.sh

Here you can change Spark driver and executor configuration, as well as shuffle manager.

1. cobalt/daos\_agent.xml, cobalt/daos-site.xml, cobalt/core-site.xml, see [“cobalt” Folder](cobalt#_) section.

## Appendix

## Copy dataset to DAOS container via Hadoop

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

## make sure all the config files are under the “cobalt” folder

$ cd cobalt

$ source /soft/storage/daos/spark/cobalt/bin/env\_skylake.sh

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

### verify if data copied correctly

$ hadoop fs -ls /jlse/kmeans

Then user can access the data in Spark with URI, “daos:///jlse/kmeans/ukmeans.csv”.