Конфигурирование и настройка Spark

МОДУЛЬ 8

ШКОЛА БОЛЬШИХ ДАННЫХ

Уровни конфигурирования

- операционная система
- менеджер кластера (например, **“YARN”**)
- **“spark”**

Способы конфигурирования Spark

- конфигурация “spark” (объект “**SparkConf**”)
- параметры **командной строки**
- настройка **окружения**
- методы “**SparkSession.builder**” (“master()”, “appName()”, “config()”)
- настроить необходимо **ДО** создания spark сессии
- все настройки тут - **https://spark.apache.org/docs/latest/configuration.html**

SparkConf

- самый "**полнофункциональный**" способ
- конфигурация: пары "**ключ**" - "**значение**"
- конфигурация используется при **создании сессии**
- объект **“SparkConf”**
  - ✓ загружает значения по умолчанию
  - ✓ можно дополнить своими значениями
  - ✓ методы поддерживают **“chaining”**

Параметры командной строки

- опция **“—conf”/”-c”**

  spark-submit --conf spark.eventLog.enabled=false

- наиболее важные опции вынесены отдельно, см. **“spark-submit –help”**

  spark-submit --driver-memory 1G

Настройки окружения

- можно задать в файле **“conf/spark-env.sh”**
- можно задать программно **ДО** создания сессии

os.environ["PYSPARK_PYTHON"] = "/usr/bin/python2.7"

Что чаще всего нужно конфигурировать

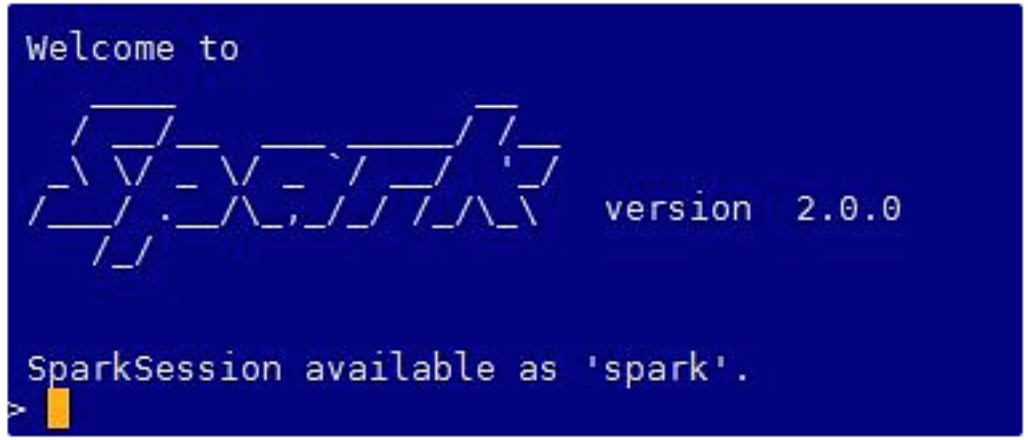
- **“spark.app.name”** - имя приложения
- **“spark.yarn.queue”** - очередь YARN
- **“spark.driver.memory”**, **”spark.executor.memory”** - объемы памяти (драйвер и экзекьюторы)
- **“spark.pyspark.python”**- версии питона
- **“spark.driver.extraClassPath”**- пути для поиска файлов драйвером
- **“spark.master”**- URL кластер менеджера
- **“spark.submit.deployMode”** - режим исполнения
- **“spark.sql.shuffle.partitions”** - количество партиций для join/groupby
- уровень логгирования (при запуске из командной строки - рассмотрим отдельно)

Dynamic Resource Allocation

- механизм **динамического** реагирования на потребность в ресурсах
- отключен (по умолчанию) (**“spark.dynamicAllocation.enabled”**)
- экзекьюторы **добавляются** когда нужны (эвристика)
- экзекьюторы **удаляются**, когда больше не нужны
- ресурсы экзекьютора
  - ✓ **“shuffle files”** - можно использовать внешние “spark.shuffle.service.enabled”
  - ✓ **“cache”** - экзекьюторы с кэшем не удаляются (возможно в будущем поступят как с “shuffle”)

Spark shells

- **“REPL”** - Read-Evaluate-Print-Loop
  - ✓ создают сессию (доступна как `spark`)
  - ✓ настраивают окружение
- языковые shells
  - ✓ Scala (**“spark-shell”**)
  - ✓ Python (**“pyspark”**)
  - ✓ SQL (**“spark-sql”**)
  - ✓ R (**“sparkR”**)



ВАЖНО: для выхода наберите **“quit()”** (или **“CTRL+D”**)

ШКОЛА БОЛЬШИХ ДАННЫХ

Запуск python приложения

- shell-ы не рассчитаны на запуск приложения
- используем **“spark-submit”**

  spark-submit myapp.py

- в приложении необходимо явно создать сессию

  spark = SparkSession.builder.master().appName().getOrCreate()

- уровень логгирования

  spark.sparkContext.setLogLevel('WARN’)

  ✓ действует "не сразу" (с момента создания сессии)

Настройка jupyter notebook

Типичная настройка включает

- **“PYTHON_PATH”** должен содержать необходимые библиотеки (pyspark, py4j)
- **“SPARK_HOME”** должен указывать на инсталляцию `spark`
- версии питона (**“PYSPARK_PYTHON”**, **“PYSPARK_DRIVER_PYTHON”**)
- параметры **“SPARK_SUBMIT”**

вся информация доступна через `pyspark`

Локальная установка pyspark

Spark может работать локально (включая pyspark)

Установка проста

- ✓ устанавливаем pyspark (pip, conda, ...)
- ✓ копируем дистрибутив spark нужной версии с сайта
- ✓ в переменной окружения SPARK_HOME сохраняем путь (выше)
- ✓ устанавливаем JDK версии 8 (важно)
- ✓ проверяем (**“pyspark”**)

❖ настраиваем jupyter notebook

❖ работаем локально (**“.master("local")”**)

Вопросы?