ML BD

Katya Koshchenko

Spring 2020

1 Лекция 3. SparkSQL

1.1 Background and Goals

Spark проблемы: низкоуровневый процедуральный код и отсутсвие оптимизаций.

Shark — первая попытка сделать резляционный интерфейс Spark, сделал так, чтобы Apache Hive System запускалась на Spark. Но в нем команды можно было писать только именно SQL строкой, оптимайзер был настроен именно на MR, не расширить вообще было.

Заметили, что большинство пайплайнов — комбинации реляционных и процедуральных алгоритмов. И сделали SparkSQL — новый модуль в Apache Spark. В нем DataFrame — коллекции структурированных записей, которыми можно манипулировать, пользуясь Spark APIs и процедурацльный, и реляционные. Их можно создать напрямую из RDD.

1.2 Programming interface

DataFrame — распределнная коллекция строк с одинаковой схемой. Он эквивалентен таблице в реляционной базе данных. Можно им пользоваться как RDD.

Строить фреймы можно из внешнего источника (HDFS, Hive) или существующей RDD. Вообще его можно рассматривать как RDD объектов-строк, так что над ним всякие процедуральные операции как map можно делать. Реляционные операции выполнять можно, используя DSL (domain-specific language), похожий на Pandas в питоне.

Фреймы тоже ленивые, так что оптимизировать их в радость. Spark строит перед их запуском логический план, а потом физичекский план. В отличае от оригинального Spark, этот строит AST выражения, которое передается в Catalyst для оптимизации.

Всякие штуки. Cache() <-> persist(), кэширование может быть полезно для интерактивных запросов и итеративных алгоритмов мл. UDF — функция, определяемая юзером, которая выполнится над фреймом.

1.3 Catalyst

В Каталисте лежат какие-то базовые библиотеки для представления и правила манипулирования с деревьями. Самый популярный подход к составлению правил: найти поддерево определенной структуры pattern matching функциями и заменить их на что-то. Каталист группирует правила в батчи и запускает каждый, пока он не достигнет точки фиксации, то есть пока дерево не перестанут меняться.