BIGDATA

Лекция 6

ПЛАН

- HBase: остатки
- Spark

ТЕРМИНОЛОГИЯ HBASE

- namespace - набор логически взаимосвязанных таблиц
- Аналог "database" в postgres
- table - набор строк с разреженным набором колонок
- Схемы нет в принципе

ТЕРМИНОЛОГИЯ HBASE

- Column Family - набор колонок
- Логически больше похоже на SQL таблицу, чем HBase-таблица
- Но не один в один
- Физически - близки по времени обновления колонки одной строки из одной CF будет физически храниться рядом

ТЕРМИНОЛОГИЯ HBASE

- Давно обновленная ячейка строки может оказаться подальше
- Но для колонок из разных family ситуация может быть похуже
- Иногда можно выделить в CF колонки с определенным шаблоном жизненного цикла
- Например, редко изменяемые и часто запрашиваемые можно выделить в отдельное family
- Но не сильно увлекаемся

ТЕРМИНОЛОГИЯ HBASE

- Column qualifier - имя колонки внутри column family
- Набор column family - свойство таблицы
- Их обычно немного - редко когда двузначное число
- Column qualifier - могут отличаться у разных строк

ТЕРМИНОЛОГИЯ HBASE

- Cell - "ячейка" таблицы
- Определяется ключом строки, CF и CQ
- Содержит значение и timestamp
- Можно хранить несколько версий колонки

ОСНОВНЫЕ ОПЕРАЦИИ

- Создание таблицы: указывается имя таблицы и перечень CF
- put: установить значение колонки в строке
- get: узнать значение заданной колонки в заданной строке
- Вариант: все колонки строки (может быть заметно дороже)
- get: можно ограничивать CF

ОСНОВНЫЕ ОПЕРАЦИИ

- scan: пройтись по диапазону колонок
- Можно уточнить диапазон колонок
- Ограничить набор колонок
- Еще есть удаление

КАК ОНО РАБОТАЕТ

- Работает поверх HDFS
- В чем-то повторяет распределение ролей
- HMaster - аналог NameNode
- RegionServer - аналог DataNode

Картинка

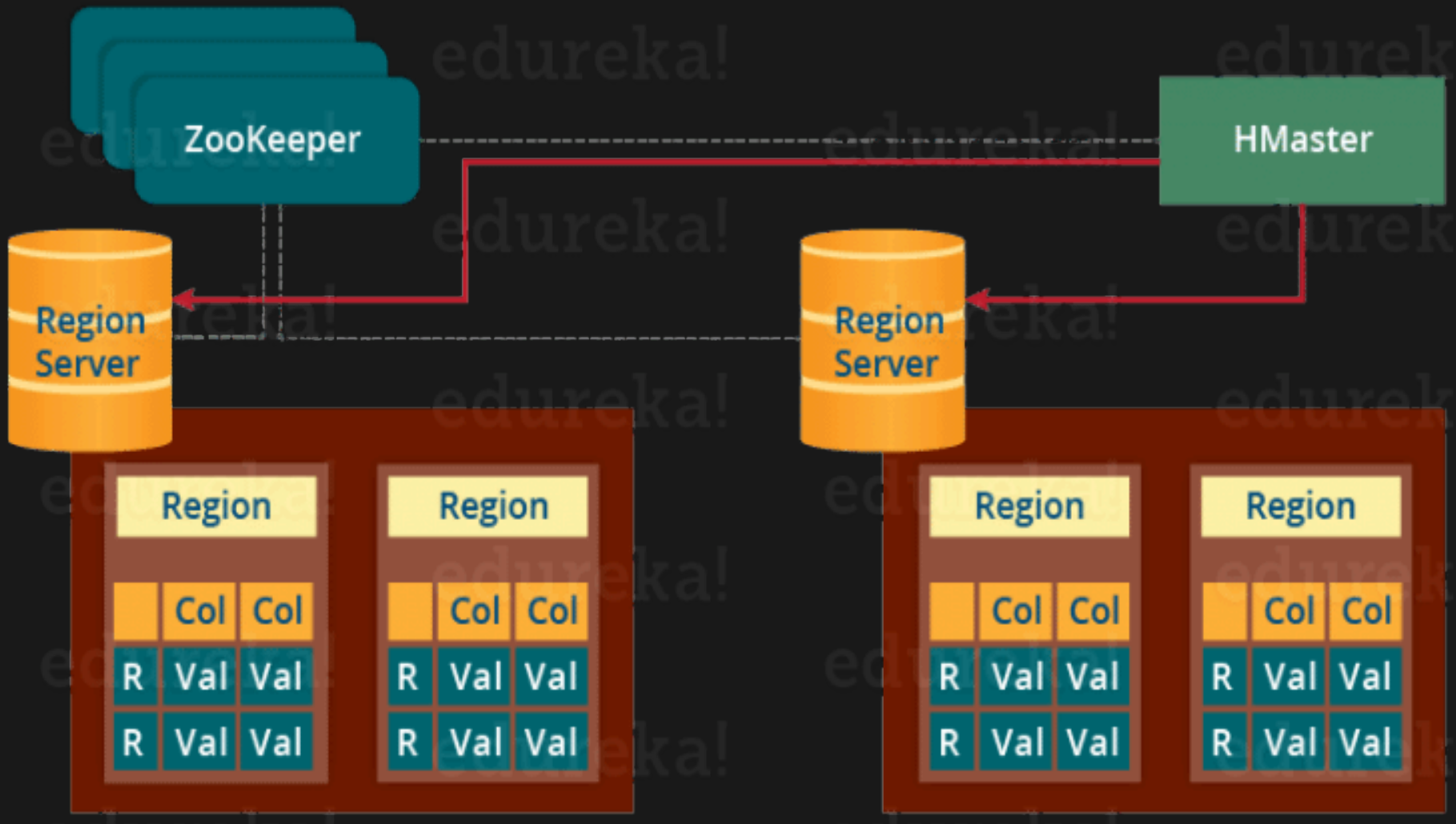

Figure: Components of HBase

КАК ОНО РАБОТАЕТ

- Таблица партиционируется по ключу строки
- Диапазон ключей назначается своему RegionServer-у
- RegionServer - это процесс, запускаемый на физическом узле
- В принципе - можно на любом, осмысленно - на HDFS-узле

КАК ОНО РАБОТАЕТ

- Пустая таблица сначала живет на одном RegionServer-е
- По мере роста происходит разбиение на два и выделение нового RegionServer-а
- HMaster хранит таблицу META
- В ней хранится знание о диапазонах ключей, назначенных RegionServer-ам

Картинка

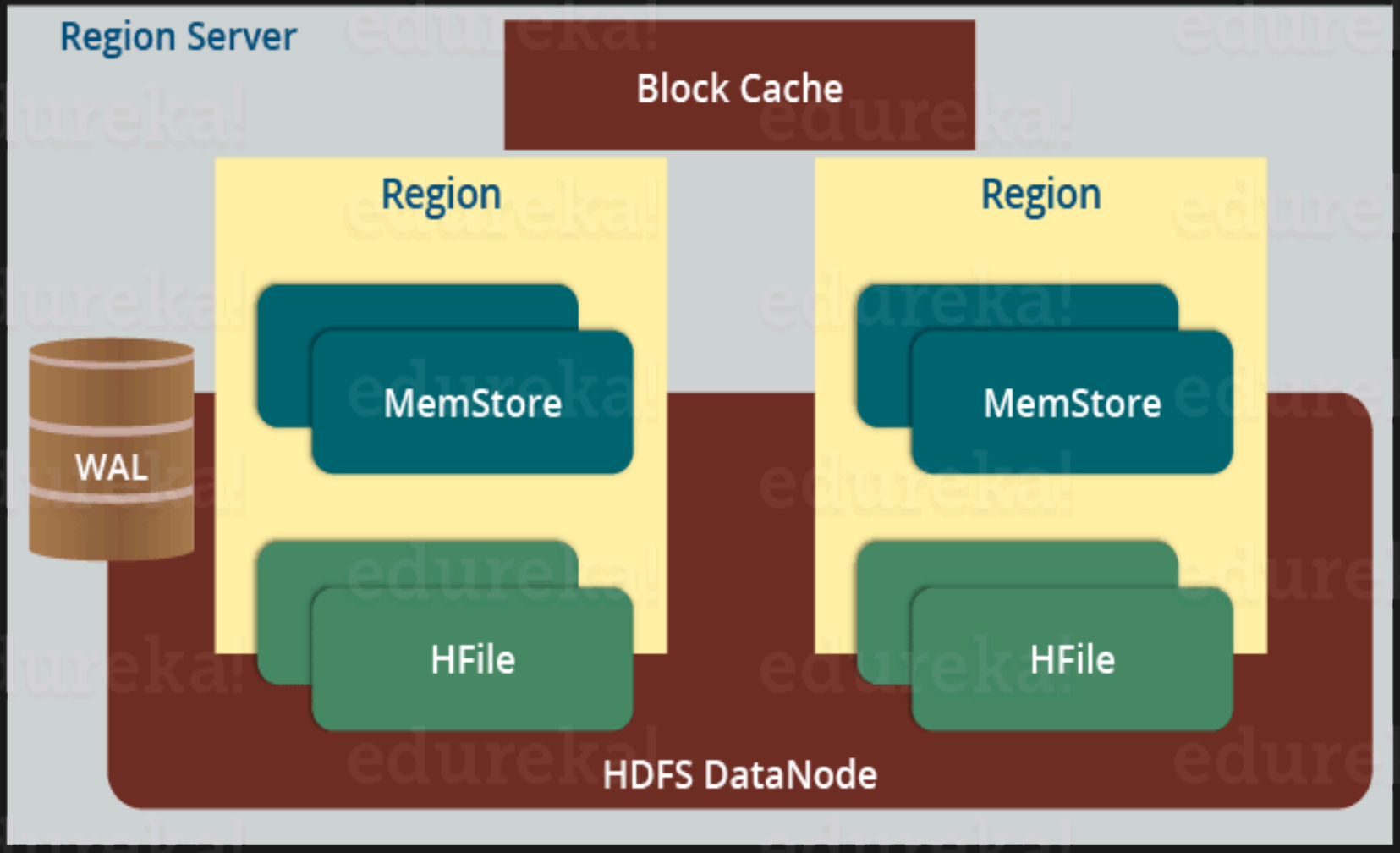

Figure: Region Server Components

КАК ОНО РАБОТАЕТ

- WAL - write-ahead log
- Помогает восстанавливаться при выключении
- Разные степени использования - настраивается на уровне таблицы
- Memstore - структура в памяти, что-то вроде дерева поиска

КАК ОНО РАБОТАЕТ

- Каждому CF соответствует один Memstore
- И много HFile-ов
- После записи в MemStore клиенту сообщается об успешной записи
- Когда MemStore становится большим - записывается в новый HFile

ПРОМЕЖУТОЧНЫЙ ИТОГ

- Пририсовывается схема работы get/scan
- Беспокоит хранение неактуальных копий значения ячейки
- И рост числа файлов
- И в целом монотонность роста
- Мы вообще удалять что-то собираемся ? И как ?

КАК ЧТО-ТО УДАЛЯЕТСЯ

- Логически - есть два варианта: явное удаление и по указанию времени жизни
- Физически организовать явное удаление проблематично
- Можно реализовать как вариант put-операции
- Установить на ячейку признак ее удаления - "thomb stone"

КАК ЧТО-ТО УДАЛЯЕТСЯ

- При запросах будем его находить
- Но это не решает проблему физического разрастания
- Скорее даже наоборот
- Нужна еще одна системная активность

Spark

- Высокоуровневая надстройка над механизмами распределенной обработки
- Прячет детали
- Предлагает удобные абстракции
- Претендует на хорошую оптимизацию
- В основе - Scala-ориентированная
- Существует в Java- и Python-изводе

Spark

- Предлагает несколько моделей
- Чуть более историческая - RDD
- Resilient Distributed Dataset
- Это такая распределенная коллекция
- С параллельной обработкой
- И устойчивостью к падениям

RDD

- Из чего можно создать RDD
  - Из обычной программной коллекции
  - Из обычного файла
  - Из HDFS-файла
  - Из всего, что выражается через InputFormat
  - Из HBase-таблицы
  - Из Hive-таблицы
  - Из всякого разного: Cassandra, S3 и т.п.

RDD

- Два типа операций: преобразования (transformations) и действия(actions)
- Пример преобразования - метод map
- Это не тот, который из MapReduce
- А в смысле FP
- Поэлементное преобразование
- Результат преобразования - новый RDD

RDD

- Пример действия - метод reduce
- Это не тот, который из MapReduce
- Сильно не тот
- А в смысле FP
- Агрегация над коллекцией
- Результат действия - значение

RDD

- Есть еще метод reduceByKey
- Вот он больше похож на reduce из MR
- Редукция по ключам
- И результат - RDD
- Как и в MR
- То есть reduceByKey - преобразование

Преобразования

- Все преобразования - ленивые
- Действия являются триггером для материализации преобразований
- Если сохранить преобразование в переменной
- И применить отдельно два действия
- Оно может быть пересчитано
- Но можно явно заказать его переиспользование
- Методы cache и persist

Пример на Scala

```scala
val lines = sc.textFile("data.txt")
val lineLengths = lines.map(s => s.length)
val totalLength = lineLengths.reduce((a, b) => a + b)
```

Пример на Python

```python
lines = sc.textFile("data.txt")
lineLengths = lines.map(lambda s: len(s))
totalLength = lineLengths.reduce(lambda a, b: a + b)
```

Тонкости с функциями

- Параметрами преобразований и действий являются функции
- Функции работают удаленно - вообще говоря
- В духе map-reduce
- Не любой синтаксически корректный код будет корректен по смыслу
- Или будет корректен, но не оптимален

Рекомендации

- Для Scala
  - Анонимные функции
  - Статические методы в глобальном синглтоне
- Для Python
  - lambda
  - Локальные def в той же функции, где вызываем spark-метод
  - Функции верхнего уровня в том же модуле

Как оно работает

- Spark планирует каждое задание (job)
- И разбивает его на задачи (task)
- Задачи назначает исполнителям (executor)
- Учитывая, где находятся данные
- Их ожидаемый размер и т.п.

Как оно работает

- И отправляет код к месту работы
- Как правило - поближе к данным
- В частности, при наличии замыкания
- Оно корректно отправится к месту работы
- И если код в замыкании что-то поменяет
- То оно поменяется в удаленном слепке

Как оно работает

- Аналогично - с печатью
- Частая ошибка - что-то печатать
- В функциях, передаваемых в преобразования
- И удивляться, что они "ничего не печатают"
- Хотя эффект преобразования виден

Варианты преобразований

Варианты преобразований

- mapPartitionsWithIndex - с нумерацией передаваемых элементов
- sample - случайная выборка
- distinct - дедупликация
- groupByKey - над коллекцией пар
- Возвращает ключ и итератор по значениям

Варианты преобразований

- aggregateByKey - посложнее
- Разберем на примере
- Пусть есть товары
- Про каждый товар знаем его тип, разновидность и стоимость
- Например, производитель телефона, модель и стоимость