Проектирование и реализация решений интеллектуального анализа BigData с использованием стека технологий Apache Spark и методов онтологического инжиниринга

Постаногов Игорь Сергеевич
ipostanogov@outlook.com

О себе

- Инженер @ Пермский государственный университет (ПГНИУ)
- Разработчик @ Малое инновационное предприятие «ООО МИП КНОВА», http://knova.ru
- Участник международной исследовательской группы
  - Проект - «Разработка унифицированных программных средств автоматизированной трансформации традиционных информационных систем в интеллектуальные с использованием методов онтологического инжиниринга на примере двуязычной базы знаний в области Компьютинга» под рук. Е.К.Хеннера
- Магистрант 2 курса @ Кафедра математического обеспечения вычислительных систем, ПГНИУ

Учебные дисциплины в 3 семестре

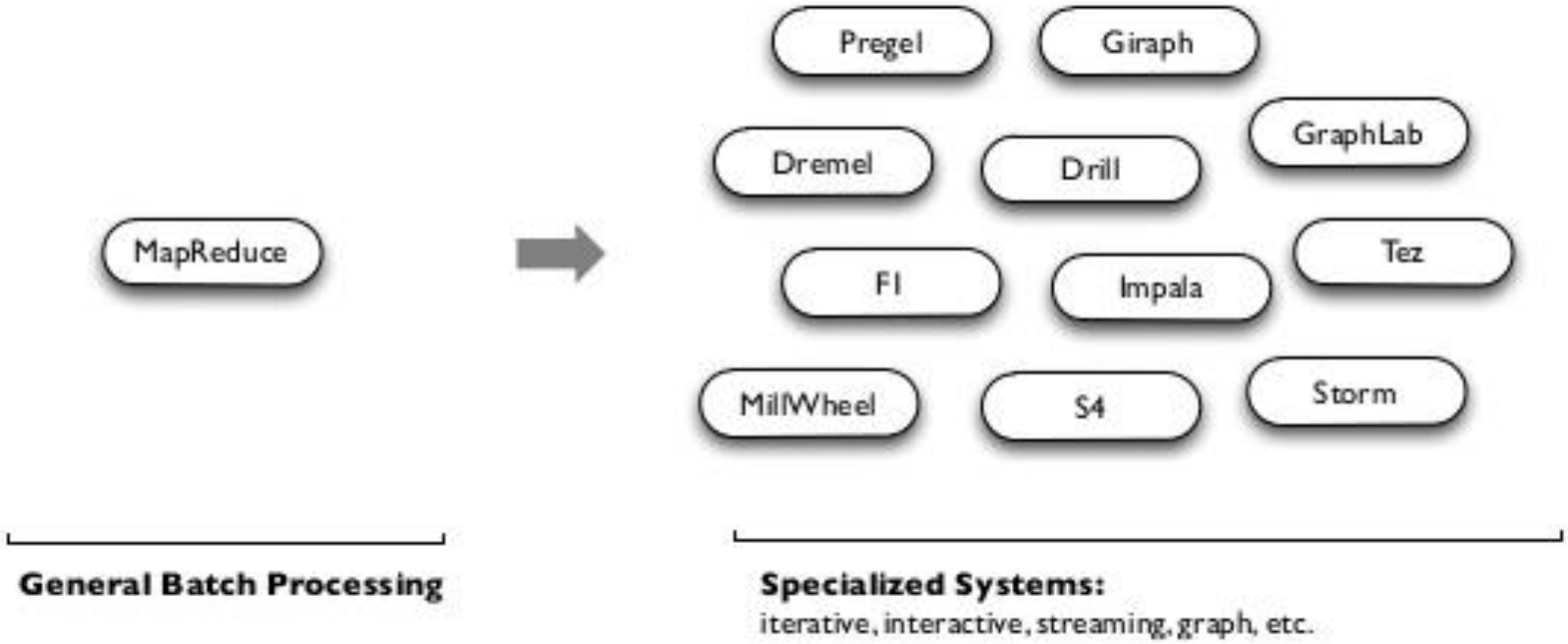
- Добыча знаний: теоретические основы и инструментальные средства Data mining
- Современные Internet-технологии

Что не так с MapReduce?

- Данные должны уже «быть» до старта
- Пакетная обработка
- Связь между задачами через HDFS
- Не поддерживаются рекурсивные и итеративные задачи
- Долгая планировка задачи

- Задачи неудобно решать в парадигме MR

Как с этим справлялись?



В чём проблема обходных решений?

- Под каждую задачу – новый фреймворк
  - Затраты на обучение
- Фреймворки всё так же ограничивают типы задач
  - Dryad & MR Merge не поддерживают циклический поток данных
- Или не адекватны среде вычислений
  - Twister не отказоустойчивый

Hadoop MapReduce Word Count

```
public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new
IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context
context
                ) throws IOException, InterruptedException
{
      StringTokenizer itr = new
StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
```

....

```
…

  public static class IntSumReducer
       extends
Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable>
values,
                       Context context
                       ) throws IOException,
InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }
```

Hadoop MapReduce Word Count

```
…
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Scala Word Count

```
object WCScala extends App {
  val text = scala.io.Source.fromFile("file.txt").getLines()
  val words = text.flatMap(line => line.split(' '))
  val wordsCount = words.foldLeft(Map.empty[String,
Int]){
    (count, word) => count + (word ->
(count.getOrElse(word, 0) + 1))
  }
}
```

Spark Word Count

```
object WCSpark extends App {
  val conf = new SparkConf().setAppName("Word Count")
  val sc = new SparkContext(conf)
  val words = sc.textFile("hdfs://...").flatMap(line =>
line.split(" "))
  val wordsCount = words.map(word => (word,
1)).reduceByKey((a, b) => a + b)
}
```

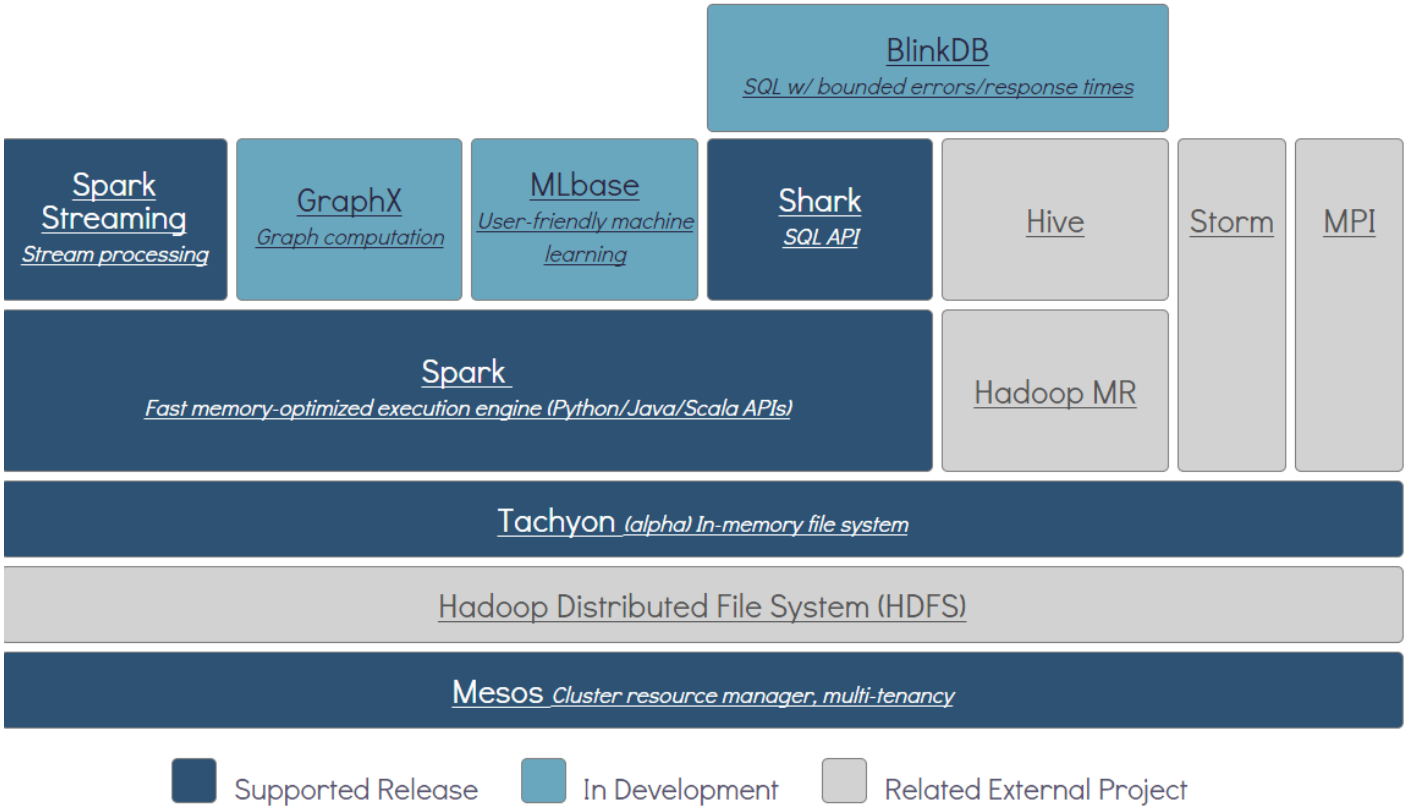
Что есть Apache Spark

- Быстрая и универсальная система кластерных вычислений
- + набор стандартных расширений
  - Spark SQL (обработка SQL и структурированных данных)
  - MLlib (машинное обучение)
  - GraphX (обработка графов)
  - Spark Streaming (потоковая обработка)
- Высокоуровневое API для Java, Scala, Python

Кто за всем стоит?

- Изначально (с 2009 г.)
  - AMPLab – UC Berkeley
- Теперь
  - Apache Software Foundation (с февраля 2014 г.)
  - 397 участников, 9382 коммитов (на 23.01.2015)

Berkeley Data Analytics Stack



Что об этом говорят?

Организациям, сталкивающимся с проблемами при работе с Big Data, включая сбор, ETL, хранение, исследование и анализ, следует рассмотреть Spark ввиду высокой производительности in-memory вычислений с помощью него, а также широте возможностей его применения. Он поддерживает передовые решения поверх Hadoop-кластеров, включая итеративные модели, необходимые для машинного обучения и анализа графов.

Gartner, Advanced Analytics and Data Science (2014)

Кто это использует?

Для чего используют?

- Im-memory аналитика и поиск аномалий (Conviva)
- Интерактивные запросы к потокам данных (Quatifind)
- Анализ логов (Foursquare)
- Оценка загруженности дорог по информации с мобильных GPS (Mobile Millennium)
- Классификация спама в Twitter (Monarch)
- …

Ключевые концепции Spark - RDD

- Resilient Distributed Dataset (RDD)
  - Устойчивый распространенный набор данных
  - Абстракция распределённой памяти
  - Ленивый (lazy) и эфемерный read-only набор
  - Распределен по узлам кластера
  - Отказоустойчив (поддерживает *родословную*)

Откуда берутся RDD?

- Из файлов в файловой системе (в т.ч. из HDFS)
- С помощью функции parallelize коллекций из Scala (массивов, списков, …)
- Из других RDD с помощью операции flatMap
  - А, следовательно, и её производных (map, filter, …)
- Изменяя персистентность исходной RDD
  - Кэширование в памяти
  - Сохранение в файловую систему
  - + в последних версиях – их комбинация

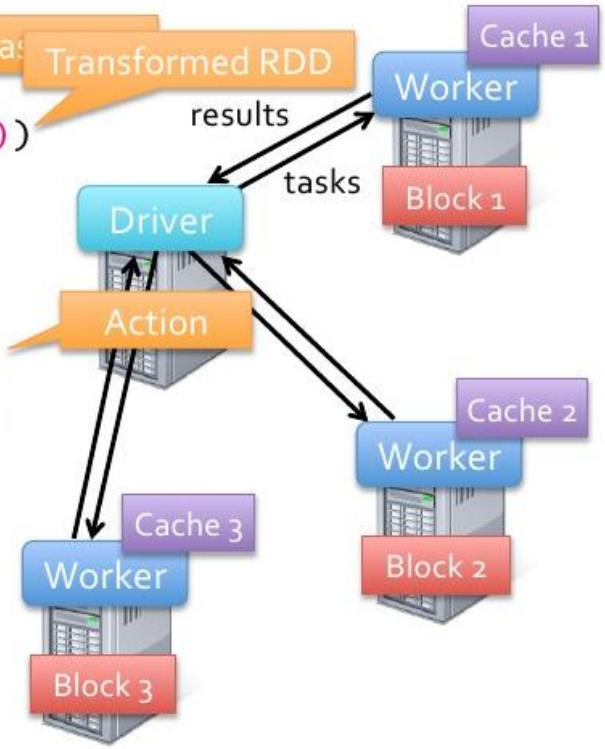
Операции над RDD

map	reduce	sample
filter	count	cogroup
groupBy	reduceByKey	take
sort	groupByKey	partitionBy
join	first	pipe
leftOuterJoin	union	save
rightOuterJoin	cross	...

Пример: Log Mining

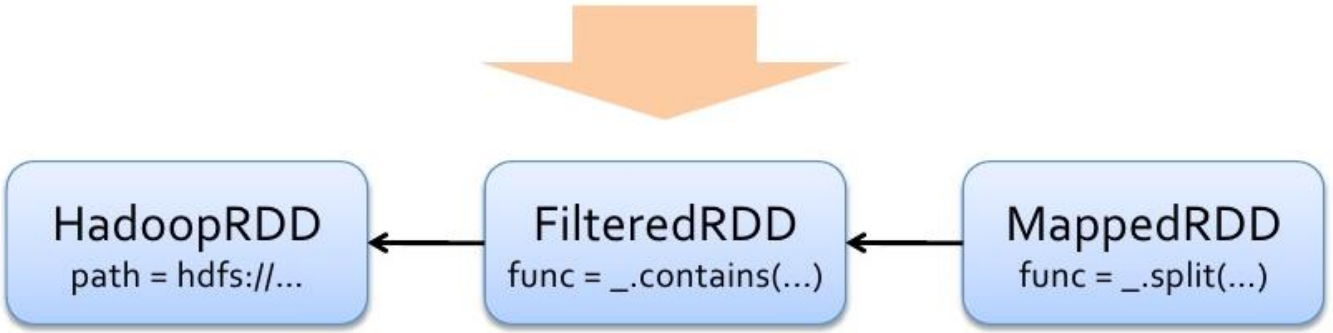
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(_.startsWith("ERROR"))
messages = errors.map(_.split('\t')(2))
cachedMsgs = messages.cache()

cachedMsgs.filter(_.contains("foo")).count
cachedMsgs.filter(_.contains("bar")).count
. . .
```

Result: scaled to 1 TB data in 5-7 sec
(vs 170 sec for on-disk data)



Отказоустойчивость (поддержка родословной)

E.g:
```
messages = textFile(...).filter(_.contains("error"))
                        .map(_.split('\t')(2))
```



Пример: логистическая регрессия

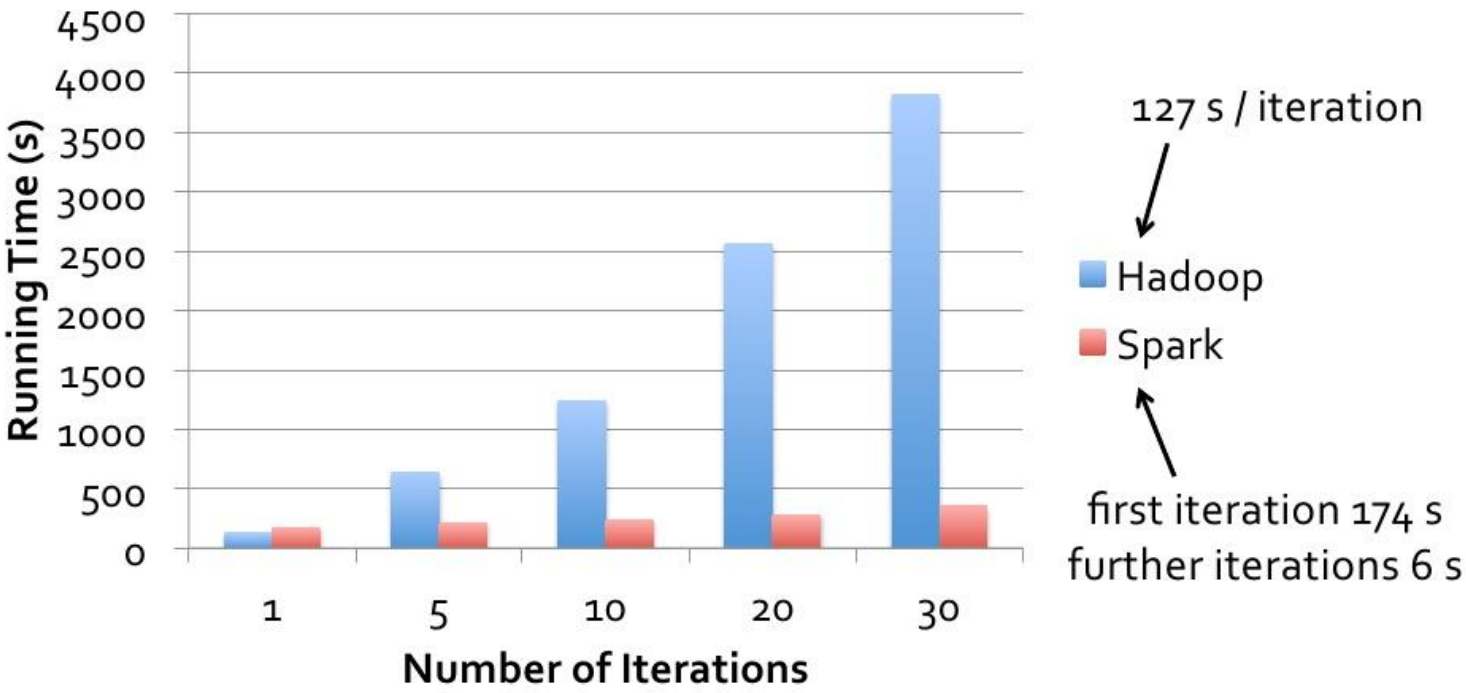
```
val data = spark.textFile(...).map(readPoint).cache()

var w = Vector.random(D)

for (i <- 1 to ITERATIONS) {
  val gradient = data.map(p =>
    (1 / (1 + exp(-p.y*(w dot p.x))) - 1) * p.y * p.x
  ).reduce(_ + _)
  w -= gradient
}

println("Final w: " + w)
```

Load data in memory once

Initial parameter vector

Repeated MapReduce steps to do gradient descent

Производительность логистической регрессии



Интегрированные решения

- Databricks Cloud
- Spark + Hadoop + Hbase
- Spark + PyData
- Spark + Kafka + Cassandra
- Spark + ElasticSearch
- Spark + Play
- Spark + Mesos + Mesosphere + Google Cloud Platform

Встроенные библиотеки

- Spark Streaming
- GraphX
- Spark SQL
- MLlib

Предложенное задание

- Построить индекс типа RDD[(A, B)], в котором ключом является слово, а значением - список ссылок, на страницах которых встречается это слово.
- Построить временную Spark SQL таблицу, в которой есть 2 столбца: (значение метаданного; ссылка на страницу, имеющую это значение метаданного).
- Реализовать поиск по словам на странице.
- Реализовать поиск по метаданным страниц.

Применение в магистерских

- Apache Spark
  - Распределённый параллелизм
- GraphX
  - Обработка графа Википедии

Применение на работе

- Spark SQL & MLlib
  - Анализ автомобильного потока
- Apache Spark
  - Отслеживание интернет-трафика компании

Источники

- https://spark.apache.org/docs/1.1.0/
- http://www.slideshare.net/pacoid/how-spark-fits-into-the-big-data-landscape
- http://www.ibm.com/developerworks/ru/library/os-spark/
- Spark: Cluster Computing with Working Sets. Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica. *HotCloud 2010*. June 2010.
- https://twitter.github.io/scala_school/ru/collections.html
- http://www.slideshare.net/AnatoliyNikulin/apache-hive-34230672
- http://www.slideshare.net/Hadoop_Summit/spark-and-shark