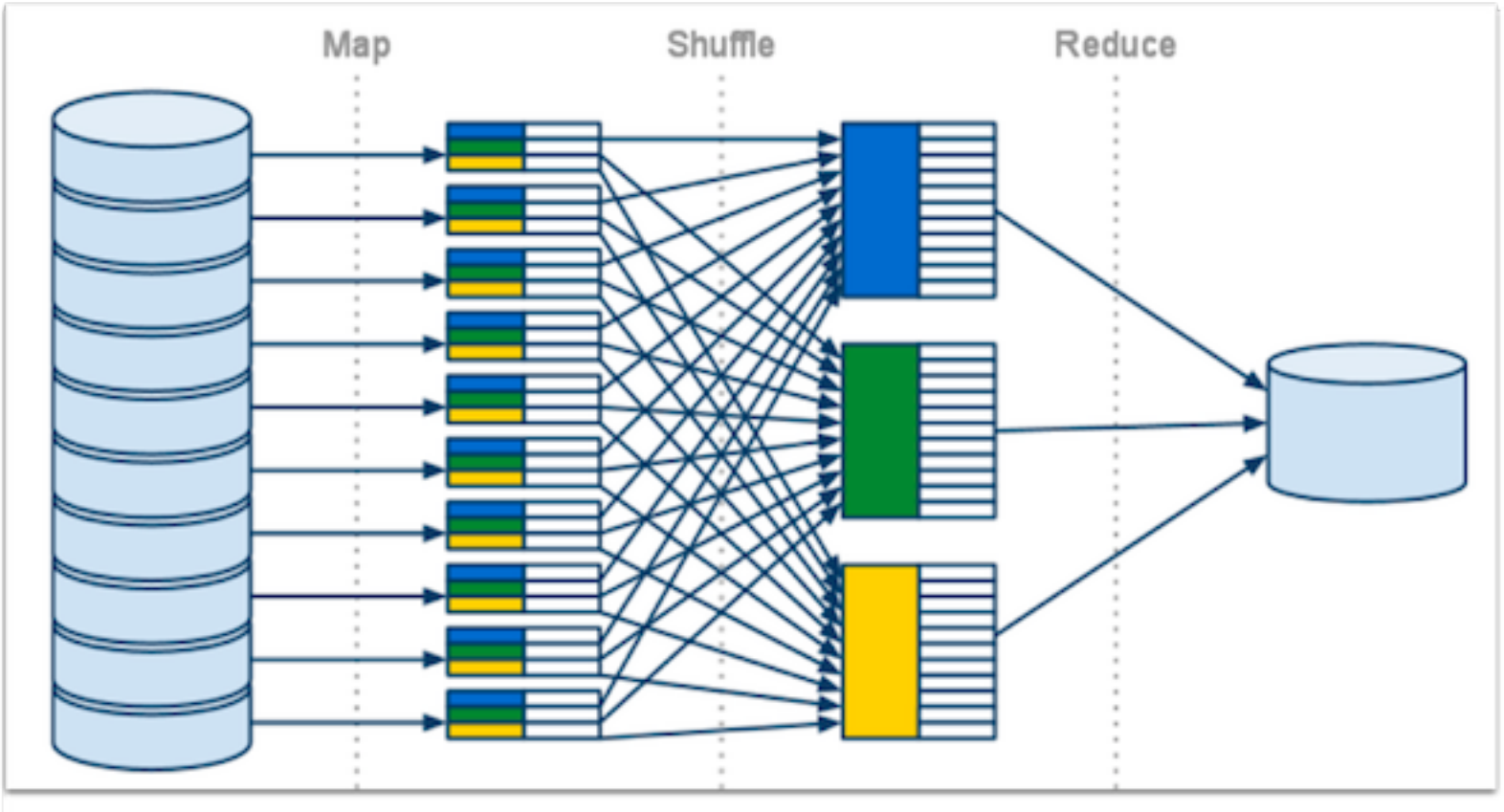
Схема выполнения MapReduce задачи



«Хорошие» задачи для MapReduce

- batch обработка
- grep запросы
- group by запросы

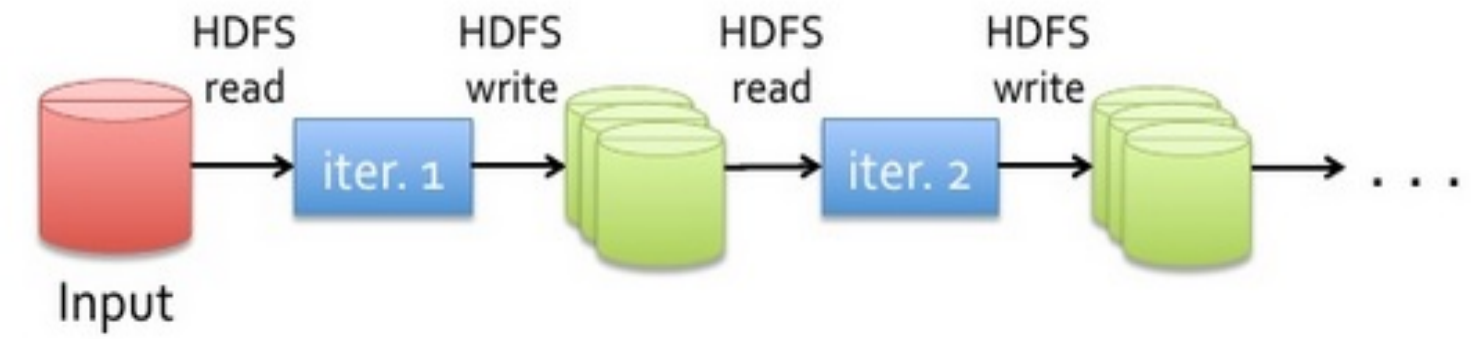
«Проблемные» задачи для MapReduce

«Проблемные» задачи для MapReduce

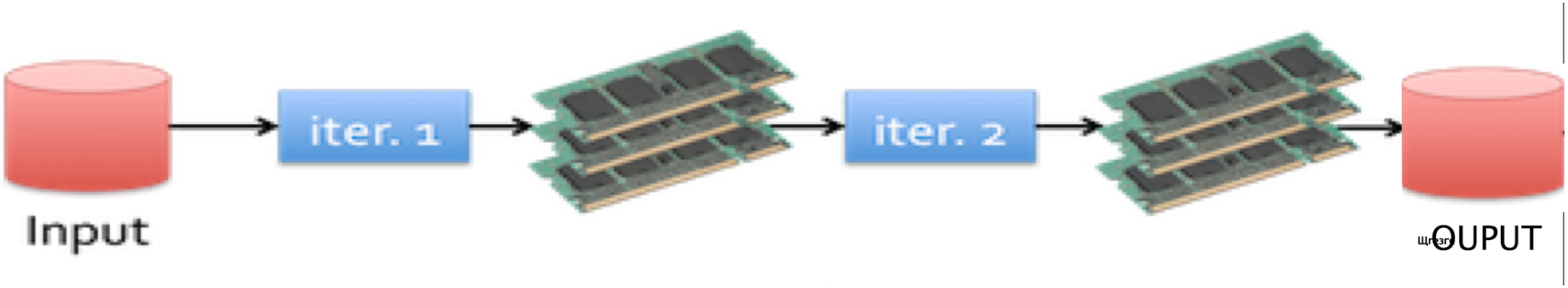
Хочу вычислить PageRank:

```
…
for (int i = 0; i < numberOfIterations; ++i) {
        String pagerankOutputDir = outputPath + "/pagerank" + ((i<10)?"0":"") + i;
        conf.set("input.pagerank", pagerankInputDir);
        Job job = new Job(conf);
        job.setJarByClass(PagerankDriver.class);
        job.setMapperClass(PagerankMapper.class);
        job.setReducerClass(PagerankReducer.class);
        job.setJobName ("pagerank iteration " + i);


        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(pagerankOutputDir));

        if (!job.waitForCompletion(true)) {
           return -1;
        }
        pagerankInputDir = pagerankOutputDir;
      }
…
```

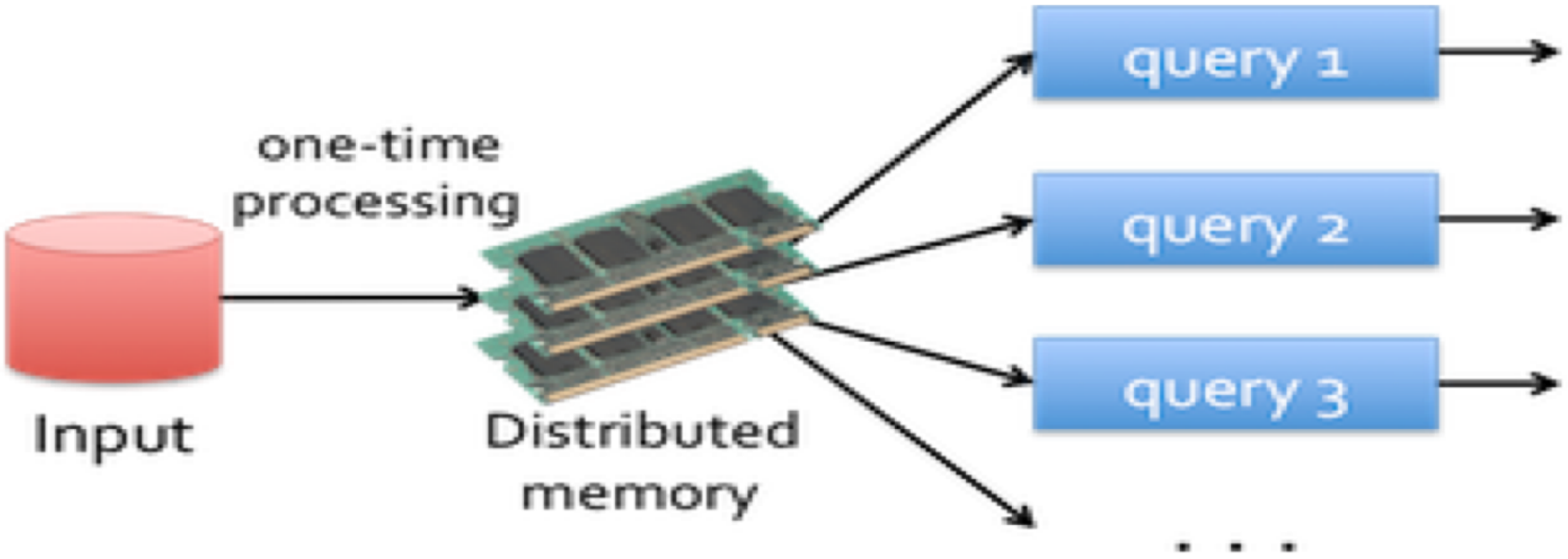
«Проблемные» задачи для MapReduce



Идея сохранения промежуточных результатов в оперативной памяти



Идея сохранения промежуточных результатов в оперативной памяти



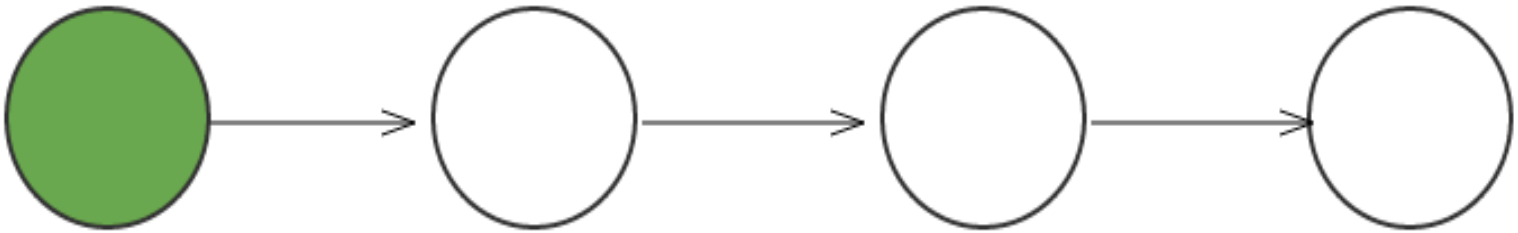
Есть ли какие-то проблемы в предложенной схеме?

Отказоустойчивость

Решение данной проблемы

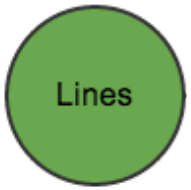
Сохранение метаданных о проведенных вычислениях.

Сохранение цепочки вычислений над данными



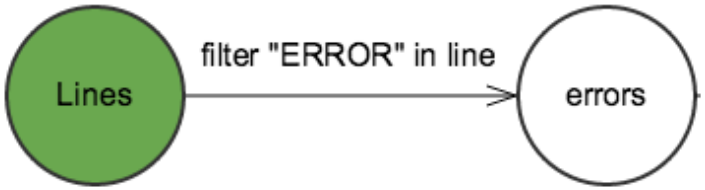
Трансляция вычислений в направленный граф

```
>>> lines = sc.textFile("hdfs://…”)
```



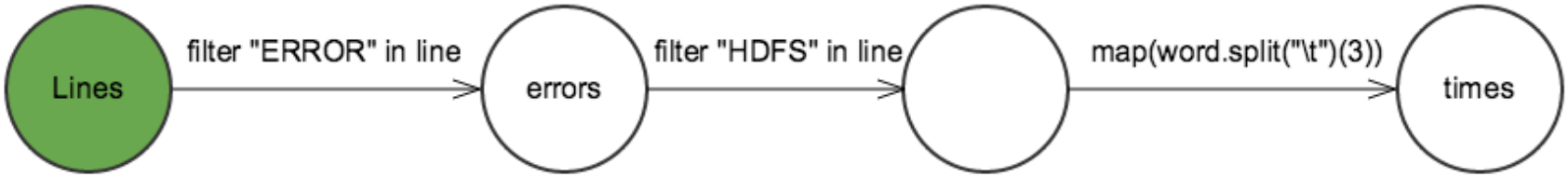
Трансляция вычислений в направленный граф

```
>>> lines = sc.textFile("hdfs://...")
>>> errors = lines.filter(lambda line: "ERROR" in line)
```



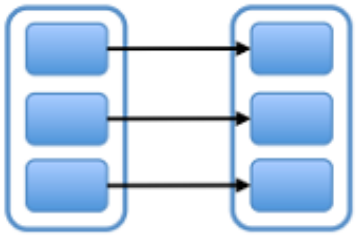
Трансляция вычислений в направленный граф

```
>>> lines = sc.textFile("hdfs://...")
>>> errors = lines.filter(lambda line: "ERROR" in line)
>>> errors.filter(lambda line: "HDFS" in line)
      .map(lambda word: word.split("\t")(3)) \
          .collect()
```

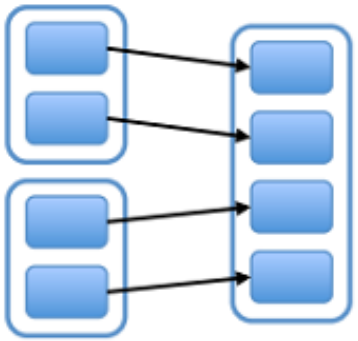


Всегда ли будут линейная цепочка вычислений?

Всегда ли будут линейная цепочка вычислений?

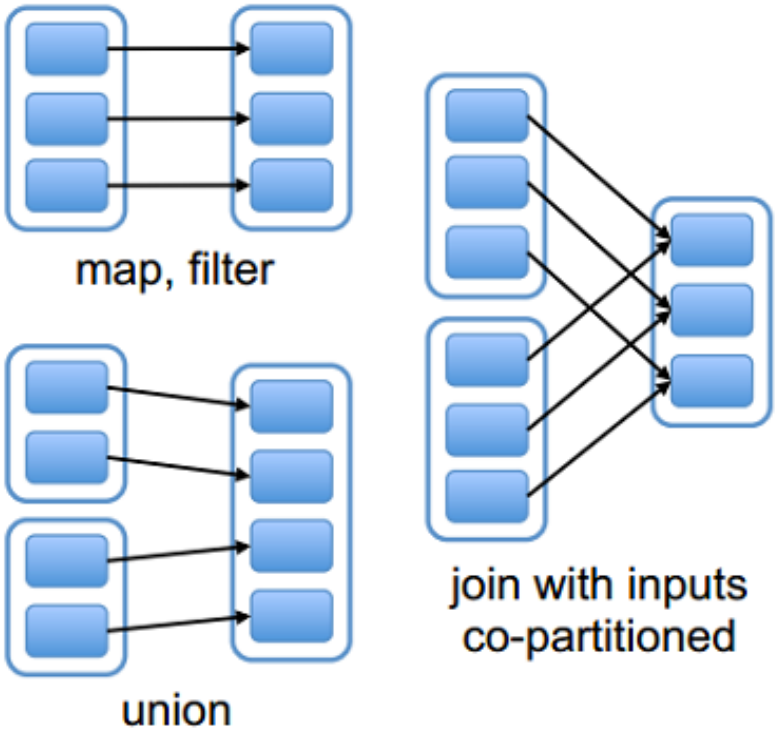


map, filter

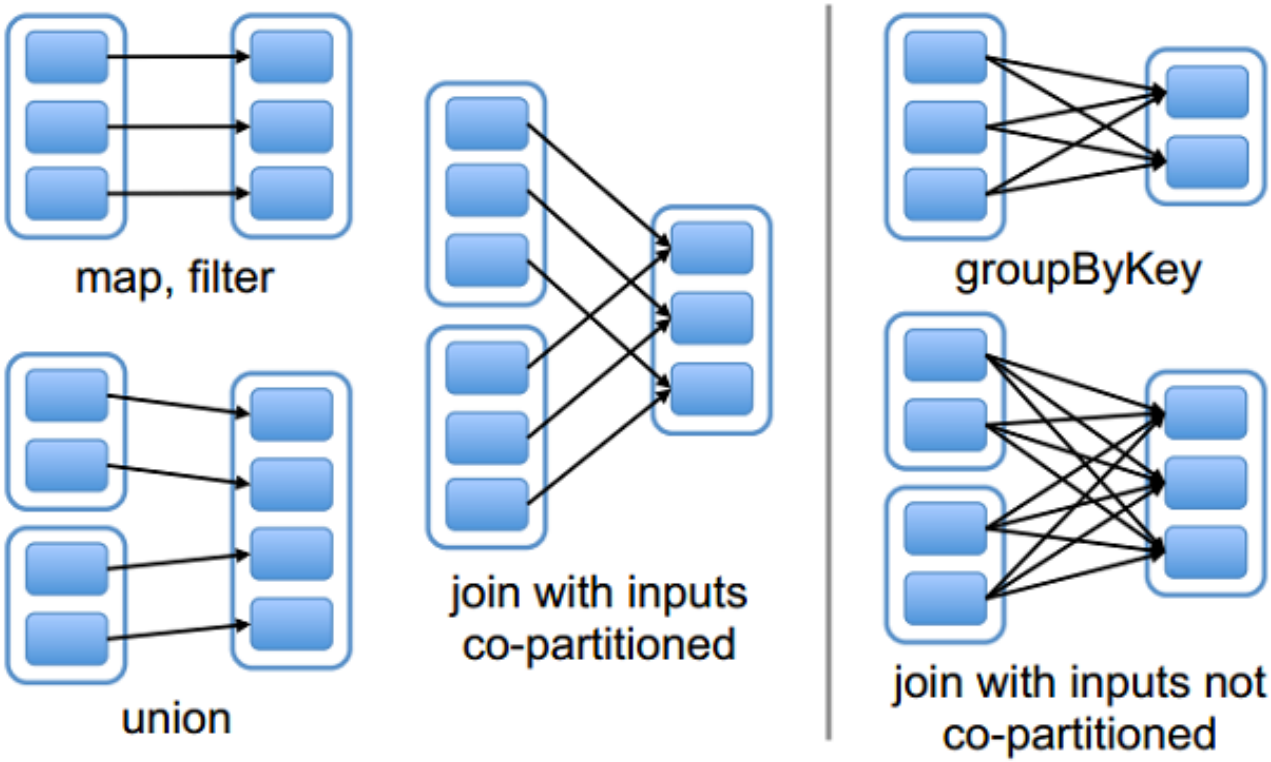


union

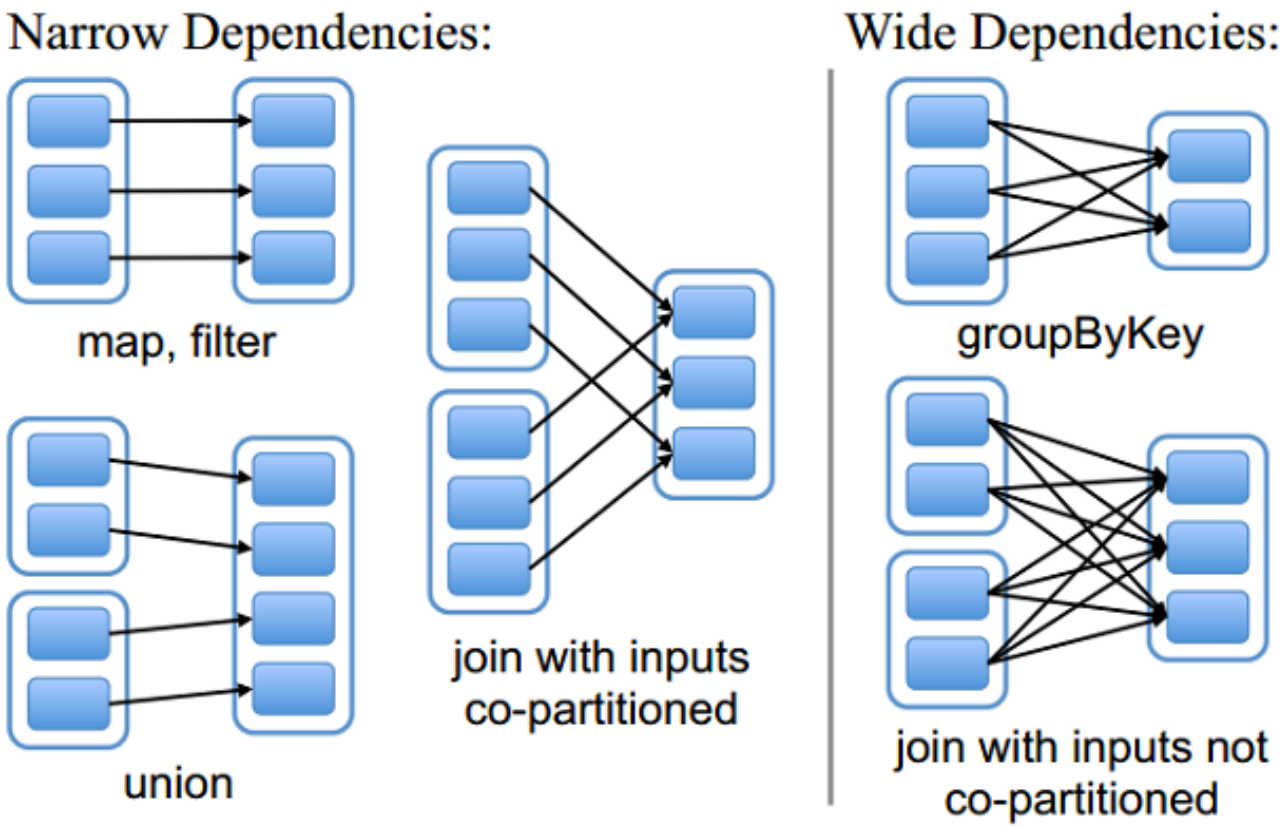
Всегда ли будут линейная цепочка вычислений?



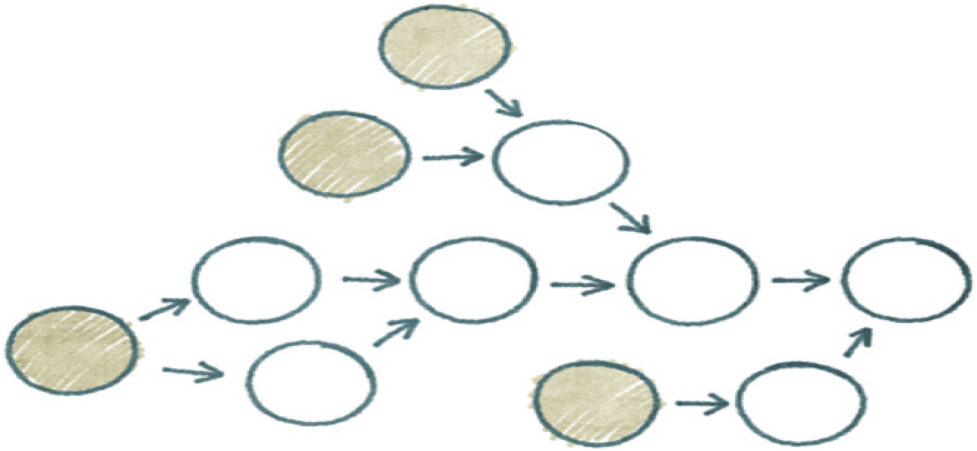
Всегда ли будут линейная цепочка вычислений?



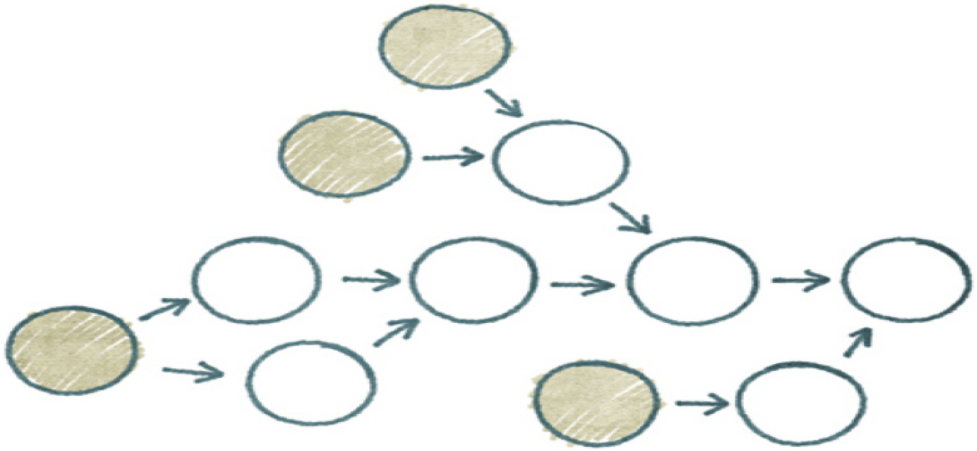
Типы зависимостей для различных операций



Вычисления по принципу направленного графа



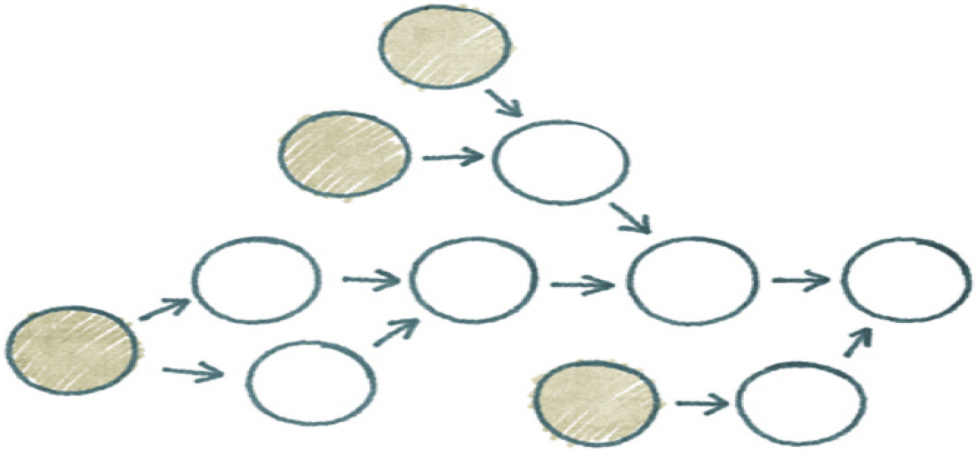
Вычисления по принципу направленного графа



DAG-граф вычислений:

- направленный

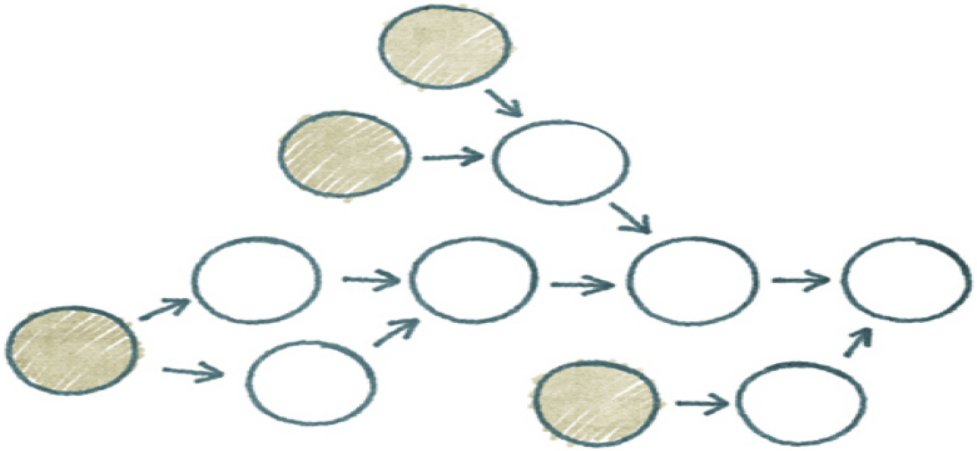
Вычисления по принципу направленного графа



DAG-граф вычислений:

- направленный
- не содержит циклов

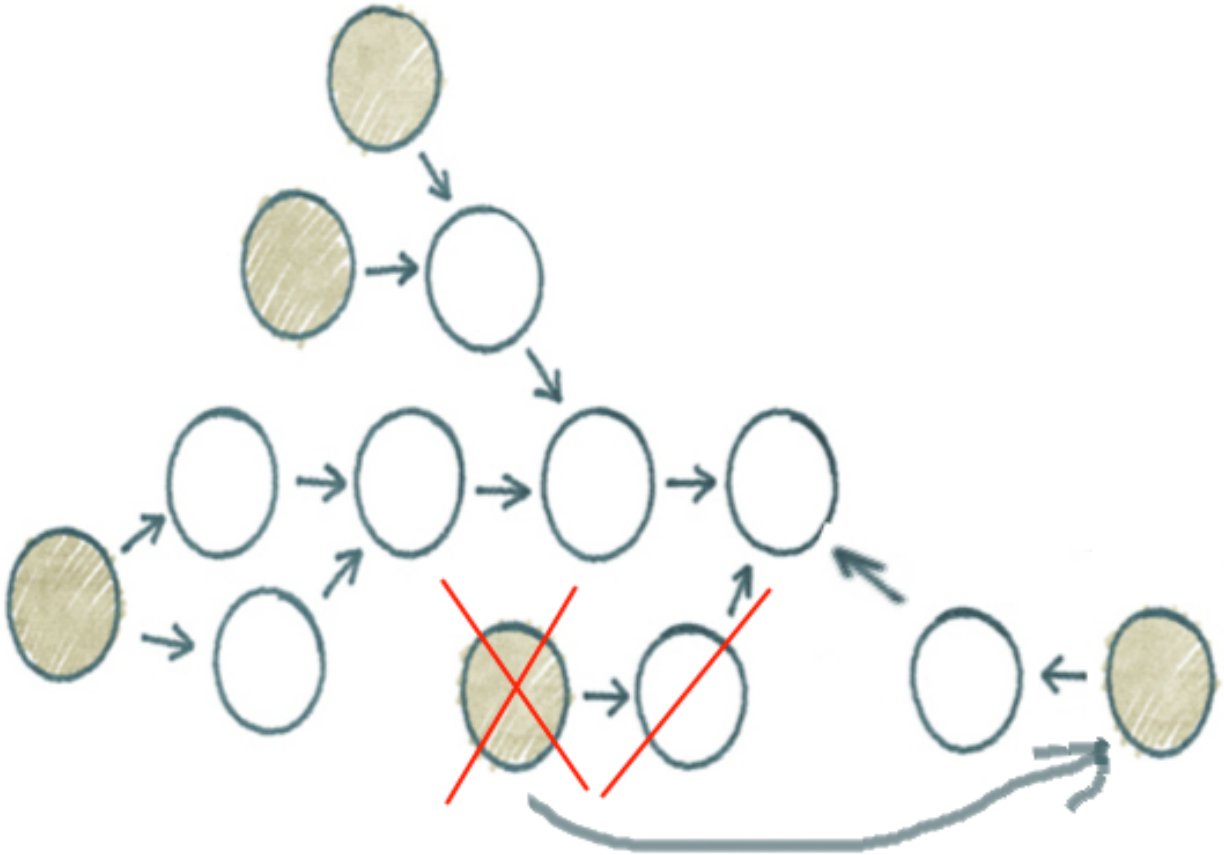
Вычисления по принципу направленного графа



DAG-граф вычислений:

- направленный
- не содержит циклов
- служит основой для ленивых вычислений

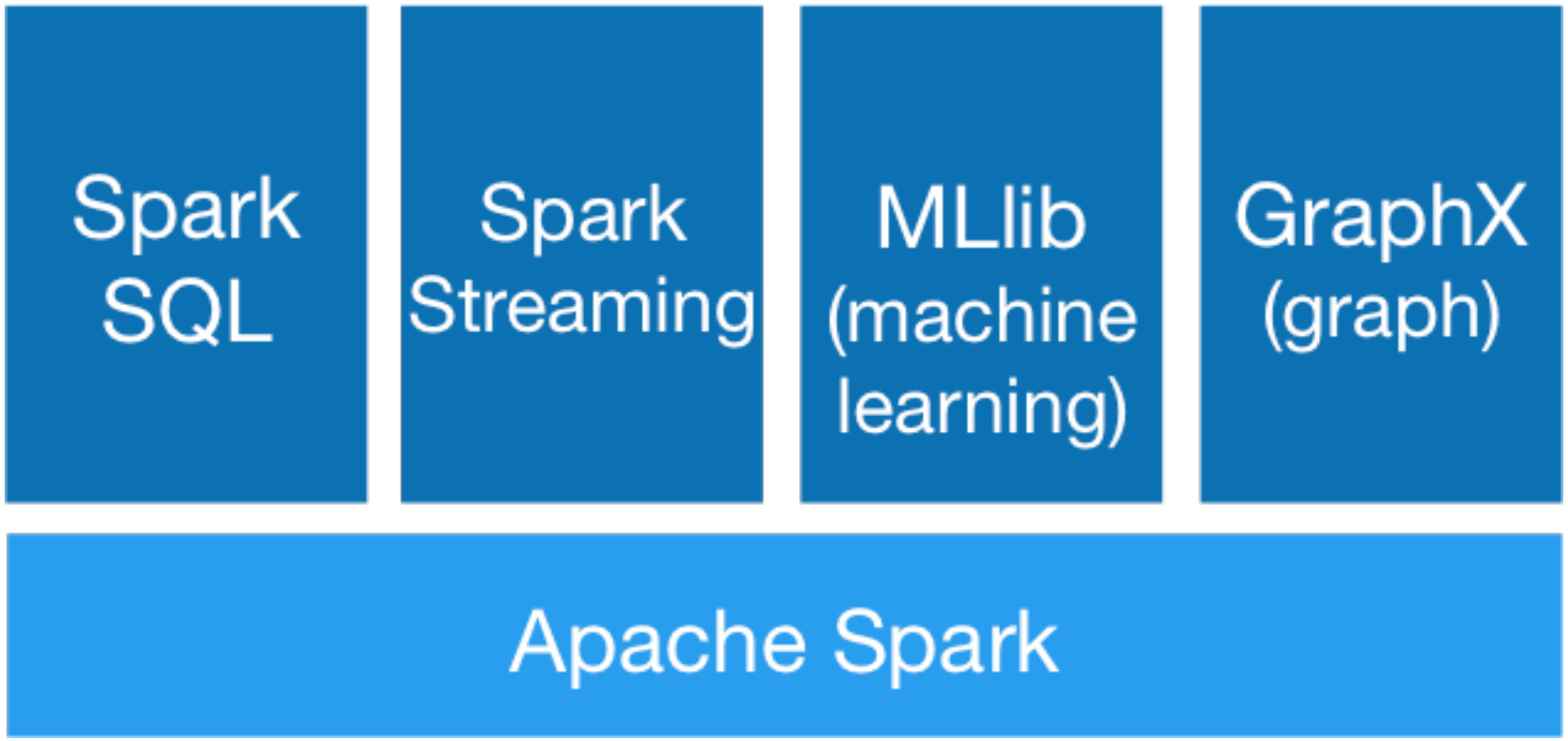
Обеспечение отказоустойчивости



Проект 

- Исследовательский проект AMPLab, UC Berkeley
- Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, 2012
- https://spark.apache.org/

Экосистема Apache Spark



Популярность Apache Spark

- с февраля 2014 года первичный проект Apache
- Convia Inc, video company (Ad-hoc queries)
- Researchers in Berkeley, Mobile Millenium project (estimate road traffic congestion)
- Monarch project in Berkeley (social network spam classification)

Элегантность Apache Spark

- интерактивный командный интерпретатор
- возможность разработки на Java,Python,Scala
- простота реализации задач

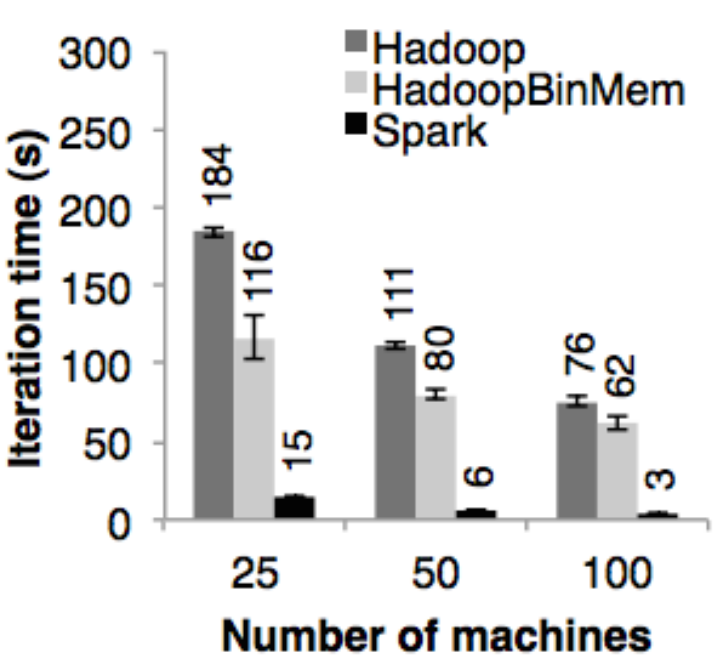
Элегантность Apache Spark

```
>>> file = sc.textFile("hdfs://user/shtokhov/data.txt")
>>> counts = file.flatMap(lambda line: line.split(" ")) \
        .map(lambda word: (word, 1)) \
        .reduceByKey(lambda a, b: a + b)
>>> counts.saveAsTextFile("hdfs://user/shtokhov/result.txt")
```

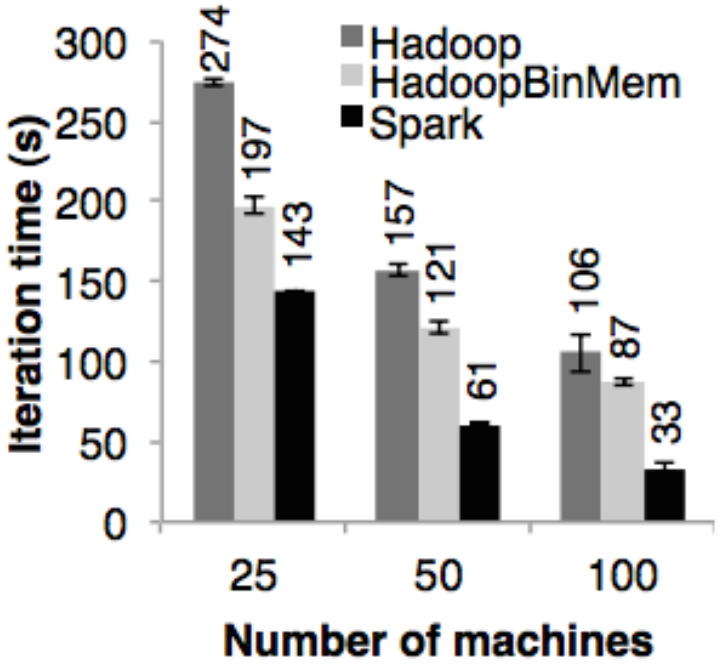
P.S. Пусть к файлам необходимо указывать так:

hdfs://hadoop2-00.yandex.ru/user/shtokhov/data.txt

Скорость работы Apache Spark



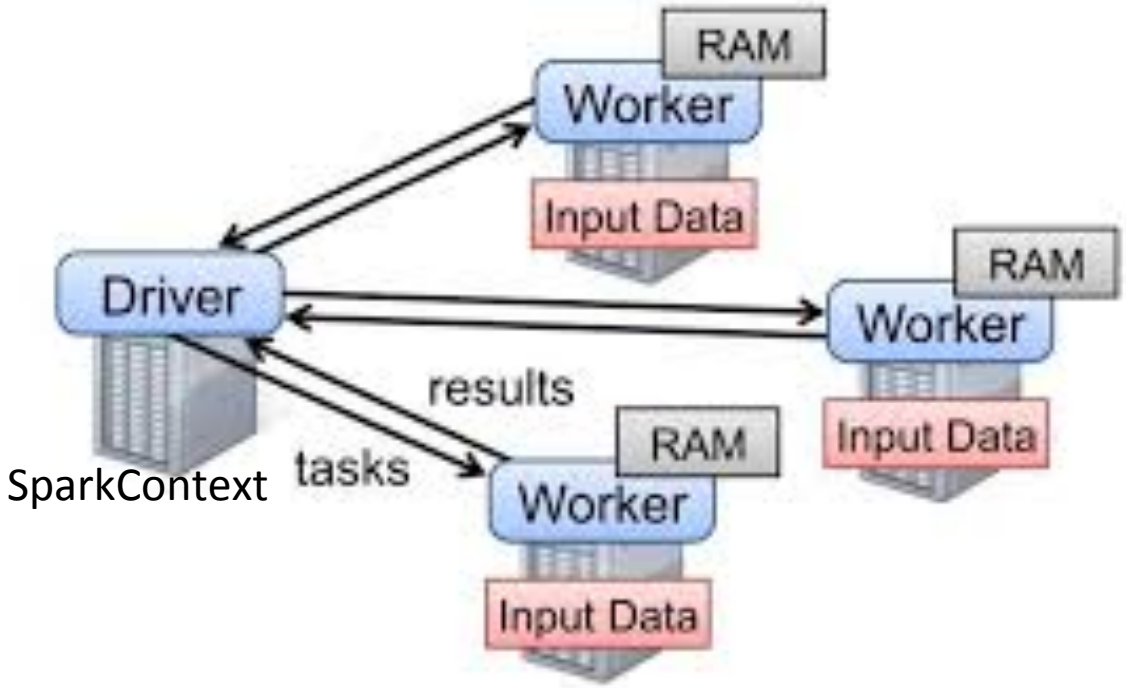
(a) Logistic Regression



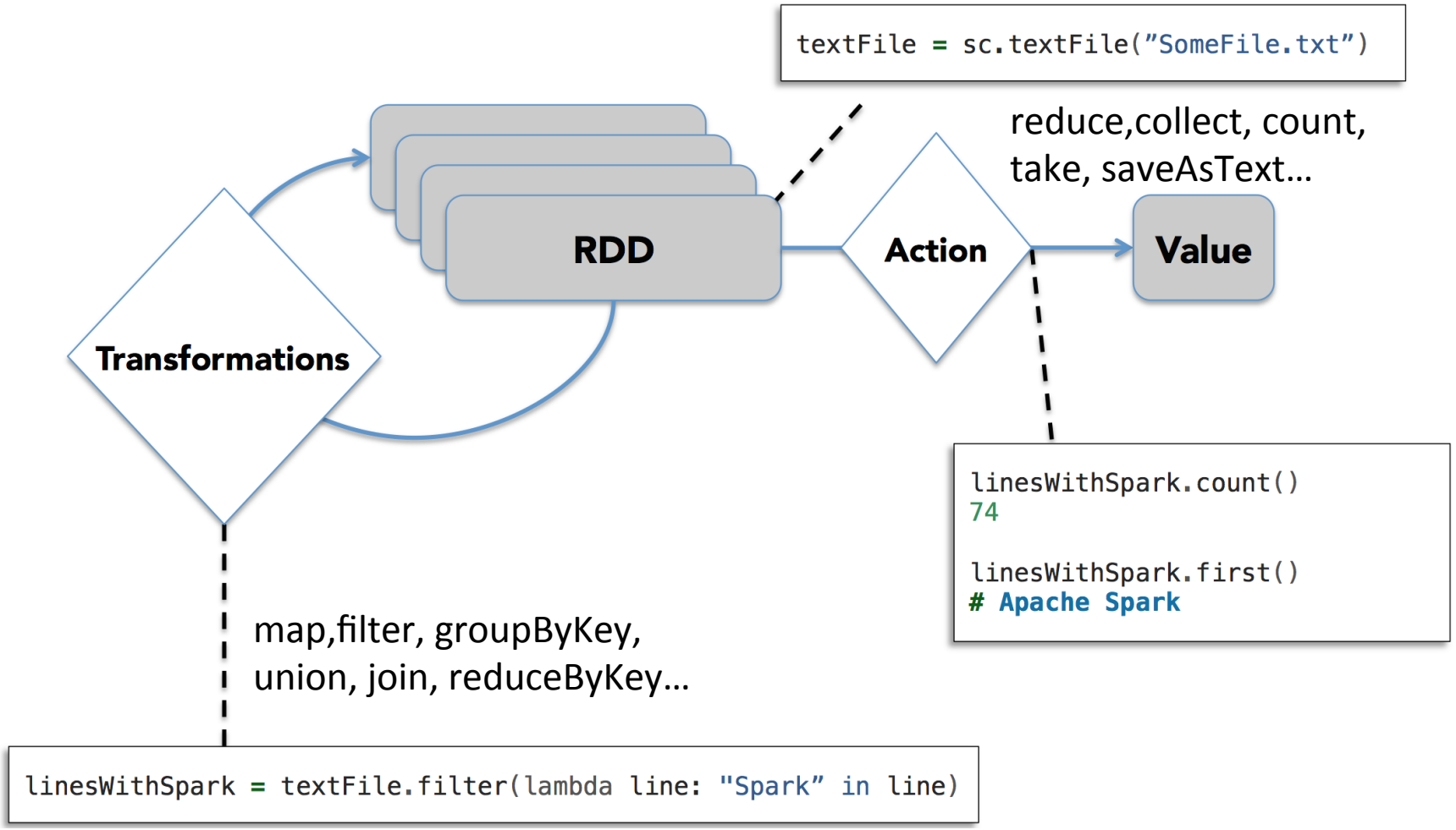
(b) K-Means

Архитектура работы Apache Spark

- от Driver передается информация о partitions и iterator (map step)
- после выполнения результат передается обратно на Driver (reduce step)
- идея локальности выполнения



Операции с RDD. Трансформации и действия



Создание RDD

- По данным

```
>>> lines = sc.textFile("hdfs://...")
>>> lines = sc.parallelize(["pandas", "i like pandas"])
```

- Трансформацией из другого RDD

```
>>> errorsRDD = inputRDD.filter(lambda x: "error" in x)
```

Shared переменные. Broadcast

```
def loadCallSignTable():
        f = open("callsign_tbl_sorted.txt", "r")
        return f.readlines()

def lookupCountry(line):
    pair = line.split(‘\t');
    country = ''
    for value in callsSignTable.value:
        valPair = value.split(',')
        if (pair[0] == valPair[0]):
                country = valPair[1]
    return country


>>>callsSignTable = sc.broadcast(loadCallSignTable())
>>>file = sc.textFile("hdfs://…")
>>>values = file.map(lambda line: lookupCountry(line))
>>>values.take(10)
```

Shared переменные. Accumulator

```
def parseAcc(line):
    global seventeens
    if (line.startswith('17')):
      seventeens += 1
    return line.split('\t’)

>>>seventeens = sc.accumulator(0)
>>>file = sc.textFile("hdfs://…")
>>>usersIds = file.flatMap(parseAcc)
>>>usersIds.saveAsTextFile("hdfs://…")
>>>print "Lines starts 17 count %d" % seventeens.value
```

Примеры кода PageRank

```
>>> sc = SparkContext(”PageRank")
>>> lines = sc.textFile(“hdfs://…”, 1)
>>> links = lines.map(lambda urls: parseNeighbors(urls)).distinct().groupByKey().cache()
>>> ranks = links.map(lambda (url, neighbors): (url, 1.0))
>>> for iteration in xrange(int(sys.argv[2])):
        contribs = links.join(ranks).flatMap(
            lambda (url, (urls, rank)): computeContribs(urls, rank))
        ranks = contribs.reduceByKey(add).mapValues(lambda rank: rank * 0.85 + 0.15)
>>> for (link, rank) in ranks.collect():
        print "%s has rank: %s." % (link, rank)
>>> sc.stop()
```