

Министерство образования и науки Российской Федерации

Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

Ю.С. Белов, С.С. Гришунов

**ОСНОВЫ SPARK. УСТАНОВКА SPARK. ОСНОВНЫЕ КОМАНДЫ
ДЛЯ РАБОТЫ С RDD**

Методические указания к выполнению домашней работы
по курсу «Технологии обработки больших данных»

Калуга - 2018

УДК 004.62
ББК 32.972.5
Б435

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационных технологий и прикладной математики».

Методические указания рассмотрены и одобрены:

- Кафедрой «Программного обеспечения ЭВМ, информационных технологий и прикладной математики» (ФН1-КФ) протокол № 6 от « 12 » января 2018 г.


Зав. кафедрой ФН1-КФ  д.ф.-м.н., профессор Б.М. Логинов

- Методической комиссией факультета ФНК протокол № 1 от « 30 » 01 2018 г.

Председатель методической комиссии факультета ФНК  к.х.н., доцент К.Л. Анфилов

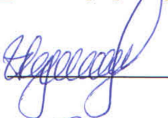
- Методической комиссией КФ МГТУ им.Н.Э. Баумана протокол № 1 от « 06 » 02 2018 г.

Председатель методической комиссии КФ МГТУ им.Н.Э. Баумана

 д.э.н., профессор О.Л. Перерва

Рецензент:

к.т.н., зав. кафедрой ЭИУ2-КФ

 И.В. Чухраев

Авторы

к.ф.-м.н., доцент кафедры ФН1-КФ
ассистент кафедры ФН1-КФ

 Ю.С. Белов
 С.С. Гришунов

Аннотация

Методические указания по выполнению домашней работы по курсу «Технологии обработки больших данных» содержат краткие теоретические сведения о платформе Apache Spark, описание основных модулей, описание установки и настройки Apache Spark. Рассмотрен синтаксис основных команд и примеры скриптов для работы с данными.

Предназначены для студентов 4-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

© Калужский филиал МГТУ им. Н.Э. Баумана, 2018 г.

© Ю.С. Белов, С.С. Гришунов, 2018 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	4
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ.....	5
КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ	6
УСТАНОВКА SPARK	7
РАБОТА С ДАННЫМИ В SPARK	10
ЗАДАНИЕ НА ДОМАШНЮЮ РАБОТУ	15
ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ.....	15
ВАРИАНТЫ ЗАДАНИЙ.....	15
КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ	21
ФОРМА ОТЧЕТА ПО ДОМАШНЕЙ РАБОТЕ.....	21
ОСНОВНАЯ ЛИТЕРАТУРА.....	22
ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА	22

ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой проведения курса «Технологии обработки больших данных» на кафедре «Программное обеспечение ЭВМ, информационные технологии и прикладная математика» факультета фундаментальных наук Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 4-го курса направления подготовки 09.03.04 «Программная инженерия», описание установки и настройки Apache Spark, синтаксис основных команд, примеры скриптов и задание на выполнение домашней работы.

Методические указания составлены для ознакомления студентов с платформой Apache Spark. Для выполнения домашней работы студенту необходимы минимальные знания по программированию на высокоуровневом языке программирования, имеющего интеграцию с Apache Spark (Java, Python, Scala и др.).

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения домашней работы является формирование практических навыков работы с платформой Apache Spark для обработки больших данных.

Основными задачами выполнения домашней работы являются:

1. Изучить основы Apache Spark.
2. Научиться устанавливать и конфигурировать Spark.
3. Уметь работать с RDD.
4. Получить навыки написания программ для обработки больших данных.

Результатами работы являются:

- Входные файлы с данными
- Программа, реализующая рекомендательную систему
- Выходные файлы с результатами вычислений
- Подготовленный отчет

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Apache Spark — программная платформа с открытым исходным кодом для реализации распределённой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop. В отличие от классического обработчика из ядра Hadoop, реализующего двухуровневую концепцию MapReduce с дисковым хранилищем, использует специализированные примитивы для рекуррентной обработки в оперативной памяти, благодаря чему позволяет получать значительный выигрыш в скорости работы для некоторых классов задач, в частности, возможность многократного доступа к загруженным в память пользовательским данным делает библиотеку привлекательной для алгоритмов машинного обучения.

Spark в настоящее время предоставляет API для Java, Scala, Python, R. Изначально написан на Scala, впоследствии добавлена существенная часть кода на Java для предоставления возможности написания программ непосредственно на Java.

Состоит из ядра и нескольких расширений, таких как:

1. Spark SQL (позволяет выполнять SQL-запросы над данными)
2. Spark Streaming (надстройка для обработки потоковых данных)
3. Spark MLlib (набор библиотек машинного обучения)
4. GraphX (предназначено для распределённой обработки графов).

Может работать как в среде кластера Hadoop под управлением YARN, так и без компонентов ядра Hadoop, поддерживает несколько распределённых систем хранения — HDFS, OpenStack Swift, NoSQL-СУБД Cassandra, Amazon S3.

УСТАНОВКА SPARK

Необходимое ПО

Для установки Spark необходима операционная система Linux. Далее будет рассмотрена установка и настройка Spark 2.2.0 под ОС Ubuntu 14.0.

Для работы потребуется следующее [программное обеспечение](#): Java 8+, Python 2.7+/3.4+

Создание учетной записи для Spark

Для запуска Spark будет использоваться отдельная учетная запись Linux. Этот шаг не является обязательным. Создадим группу spark и добавим в нее пользователя sparkusr, также предоставим новому пользователю права sudo, для этого необходимо выполнить команды:

```
sudo addgroup spark
sudo adduser --ingroup spark sparkusr
sudo usermod -aG sudo sparkusr
```

Предполагается, что все дальнейшие действия будут выполняться от только что созданного пользователя.

Настройка SSH

Сгенерируем новый ssh ключ:

```
ssh-keygen -t rsa -P ""
```

Добавим созданный ключ в список авторизованных:

```
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

Для проверки подключимся к localhost:

```
hduser@ubuntu:~$ ssh localhost
```

```
The authenticity of host 'localhost (::1)' can't be established.
```

RSA key fingerprint is d7:87:25:47:ae:02:00:eb:1d:75:4f:bb:44:f9:36:26.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts

Распаковка

Скачать файлы Spark можно с официального сайта, либо выполнив следующую команду:

```
sudo wget http://apache-mirror.rbc.ru/pub/apache/spark/spark-2.2.0/  
spark-2.2.0-bin-hadoop2.7.tgz
```

Далее нужно распаковать архив и переместить файлы в каталог /usr/local/spark:

```
sudo mv spark-2.2.0-bin-hadoop2.7.tgz /usr/local/  
cd /usr/local/  
sudo tar xzf spark-2.2.0-bin-hadoop2.7.tgz  
sudo mv spark-2.2.0-bin-hadoop2.7 spark
```

Также необходимо дать пользователю sparkusr права создателя на директорию:

```
sudo chown -R sparkusr:spark spark
```

Настройка переменных окружения

В файл \$HOME/.bashrc нужно добавить следующие переменные:

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64/jre/  
export SPARK_HOME=/usr/local/spark  
export PATH=$SPARK_HOME/bin:$PATH
```

После этого можно запустить spark командой:

```
spark/bin/pyspark
```


По выполнению этой команды в терминале откроется оболочка для ввода spark-команд (рис. 1).

```
Welcome to  

██████ ██████████  

██████ ██████████ version 2.2.0  

██████ ██████████  

Using Python version 2.7.12 (default, Nov 19 2016 06:48:10)  

SparkSession available as 'spark'.  

>>>
```

Рис. 1 – Окно ввода spark-команд

Для выхода из режима ввода команд необходимо ввести команду `exit()`

РАБОТА С ДАННЫМИ В SPARK

RDD

В Spark вводится концепция RDD (*Resilient Distributed Dataset*, устойчивый распределенный набор данных) – неизменяемая отказоустойчивая распределенная коллекция объектов, которые можно обрабатывать параллельно. В RDD могут содержаться объекты любых типов; RDD создается путем загрузки внешнего набора данных или распределения коллекции из основной программы.

В RDD поддерживаются операции двух типов (рис. 2):

1. Трансформации – это операции (например, отображение, фильтрация, объединение и т.д.), совершаемые над RDD; результатом трансформации становится новый RDD, содержащий ее результат.
2. Действия – это операции (например, редукция, подсчет и т.д.), возвращающие значение, получаемое в результате некоторых вычислений в RDD.

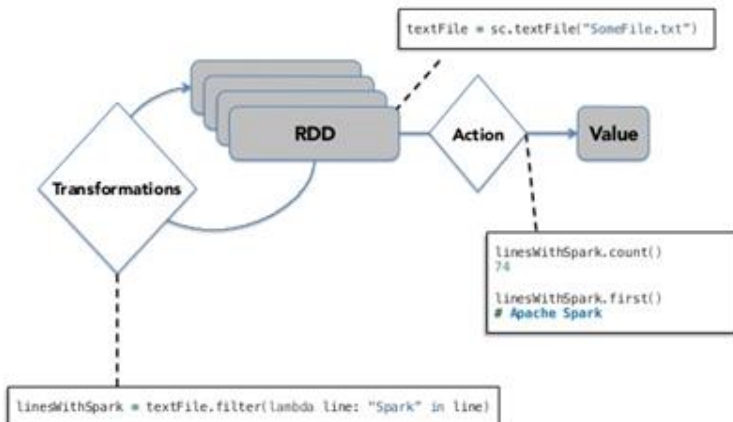


Рис. 2 – Операции над RDD

Трансформации

Результатом применения данной операции к RDD является новый RDD. Как правило, это операции, которые каким-либо образом преобразовывают элементы данного dataset. Вот неполный список самых распространенных преобразований, каждое из которых возвращает новый RDD (полный список представлен в документации Apache Spark):

- **.map(function)** — применяет функцию function к каждому элементу RDD
- **.filter(function)** — возвращает все элементы RDD, на которых функция function вернула истинное значение
- **.distinct([numTasks])** — возвращает RDD, который содержит уникальные элементы исходного RDD

Действия

Действия применяются тогда, когда необходимо материализовать результат — как правило, сохранить данные на диск, либо вывести часть данных в консоль. Вот список самых распространенных действий, которые можно применять над RDD:

- **.saveAsTextFile(path)** — сохраняет данные в текстовый файл (в hdfs, на локальную машину или в любую другую поддерживаемую файловую систему — полный список можно посмотреть в документации)
- **.collect()** — возвращает элементы RDD в виде массива. Как правило, это применяется в случаях, когда данных в RDD уже мало (применены различные фильтры и преобразования) — и необходима визуализация, либо дополнительный анализ данных.
- **.take(n)** — возвращает в виде массива первые n элементов RDD
- **.count()** — возвращает количество элементов в RDD
- **.reduce(function)** — знакомая операция для тех, кто знаком с MapReduce. Из механизма этой операции следует, что функция function (которая принимает на вход 2 аргумента возвращает одно значение) должна быть обязательно коммутативной и ассоциативной.

Spark Shell

[Spark Shell](#) позволяют обрабатывать данные, находящиеся на разных узлах кластера, и Spark автоматически распределяет эту обработку.

При запуске Spark необходимо создать **SparkContext** (это объект, который отвечает за реализацию более низкоуровневых операций с кластером), который при запуске **Spark Shell** создается автоматически и доступен сразу (объект **sc**)

Для Python используется оболочка \$SPARK_INSTALL/bin/pyspark, для Scala \$SPARK_INSTALL/bin/spark-shell

Загрузка данных

Загружать данные в Spark можно двумя путями:

1. Непосредственно из локальной программы с помощью функции **.parallelize(data)**:

```
>>> data = [1,2,3,4,5]
>>> myRDD = sc.parallelize(data)
```

2. Из поддерживаемых хранилищ (например, HDFS) с помощью функции **.textFile(path)**:

```
>>> myRDD = sc.textFile("hdfs://...")
```

Пример

Рассмотрим стандартный пример для обработки больших данных – приложение для подсчета количества слов. Создадим файл example.py:

```
import sys
from pyspark.sql import SparkSession
spark = SparkSession\
    .builder\
    .appName("WordCount")\
    .getOrCreate()
```

```

text_file = spark.sparkContext.textFile("../file.txt")
counts = text_file.flatMap(lambda line: line.split(",")) \
    .map(lambda word: (word, 1)) \
    .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("./result")
spark.stop()

```

Данный код не только легче и проще писать и читать, но также он выполняется на кластере гораздо быстрее, чем стандартный Hadoop MapReduce, т.к. Hadoop MapReduce записывает промежуточные результаты каждой операции на диск, а следующая операция по новой их считывает – работа с дисками является самым узким местом Hadoop MapReduce – в то время как Apache Spark, минимизирует количество операций чтения/записи на диск и сохраняет промежуточные результаты в оперативной памяти серверов.

Рассмотрим более подробно использованные команды:

- `text_file = spark.sparkContext.textFile("../file.txt")` – [загрузка данных](#) из файла `file.txt`
- `text_file.flatMap(lambda line: line.split(","))` – разбиение строки на слова
- `.map(lambda word: (word, 1))` – создаем пары ключ-значение, каждому ключу присваиваем значение 1
- `.reduceByKey(lambda a, b: a + b)` – объединяем пары с одинаковыми ключами, складывая значения
- `counts.saveAsTextFile("./result")` – результат [сохраняем](#) в файл директории `result`

Для запуска необходимо команду `spark-submit` с указанием скрипта и файла с входными данными:

```
$SPARK_INSTALL/bin/spark-submit example.py file.txt
```

Результаты выполнения скрипта будут сохранены в файл bin/result/part-00000.

При содержимом файла file.txt:

sun, fun, cat, dog, sun, cat

fun, fun, fun, fun, fun, fun

Будет получен следующий результат:

(u'fun', 7)

(u'sun', 2)

(u'dog', 1)

(u'cat', 2)

ЗАДАНИЕ НА ДОМАШНЮЮ РАБОТУ

Написать скрипт для платформы Apache Spark для решения задачи, указанной в варианте. В качестве входных текстовых файлов можно использовать книги в txt формате из библиотеки Project Gutenberg: <https://www.gutenberg.org>.

Список стоп-слов: <http://xpro6.com/wp-content/uploads/2015/01/stop-wordlist.csv>

ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ

Приложение может быть реализовано на любом языке высокого уровня, имеющем интеграцию с платформой Apache Spark (Java, Python, Scala и др.). В качестве источника данных можно пользоваться как файлами, размещенными в HDFS, так и файлами в локальной файловой системе.

ВАРИАНТЫ ЗАДАНИЙ

1. Подсчитать средний рейтинг фильма. Входный файл рейтингов имеет формат:

userId, movieId, rating, timestamp

Выполнить операцию объединения с файлом, содержащим названия фильмов. Данный файл имеет формат:

movieId, title, genres

Если для фильма указано несколько жанров, оставить только первый. Сгруппировать записи по жанру и подсчитать средний рейтинг жанра. Результат сохранить в файл:

genre, av_rating

Исходные файлы:

[rating.csv](#)

[movies.csv](#)

2. Подсчитать вероятность взаимного появления двух слов в предложении. Если предложение состоит из слов “А В С”, то взаимно появляющимися парами будут:

(А, В)

(А, С)

(В, А)

(В, С)

(С, А)

(С, В)

Для оценки вероятности взаимного появления использовать метрику PMI (pointwise mutual information):

$$PMI(x, y) = \ln \frac{p(x, y)}{p(x)p(y)},$$

где $p(x, y)$ – частота взаимного появления двух слов, $p(x)$, $p(y)$ – частота появления слова в отдельности.

Для реализации использовать подход «пар» (pairs approach). То есть, результат будет представлен в виде пары ключ-значение, где ключом будет пара слов (a,b), а значением – $PMI(a,b)$.

Из результата должны быть удалены стоп-слова

3. Подсчитать вероятность взаимного появления двух слов в предложении. Если предложение состоит из слов “А В С”, то взаимно появляющимися парами будут:

(А, В)

(А, С)

(В, А)

(В, С)

(С, А)

(С, В)

Для оценки вероятности взаимного появления использовать метрику PMI (pointwise mutual information):

$$PMI(x, y) = \ln \frac{p(x, y)}{p(x)p(y)}$$

где $p(x, y)$ – частота взаимного появления двух слов, $p(x)$, $p(y)$ – частота появления слова в отдельности.

Для реализации использовать подход «полосы» (stripe approach). То есть, результат будет представлен в виде пары ключ-значение, где ключом будет слово a , а значением – список-хэш, содержащий все слова s с которыми есть взаимное появление слова a в файле (ключ – слово b , а значение – $PMI(a,b)$).

Из результата должны быть удалены стоп-слова.

4. Подсчитать средний рейтинг фильма. Входный файл рейтингов имеет формат:

userId, movieId, rating, timestamp

Выполнить операцию объединения с файлом, содержащим названия фильмов. Данный файл имеет формат:

movieId, title, genres

Результат должен быть сохранен в файле в формате:

movieId, title, av_rating

Объединение выполнять согласно подходу Job-side join. В результате должны быть представлены 20 фильмов с самым высоким средним рейтингом

Исходные файлы:

rating.csv

movies.csv

5. Для заданного списка слов (поисковый запрос) среди множества из N документов найти 3 самых похожих. Для меры схожести использовать метрику $TF \cdot IDF$. Результат не должен учитывать стоп-слова.

Метрика $TF \cdot IDF$ для слова s в документе d вычисляется:

$$TF \cdot IDF = (1 + \ln TF) * \ln \frac{N}{IDF}$$

где N – общее количество документов, TF – количество встречения слова s в документе d , IDF – количество документов, в которых встречается слово s .

Возможные этапы реализации:

Стадия 1:

Подсчет всех слов в каждом документе (по аналогии с WordCount).

Стадия 2:

Подсчет tf-idf каждого слова для каждого документа. Входными данными являются результаты стадии 1.

Стадия 3:

Подсчет нормализованных значений TF·IDF. Входными данными являются результаты стадии 2.

Стадия 4:

Подсчет схожести документов с поисковым запросом. Входными данными являются результаты стадии 3 и файл с запросом. Для каждого документа сложить все значения TF·IDF для слов, присутствующих в запросе.

Стадия 5:

Отбор 3 самых подходящих документов.

6. Для каждого пользователя подсчитать среднюю оценку, которую он поставил всем фильмам. Входный файл рейтингов имеет формат:

userId, movieId, rating, timestamp

Не учитывать пользователей, которые поставили менее 20 оценок.

Результат отсортировать по убыванию рейтинга фильма и сохранить в файле в формате:

userId, av_rating

Исходные файлы:

rating.csv

7. Для двух текстовых файлов подсчитать количество слов, которые встречаются одновременно и в первом и во втором файле. Результат сохранить в файл в виде пар ключ-значение, где ключ – количество общих слов, значение – само слово.

8. Реализовать алгоритм PageRank. Входные данные – ориентированный граф (V, E) , где вершины V представляют web-страницу, а ребра E – ссылки на страницы. Граф представлен в виде списка смежности – каждой странице соответствует список, состоящий из страниц, на которые есть ссылки с данной страницы. Рассчитать значение PageRank для каждой страницы, для расчета использовать как минимум 10 итераций.

$$PR(v) = \frac{1-d}{N} + d \sum_{i=1}^n \frac{PR(u_i)}{C(u_i)}$$

где $u_1, u_2 \dots$ – страницы, на которые есть ссылки со страницы v ,

$C(u_i)$ – Количество ребер, выходящих из вершины u_i ,

d – константа, принять равной 0,85

N – общее количество страниц.

Начальное значение PR для всех страниц задать равным 1.

Вывести результат в отсортированном по убыванию значения PR списке страниц.

Во входном файле первое число в каждой строке – id страницы, далее перечислены id страниц, на которые есть ссылка с исходной страницы.

Входной файл:

<http://lintool.github.io/Cloud9/docs/exercises/sample-medium.txt>

9. Реализовать алгоритм кластеризации K-means.

Стадия map связывает каждую точку с ближайшим центром. Стадия Reduce выполняет пересчет положения центра кластера. Условие остановки алгоритма – точки перестали перемещаться между кластерами, либо число итераций достигло заданного максимального значения.

Формат входного файла: в каждой отдельной строке координаты точки

x, y

Формат выходного файла: каждая отдельная строка – описание кластера, x_c , y_c – координаты центра кластера, далее перечислены все точки, входящие в кластер

$((x_c, y_c) ((x, y), (x, y), \dots))$

10. Реализовать алгоритм поиска в ширину на графе. Граф представлен в виде списка смежности – каждой вершине соответствует список, состоящий из вершин, до которых есть ребра, выходящие из данной вершины. Результат представить в виде списка вершин в порядке их обхода алгоритмом.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Дайте определение Apache Spark.
2. Раскройте преимущества Apache Spark перед Hadoop MapReduce?
3. Перечислите модули, из которых состоит Apache Spark.
4. Опишите основные этапы установки Apache Spark.
5. Приведите команду для запуска Spark Shell.
6. Дайте определение RDD.
7. Перечислите основные типы операций над RDD.
8. Перечислите основные трансформации, производимые над RDD.
9. Перечислите основные действия, выполняемые над RDD.
10. Приведите команды для загрузки данных в RDD.

ФОРМА ОТЧЕТА ПО ДОМАШНЕЙ РАБОТЕ

На выполнение домашней работы отводится 10 часов.

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания (вариант), этапы выполнения работы (со скриншотами), результаты выполнения работы. выводы.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Федин Ф.О. Анализ данных. Часть 1. Подготовка данных к анализу [Электронный ресурс] : учебное пособие / Ф.О. Федин, Ф.Ф. Федин. — Электрон. текстовые данные. — М. : Московский городской педагогический университет, 2012. — 204 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/26444.html>
2. Федин Ф.О. Анализ данных. Часть 2. Инструменты Data Mining [Электронный ресурс] : учебное пособие / Ф.О. Федин, Ф.Ф. Федин. — Электрон. текстовые данные. — М. : Московский городской педагогический университет, 2012. — 308 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/26445.html>
3. Чубукова, И.А. Data Mining [Электронный ресурс] : учеб. пособие — Электрон. дан. — Москва : , 2016. — 470 с. — Режим доступа: <https://e.lanbook.com/book/100582>. — Загл. с экрана.
4. Воронова Л.И. Big Data. Методы и средства анализа [Электронный ресурс] : учебное пособие / Л.И. Воронова, В.И. Воронов. — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2016. — 33 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/61463.html>

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

5. Волкова Т.В. Разработка систем распределенной обработки данных [Электронный ресурс] : учебно-методическое пособие / Т.В. Волкова, Л.Ф. Насейкина. — Электрон. текстовые данные. — Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2012. — 330 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/30127.html>
6. Кухаренко Б.Г. Интеллектуальные системы и технологии [Электронный ресурс] : учебное пособие / Б.Г. Кухаренко. — Электрон. текстовые данные. — М. : Московская государственная академия водного транспорта, 2015. — 116 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/47933.html>

7. Воронова Л.И. Интеллектуальные базы данных [Электронный ресурс] : учебное пособие / Л.И. Воронова. — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2013. — 35 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/63324.html>
8. Николаев Е.И. Базы данных в высокопроизводительных информационных системах [Электронный ресурс] : учебное пособие / Е.И. Николаев. — Электрон. текстовые данные. — Ставрополь: Северо-Кавказский федеральный университет, 2016. — 163 с. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/69375.html>

Электронные ресурсы:

9. <http://hadoop.apache.org/> (англ.)
10. <https://spark.apache.org/> (англ.)