Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Отчет по лабораторной работе MapReduce по дисциплине

«Технология обработки больших данных»

ИСПОЛНИТЕЛЬ:

Березин И.С.

Группа ИУ5-43М

_____

"__"_____2020 г.

Москва 2020

Задание:

1) **Сгенерировать текстовый файл размером 8ГБ, каждое слово которого содержит от 3 до 6 символом из латинского алфавита в произвольном порядке**
2) **Написать свой простейший MapReduce с разбивкой по частям большого файла при его считывании, подсчитать количество совпадений слова в исходном файле.**

MapReduce – это модель распределённых вычислений от компании Google, используемая в технологиях Big Data для параллельных вычислений над очень большими (до нескольких петабайт) наборами данных в компьютерных кластерах, и фреймворк для вычисления распределенных задач на узлах (node) кластера [1].

Назначение и области применения

MapReduce можно по праву назвать главной технологией Big Data, т.к. она изначально ориентирована на параллельные вычисления в распределенных кластерах. Суть MapReduce состоит в разделении информационного массива на части, параллельной обработки каждой части на отдельном узле и финального объединения всех результатов.

Программы, использующие MapReduce, автоматически распараллеливаются и исполняются на распределенных узлах кластера, при этом исполнительная система сама заботится о деталях реализации (разбиение входных данных на части, разделение задач по узлам кластера, обработка сбоев и сообщение между распределенными компьютерами). Благодаря этому программисты могут легко и эффективно использовать ресурсы распределённых Big Data систем.

Технология практически универсальна: она может использоваться для индексации веб-контента, подсчета слов в большом файле, счётчиков частоты обращений к заданному адресу, вычисления объём всех веб-страниц с каждого URL-адреса конкретного хост-узла, создания списка всех адресов с необходимыми данными и прочих задач обработки огромных массивов распределенной информации. Также к областям применения MapReduce

относится распределённый поиск и сортировка данных, обращение графа веб-ссылок, обработка статистики логов сети, построение инвертированных индексов, кластеризация документов, машинное обучение и статистический машинный перевод. Также MapReduce адаптирована под многопроцессорные системы, добровольные вычислительные, динамические облачные и мобильные среды [2].

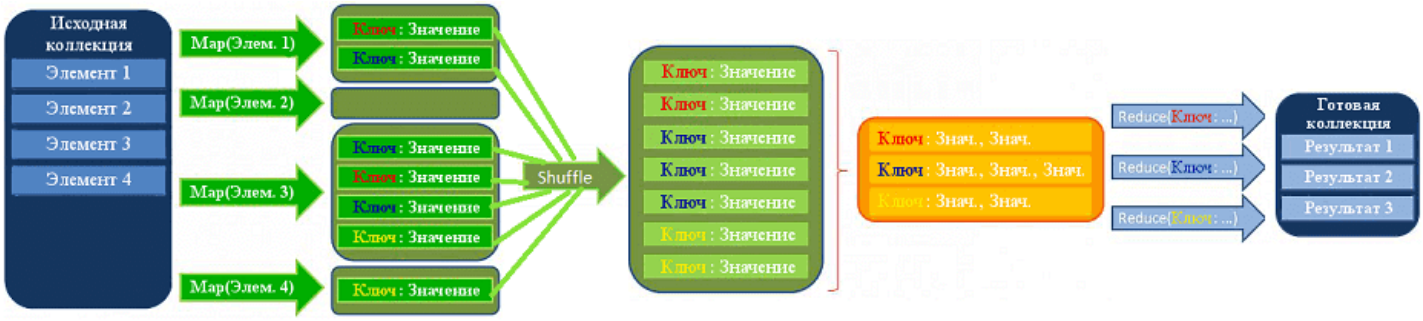
История развития главной технологии Big Data

Авторами этой вычислительной модели считаются сотрудники Google Джеффри Дин (Jeffrey Dean) и Санджай Гемават (Sanjay Ghemawat), взявшие за основу две процедуры функционального программирования: map, применяющая нужную функцию к каждому элементу списка, и reduce, объединяющая результаты работы map [3]. В процессе вычисления множество входных пар ключ/значение преобразуется в множество выходных пар ключ/значение [4].

Изначально название MapReduce было запатентовано корпорацией Google, но по мере развития технологий Big Data стало общим понятием мира больших данных. Сегодня множество различных коммерческих, так и свободных продуктов, использующих эту модель распределенных вычислений: Apache Hadoop, Apache CouchDB, MongoDB, MySpace Qizmt и прочие Big Data фреймворки и библиотеки, написанные на разных языках программирования [2]. Среди других наиболее известных реализаций MapReduce стоит отметить следующие [5]:

- Greenplum — коммерческая реализация с поддержкой языков Python, Perl, SQL и пр.;
- GridGain — бесплатная реализация с открытым исходным кодом на языке Java;
- Phoenix — реализация на языке C с использованием разделяемой памяти;
- MapReduce реализована в графических процессорах NVIDIA с использованием CUDA;

- Qt Concurrent — упрощённая версия фреймворка, реализованная на C++, для распределения задачи между несколькими ядрами одного компьютера;
- CouchDB использует MapReduce для определения представлений поверх распределённых документов;
- Skynet — реализация с открытым исходным кодом на языке Ruby;
- Disco — реализация от компании Nokia, ядро которой написано на языке Erlang, а приложения можно разрабатывать на Python;
- Hive framework — надстройка с открытым исходным кодом от Facebook, позволяющая комбинировать подход MapReduce и доступ к данным на SQL-подобном языке;
- Qizmt — реализация с открытым исходным кодом от MySpace, написанная на C#;
- DryadLINQ — реализация от Microsoft Research на основе PLINQ и Dryad.



MapReduce — это разделение, параллельная обработка и свертка распределенных результатов

Как устроен MapReduce: принцип работы

Прежде всего, еще раз поясним смысл основополагающих функций вычислительной модели [2]:

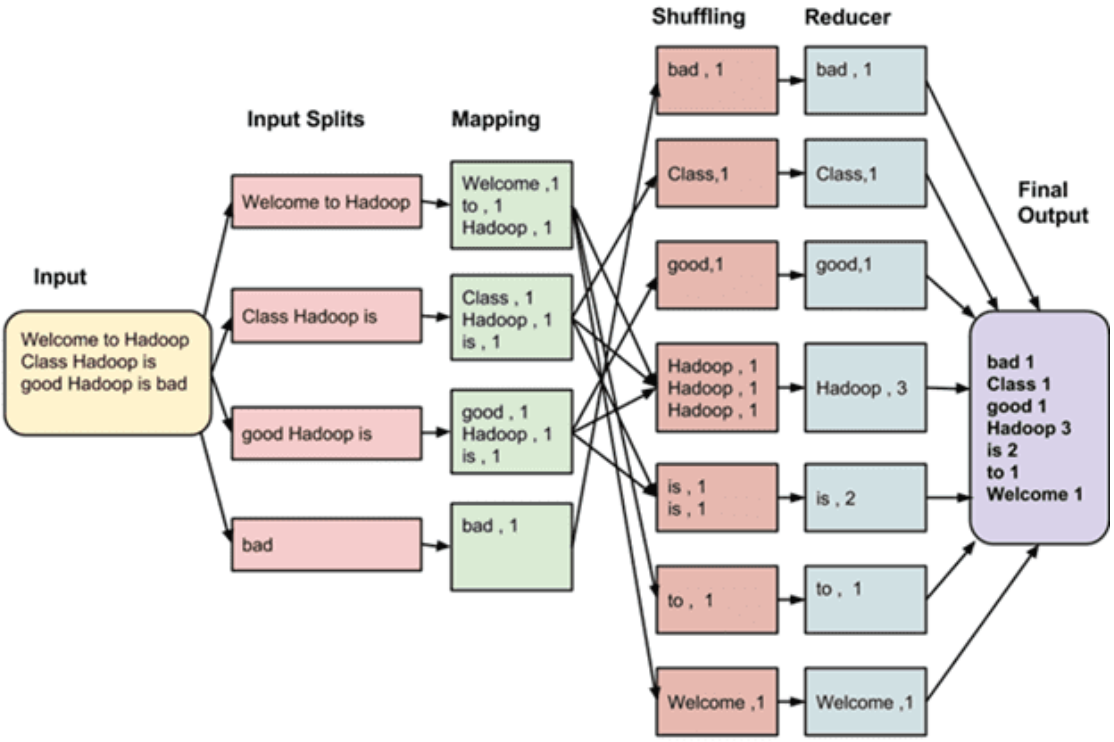
- mapпринимает на вход список значений и некую функцию, которую затем применяет к каждому элементу списка и возвращает новый список;

- reduce (свёртка) — преобразует список к единственному атомарному значению при помощи заданной функции, которой на каждой итерации передаются новый элемент списка и промежуточный результат.

Для обработки данных в соответствии с вычислительной моделью MapReduce следует определить обе эти функции, указать имена входных и выходных файлов, а также параметры обработки.

Сама вычислительная модель состоит из 3-хшаговой комбинации вышеприведенных функций [2]:

1. Map – предварительная обработка входных данных в виде большого список значений. При этом главный узел кластера (master node) получает этот список, делит его на части и передает рабочим узлам (worker node). Далее каждый рабочий узел применяет функцию Map к локальным данным и записывает результат в формате «ключ-значение» во временное хранилище.
2. Shuffle, когда рабочие узлы перераспределяют данные на основе ключей, ранее созданных функцией Map, таким образом, чтобы все данные одного ключа лежали на одном рабочем узле.
3. Reduce – параллельная обработка каждым рабочим узлом каждой группы данных по порядку следования ключей и «склейка» результатов на master node. Главный узел получает промежуточные ответы от рабочих узлов и передаёт их на свободные узлы для выполнения следующего шага. Получившийся после прохождения всех необходимых шагов результат – это и есть решение исходной задачи.



Принцип работы MapReduce

Исходный код программы:

```
import time
import random
import os
from multiprocessing import Pool

file_size = 8 * 1024 * 1024 * 1024


class Profiler(object):
    def __init__(self, name):
        self.name = name

    def __enter__(self):
        self._startTime = time.time()

    def __exit__(self, type, value, traceback):
        print("Elapsed time - {}: {:.3f} sec".format(self.name, time.time() - self._startTime))


def worker(_):
    a = ''
    for _ in range(100_000):
        a += ''.join([random.choice('abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ')
                      for _ in range(random.choice([3, 4, 5, 6]))] + ['/'])
    return a


if __name__ == '__main__':
    with Profiler('good') as p:
        with open('lr1.txt', 'a') as f:
            with Pool(processes=4) as pool:
                while True:
                    for result in pool.imap_unordered(worker, range(1_000)):
                        f.write(result)
                        if os.path.getsize('lr1.txt') >= file_size:
                            break
                    if os.path.getsize('lr1.txt') >= file_size:
                        break
```

```python
import time
from multiprocessing import Pool
from collections import Counter

CHUNK_SIZE = 1024
CHUNK_MIN_SIZE = 1024 - 7


class Profiler(object):
    def __init__(self, name):
        self.name = name

    def __enter__(self):
        self._startTime = time.time()

    def __exit__(self, type, value, traceback):
        print("Elapsed time - {}: {:.3f} sec".format(self.name, time.time() - self._startTime))


class Chunk:
    @classmethod
    def split(cls, file_name):
        with open(file_name, 'rb') as f:
            chunk_end = f.tell()
            delta = 0
            while True:
                chunk_start = chunk_end
                delta = cls._EOC(f, delta)
                chunk_end = f.tell() - delta
                yield chunk_start, chunk_end
                if delta > 6:
                    break

    @staticmethod
    def _EOC(f, delta):
        f.read(CHUNK_MIN_SIZE - delta)
        chunk_tail = f.read(7)
        return 6 - chunk_tail.rfind(b'/')  # число символов лишнего считывания

    @staticmethod
    def _EOC(f, delta):
        f.read(CHUNK_MIN_SIZE - delta)
        chunk_tail = f.read(7)
        return 6 - chunk_tail.rfind(b'/')  # число символов лишнего считывания

    @staticmethod
    def read(a, d, g):
        g.seek(a)
        return g.read(d-a)

    @staticmethod
    def parse(chunk_string):
        d = dict(Counter(chunk_string.split('/')))
        d.pop('', None)
        return d
```

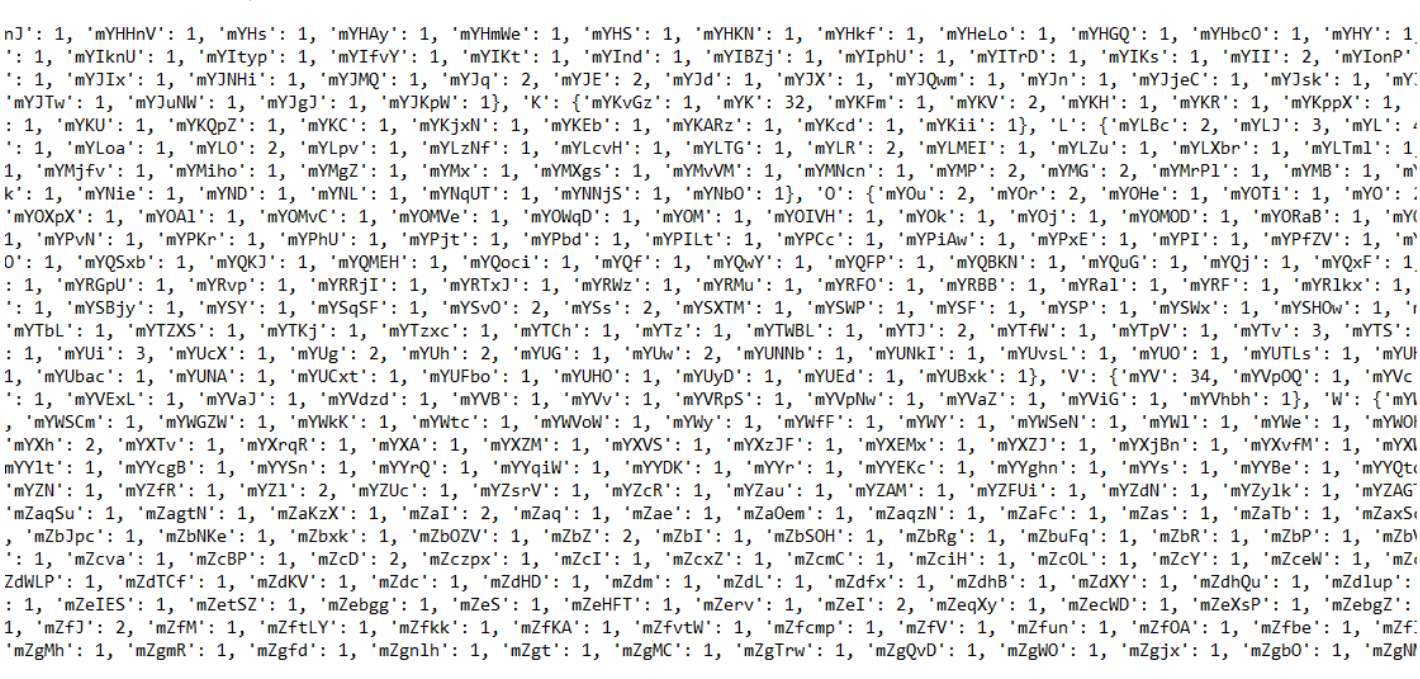
```python
def gen_default_dict():
    d = {}
    for i in 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ':
        d[i] = {}
        for j in 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ':
            d[i][j] = {}
            for k in 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ':
                d[i][j][k] = {}
    return d

def gen_default_dict():
    d = {}
    for i in 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ':
        d[i] = {}
        for j in 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ':
            d[i][j] = {}
            for k in 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ':
                d[i][j][k] = {}
    return d


def worker_map(args):
    with open('lr1.txt', 'r') as g:
        return Chunk.parse(Chunk.read(*args, g))


if __name__ == '__main__':
    with Profiler('lr1') as p:
        with Pool(processes=4) as pool:
            workers = []
            d = gen_default_dict()
            for d2 in pool.imap_unordered(worker_map, Chunk.split('lr1.txt')):
                for key, value in d2.items():
                    if d[key[0]][key[1]][key[2]].get(key):
                        d[key[0]][key[1]][key[2]][key] += value
                    else:
                        d[key[0]][key[1]][key[2]][key] = value
            with open('lr1_result.txt', 'w') as q:
                q.write(str(d))
```

Результаты выполнения:



nJ': 1, 'mYHHnV': 1, 'mYHs': 1, 'mYHAy': 1, 'mYHmWe': 1, 'mYHS': 1, 'mYHKN': 1, 'mYHkf': 1, 'mYHeLo': 1, 'mYHGQ': 1, 'mYHbcO': 1, 'mYHY': 1,
': 1, 'mYIknU': 1, 'mYItyp': 1, 'mYIfvY': 1, 'mYIKt': 1, 'mYInd': 1, 'mYIBZj': 1, 'mYIphU': 1, 'mYITrD': 1, 'mYIKs': 1, 'mYII': 2, 'mYIonP'
': 1, 'mYJIx': 1, 'mYJNHi': 1, 'mYJMQ': 1, 'mYJq': 2, 'mYJE': 2, 'mYJd': 1, 'mYJX': 1, 'mYJQwm': 1, 'mYJn': 1, 'mYJjeC': 1, 'mYJsk': 1, 'mY
'mYJTw': 1, 'mYJuNW': 1, 'mYJgJ': 1, 'mYJKpW': 1}, 'K': {'mYKvGz': 1, 'mYK': 32, 'mYKFm': 1, 'mYKV': 2, 'mYKH': 1, 'mYKR': 1, 'mYKppX': 1,
: 1, 'mYKU': 1, 'mYKQpZ': 1, 'mYKC': 1, 'mYKjxN': 1, 'mYKEb': 1, 'mYKARz': 1, 'mYKcd': 1, 'mYKii': 1}, 'L': {'mYLBc': 2, 'mYLJ': 3, 'mYL': 4
': 1, 'mYLoa': 1, 'mYLO': 2, 'mYLpv': 1, 'mYLzNf': 1, 'mYLcvH': 1, 'mYLTG': 1, 'mYLR': 2, 'mYLMEI': 1, 'mYLZu': 1, 'mYLXbr': 1, 'mYLTml': 1,
1, 'mYMjfv': 1, 'mYMiho': 1, 'mYMgZ': 1, 'mYMx': 1, 'mYMXgs': 1, 'mYMvVM': 1, 'mYMNcn': 1, 'mYMP': 2, 'mYMG': 2, 'mYMrPl': 1, 'mYMB': 1, 'mY
k': 1, 'mYNie': 1, 'mYND': 1, 'mYNL': 1, 'mYNqUT': 1, 'mYNNjS': 1, 'mYNbO': 1}, 'O': {'mYOu': 2, 'mYOr': 2, 'mYOHe': 1, 'mYOTi': 1, 'mYO': 2
'mYOXpX': 1, 'mYOAl': 1, 'mYOMvC': 1, 'mYOMVe': 1, 'mYOWqD': 1, 'mYOM': 1, 'mYOIVH': 1, 'mYOk': 1, 'mYOj': 1, 'mYOMOD': 1, 'mYORaB': 1, 'mYO
1, 'mYPvN': 1, 'mYPKr': 1, 'mYPhU': 1, 'mYPjt': 1, 'mYPbd': 1, 'mYPILt': 1, 'mYPCc': 1, 'mYPiAw': 1, 'mYPxE': 1, 'mYPI': 1, 'mYPfZV': 1, 'mY
O': 1, 'mYQSxb': 1, 'mYQKJ': 1, 'mYQMEH': 1, 'mYQoci': 1, 'mYQf': 1, 'mYQwY': 1, 'mYQFP': 1, 'mYQBKN': 1, 'mYQuG': 1, 'mYQj': 1, 'mYQxF': 1,
: 1, 'mYRGpU': 1, 'mYRvp': 1, 'mYRRjI': 1, 'mYRTxJ': 1, 'mYRWz': 1, 'mYRMu': 1, 'mYRFO': 1, 'mYRBB': 1, 'mYRal': 1, 'mYRF': 1, 'mYRlkx': 1,
': 1, 'mYSBjy': 1, 'mYSY': 1, 'mYSqSF': 1, 'mYSvO': 2, 'mYSs': 2, 'mYSXTM': 1, 'mYSWP': 1, 'mYSF': 1, 'mYSP': 1, 'mYSWx': 1, 'mYSHOw': 1, 'm
'mYTbL': 1, 'mYTZXS': 1, 'mYTKj': 1, 'mYTzxc': 1, 'mYTCh': 1, 'mYTz': 1, 'mYTWBL': 1, 'mYTJ': 2, 'mYTfW': 1, 'mYTpV': 1, 'mYTv': 3, 'mYTS':
: 1, 'mYUi': 3, 'mYUcX': 1, 'mYUg': 2, 'mYUh': 2, 'mYUG': 1, 'mYUw': 2, 'mYUNNb': 1, 'mYUNkI': 1, 'mYUvsL': 1, 'mYUO': 1, 'mYUTLs': 1, 'mYUH
1, 'mYUbac': 1, 'mYUNA': 1, 'mYUCxt': 1, 'mYUFbo': 1, 'mYUHO': 1, 'mYUyD': 1, 'mYUEd': 1, 'mYUBxk': 1}, 'V': {'mYV': 34, 'mYVpOQ': 1, 'mYVc
': 1, 'mYVExL': 1, 'mYVaJ': 1, 'mYVdzd': 1, 'mYVB': 1, 'mYVv': 1, 'mYVRpS': 1, 'mYVpNw': 1, 'mYVaZ': 1, 'mYViG': 1, 'mYVhbh': 1}, 'W': {'mYl
, 'mYWSCm': 1, 'mYWGZW': 1, 'mYWkK': 1, 'mYWtc': 1, 'mYWVoW': 1, 'mYWy': 1, 'mYWfF': 1, 'mYWY': 1, 'mYWSeN': 1, 'mYWl': 1, 'mYWe': 1, 'mYWOh
'mYXh': 2, 'mYXTv': 1, 'mYXrqR': 1, 'mYXA': 1, 'mYXZM': 1, 'mYXVS': 1, 'mYXzJF': 1, 'mYXEMx': 1, 'mYXZJ': 1, 'mYXjBn': 1, 'mYXvfM': 1, 'mYXl
mYYlt': 1, 'mYYcgB': 1, 'mYYSn': 1, 'mYYrQ': 1, 'mYYqiW': 1, 'mYYDK': 1, 'mYYr': 1, 'mYYEKc': 1, 'mYYghn': 1, 'mYYs': 1, 'mYYBe': 1, 'mYYQtc
'mYZN': 1, 'mYZfR': 1, 'mYZl': 2, 'mYZUc': 1, 'mYZsrV': 1, 'mYZcR': 1, 'mYZau': 1, 'mYZAM': 1, 'mYZFUi': 1, 'mYZdN': 1, 'mYZylk': 1, 'mYZAG
'mZaqSu': 1, 'mZagtN': 1, 'mZaKzX': 1, 'mZaI': 2, 'mZaq': 1, 'mZae': 1, 'mZaOem': 1, 'mZaqzN': 1, 'mZaFc': 1, 'mZas': 1, 'mZaTb': 1, 'mZaxSc
, 'mZbJpc': 1, 'mZbNKe': 1, 'mZbxk': 1, 'mZbOZV': 1, 'mZbZ': 2, 'mZbI': 1, 'mZbSOH': 1, 'mZbRg': 1, 'mZbuFq': 1, 'mZbR': 1, 'mZbP': 1, 'mZbV
': 1, 'mZcva': 1, 'mZcBP': 1, 'mZcD': 2, 'mZczpx': 1, 'mZcI': 1, 'mZcxZ': 1, 'mZcmC': 1, 'mZciH': 1, 'mZcOL': 1, 'mZcY': 1, 'mZceW': 1, 'mZc
ZdWLP': 1, 'mZdTCf': 1, 'mZdKV': 1, 'mZdc': 1, 'mZdHD': 1, 'mZdm': 1, 'mZdL': 1, 'mZdfx': 1, 'mZdhB': 1, 'mZdXY': 1, 'mZdhQu': 1, 'mZdlup':
: 1, 'mZeIES': 1, 'mZetSZ': 1, 'mZebgg': 1, 'mZeS': 1, 'mZeHFT': 1, 'mZerv': 1, 'mZeI': 2, 'mZeqXy': 1, 'mZecWD': 1, 'mZeXsP': 1, 'mZebgZ':
1, 'mZfJ': 2, 'mZfM': 1, 'mZftLY': 1, 'mZfkk': 1, 'mZfKA': 1, 'mZfvtW': 1, 'mZfcmp': 1, 'mZfV': 1, 'mZfun': 1, 'mZfOA': 1, 'mZfbe': 1, 'mZf
'mZgMh': 1, 'mZgmR': 1, 'mZgfd': 1, 'mZgnlh': 1, 'mZgt': 1, 'mZgMC': 1, 'mZgTrw': 1, 'mZgQvD': 1, 'mZgWO': 1, 'mZgjx': 1, 'mZgbO': 1, 'mZgN

Список источников

1) Материалы портала URL: https://ru.bmstu.wiki/MapReduce – дата обращения 09.03.2020.
2) Материалы портала URL: https://www.computerra.ru/183360/mapreduce/ - дата обращения 10.03.2020.
3) Материалы портала URL: https://www.computerra.ru/183360/mapreduce/ - дата обращения 10.03.2020.
4) Материалы портала URL: https://www.ibm.com/developerworks/ru/library/cl-mapreduce/index.html - дата обращения 10.03.2020.
5) Материалы портала URL: https://dic.academic.ru/dic.nsf/ruwiki/607046 - дата обращения 10.03.2020.
6) Материалы портала URL: https://habr.com/ru/post/103467/ – дата обращения 07.03.2020.
7) Материалы портала URL: https://www.bigdataschool.ru/wiki/mapreduce – дата обращения 07.03.2020.