

CMSC5741 Big Data Tech. & Apps.

Lecture 2: MapReduce and

Assignment Project Exam Help

F <https://eduassistpro.github.io/> ts

Add WeChat edu_assist_pro

Prof. Michael R. Lyu

Computer Science & Engineering Dept.

The Chinese University of Hong Kong

Outline

- Introduction
- The Hadoop Distributed File System (HDFS)
- MapReduce
- Hadoop
- Hadoop Streaming
- Problems Suited for MapReduce
- TensorFlow
- Frequent Itemsets
- Conclusion

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Introduction

- Much of the course will be devoted to learning with big data

Assignment Project Exam Help

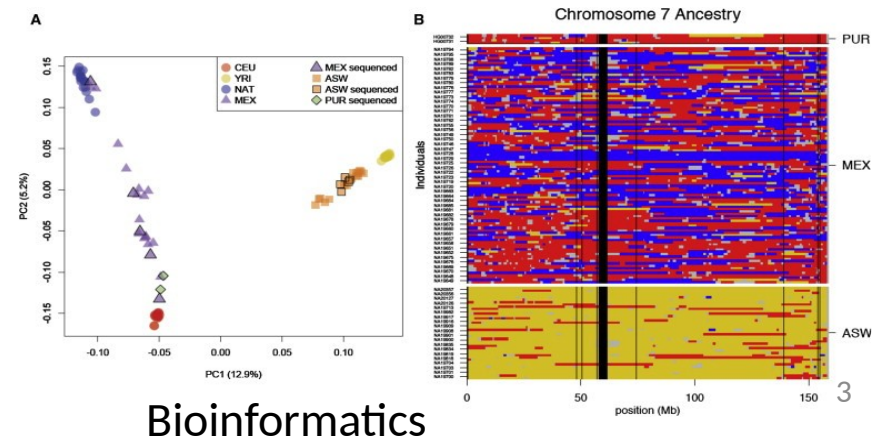
<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Facebook



Netflix



Bioinformatics

Introduction

- Challenges:
 - How to distribute computation?
 - Distributed/p is hard
- MapReduce a <https://eduassistpro.github.io/>bove
 - Google's computational/d relation model
 - Elegant way to work with Big Data

Motivation: Data Volume Now

The scale of data today and tomorrow:

- 2008: Google processes 20 PB a day
- 2009: Facebook has 2.5 PB user data + 15 TB/day
- 2009: eBay has 6.5 PB user data +
- 2013: Estimated size of digital world
- 2016: 2.5 exabytes (EB) created everyday
- 2017: Google holds 10-15 exabytes of data
- By 2020: 44ZB (10^{21}) will be produced (5.2 TB for every person)

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Motivation: Google Example

- 20+ billion web pages x 20KB = 400+ TB
- 1 computer reads 30-35 MB/sec from disk
 - ~4 months to read
- ~1,000 hard drive <https://eduassistpro.github.io/> (only text data)
- Takes even more resources to do work with the data
- Today, a **standard architecture** for such problem is emerging:
 - Cluster of commodity Linux nodes
 - Commodity network (ethernet) to connect them
 - It was estimated that Google had over 2.5M machines in 16 data centers worldwide (one center includes 9,941 miles of cable)



Large-scale Computing

- Large-scale computing for data mining problems on commodity hardware
- Challenges:
 - 1. How do you <https://eduassistpro.github.io/>
 - 2. How can we make it easy to [Add WeChat edu_assist_pro](#) ibuted programs?
 - 3. How can you handle machine failures?
 - One server may stay up 3 years (1,000 days)
 - If you have 1,000 servers, expect to lose 1/day
 - It is estimated that Google had 2.5M machines in 2016
 - 2,500 machine fails every day!

Idea and Solution

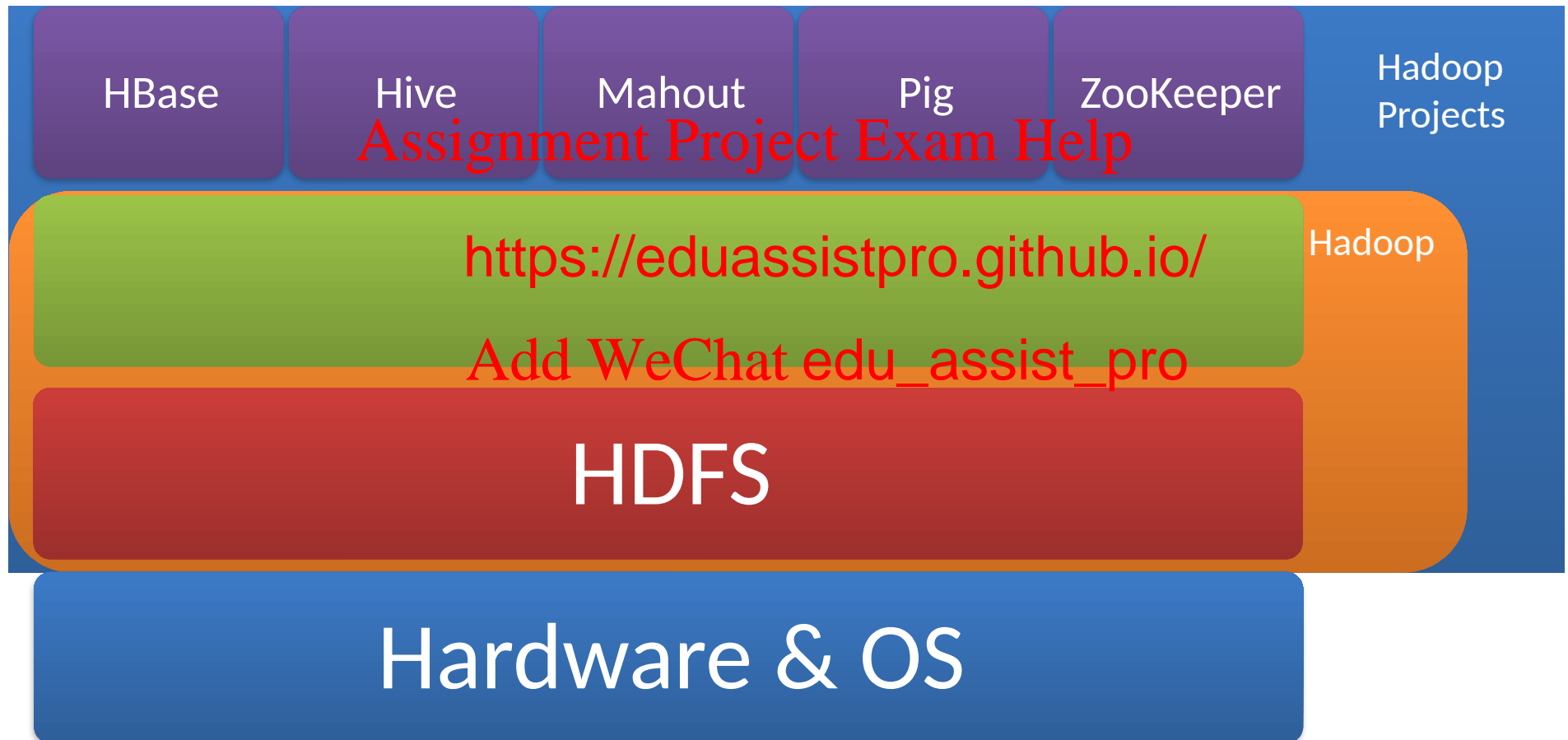
- Issue: Copying data over a network takes time
- Idea:
 - Bring computation close to the data
 - Store files multiple times
- MapReduce address
 - Google's computational/data management model
 - Elegant way to work with big data
 - Storage Infrastructure – File system
 - Google: GFS; Hadoop: HDFS
 - Programming model
 - Map-reduce

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Relationship



Outline

- Introduction
- The Hadoop Distributed File System (HDFS)
- MapReduce
- Hadoop <https://eduassistpro.github.io/>
- Hadoop Streaming [Add WeChat edu_assist_pro](#)
- Problems Suited for MapReduce
- TensorFlow
- Frequent Itemsets
- Conclusion

The Hadoop Distributed File System (HDFS)

- With hundreds of machines at hand, **failure** is the **norm** rather than exception
- Traditional file systems do not cope with the scale and failures
- The **Hadoop Distributed File System (HDFS)** is a natural solution to this problem
 - Distributed File System
 - Provides global file namespace
 - Replica to ensure data recovery

The Hadoop Distributed File System (HDFS)

- A HDFS instance may consist of thousands of server machines, each storing **part** of the file system's data.
- Since we have **Assignment Project Exam Help** components, and each component **https://eduassistpro.github.io/** probability of failure, it means that there **Add WeChat edu_assist_pro** some component that is non-functional.
- **Detection** of faults and quick, automatic **recovery** from them are a core architectural goal of HDFS.

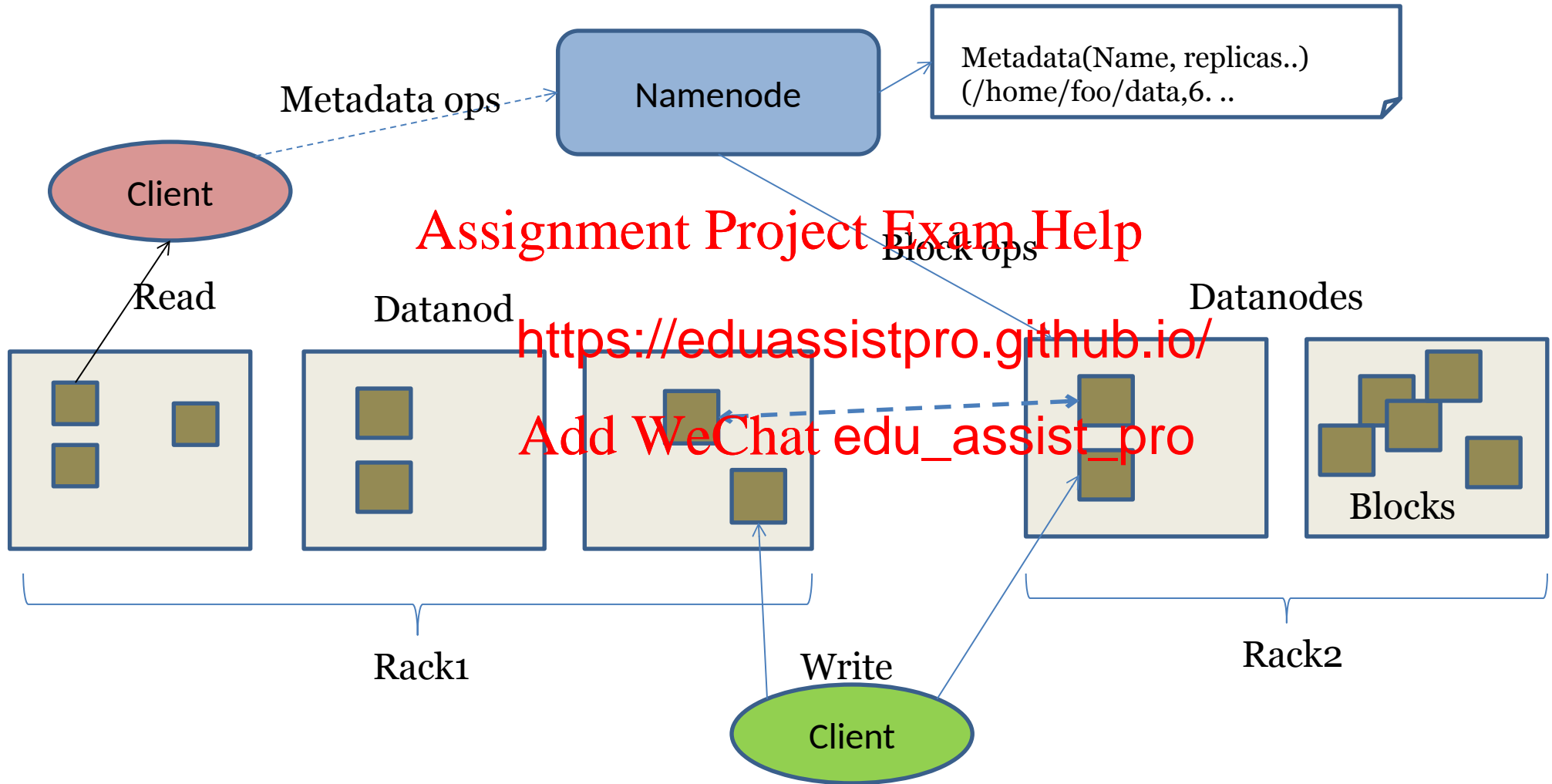
Data Characteristics

- Streaming data access
- Batch processing rather than interactive user access <https://eduassistpro.github.io/>
- Write-once-read-many: once created, written and closed, need not be changed
- This assumption simplifies coherency in concurrent accesses

HDFS Architecture

- Master/slave architecture
 - Master: NameNode
 - Slave: DataNode
- HDFS exposes <https://eduassistpro.github.io/> [pace \(NameNode\)](#)
and allows user data to be [Add WeChat edu_assist_pro](#) files.
- A file is split into one or more blocks and set of blocks are stored in [DataNodes](#).

HDFS Architecture



File System Namespace

- Namenode maintains the file system.
 - Hierarchical file system with directories and files.
 - Create, remove, move, rename, etc.
 - Any meta info <https://eduassistpro.github.io/> file system is recorded by the Namenode.
- An application can specify the number of replicas of the file needed: replication factor of the file. This information is stored in the Namenode.

Data Replication

- HDFS is designed to store very large files across machines in a large cluster.
 - Each file is a sequence of blocks.
 - All blocks in the file are of the same size.
- Blocks are replicated.
- Block size and replication factor are configurable per file.
- The NameNode receives a Heartbeat and a BlockReport from each DataNode in the cluster.
- BlockReport contains all the blocks on a DataNode.

Replica Selection

- Replica selection for read operation: HDFS tries to minimize the bandwidth consumption and latency.
- If there is a replica on the same node then that is preferred. <https://eduassistpro.github.io/>
- HDFS cluster may span multiple data centers: replica in the local data center is preferred over the remote one.

Safemode Startup

- Each DataNode checks in with Heartbeat and BlockReport.
- NameNode verifies that each block has acceptable number of replicas. [Assignment Project Exam Help](https://eduassistpro.github.io/)
- After a configurable number of replicated blocks check in with the NameNode, NameNode exits Safemode. <https://eduassistpro.github.io/>
- It then makes the list of blocks to be replicated. [Add WeChat edu_assist_pro](https://eduassistpro.github.io/)
- NameNode then proceeds to replicate these blocks to other DataNodes.

Filesystem Metadata

- The HDFS namespace is stored by NameNode.
- NameNode uses a transaction log called the EditLog to record every <https://eduassistpro.github.io/> to the filesystem meta data.
 - For example, creating a new [Add WhatsApp edu_assist_pro](#)
 - Change replication factor of a file
 - EditLog is stored in the NameNode's local filesystem

NameNode

- Keeps image of entire file system namespace.
- When the Namenode starts up
 - Gets the Fslmage and Editlog.
 - Update Fslma
 - Stores a copy of the Fslmag
- In case of crash
 - Last checkpoint is recovered.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

checkpoint.

Add WeChat edu_assist_pro

DataNode

- A DataNode stores data in files in its local file system.
 - Each block of HDFS is a separate file.
 - These files are placed in different directories.
 - Creation of new files is based on heuristics.
- When the filesystem starts
 - Generates Blockreport.
 - Sends this report to NameNode.

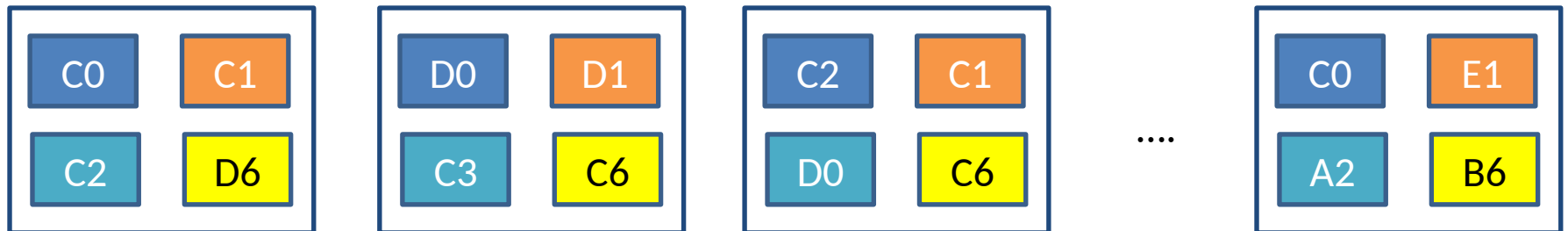
HDFS Summary

- Reliable distributed file system
- Data kept in “chunks” spread across machines
- Each chunk replicated multiple times across multiple machines and racks
 - Seamless recovery from disk failure

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Outline

- Introduction
- The Hadoop Distributed File System (HDFS)
- MapReduce **Assignment Project Exam Help**
- Hadoop **<https://eduassistpro.github.io/>**
- Hadoop Streaming **Add WeChat edu_assist_pro**
- Problems Suited for MapReduce
- TensorFlow
- Frequent Itemsets
- Conclusion

MapReduce

- Warm-up task
 - We have a huge text document
 - Count the number of distinct words that appear in the file
- Sample application
 - Analyze web server logs to find popular URLs

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Task: Word Count

- Using Unix tool chain, we can count the occurrences of words:
 - `words (doc.txt) | sort | uniq -c`
 - Where words are words in it, one per line
- This way of counting captures the essence of MapReduce
 - **Mapper** (done by words)
 - Group by keys and sort (done by sort)
 - **Reducer** (done by uniq)
 - Hadoop handles the partition and parallelization

MapReduce: Overview

- Sequentially read a lot of data
- **Map:**
– Extract some
- Group by key:
- **Reduce:**
– Aggregate, summarize, filter or transform
- Write the **result**

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

MapReduce

- Input: a set of key-value pairs
- Programmer must specify two methods:
 - **Map**(k,v) \rightarrow \langle
 - Takes a key-value pair from the set of key-value pairs
 - There is one Map call for every key \rangle
 - **Reduce** (k', $\langle v' \rangle$) \rightarrow $\langle k', v'' \rangle$
 - All values v' with the same key k' are reduced together and processed in v' order
 - There is one Reduce function call per unique key k'

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

MapReduce: Word Count Example

- Now that one document changes to a large corpus of documents

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

MapReduce: Word Count Example

```
// key: document name; value: text of the document
```

```
Map(key, value):
```

```
  for each word w in value:
```

```
    emit(w, 1)
```

```
// key: a word; value: counts
```

```
Reduce(key, value
```

```
  result = 0
```

```
  for each count v in values:
```

```
    result += v
```

```
  emit(key, result)
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

In-class Practice

- Go to [practice](#)

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

MapReduce: Environment

- MapReduce environment takes care of:
 - Partitioning the input data
 - Scheduling the execution across a set of machines
 - Performing the “group by”
 - Handling machine failures
 - Managing required inter-machine communication

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

MapReduce

Map:

Read input and
produces a set of
key-value pairs

Group by key:

Collect all pairs with
the same key

Reduce:

Collect all values
belonging to the key
and output

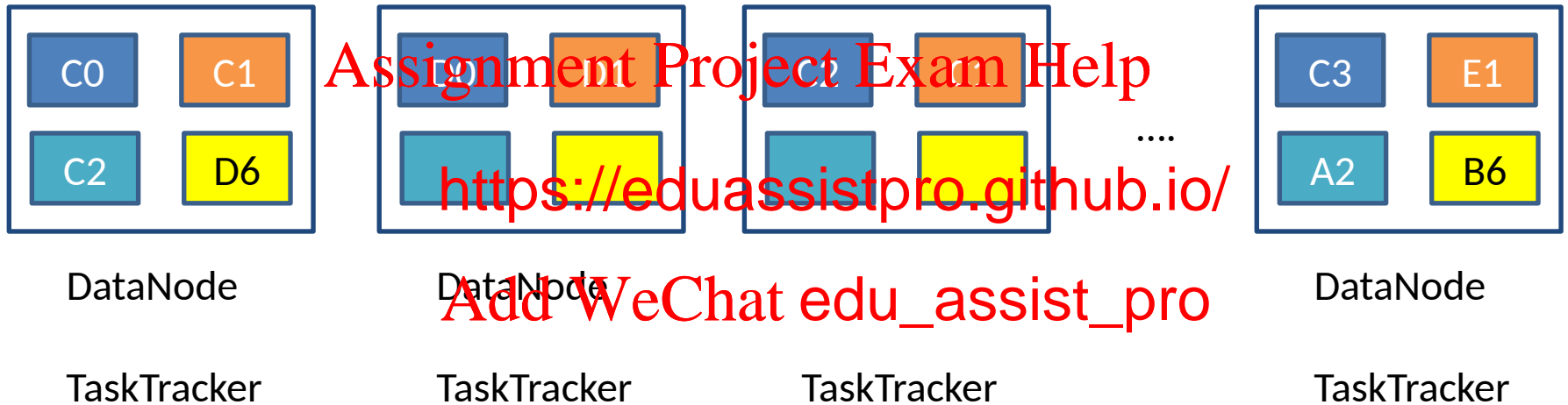
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

MapReduce

- Move computation to the data



Bring computation directly to the data!

DataNode also serve as compute servers

Data Flow

- Input and final output are stored on a distributed file system (FS):
 - Scheduler tries to keep “close” to physical storage
- Intermediate results are stored in a local FS of Map and Reduce workers
- Output is often input to another MapReduce task

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Coordination: Master

- Master node takes care of coordination:
 - Task status: (idle, in-progress, completed)
 - Idle tasks get <https://eduassistpro.github.io/> become available
 - When a map <https://eduassistpro.github.io/> adds the master the location and sizes of its R_i files, one for each reducer
 - Master pushes this info to reducers
- Master pings workers periodically to detect failures

Dealing with Failures

- Map worker failure
 - Map tasks completed or in-progress at worker are reset to idle
 - Reduce workers are notified when task is rescheduled on another worker
- Reduce worker failure
 - Only in-progress tasks are reseeded
 - Reduce task is restarted
- Master failure
 - MapReduce task is aborted and client is notified.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

How Many Map and Reduce Jobs?

- M map tasks, R reduce tasks
- Rule of a thumb:
 - Make M much greater than R, number of nodes in the cluster
 - One chunk per map is common
 - Improves dynamic load balancing and speeds up recovery from worker failures
- Usually R is smaller than M
 - Output is spread across R files

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Refinement: Combiners

- Often a Map task will produce many pairs of the form $(k, v_1), (k, v_2), \dots$ for the same key k
 - E.g., popular words in the word count example
- Can save network traffic by aggregating values in the mapper:
 - $\text{Combine}(k, \text{list}(v)) \rightarrow v_2$
 - Combiner is usually the same as the reduce function
- Works only if reduce function is **commutative** and **associative**

Refinement: Partition Function

- Want to control how keys get partitioned
 - Inputs to map tasks are created by contiguous splits of input file
 - Reduce needs to process all intermediate keys with the same reducer
- System uses a default partitioning function:
 - $\text{Hash}(\text{key}) \bmod R$
- Sometimes useful to override the hash function:
 - E.g., `hash(hostname(URL)) mod R` ensures URLs from a host end up in the same output file



Outline

- Introduction
- The Hadoop Distributed File System (HDFS)
- MapReduce
- Hadoop <https://eduassistpro.github.io/>
- Hadoop Streaming [Add WeChat edu_assist_pro](#)
- Problems Suited for MapReduce
- TensorFlow
- Frequent Itemsets
- Conclusion

Hadoop

- Hadoop is an open source implementation of MapReduce framework
 - Hadoop Distributed File System (HDFS) as storage
 - Hadoop handles data distribution, task monitoring and failure recovery
 - All you need to do is to write two Java classes
 - Mapper
 - Reducer

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Hadoop

- Follow the MapReduce architecture, the Hadoop has a master/slave design

Assignment Project Exam Help

slave		
MapReduce	jobtrack	ktracker
HDFS	namenode	datanode

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Word Count in Hadoop

- Mapper

```
public static class MapClass extends MapReduceBase
implements Mapper {
    private final static IntWritable one = new IntWritable(1);
    private Text word

    public void map(WritableComparable value,
OutputCollector output, Reporter reporter)
throws IOException {
    String line = ((Text)value).toString();
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        output.collect(word, one);
    }
}
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Word Count in Hadoop

- Reducer

```
public static class Reduce extends MapReduceBase implements
Reducer {
    public void reduce(Object key, Iterator
values, OutputCollector reporter)
throws IOException {
    int sum = 0;
    while (values.hasNext()) {
        sum += ((IntWritable) values.next()).get();
    }
    output.collect(key, new IntWritable(sum));
}
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Word Count in Hadoop

- Main

```
public static void main(String[] args) throws IOException {  
    //checking goes here  
    JobConf conf = new JobConf();  
  
    conf.setOutputKey(  
    conf.setOutputValue(  
  
    conf.setMapperClass(MapClass.class);  
    conf.setCombinerClass(Reduce.class);  
    conf.setReducerClass(Reduce.class);  
  
    conf.setInputPath(new Path(args[0]));  
    conf.setOutputPath(new Path(args[1]));  
  
    JobClient.runJob(conf);  
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Outline

- Introduction
- The Hadoop Distributed File System (HDFS)
- MapReduce
- Hadoop
- Hadoop Streaming
- Problems Suited for MapReduce
- TensorFlow
- Frequent Itemsets
- Conclusion

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Hadoop Streaming

- To enjoy the convenience brought by Hadoop, one has to implement mapper and reducer in Java
 - Hadoop defines a simple and complex class hierarchy <https://eduassistpro.github.io/>
 - There is a learning curve [Add WeChat edu_assist_pro](#)
- Hadoop streaming allows you to use any language to write the mapper and reducer

Hadoop Streaming

- Using Hadoop Streaming, you need to write

- Mapper

- Read input from standard input (STDIN)
 - Write map r <https://eduassistpro.github.io/> (STDOUT)
 - Key value are separated using

- Group by key

- Done by Hadoop

- Reducer

- Read input (Mapper's output) from standard input (STDIN)
 - Write output (Final result) to standard output (STDOUT)

Hadoop Streaming

- Allows you to start writing MapReduce application that can be readily deployed without having to learn Hadoop data types
- Speed up dev <https://eduassistpro.github.io/>
- Utilize rich features and libraries from other languages (Python, Ruby)
- Efficiency critical application can be implemented in efficient language (C, C++)

Hadoop Streaming: Word Count Mapper

```
#!/usr/bin/env python
```

```
import sys
```

```
# input comes from STDIN (standard input)
```

```
for line in sys.stdin:
```

```
    # remove leading and trailing space
```

```
    line = line.strip()
```

```
    # split the line into words
```

```
    words = line.split()
```

```
    # increase counters
```

```
    for word in words:
```

```
        # write the results to STDOUT (standard output);
```

```
        # what we output here will be the input for the
```

```
        # Reduce step, i.e. the input for reducer.py
```

```
        #
```

```
        # tab-delimited; the trivial word count is 1
```

```
        print '%s\t%s' % (word, 1)
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Hadoop Streaming: Word Count Reducer

```
#!/usr/bin/env python
from operator import itemgetter
import sys
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    line = line.strip()
    word, count = line.split()
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s\t%s' % (current_word, current_count)
            current_count = count
            current_word = word
        if current_word == word:
            print '%s\t%s' % (current_word, current_count)
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Hadoop Streaming: How to Run?

- To run the sample code

```
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-streaming.jar \  
-input inputPathonHDFS \  
-output outputPathonHDFS \  
-file pathToMapper.p \  
-mapper mapper.py \  
-file pathToReducer.py \  
-reducer reducer.py
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

- -file caches the argument to every tasktracker
- The above command distribute the mapper.py and reducer.py to every tasktracker

Hadoop Streaming: Word Count

```
#!/usr/bin/env python
"""A more advanced Mapper, using Python iterators and generators."""

import sys
def read_input(file):
    for line in file:
        yield line.split()

def main(separator='\t'):
    # input comes from STDIN (standard input)
    data = read_input(sys.stdin)
    for words in data:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        for word in words:
            print '%s%s%d' % (word, separator, 1)

if __name__ == "__main__":
    main()
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Hadoop Streaming: Word Count

```
#!/usr/bin/env python
"""A more advanced Reducer, using Python iterators and generators."""

from itertools import groupby
from operator import itemgetter
import sys

def read_mapper_output(file, separator='\t'):
    for line in file:
        yield line.rstrip().split(separator, 1)

def main(separator='\t'):
    # input comes from STDIN
    data = read_mapper_output(sys.stdin, separator)
    # groupby groups multiple word-count pairs by word
    # and creates an iterator that returns consecutive groups
    # current_word - string containing a word (the key)
    # group - iterator yielding all ["<current_word>", "<count>"] items
    for current_word, group in groupby(data, itemgetter(0)):
        try:
            total_count = sum(int(count) for current_word, count in group)
            print "%s%s%d" % (current_word, separator, total_count)
        except ValueError:
            # count was not a number, so silently discard this item
            pass

if __name__ == "__main__":
    main()
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Demo

- Given a list of academic paper authors and their papers, we try to output:
 - The most use the title for each author <https://eduassistpro.github.io/>
- We use a python based ML framework implementation called mincemeat.

Demo

- Input Data

- Books, Ph.D. Thesis, web pages, academic papers

- Input format

- Publication t <https://eduassistpro.github.io/>

- Affiliation [Add WeChat edu_assist_pro](#)

- Abbreviation code

- Authors

- Title

tr/gte/TM-0014-06-88-165::Frank Manola::Distributed Object
Management Technology.

tr/ibm/IWBS94::Christoph Beierle::Udo Pletat::The Algebra of
Feature Graph Specifications

Demo

- To run the demo:
 - In terminal 1, type:
 - `python demo.py`
 - This set the map in program
 - In terminal 2, type:
 - `python mincemeat.py -p changeme`
 - “changeme” is the authentication password
 - 127.0.0.1 is the server address
 - This starts the Map-Reduce framework

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Demo

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Outline

- Introduction
- The Hadoop Distributed File System (HDFS)
- MapReduce
- Hadoop
- Hadoop Streaming
- **Problems Suited for MapReduce**
- TensorFlow
- Frequent Itemsets
- Conclusion

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Example: Host Size

- Suppose we have a large web corpus
- Look at the metadata file
 - Lines of the form `host page_id`
- For each host, $\sum_{page} \text{size of page}$
 - That is, the sum of the page sizes from that particular host
- Other examples:
 - Link analysis and graph processing
 - Machine learning algorithms

Example: Language Model

- Statistical machine translation:
 - Need to count number of times every 5-word sequence occurs in a large corpus of documents
- Very easy with <https://eduassistpro.github.io/>
 - Map: Add WeChat edu_assist_pro
 - Extract (5-word sequence, count) from document
 - Reduce:
 - Combine the counts

Example: Join By MapReduce

- Compute the natural join $R(A,B) \times S(B,C)$
- R and S are each stored in files
- Tuples are pairs

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

A	B
a1	b1
a2	b1
a3	b2
a4	b3

x

B	C
b2	c1
b2	c2
b3	c3

=

A	C
a3	c1
a3	c2
a4	c3

Note – Other relational-algebra operations: Selection, Projection, Union/Intersection/Difference, Grouping/Aggregation

MapReduce Join

- Use a hash function from B-values to
- A Map process turns:
 - Each input tup
 - Each input tup
- Map processes send each pair with key to Reduce process
 - Hadoop does this automatically; just tell it what is
- Each Reduce process matches all the pairs with all and outputs .

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WhatsApp edu_assist_pro

Cost Measures for Algorithms

- In MapReduce we quantify the cost of an algorithm using
 - Communication cost
 - total I/O of all processes
 - Elapsed communication cost
 - Max of I/O along any path
 - (Elapsed) computation cost
 - Analogous, but count only running time of processes
 - Note that here the big-O notation is not the most useful (adding more machines is always an option)

Example: Cost Measures

- For a MapReduce algorithm:
- **Communication cost** = input file size + 2 (sum of the sizes of all map processes to the Reduce process + the output sizes of the Reduce processes).
<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro
- **Elapsed communication cost** is the sum of the largest input + output for any map process, plus the same for any reduce process

What Cost Measures Mean

- Either the I/O (communication) or processing (computation) cost dominates
 - Ignore one or
- Total cost tell <https://eduassistpro.github.io/> nt from your friendly neighborhood [Add WeChat edu_assist_pro](#)
- Elapsed cost is wall-clock time using parallelism

Cost of MapReduce Join

- Total communication cost
 - $O(|R| + |S| + |RS|)$
- Elapsed communication cost $= O(s)$, where s is the I/O limit
 - We're going to pick s processes so that the I/O limit s is respected
 - We put a limit s on the amount of input that any one process can have
 - s could be:
 - What fits in main memory
 - What fits on local disk
- With proper indexes, computation cost is linear in the input + output size
 - So computation cost is like communication cost



Outline

- Introduction
- The Hadoop Distributed File System (HDFS)
- MapReduce
- Hadoop <https://eduassistpro.github.io/>
- Hadoop Streaming [Add WeChat edu_assist_pro](#)
- Problems Suited for MapReduce
- TensorFlow
- Frequent Itemsets
- Conclusion

TensorFlow

- Interface for expressing machine learning algorithms, and an implementation for executing large-scale algorithms
- Dataflow framework that can be ported to native CPU / GPU code
- Drastic reduction in development time
- Visualization (TensorBoard)

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Programming Model

- Express a numeric computation as a **graph**
 - Graph nodes are **operations** which have any number of inputs and outputs
 - Graph edges <https://eduassistpro.github.io/> between nodes

Add WeChat edu_assist_pro

Big Data: Distributed Environment

- Portability: deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Outline

- Introduction
- The Hadoop Distributed File System (HDFS)
- MapReduce
- Hadoop <https://eduassistpro.github.io/>
- Hadoop Streaming [Add WeChat edu_assist_pro](#)
- Problems Suited for MapReduce
- TensorFlow
- Frequent Itemsets
- Conclusion

Frequent Itemsets

- Simple question: Find sets of items that appear together “frequently” in baskets
- **Support** for itemset: Number of baskets containing all items in
– Often expressed as of baskets
- Given a **support threshold** <https://eduassistpro.github.io/> that appear in at least baskets are called **frequent itemsets**
Add WeChat edu_assist_pro

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Support of {Beer, Bread} = 2

Example: Frequent Itemsets

- Items = {milk, coke, pepsi, beer, juice}
- Minimum support = 3 baskets

B1 =

B3 =

B5 = {m, p, b}

B7 = {c, b, j}

B6

B8

- Frequent itemsets: {m}, {c}, {b}, {j},
{m,b}, {b,c}, {c,j}

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Itemsets: Computation Model

- Typically, data is kept in flat files rather than in a database system:
 - Stored on disk
 - Stored basket by basket
 - Baskets are small but we have many baskets and many items
 - Expand baskets into pairs, triples, etc. as you read baskets
 - Use k nested loops to generate all sets of size k

Item
Item
Item
Item
Item
Item
Item
Item
Item
Item
Item
Etc.

Items are positive integers, and boundaries between baskets are -1.

Computation Model

- The true cost of mining disk-resident data is usually the **number of disk I/O's**
- In practice, as algorithms read the data in **passes** <https://eduassistpro.github.io/> in turn
- We measure the cost by the **number of passes** an algorithm makes over the data

Main-Memory Bottleneck

- For many frequent-itemset algorithms, **main-memory** is the critical resource
 - As we read b...nt something, e.g., occurrences <https://eduassistpro.github.io/>
 - The number of different th...n count is limited by main memory
 - Swapping counts in/out is a disaster

Naïve Algorithm to Count Pairs

- Read file once, counting in main memory the occurrences of each pair:
 - From each basket of n items, generate its $n(n-1)/2$ pairs by two nested loops
- Fails if $(\text{\#items})^2$ exceeds memory
 - Remember: #items can be 100K (Wal-Mart) or 10B (Web pages)
 - Suppose 10^5 items, counts are 4-byte integers
 - Number of pairs of items: $10^5(10^5-1)/2 = 5 \cdot 10^9$
 - Therefore, $2 \cdot 10^{10}$ (20 gigabytes) of memory needed

A-Priori Algorithm

- A **two-pass** approach called **a-priori** limits the need for main memory
- Key idea: **monotonicity**
 - If a set of items appears in s baskets, then every **subset** of it also appears in s baskets
- Contrapositive for pairs:
 - If item i does not appear in s baskets, then no pair including i can appear in s baskets

A-Priori Algorithm

- Pass 1: Read baskets and count in main memory the occurrences of each individual item
 - Requires only memory proportional to #items
- Items that appear in frequent items
- Pass 2: Read baskets in main memory only those pairs where both elements are frequent (from Pass 1)
 - Requires memory proportional to square of frequent items only (for counts)
 - Plus a list of the frequent items (so you know what must be counted)

Main-Memory: Picture of A-Priori

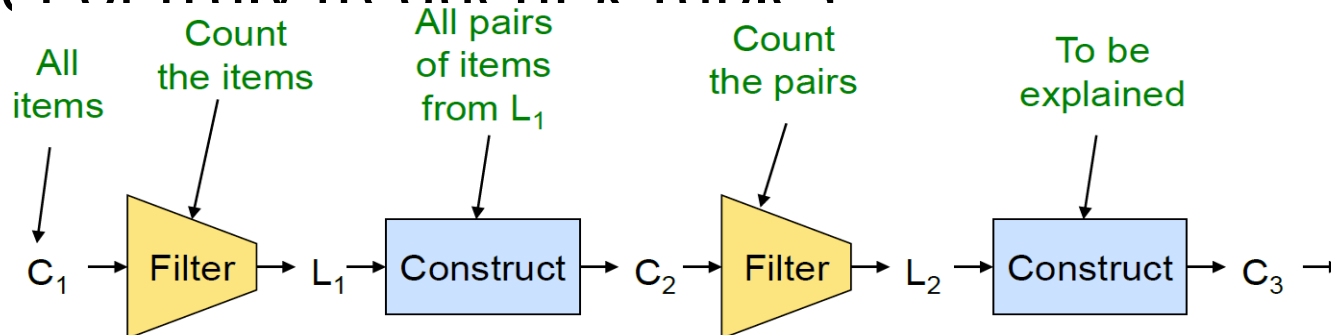
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Frequent Triplets, Etc.

- Now we know how to find frequent pairs, how about frequent triplets and frequent k-tuples?
- For each k, we construct two sets of k-tuples (sets of size k):
<https://eduassistpro.github.io/>
 = candidate k-tuples = those that be frequent sets (support > s) based on information the pass for k-1
- = the set of truly frequent k-tuples



Example

- Hypothetical steps of the A-Priori algorithm
- $= \{ \{b\} \{c\} \{j\} \{m\} \{n\} \{p\} \}$
- Count the support of itemsets in
- Prune non-frequent
- Generate $= \{ \{b,c\} \{b,j\} \{b,m\} \{c,j\} \}$
- Count the support of itemsets in
- Prune non-frequent: $= \{ \{b,m\} \{b,c\} \{c,m\} \{c,j\} \}$
- Generate $= \{ \{b,c,m\} \{b,c,j\} \{b,m,j\} \{c,m,j\} \}$
- Count the support of itemsets in
- Prune non-frequent: $= \{ \{b,c,m\} \}$

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

A-Priori for All Frequent Itemsets

- One pass for each k (itemset size)
 - Needs room in main memory to count each candidate k -tuple
 - For typical market basket data and reasonable support (e.g., 1%), $k = 2$ requires
 - Many possible e Lo port s as itemset gets bigger
- Association rules with intervals:
- For example: Men over 65 have 2 cars
- Association rules when items are in a taxonomy
- Bread, Butter \rightarrow FruitJam
 - BakedGoods, MilkProduct \rightarrow PreservedGoods

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Map-Reduce Implementation

- Divide the file in which we want to find frequent itemsets into equal chunks randomly.
- Solve the frequent itemset problem for the smaller chunk at each node ($n = \frac{\text{total number of chunks}}{\text{number of machines}}$ is the entire dataset)
<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro
- Given:
 - Each chunk is fraction $\frac{1}{n}$ of the whole input file (total n chunks)
 - σ is the support threshold for the algorithm
 - σ_c is the threshold as we process a chunk

Map-Reduce Implementation

- At each node, we can use A-Priori algorithm to solve the smaller problem
- Take the groups that have been found frequent chunks.
– Every itemset that is frequent in at least one chunk
– All the true frequent itemsets are among the candidates

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Map-Reduce Implementation

- We can arrange the aforementioned algorithm in a two-pass Map-Reduce framework
 - First Map-Reduce cycle: generate candidate itemsets
<https://eduassistpro.github.io/>
 - Second Map-Reduce cycle: filter the true frequent itemsets.
Add WeChat edu_assist_pro

Map-Reduce Implementation

First Mapper

- Run A-Priori algorithm on the chunk using support threshold
- Output the frequency for that chunk (F, c the key (itemset) and its count (or proportion)

First Reducer

- Output the candidate itemsets to verify in the second pass
- If c is not a subset of all c_a , discard c and its sets

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Map-Reduce Implementation

Second Mapper

- For all the candidate itemsets produced by first reducer, count the frequency of each itemset in the chunk

Second Reducer

- Aggregate the output of second mapper, and sum the frequency of each itemset in the output file
- Filter the itemsets with support smaller than

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro



Outline

- Introduction
- The Hadoop Distributed File System (HDFS)
- MapReduce
- Hadoop <https://eduassistpro.github.io/>
- Hadoop Streaming [Add WeChat edu_assist_pro](#)
- Problems Suited for MapReduce
- TensorFlow
- Frequent Itemsets
- Conclusion

Conclusion

- HDFS is a reliable distributed file system
- MapReduce is a distributed computing environment
 - Hadoop is an open source implementation of MapReduce
 - Hadoop Streaming <https://eduassistpro.github.io/> is a simple language to write MapReduce code
- Frequent Itemsets problem can be solved efficiently using its monotonicity property
 - A-Priori algorithm

One-Slide Takeaway

- HDFS is a distributed file system built with robust in mind
- MapReduce is a convenient paradigm to implement parallel program
- Hadoop is an open source implementation of MapReduce
- Hadoop streaming allows you to write mapper and reducer in any language to
- Monotonicity property enable efficient algorithms for Frequent Itemsets problem

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

References

- MapReduce: Simplified Data Processing on Large Clusters, Jeffrey Dean and Sanjay Ghemawat
- The Hadoop Distributed File System: Architecture and Design by Apache Foundation
- Hadoop in Action, <https://eduassistpro.github.io/>
- Hadoop File System, ppt by B. Ram
- Intro & MapReduce, pdf by Jure Leskovec
- Association Rules, pdf by Jure Leskovec
- Writing an Hadoop MapReduce Program in Python,
<http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/>



In-Class Practice

Given the following input:

I spent long spells at sea on all types of vessel; I followed officer training with the Surface Fleet and with the Royal Marines.

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

1. Write the output of the word count mapper's above input.
2. Write the output of the word count mapper's output after the shuffle process.
3. Write the output of the word count reducer's output for the above input.