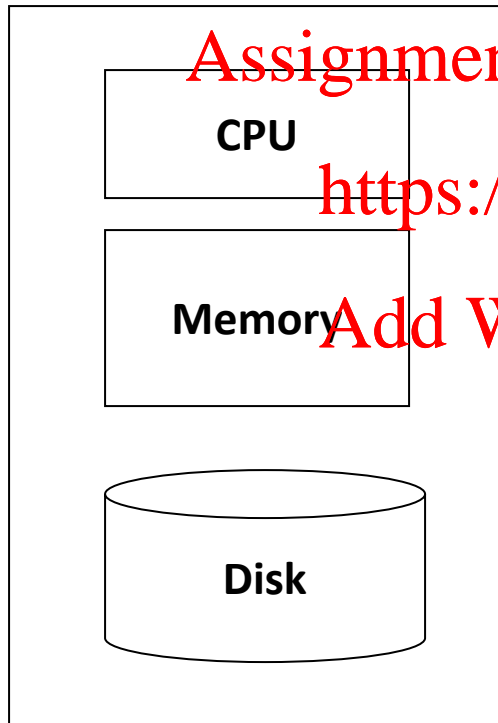


Assignment Project Exam Help
Map-Reduce

<https://powcoder.com>

Add WeChat powcoder

Single Node Architecture



Assignment Project Exam Help

<https://powcoder.com>

Machine Learning, Statistics

Add WeChat powcoder

“Classical” Data Mining

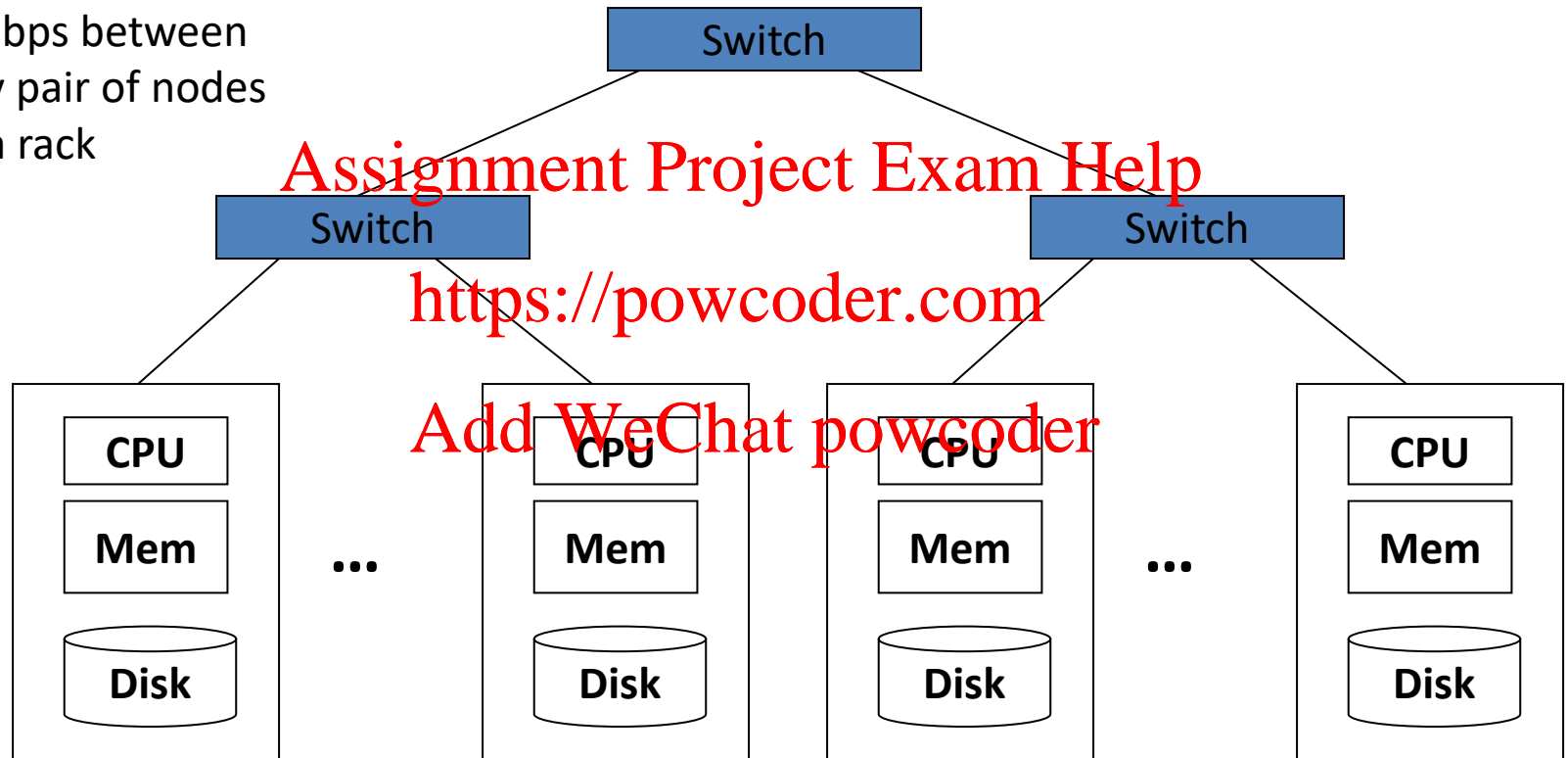
Motivation: Google Example

- 20+ billion web pages x 20KB = 400+ TB
- 1 computer reads 30-35 MB/sec from disk
 - ~4 months to read the web
- ~1,000 hard drives to store the web
- Takes even more to do something useful with the data
- **Today, a standard architecture for such problems is emerging:**
 - Cluster of commodity Linux nodes
 - Commodity network (ethernet) to connect them

Cluster Architecture

2-10 Gbps backbone between racks

1 Gbps between
any pair of nodes
in a rack



Each rack contains 16-64 nodes

In 2011 it was guestimated that Google had 1M machines, <http://bit.ly/Shh0RO>



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Distribute File System

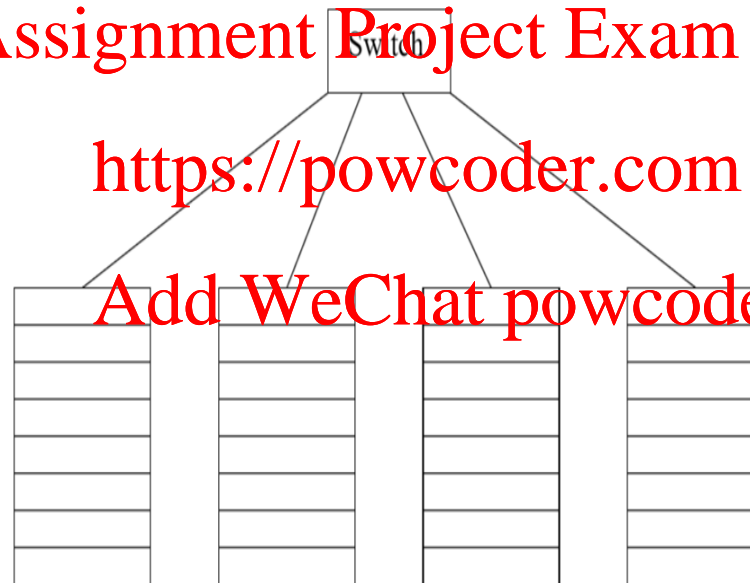
2.1. DISTRIBUTED FILE SYSTEMS

23

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Racks of compute nodes

Large-scale Computing

- **Large-scale computing for data mining problems on commodity hardware**
- **Challenges:**
 - **How do you distribute computation?**
<https://powcoder.com>
 - **How can we make it easy to write distributed programs?**
Add WeChat powcoder
 - **Machines fail:**
 - One server may stay up 3 years (1,000 days)
 - If you have 1,000 servers, expect to loose 1/day
 - People estimated Google had ~1M machines in 2011
 - 1,000 machines fail every day!

Idea and Solution

- **Issue:** Copying data over a network takes time
- **Idea:**
 - Bring computation close to the data
 - Store files multiple times for reliability
- **Map-reduce** addresses these problems
 - Google's computational/data manipulation model
 - Elegant way to work with big data
 - **Storage Infrastructure – File system**
 - Google: GFS. Hadoop: HDFS
 - **Programming model**
 - Map-Reduce

J. Leskovec, A. Rajaraman, J. Ullman:

Mining of Massive Datasets,

<http://www.mmds.org>

Storage Infrastructure

- **Problem:**
 - If nodes fail, how to store data persistently?
- **Answer:** **Assignment Project Exam Help**
 - **Distributed File System:** **<https://powcoder.com>**
 - Provides global file namespace
 - Google GFS; Hadoop HDFS, **Add WeChat powcoder**
- **Typical usage pattern**
 - Huge files (100s of GB to TB)
 - Data is rarely updated in place
 - Reads and appends are common

Distributed File System

- **Chunk servers**

- File is split into contiguous chunks
- Typically each chunk is 16-64MB
- Each chunk replicated (usually 2x or 3x)
- Try to keep replicas in different racks

- **Master node**

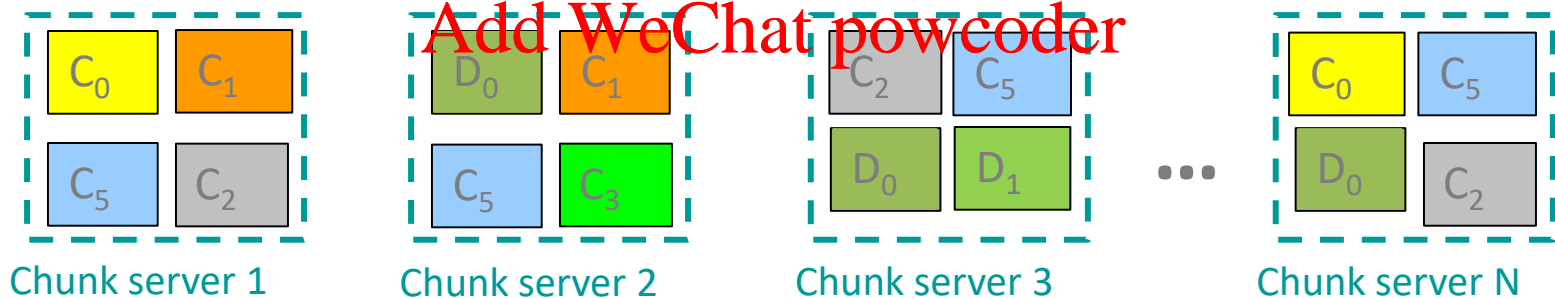
- a.k.a. Name Node in Hadoop's HDFS
- Stores metadata about where files are stored
- Might be replicated

- **Client library for file access**

- Talks to master to find chunk servers
- Connects directly to chunk servers to access data

Distributed File System

- **Reliable distributed file system**
- Data kept in “chunks” spread across machines
- Each chunk **replicated** on different machines
 - Seamless recovery from disk or machine failure



Bring computation directly to the data!

Chunk servers also serve as compute servers

Cryptographic Hash Functions

- Maps an arbitrary length input to a fixed-size output.
- Was originally proposed to generate input to digital signatures.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Desirable features of hash function

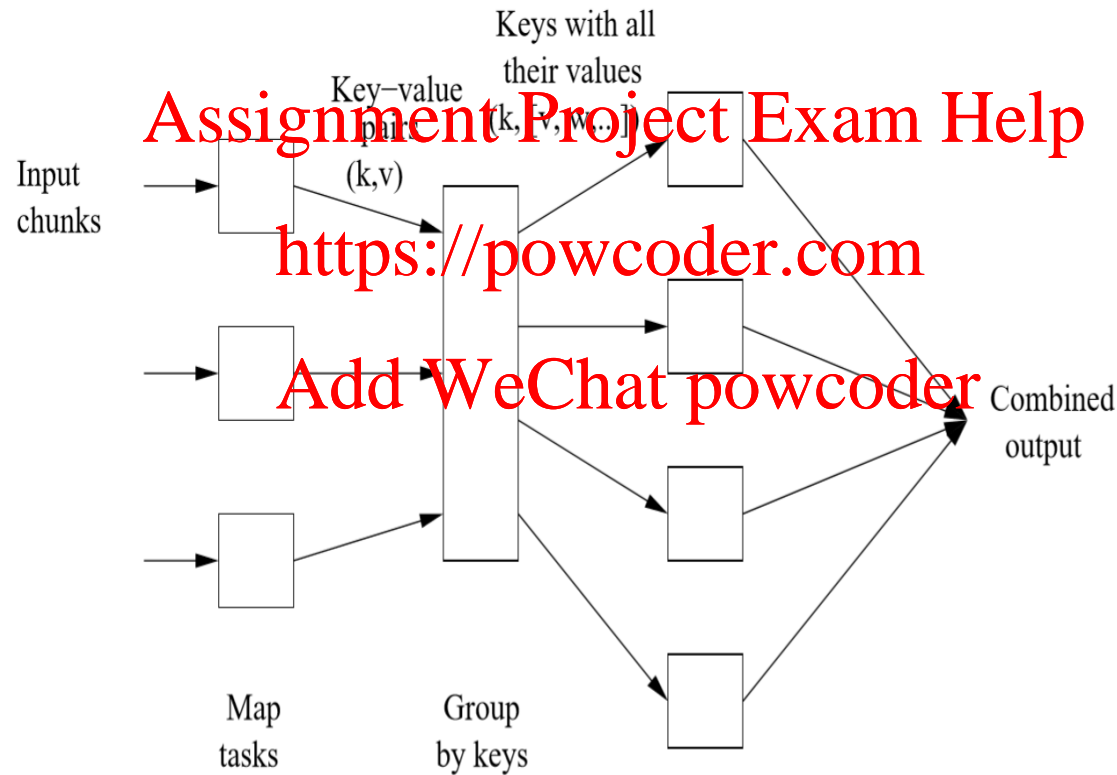
- Deterministic
- Quick Computation
- Pre-Image Resistance (one way)
- Small Changes in The Input Changes the Hash
- Collision Resistant (pseudo random)
- Constant length
 - E.g: <https://emn178.github.io/online-tools/sha256.html>

Map-Reduce: Key Value Pair

- **Input:** a set of key-value pairs
- Programmer specifies two methods:
 - **Map(k, v)** $\rightarrow <k', v'>^*$
 - Takes a key-value pair and outputs a set of key-value pairs
 - E.g., key is the filename, value is a single line in the file
 - There is one Map call for every (k, v) pair
 - **Reduce($k', <v'>^*$)** $\rightarrow <k', v''>^*$
 - All values v' with same key k' are reduced together and processed in v' order
 - There is one Reduce function call per unique key k'

Map- Reduce

Figure 2.2 suggests this computation.



Map Reduce Execution

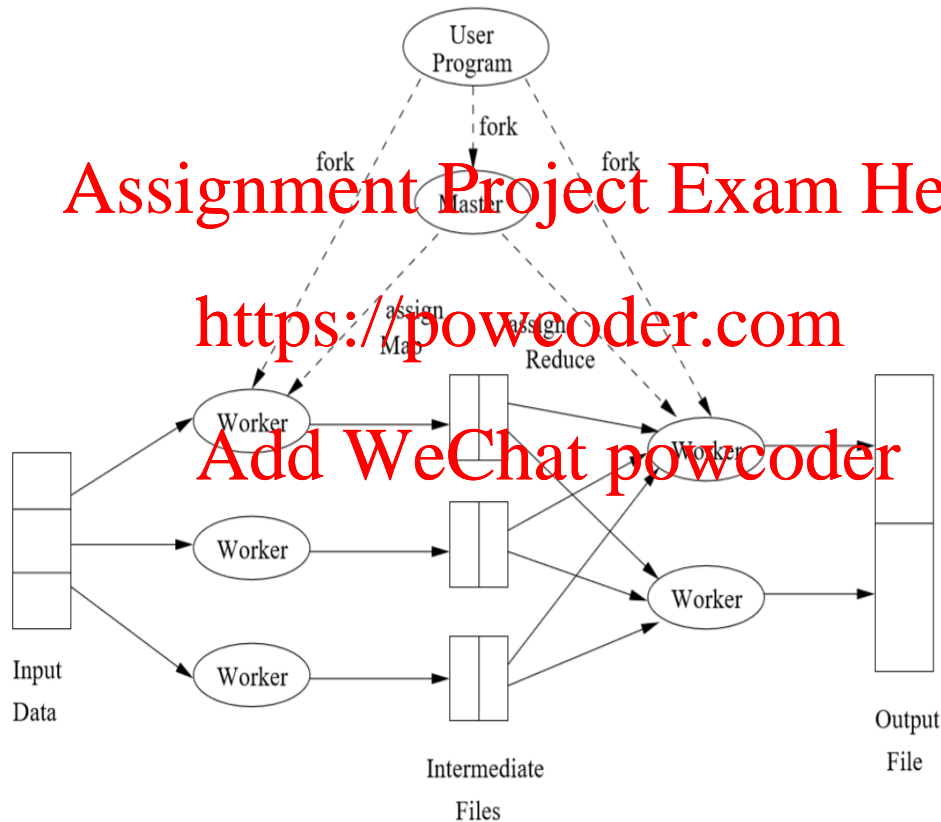


Figure 2.3: Overview of the execution of a MapReduce program

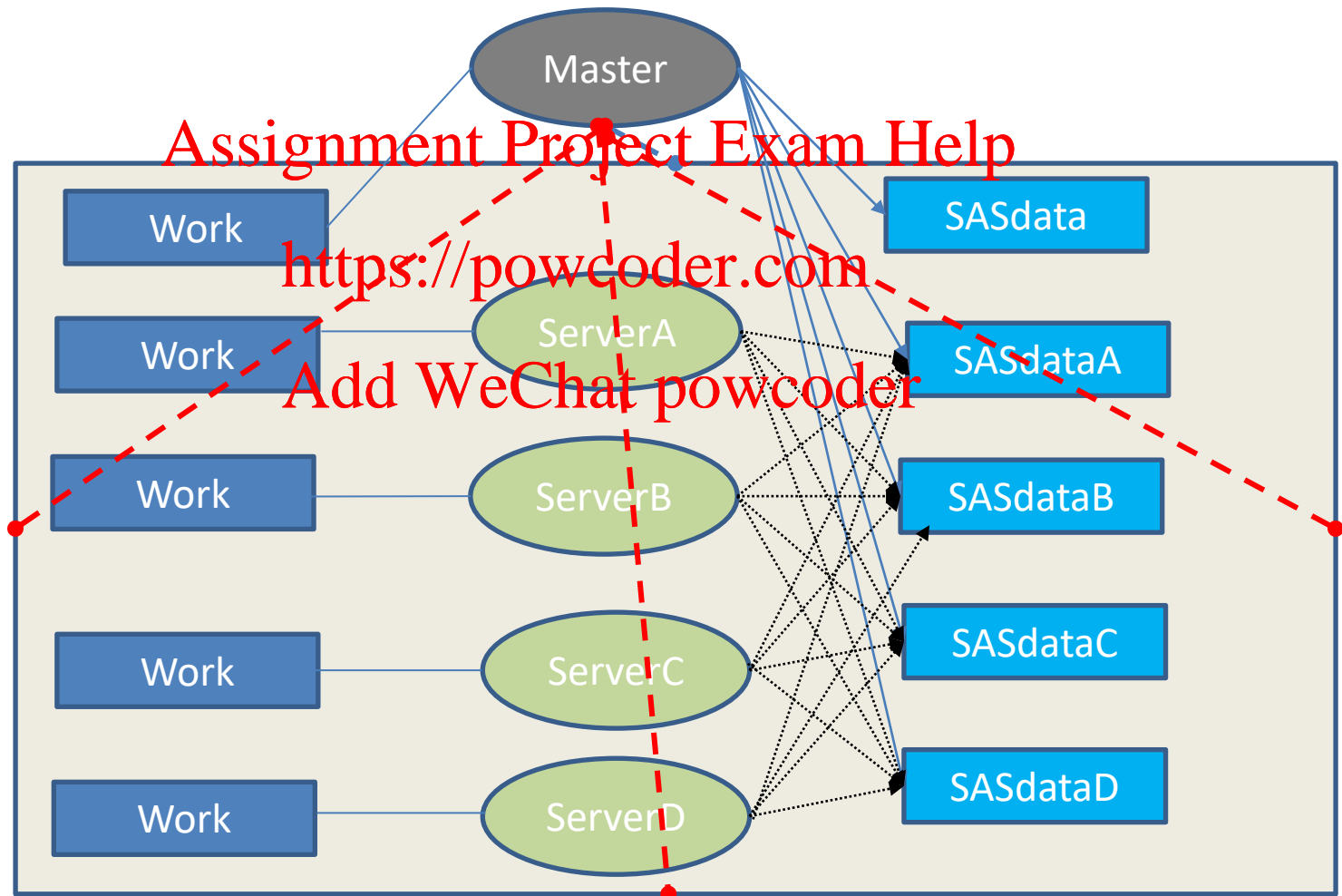
Map-Reduce: Key Value Pair

Simple Example

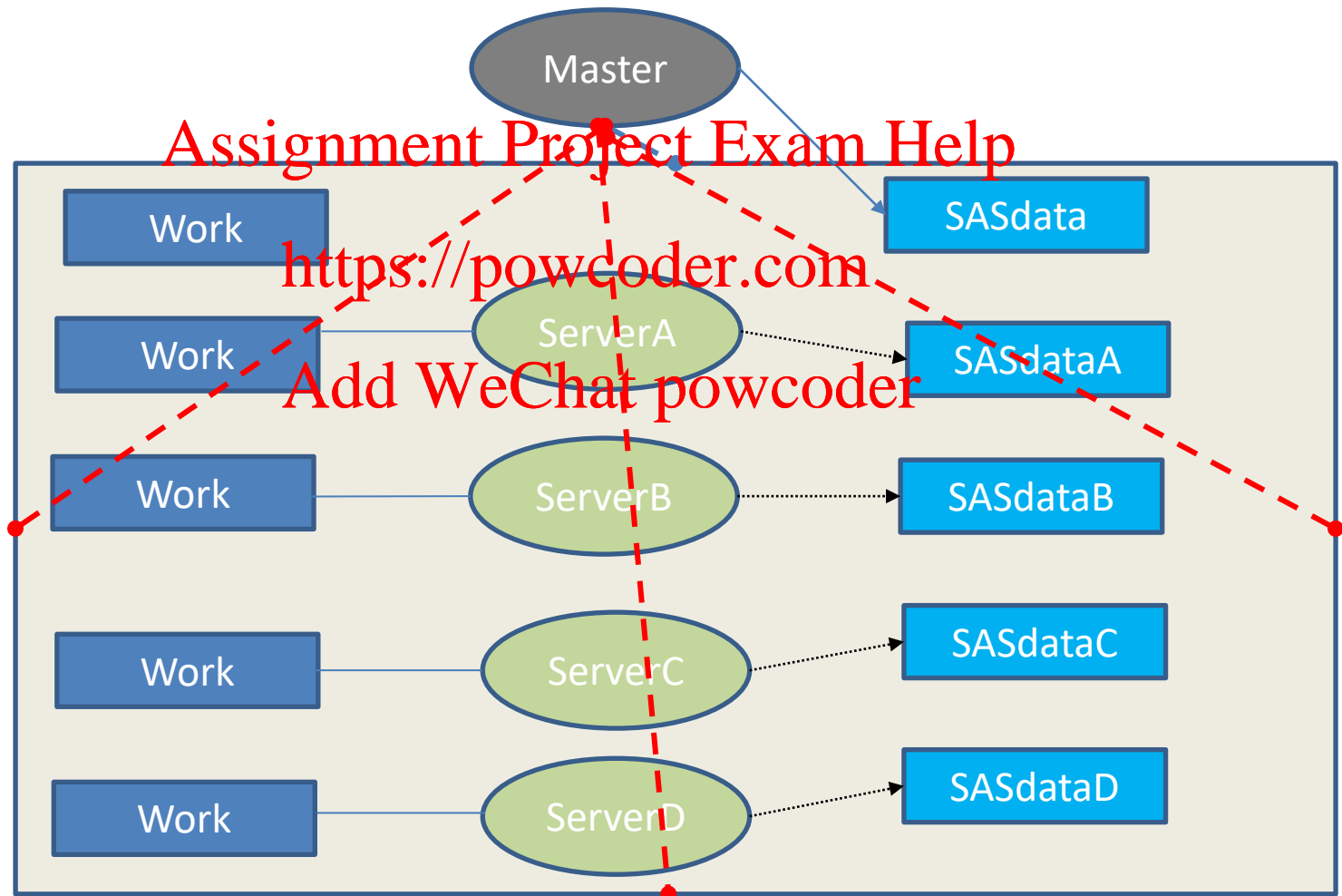
- **Input:** Individual Income
- Map:
 - Map(Account, MaxIncome)
 - Reduce(Account, Max<MaxIncome>*)

Add WeChat powcoder

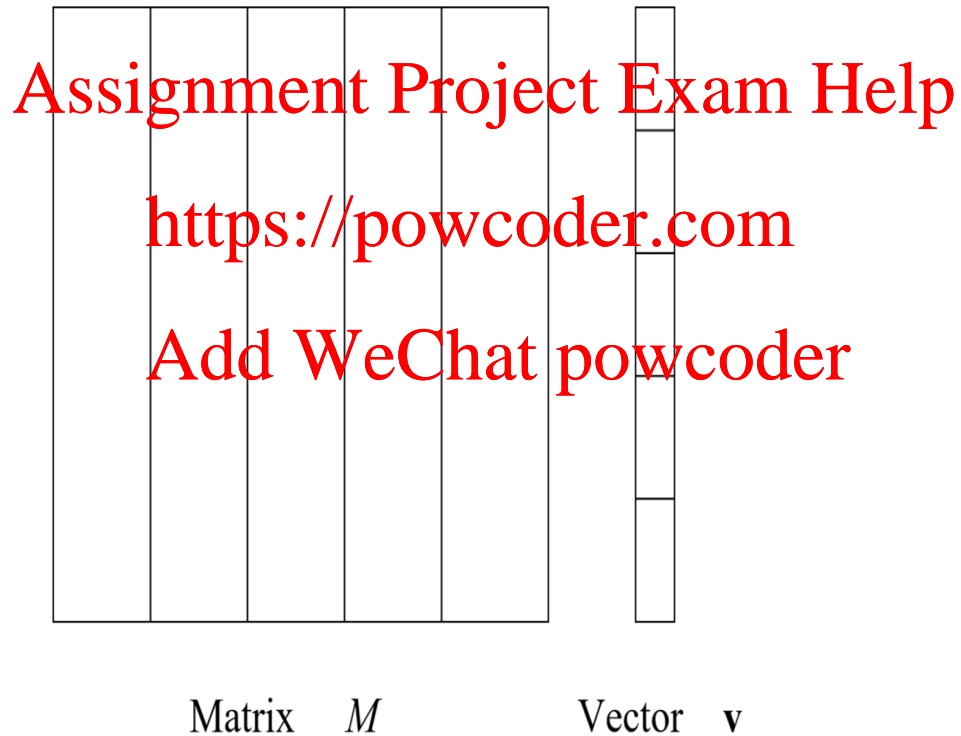
Pseudo Map Reduce Execution in SAS



Pseudo Map Reduce Execution in SAS



Matrix Multiplication



MapReduce: Word Counting

Provided by the programmer

MAP:

Read input and produces a set of key-value pairs

Group by key:

Collect all pairs with same key

Provided by the programmer

Reduce:

Collect all values belonging to the key and output

(The, 1)

(crew, 1)

(of, 1)

(the, 1)

(space, 1)

(shuttle, 1)

(Endeavor, 1)

(recently, 1)

....

(crew, 1)

(crew, 1)

(space, 1)

(the, 1)

(the, 1)

(the, 1)

(shuttle, 1)

(recently, 1)

...

(crew, 2)

(space, 1)

(the, 3)

(shuttle, 1)

(recently, 1)

...

The crew of the space shuttle Endeavor recently returned to Earth as ambassadors, harbingers of a new era of space exploration. Scientists at NASA are saying that the recent assembly of the Dextre bot is the first step in a long term space based man/machine partnership. "The work we're doing now -- the robotics we're doing - is what we're going to need

Big document

(key, value)

(key, value)

(key, value)

Only sequential reads

J. Leskovec, A. Rajaraman, J. Ullman:

Mining of Massive Datasets,

<http://www.mmms.org>

Word Count Using MapReduce

map(key, value):

```
// key: document name; value: text of the document
for each word w in value:
    emit(w, 1)
```

Assignment Project Exam Help

<https://powcoder.com>

reduce(key, values):

```
// key: a word; value: an iterator over counts
result = 0
for each count v in values:
    result += v
emit(key, result)
```