Big Data - Hadoop/MapReduce

Logistics

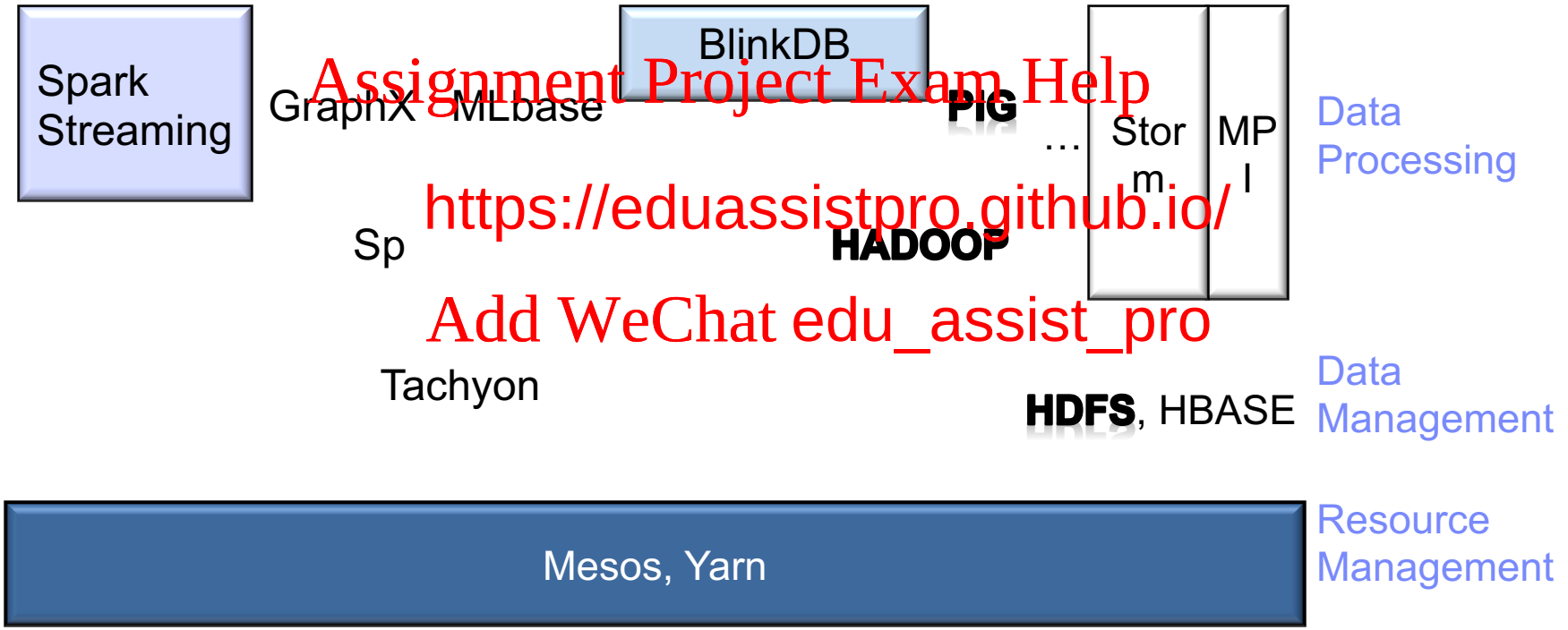
- Quiz 1 next week. Sample released.
- Assignment 2 (tentative) draft uploaded.
- In-depth exploration topics
  - Kubernets and DevOps: you will explore containers, kubernetes and its application to cloud application devOps pipeline.
  - Kafka and Events/Logs: yo m used for handling events/logs including internals, use cas
  - Elastic Search and data pi h how it is used for handling unstructured data and build data pipeline to process ata efficiently.
  - Spark and cluster computing: learn how to effectivel le and build use cases for demonstrating compute in a cluster including scale, rformance tuning.
- Paper reviews

Cluster Computing

- Designing a cluster
- Automation of tasks in a cluster
- Programming in a cluste
  - Hadoop/MapRedue
  - Spark

Big Data System Architecture

- Current system modules for each software layer



Agenda

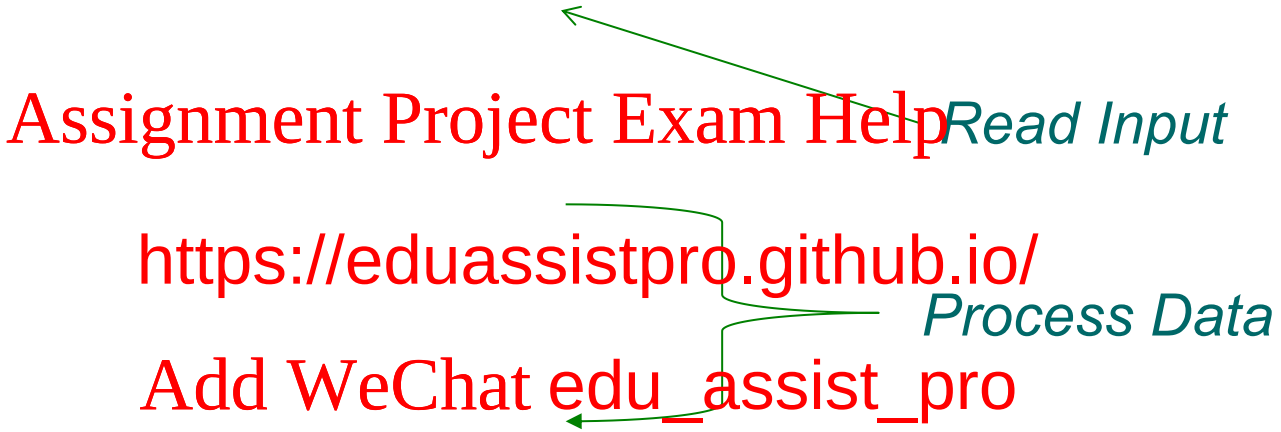
- Why Big Data?

- Apache Hado
  - Introduction
  - Architecture
  - Programming

Hypothetical Job

- You just got an awesome job at data-mining start-up .. Congratulations !!
  - Free Snacks, Soda and Coffee -- Yay!
- Your first day of work you are gi
  - The company has a new algorithm they want you to test.
  - Your boss gives you
    - The algorithm library
    - A test machine and
    - 1GB input data file

Java Document Scorer Program



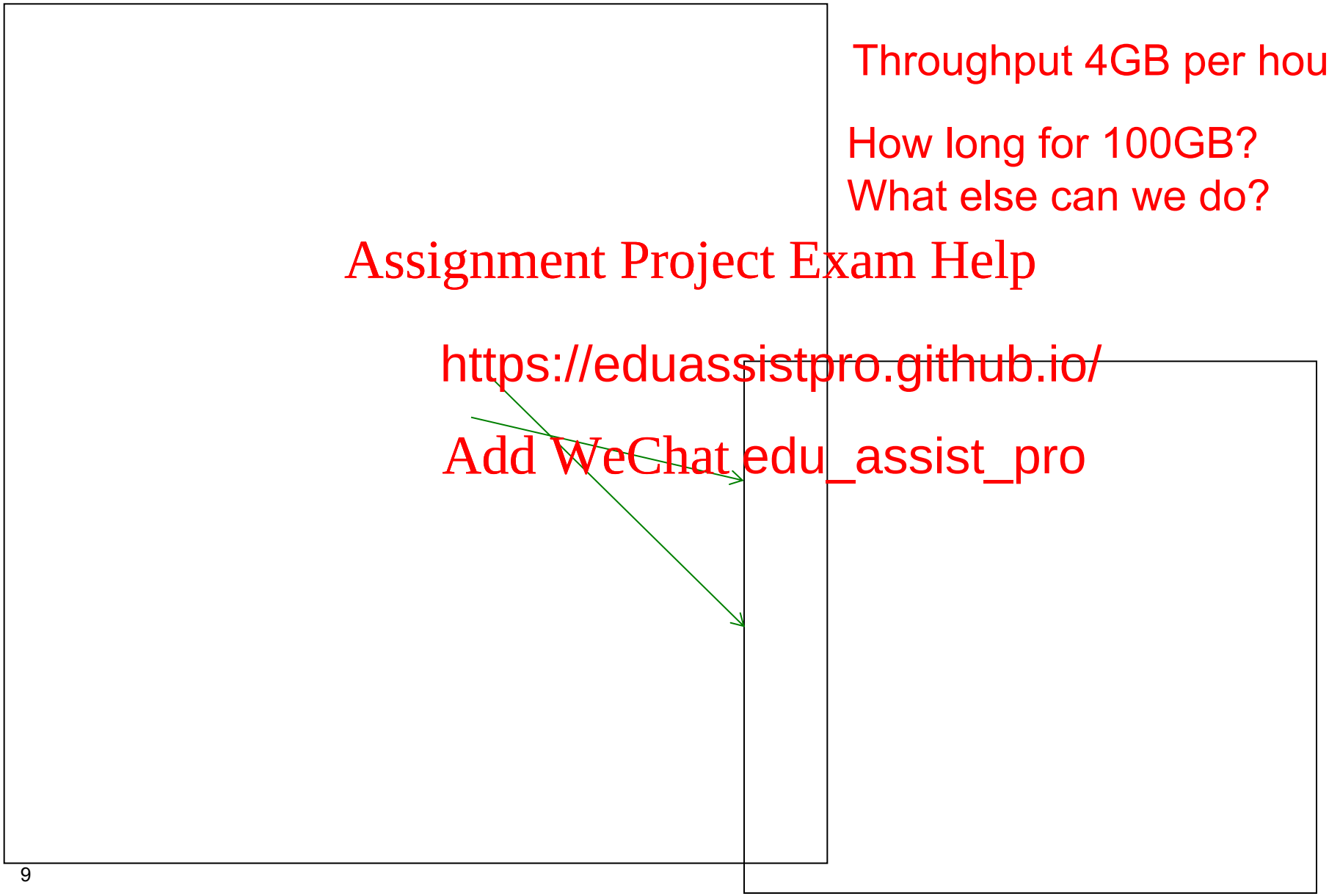
Throughput 1GB per hour.

What if we wanted to process 10GB data set? 10hours!!
How can we improve the performance?

Some Options

1. Faster CPU
2. More Memory
3. Increase the number of cores
4. Increase the number o
5. Increase the number of threads and cores

Java Document Scorer Program – Multi Threaded



Throughput 4GB per hour.

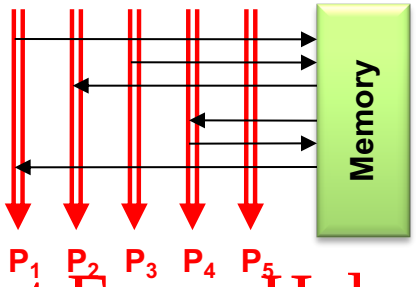
How long for 100GB?
What else can we do?

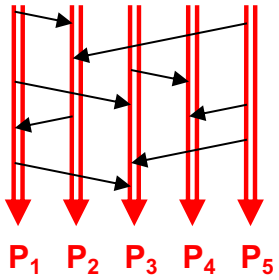
Get An Even Faster Machine with more Cores?

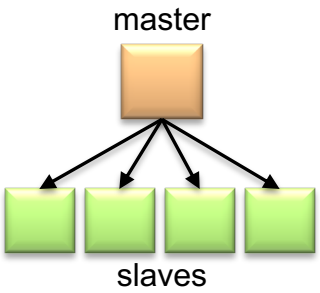
Source: MIT Open Courseware

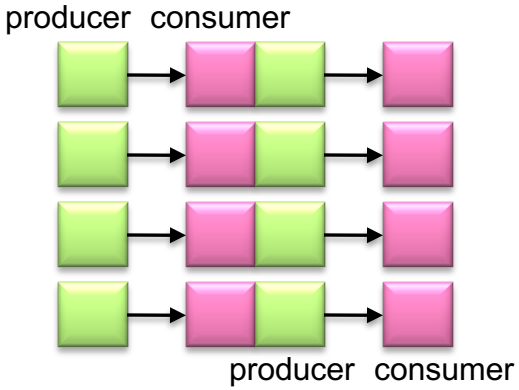
Current Tools

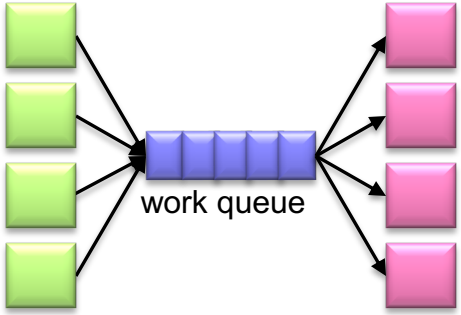
- Programming models
  - Shared memory (pthreads)
  - Message passing (MPI)
- Design Patterns
  - Master-slaves
  - Producer-cons
  - Shared work queues











Where the rubber meets the road

- Concurrency is difficult to reason about
- Concurrency is even more difficult to reason about
  - At the scale of datacenters (even across datacenters)
  - In the presence of failures
  - In terms of mul
- Not to mention
- The reality:
  - Lots of one-off solutions, custom code
  - Write you own dedicated library, then program with it
  - Burden on the programmer to explicitly manage everything

What's the common theme?

- To improve performance, you have to re-write the code
- The code has to adapt to the expected performance.
  - This doesn't work since you may not know the amount of data beforehand.
- The actual Intellectual Propert e company is the analytic algorithm
  - However a lot of effort is spent on scaling the analytic

Big Data - Motivation

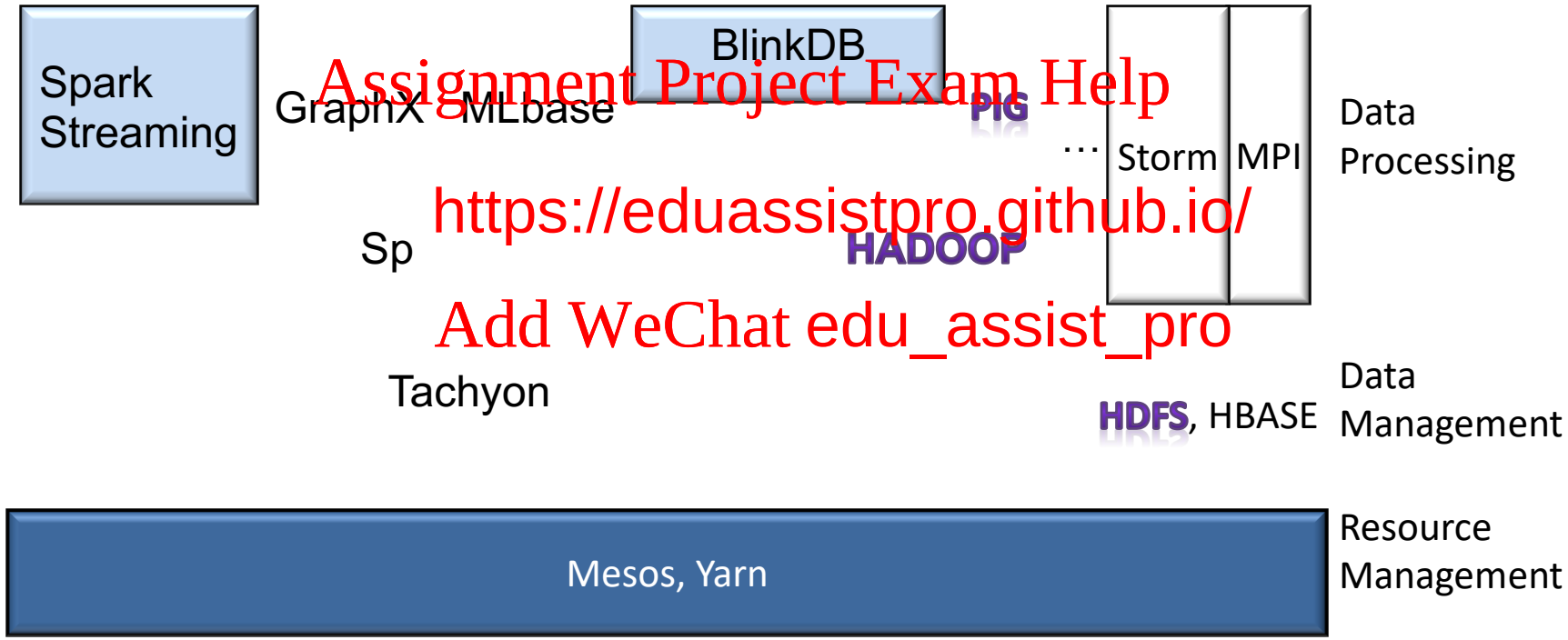
- Google processes 20 PB a day (2008)
- Wayback Machine has 3 PB + 100 TB/month (3/2009)
- Facebook has 2.5 PB of user data + 15 TB/day (4/2009)
- eBay has 6.5 PB o /2009)
- CERN's LHC will g

Enter .. Apache Hadoop

- Hadoop is a high-level Open Source project
  - Under Apache Software Foundation
  - Inspired by Google's MapReduce and GFS papers
- It contains several individual projects
  - HDFS
  - MapReduce
  - Yarn
- It also has a slew of related projects
  - PIG
  - HIVE
  - Hbase
- Has been implemented for the most part in Java.

Big Data System Architecture

- Current system modules for each software layer

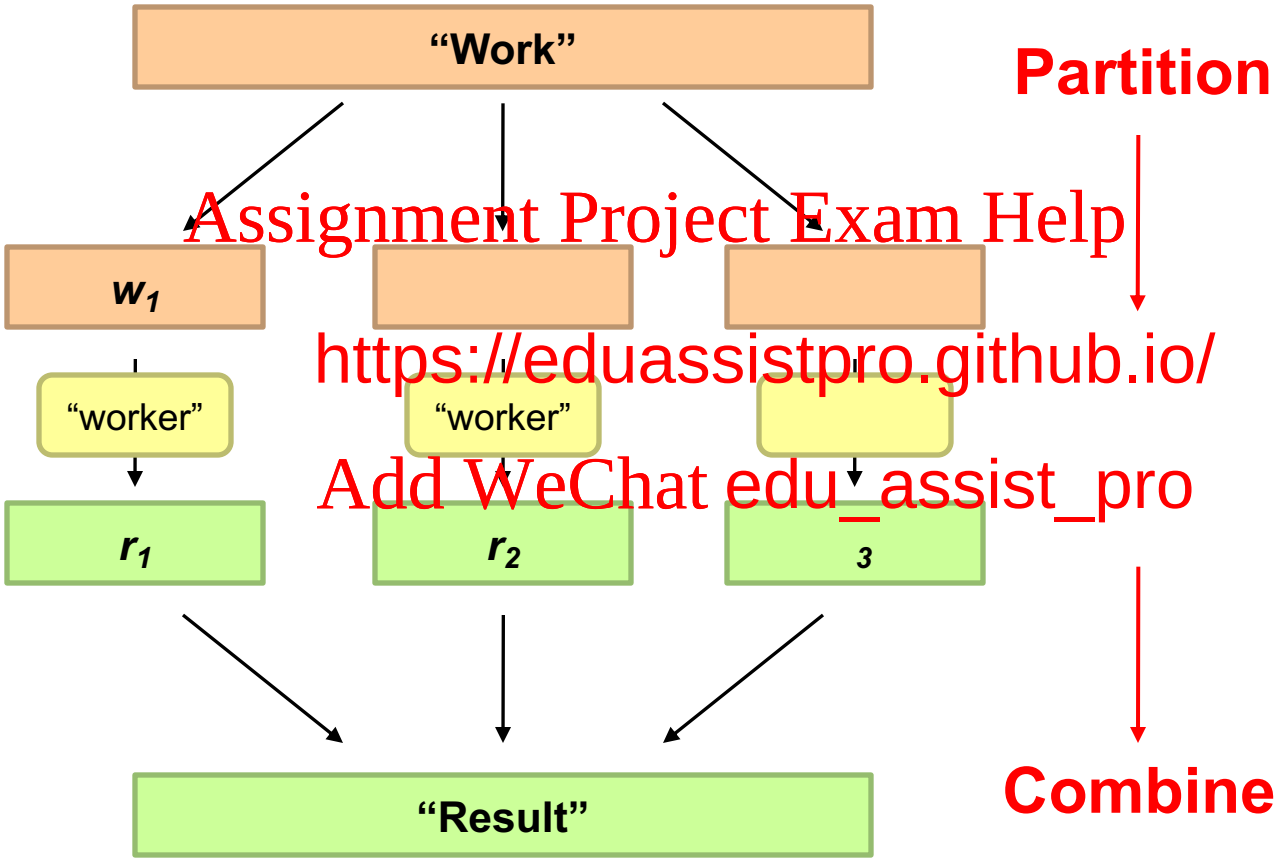


A closer look

tion Work

Combine Results

Divide and Conquer



Parallelization Challenges

- How do we assign work units to workers?
- What if we have more work units than workers?
- What if workers need to share partial results?
- How do we agg
- How do we kno inished?
- What if workers die?

What is the common theme of all of these problems?

What's the point?

- It's all about the right level of abstraction
  - The von Neumann architecture has served us well, but is no longer appropriate for the multi-core/cluster environment
- Hide system-level details from the developers
  - No more race tc.
- Separating the
  - Developer specifies the comput ds to be performed
  - Execution framework ("runtime") al execution

The datacenter *is* the computer!

“Big Ideas”

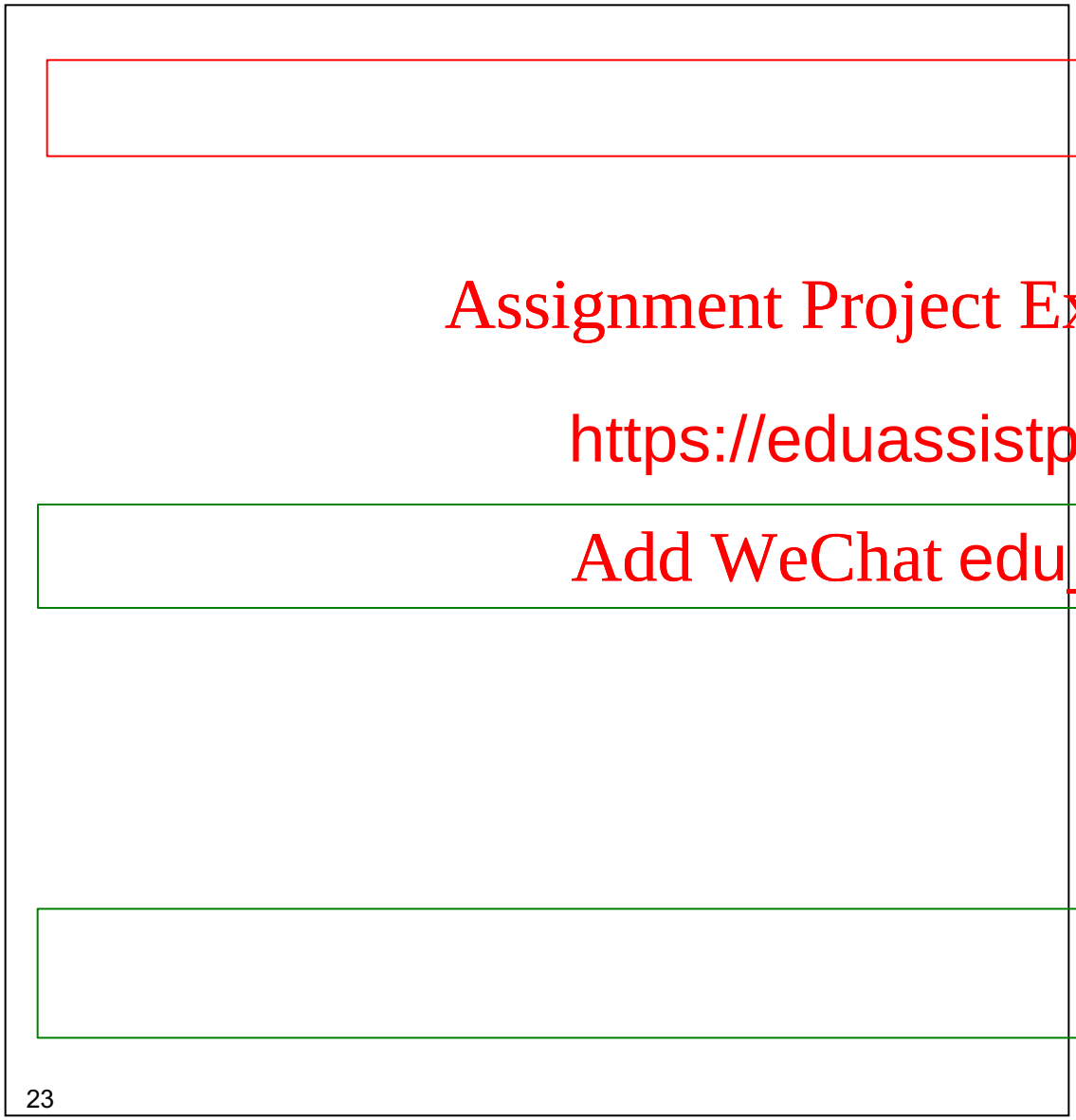
- Scale “out”, not “up”
  - Limits of SMP and large shared-memory machines
- Move processing to the data
  - Cluster have limited bandwidth
- Process data s m access
  - Seeks are exp asonable
- Seamless scalability
  - From the mythical man-month to the tradable machine-hour

Hadoop

- Platform for distributed **storage** and **computation**
  - **HDFS**
  - **MapReduce**
  - **Ecosystem**

Source: **Hadoop in Practice**, Alex Holmes, Manning Publications Co., 2012

What are we missing here?

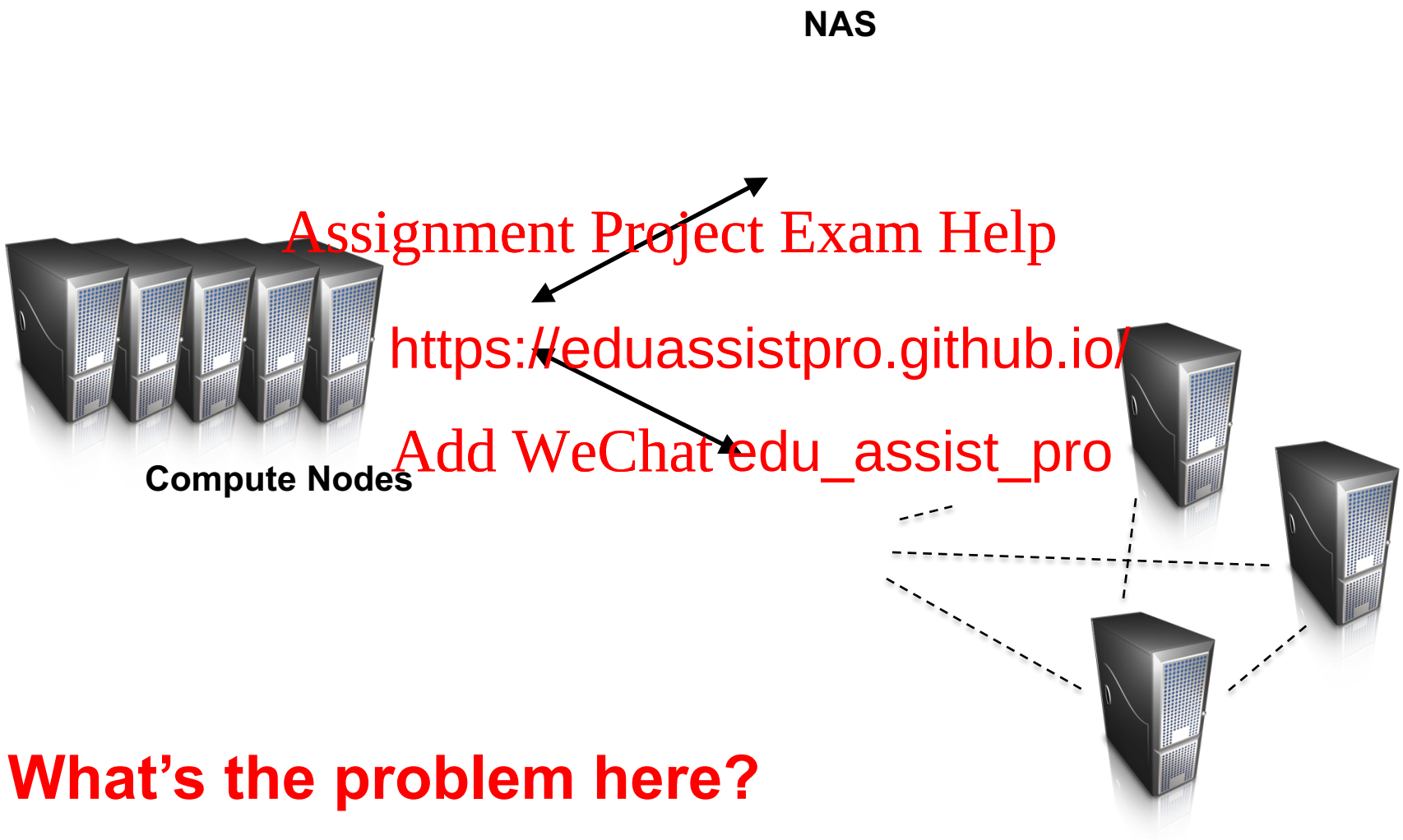


Hadoop

- Platform for distributed **storage** and **computation**
  - **HDFS**
  - **MapReduce**
  - **Ecosystem**

Source: **Hadoop in Practice**, Alex Holmes, Manning Publications Co., 2012

How do we get data to the workers?

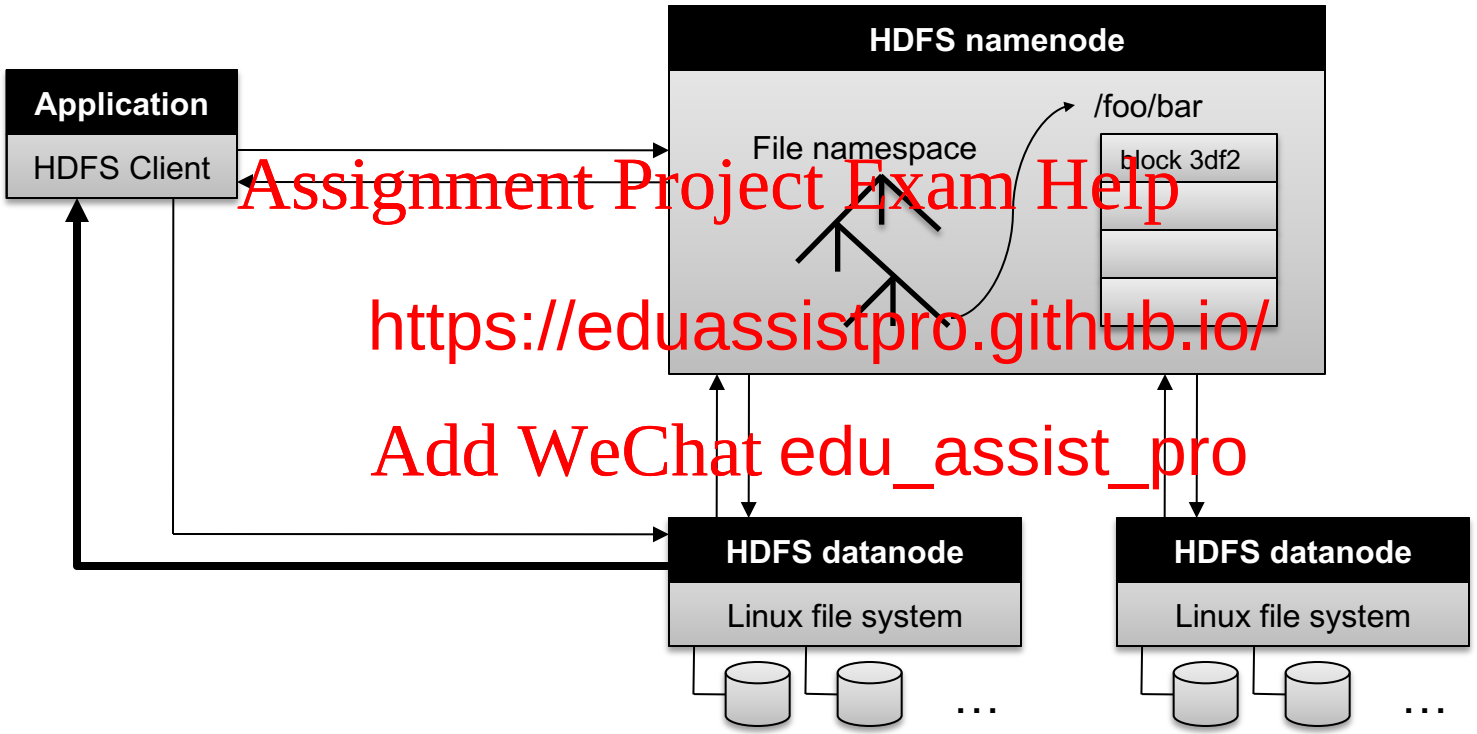


What's the problem here?

HDFS: Assumptions

- Commodity hardware over “exotic” hardware
  - Scale “out”, not “up”
- High component failure rates
  - Inexpensive commodity components fail all the time
- “Modest” number of huge files
  - Multi-gigabyte files are
- Files are write-once, rea
  - Perhaps concurrently
- Large streaming reads over random access
  - High sustained throughput over low latency

HDFS Architecture



Adapted from (Ghemawat et al., SOSP 2003)

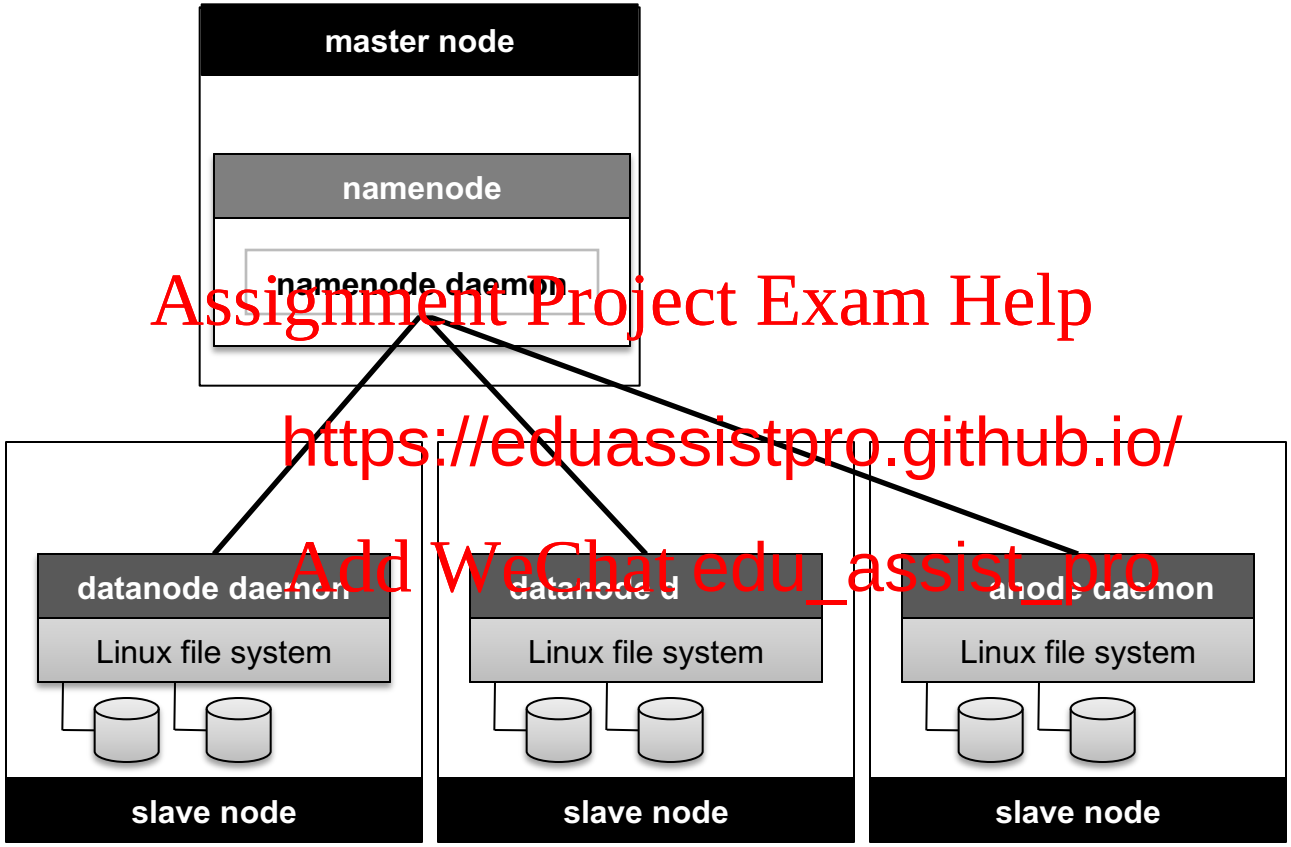
How HDFS works

- When an input file is added to HDFS
  - File is split into smaller blocks of fixed size
  - Each block is replicated
  - Each replicated block is stored on a different host
- Block size is configurable. Default is 128/256MB.
- Replication level is c
  - Replication is nec
    - Scaling
    - High Availability
- In case a host crashes or is removed
  - All blocks on that host are automatically replicated to other hosts
- In case a host is added
  - Blocks will be rebalanced so that some blocks from other hosts will be placed on the new host

HDFS Component Responsibilities

- Name Node
  - Managing the file system namespace:
    - Holds file/directory structure, metadata, file-to-block mapping, access permissions, etc.
  - Coordinating file operations:
    - Directs clients to datanodes for reads and writes
    - No data is moved through the namenode
  - Maintaining overall heal
    - Periodic communicatio
    - Block re-replication an
    - Garbage collection
- Data Node
  - Actual storage and management of data block on a single host
  - Provides clients with access to data

HDFS Components in Cluster



Hadoop

- Platform for distributed **storage** and **computation**
  - **HDFS**
  - **MapReduce**
  - **Ecosystem**

Source: **Hadoop in Practice**, Alex Holmes, Manning Publications Co., 2012

MapReduce (MR) can refer to…

- The execution framework (aka “runtime”)
- The programming model
- The specific implementation

Usage is usually clear from context!

MR Framework Components

- Job Tracker
  - Central component responsible for managing job lifecycles
  - One Job Tracker per MR framework instance
  - Accepts job submissions, queries etc. from clients
  - Enqueues jobs and schedules individual tasks.
  - Communicates wit nd run tasks
  - Attempts to assign ity.

- Task Tracker
  - One Task Tracker per host
  - Runs and manages individual tasks
  - Communicates progress of tasks back to Job Tracker.

MR Programming Model

- Programmers specify two functions:

  map (k, v) → <k', v'>*

  reduce (k', v') → <k', v'>*

  - All values with the same key are sent to the same reducer

- The MR Execution framework handles everything else…

MapReduce

- Everything Else

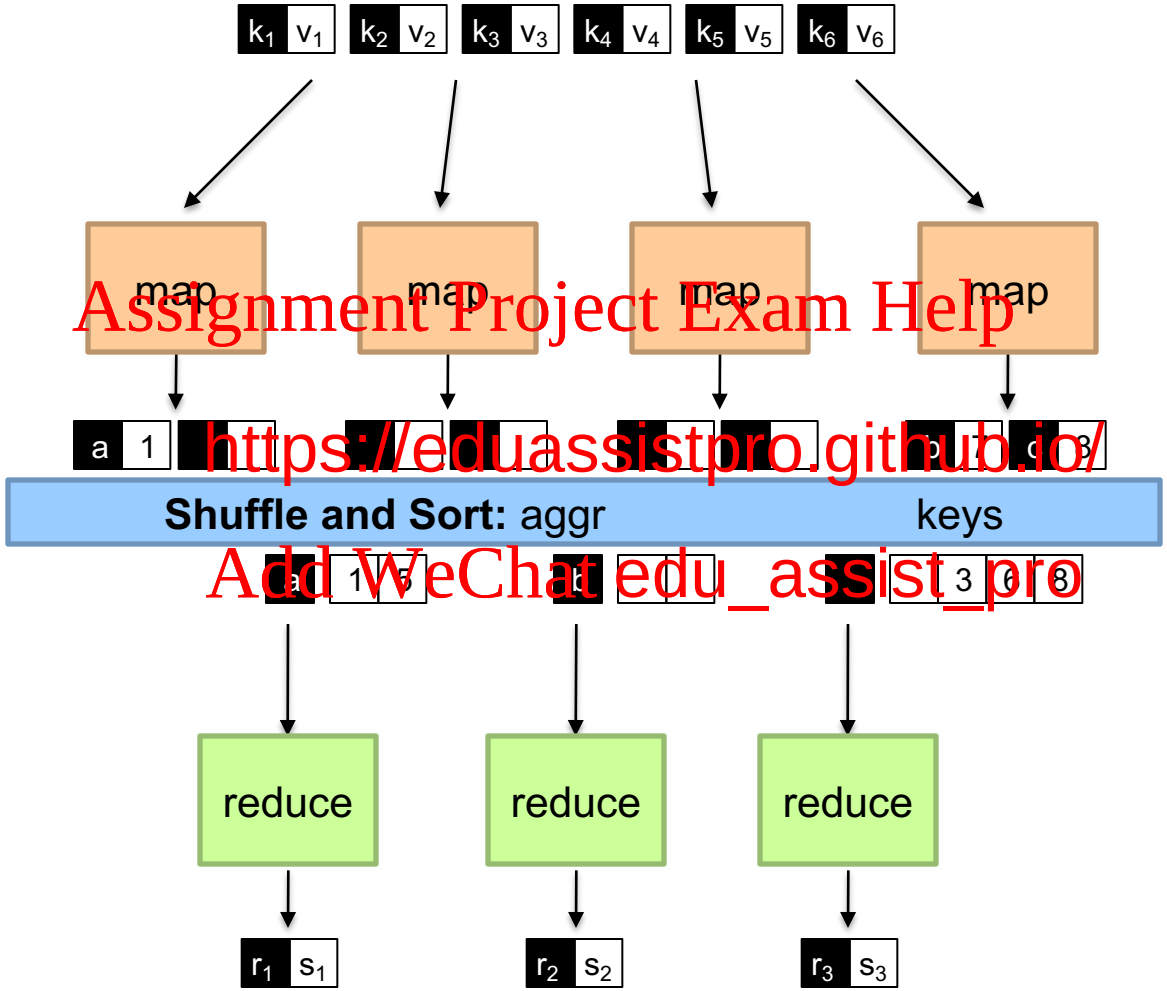
- Handles scheduling
  - Assigns workers to map and reduce tasks
- Handles "data distribution"
  - Moves processes to dat
- Handles synchronization
  - Gathers, sorts, and shuffles intermediate data
- Handles errors and faults
  - Detects worker failures and restarts
- Everything happens on top of a distributed FS (HDFS)

Our Scoring Algorithm as a Map Reduce Program

Basic Hadoop API*

- Mapper
  - void map(K1 key, V1 value, OutputCollector<K2, V2> output, Reporter reporter)
  - void configure(JobConf job)
  - void close() throws IOException
- Reducer/Comb
  - void reduce(K2 key, Iterator<V2 OutputCollector<K3,V3> output, orter)
  - void configure(JobConf job)
  - void close() throws IOException
- Partitioner
  - void getPartition(K2 key, V2 value, int numPartitions)

*Note: forthcoming API changes…



Lets Talk Numbers

- How many mappers?
  - Depends on the size of input data
  - Typically 1 mapper per data block
  - So 1 GB input data will have around 8 Mappers
    - Assuming 128MB block size
- How many red
  - Depends on cluster reducer cap
  - Can be set depending on the ex er of keys
  - For large data sets, set it to cluster reducer capacity

MapReduce

- Programmers specify two functions:

  map (k, v) → <k', v'>*

  reduce (k', v') → <k', v'>*

  - All values with the same key are reduced together

- The execution framework handles everything else…

- Not quite…usu specify:

  combine (k', v') → <k', v'>*

  - Mini-reducers that run in memor p phase
  - Used as an optimization to reduce network traffic

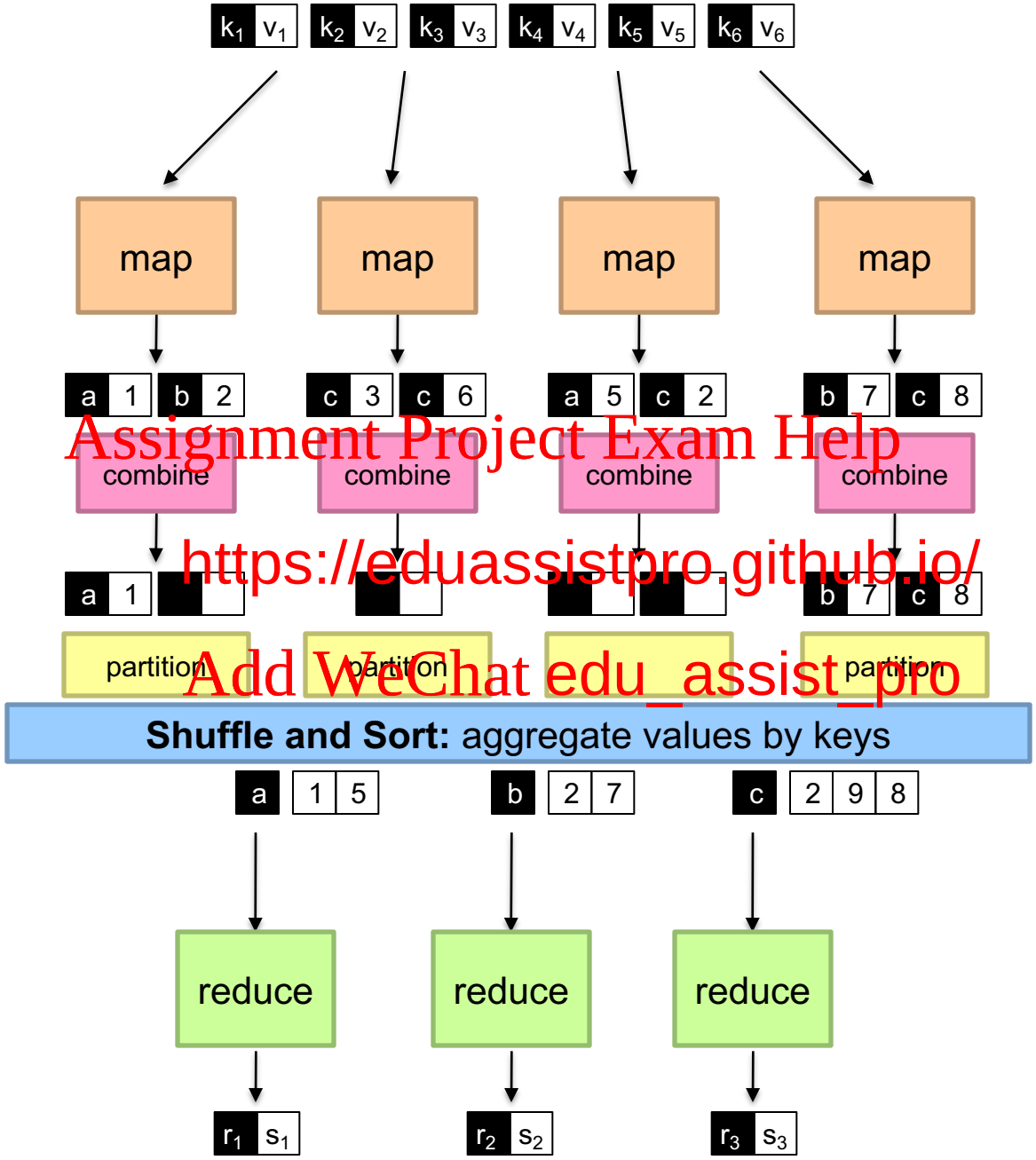
  partition (k', number of partitions) → partition for k'

  - Often a simple hash of the key, e.g., hash(k') mod n
  - Divides up key space for parallel reduce operations

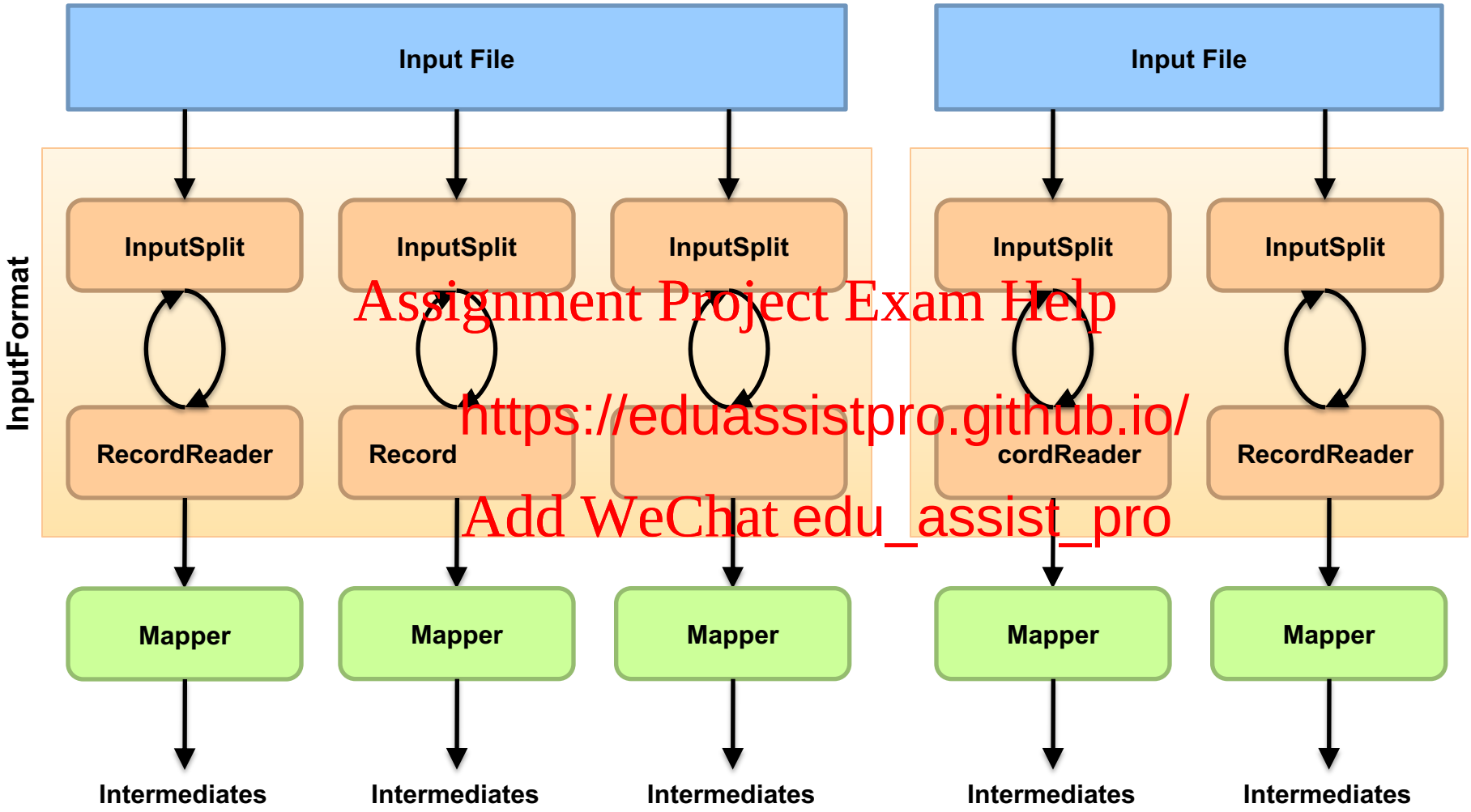
Two more details…

- Barrier between map and reduce phases
  - But we can begin copying intermediate data earlier

- Keys arrive at each reducer in sorted order
  - No enforced or

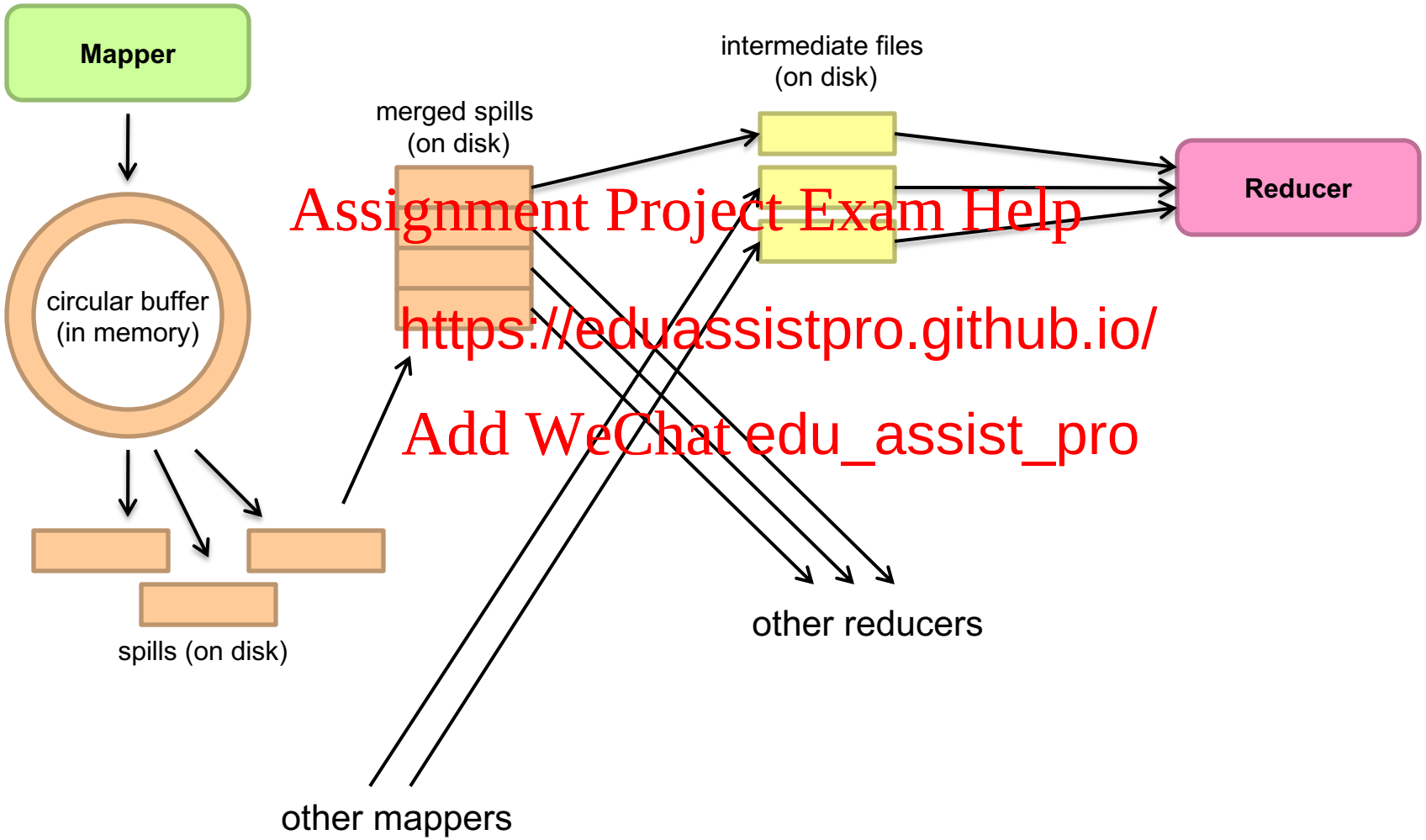


Input To Mappers



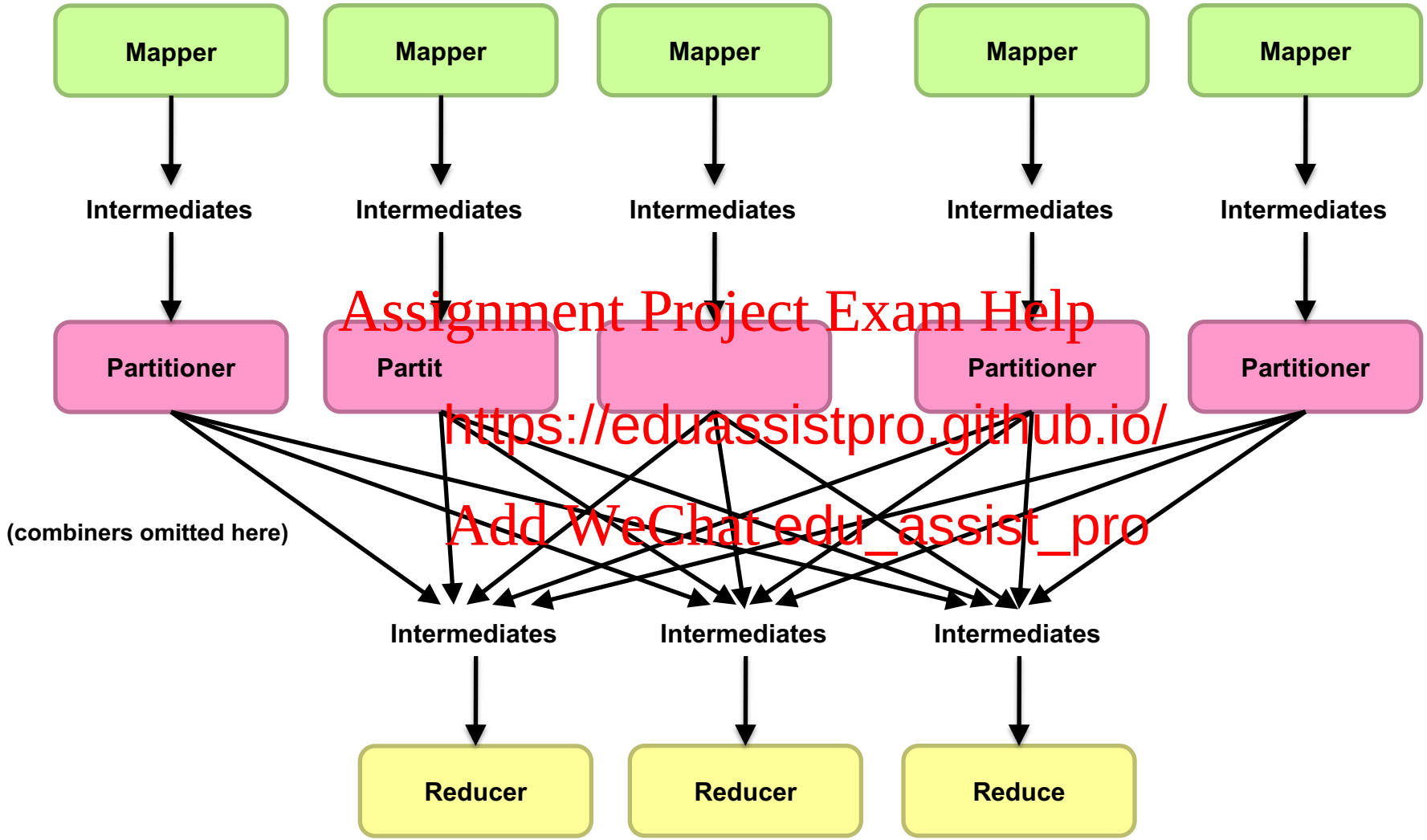
Source: redrawn from a slide by Cloduera, cc-licensed

Shuffle and Sort



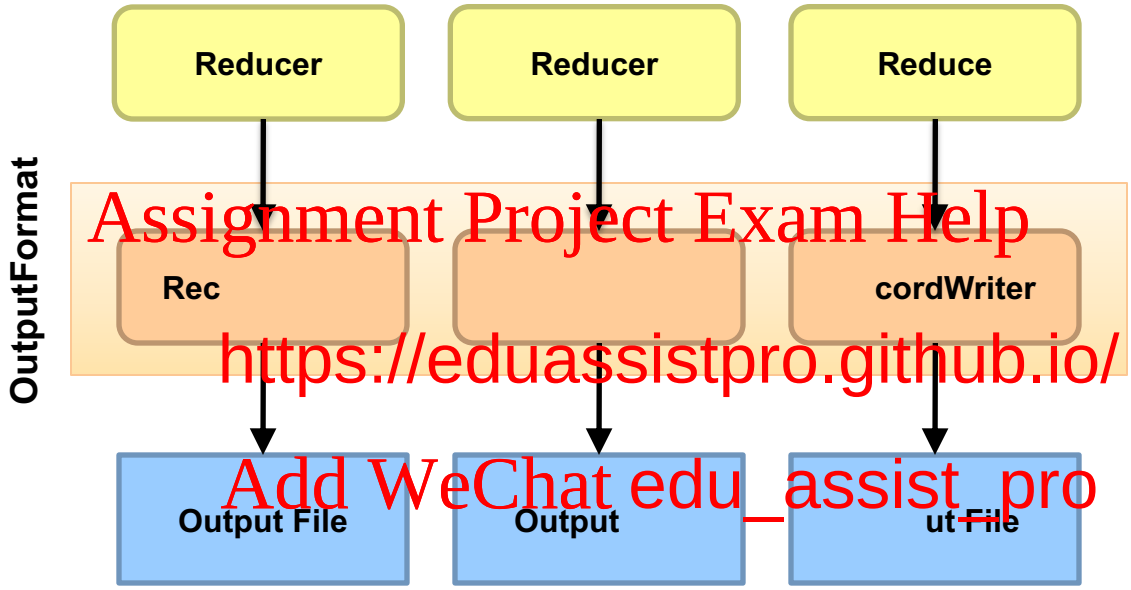
Shuffle and Sort in Hadoop

- Probably the most complex aspect of MapReduce!
- Map side
  - Map outputs are buffered in memory in a circular buffer
  - When buffer reaches threshold, contents are "spilled" to disk
  - Spills merged i rted within each partition): com
- Reduce side
  - First, map outputs are copied ov machine
  - "Sort" is a multi-pass merge of map outputs (happens in memory and on disk): combiner runs here
  - Final merge pass goes directly into reducer



Source: redrawn from a slide by Cloduera, cc-licensed

Reducer to Output

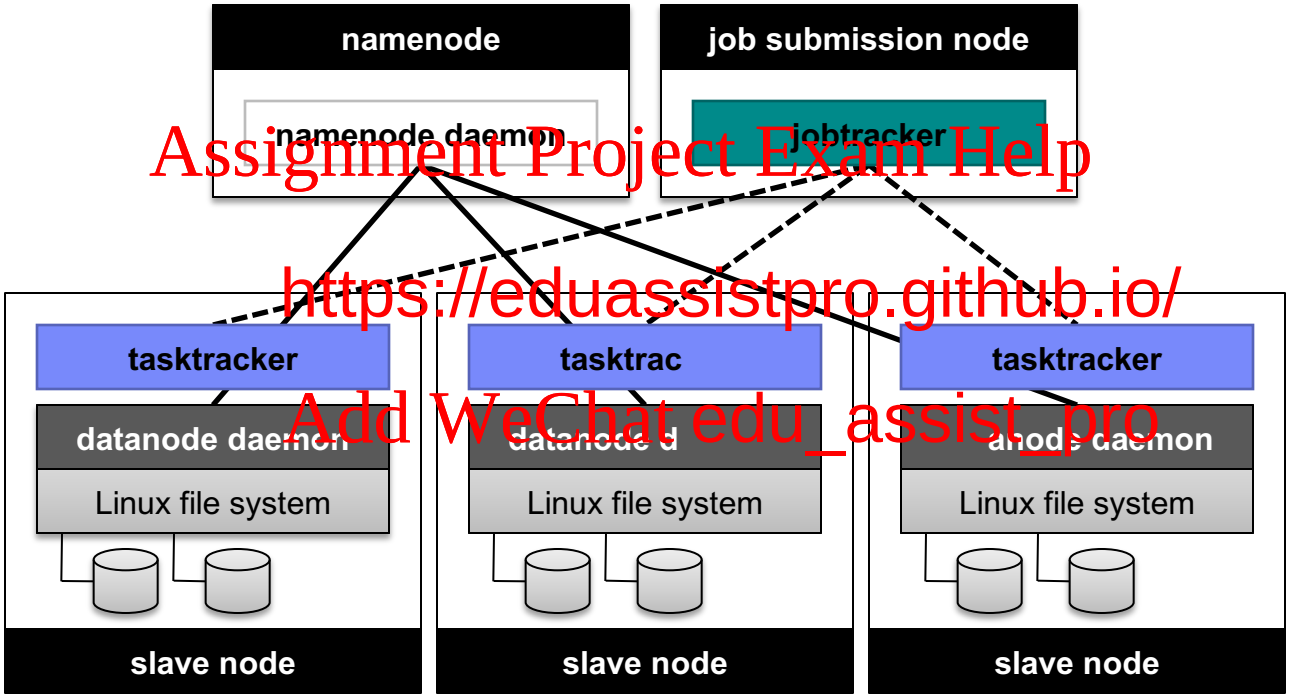


Source: redrawn from a slide by Cloduera, cc-licensed

Input and Output

- InputFormat:
  - TextInputFormat
  - KeyValueTextInputFormat
  - SequenceFileInputFormat
  - …
- OutputFormat:
  - TextOutputFormat
  - SequenceFileOutputFormat
  - …

Putting everything together…



HADOOP Architecture

- Master
  - NameNode
  - JobTracker
- Slaves
  - Data Node
  - Compute Node
  - Why together?
    - **Data Locality**

One More Thing

- Distributed Cache
  - Usually used for files of small size
  - Provides a convenient way to propagate applications and configuration files
  - HDFS is not used handle such files due to their small size
  - Shared across all ster

Dizzy Yet?

- OK, we went through a lot of details
- Whatever happened to the simplicity of programming??
- Do I really have to write [illegible] me I want to run a new analytic?

We went from..

Multi-Threaded

Map-Reduce

Enter PIG … Oink!

- High Level Languages for Map-Reduce
  - PIG
    - Developed by Yahoo
  - HIVE
    - Developed by Facebook
  - JAQL
    - Developed by IBM
- All of these languages p
- All of them allow users to plug in their own nctions (UDFs)

Lets get Practical – From Setup to Results

Setting up a Hadoop Cluster

- Minimum recommended configuration (4 Hosts)
  - 1 Host Dedicated for Management Services (Job Tracker, Name Node etc)
  - 3 Hosts as Slave nodes (Data Node , Task Trackers)
- Data nodes should have ed.
  - This is where all your d
- How much total disk space?
  - Depends on input data to be processed
  - Effective Storage Space Recommended: Typ ur input data size
  - Actual Storage Space: Effective Storage Space * 3 (replication level)
- Single node installation is fine for development/testing on very small data
  - Perhaps not the best for testing performance
- Installation instructions vary from provider to provider

Some cluster configuration parameters

- HDFS configuration parameters
  - Stored in hdfs-site.xml
  - Block size
  - Default replication count
- MapReduce configuratio
  - Stored In “mapred-site.xml”
  - Java heap size for mappers/reducers
  - Number of mappers/reducers per host
    - See http://wiki.apache.org/hadoop/HowManyMapsAndReduces
- *IMPORTANT*
  - Job Tracker URL: http://<masterhost>:50030
  - Name Node URL: http://<masterhost>:50070

Job Tracker Web Page (port 50030)

Working with data

- Lets say you have 1 GB of data in your local filesystem (mydata.txt)
- Load into HDFS
  - hadoop fs –mkdir /path/mydirectory
  - hadoop fs –put mydata.txt /path/mydirectory
  - where /path/mydirectory is in HDFS
- List the file you just uplo
  - hadoop fs –ls /path/mydirectory
- “hadoop fs” works similar to linux filesystem commands
  - However HDFS is not POSIX compliant.
  - It cannot be mounted as a regular filesystem

Writing your program .. see the simplicity!!

- JAQL program for running our scorer

- PIG program for running our scorer

All languages provide similar functionality

- LOAD (various data formats)
- JOIN
- FOR-EACH
- GROUP
- SORT
- FILTER
- Pluggable UDFs

Hadoop Programming Tips

- Thinking at scale
  - Filter unwanted data earlier in the flow
  - Store intermediate data
  - Use “sequence” format for storing data.
- These are not iterativ
  - i.e. No *for* or *while* lo
- Watch out for obvious bottlenecks
  - Single key for all mapper output will sen educer
  - Too much data sent to a UDF will result in OOM errors

Submitting a Job

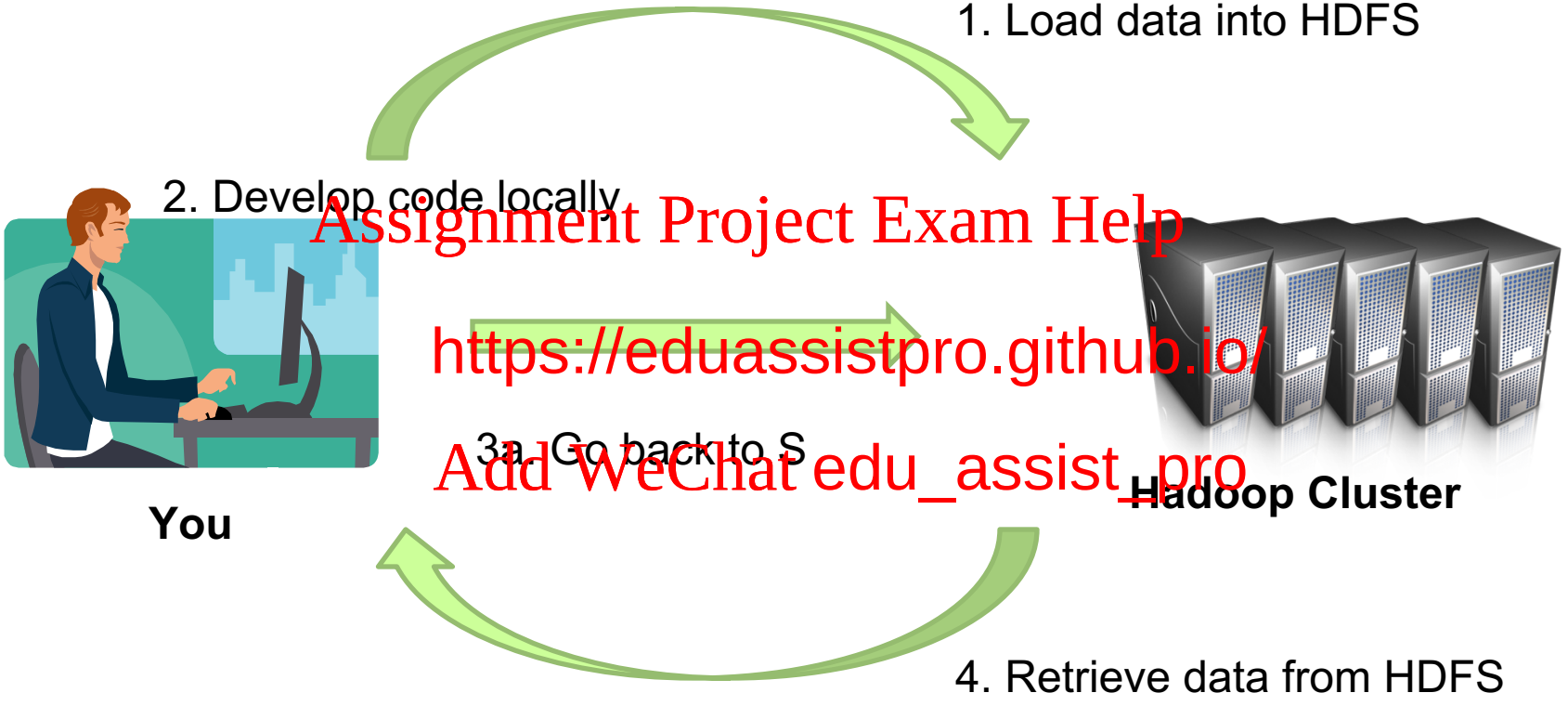
- Create and save your PIG script (myscript.pig)
- To deploy (pig command will be in your installation)
  - pig –f myscipt.pig
  - Command will complete once your job completes
- To check the status of y
  - Use the Job Tracker U
  - hadoop job –list (will print all job ids)
  - hadoop job –status <jobid> (will print the job
- To get the results
  - hadoop fs –get /path/results.txt .

Anatomy of a Job

- MapReduce program in Hadoop = Hadoop job
  - Jobs are divided into map and reduce tasks
  - An instance of running a task is called a task attempt
  - Multiple jobs can be composed into a workflow
- Job submission
  - Client (i.e., driv onfigures it, and submits it to job tracker
  - JobClient computes input splits ( )
  - Job data (jar, configuration XML) are sent to JobTracker
  - JobTracker puts job data in shared location, enqueues tasks
  - TaskTrackers poll for tasks
  - Off to the races…

A simple illustration of MR process

Hadoop Workflow



Larger View...

MR Patterns Examples

- Jimmy Lin's book
- Jeffrey Ulman's book
- An excellent blog: https://highlyscalable.wordpress.com/2012/02/01/mapreduce-patterns/

Uh Oh.. My Job Failed...Now what?

- First, take a deep breath
- Start small, start locally
- Strategies
  - Learn to use the webapp
  - Where does pr
  - Don't use printl
  - Throw RuntimeExceptions
- Logs are most easily accessible via the Job Tracker URL

Time for a Raise

- Finally you have mastered Hadoop Big Data
- Your applications are scaling.
  - You deserve a raise!!
- Boss
  - Can we query t
  - How long will t
- Problem
  - Remember this is still sequential access
  - To find a specific entity, you still need to read the entire data set.
- What now?
  - How is this solved in traditional systems?

Databases

Other projects based on Hadoop

- HBase
- Hive
- PIG
- Spark
- Mahout

Hive – a SQL-like data warehouse on Hadoop

https://cwiki.apache.org/confluence/display/Hive/Tutorial

- Supports a SQL-like data warehouse on top of Hadoop – began at Facebook
- Provides SQL users the capability of big data without requiring lower level programming for a wide range of tasks
- Fewer lines of code!
- /bin/hive –help


```
hive> LOAD DATA LOCAL INPATH 'cite75_99.txt'
    > OVERWRITE INTO TABLE cite;
Copying data from file:/root/cite75_99.txt
Loading data to table cite
OK
Time taken: 9.51 seconds
```

Wordcount Example

Hive – SQL-like data warehouse on top of Hadoop

- A data warehouse with SQL like (HiveQL) interface on Hadoop for processing and managing structured data
- Hive hides the complexity of MR programming details from user with interest for processing structured data


```
INSERT OVERWRITE TABLE user_active
SELECT user.*
FROM user
WHERE user.active =1;
```

- Key components
  - Metastore
  - Parser/planner/optimizer
  - Interface

Hive Installation and config

- Just download recent Hive tarball and extract it.
- Need to set up few directories
  - bin/hadoop fs -mkdir /tmp
  - bin/hadoop fs -mkdir /user/hive/warehouse
- Hive manages all the data under this directory
- Hive stores metad atabase

Hive examples

- 11.1.2

Data model – column partition

- Tables as fundamental data model – stored under /usr/hive/warehouse
- Hive uses concept of partition columns – partitions data by column index, i.e., state column, date column partition..

/user/hive/warehouse/users/date=20090901/state=CA
/user/hive/warehouse/users/date=20090901/state=NY
/user/hive/warehouse/users/date=20090901/state=TX ...

/user/hive/warehouse/users/d
/user/hive/warehouse/users/d
/user/hive/warehouse/users/d

/user/hive/warehouse/users/date=20090903/state=CA
/user/hive/warehouse/users/date=20090903/state=NY
/user/hive/warehouse/users/date=20090903/state=TX ...

Hive Data Model – partition and cluster

- Tables stored under user/hive/warehouse in HDFS
- Partition columns

- Buckets – allows to create smaller range partiti


```
CREATE TABLE page_view(viewTime INT, userid BIGINT,
                page_url STRING, referrer_url STRING,
                ip STRING COMMENT 'IP Address of the User')
COMMENT 'This is the page view table'
PARTITIONED BY (dt STRING, country STRING)
CLUSTERED BY (userid) INTO 32 BUCKETS
ROW FORMAT DELIMITED
        FIELDS TERMINATED BY '\t'
        LINES TERMINATED BY '\n'
STORED AS SEQUENCEFILE;
```

Data model - buckets

- concept of *buckets*, which provide efficiency to queries that can work well on a random sample of data
- Bucketing divides data into a specified number of files based on the hash of the bucket column

/user/hive/warehouse/users/date=20090901/state=CA/part-00000 ...

/user/hive/warehou 00031

/user/hive/warehou 00000 ...

/user/hive/warehouse/users/date=20090901 31

/user/hive/warehouse/users/date=20090901 00

- Example - pageview

```
CREATE TABLE page_view(viewTime INT, userid BIGINT,
                page_url STRING, referrer_url STRING,
                ip STRING COMMENT 'IP Address of the User')
COMMENT 'This is the page view table'
PARTITIONED BY (dt STRING, country STRING)
CLUSTERED BY (userid) INTO 32 BUCKETS
ROW FORMAT DELIMITED
        FIELDS TERMINATED BY '\t'
        LINES TERMINATED BY '\n'
STORED AS SEQUENCEFILE;
```

Resources

- Papers
  - Google File System, 2003
  - Google MapReduce, 2004
  - Google Bigtable, 2006
- URLS
  - Apache Hadoo
- Available Hadoop Distribution
  - Apache, IBM, Cloudera, Hortonworks

Hive Architecture

- Main components
  - SQL interface
  - Parser/Planner
  - Metastore
  - Driver