# Introduction to Big Data



wi ark







BerkeleyX

# This Lecture

The Big Data Problem

Hardware for Big Data

Distributing Wo

Handling Failures and Slow Ma

Map Reduce and Complex Jobs

Apache Spark

# Some Traditional Analysis Tools

- Unix shell commands, Pandas, R



All r

single machine

# The Big Data Problem

- Data growing faster than computation speeds
- Growing data s
  - » Web, mobile, sci
- Storage getting cheaper
  - » Size doubling every 18 months
- But, stalling CPU speeds and storage bottlenecks

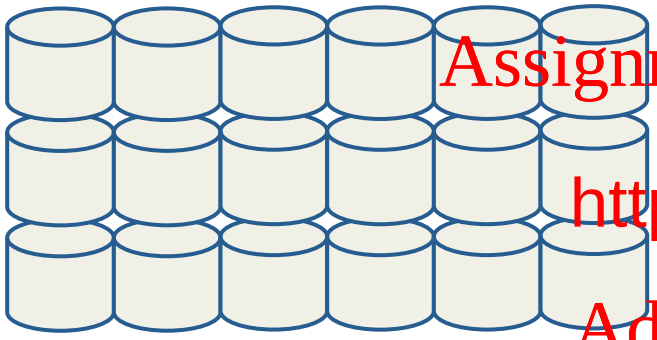
# Big Data Examples

- Facebook's daily logs: 60 TB
- 1,000 genom
- Google web i
- Cost of 1 TB of disk: ~$35
- Time to read 1 TB from disk: 3 hours (100 MB/s)

# The Big Data Problem

- A single machine can no longer process or even store all the data!
- Only solution er large clusters

# Hardware for Big Data





Lots of hard drives … a

# Hardware for Big Data

One big box?
(1990's solution)



But, expensive
- » Low volume
- » All "premium" hardware

*And, still not big enough!*

Image: Wikimedia Commons / User:Tonusamuel

# Hardware for Big Data

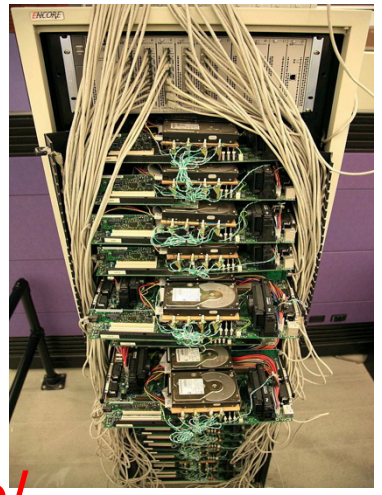


Image: Steve Jurvetson/Flickr

Consumer-grade hardware

- Not "gold plated"

Many desktop-lik

- Easy to add capacity
- Cheaper per CPU/disk

Complexity in software

# Problems with Cheap Hardware

**Failures**, Google's numbers:

- 1-5% hard drives/year
- 0.2% DIMMs/

**Network** speeds versus shared

- *Much* more latency
- Network slower than storage

**Uneven** performance

# What's Hard About Cluster Computing?

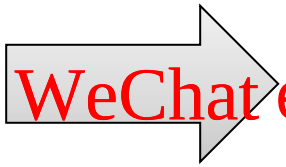
- How do we split work across machines?

# How do you count the number of occurrences of each word in a document?



"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and ham?"





you: 1
like: 1
…

# One Approach: Use a Hash Table

"I am Sam
I am Sa
Sam I a
Do you like
Green eggs and ham?"

{}

# One Approach: Use a Hash Table

"I am Sam {I : 1}

I am Sa

Sam I a

Do you like

Green eggs and ham?"

# One Approach: Use a Hash Table

"I am Sam
I am Sa
Sam I a
Do you like
Green eggs and ham?"

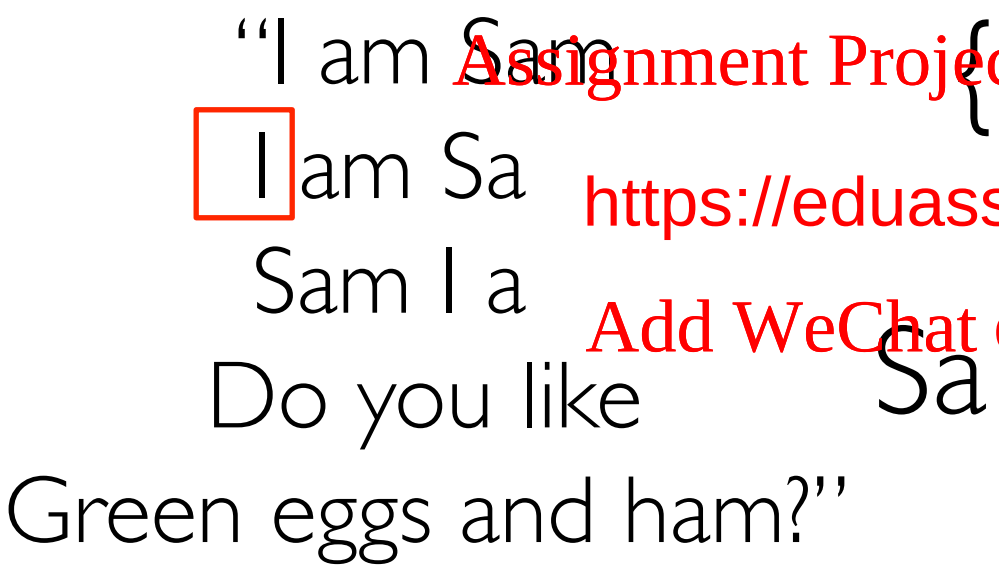
{I: 1,

# One Approach: Use a Hash Table

"I am Sam
I am Sa
Sam I a
Do you like
Green eggs and ham?"

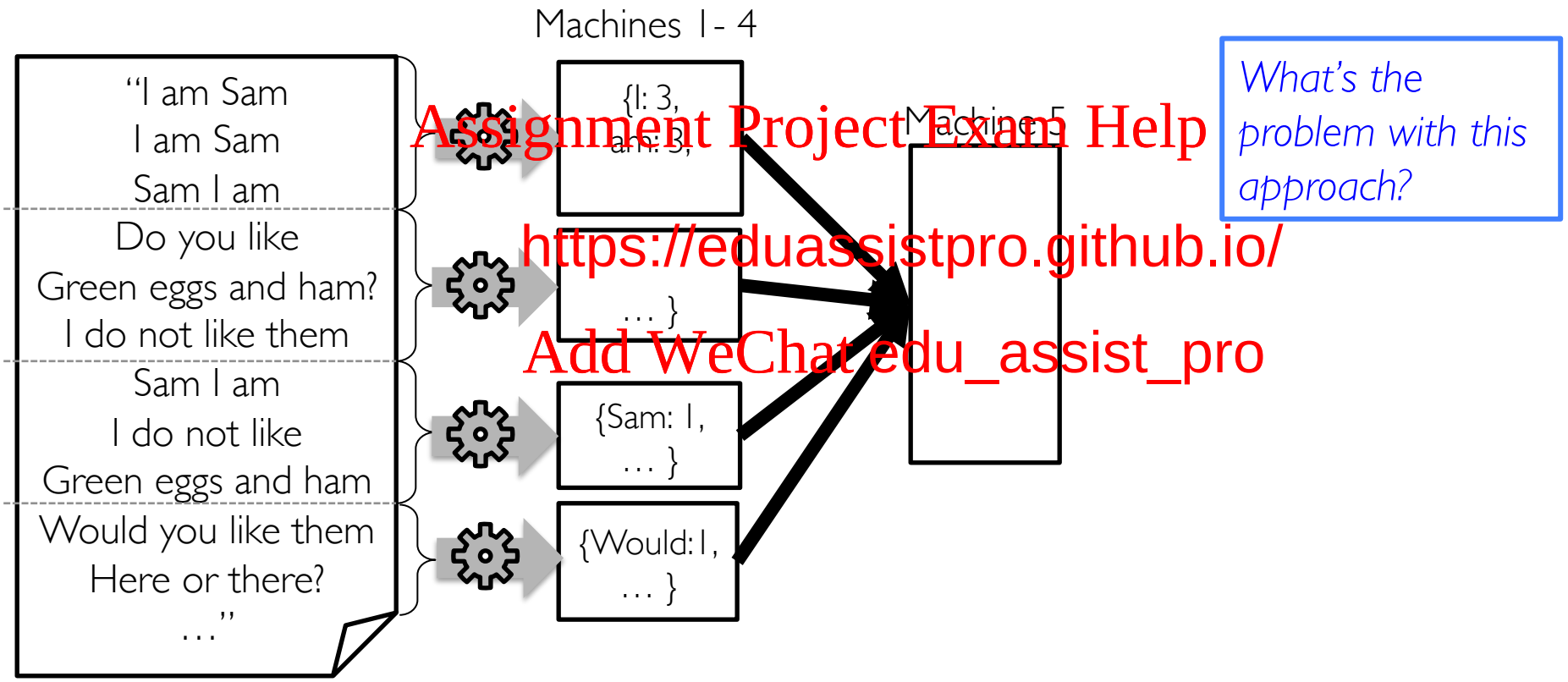
{I: 1,
Sa

# One Approach: Use a Hash Table

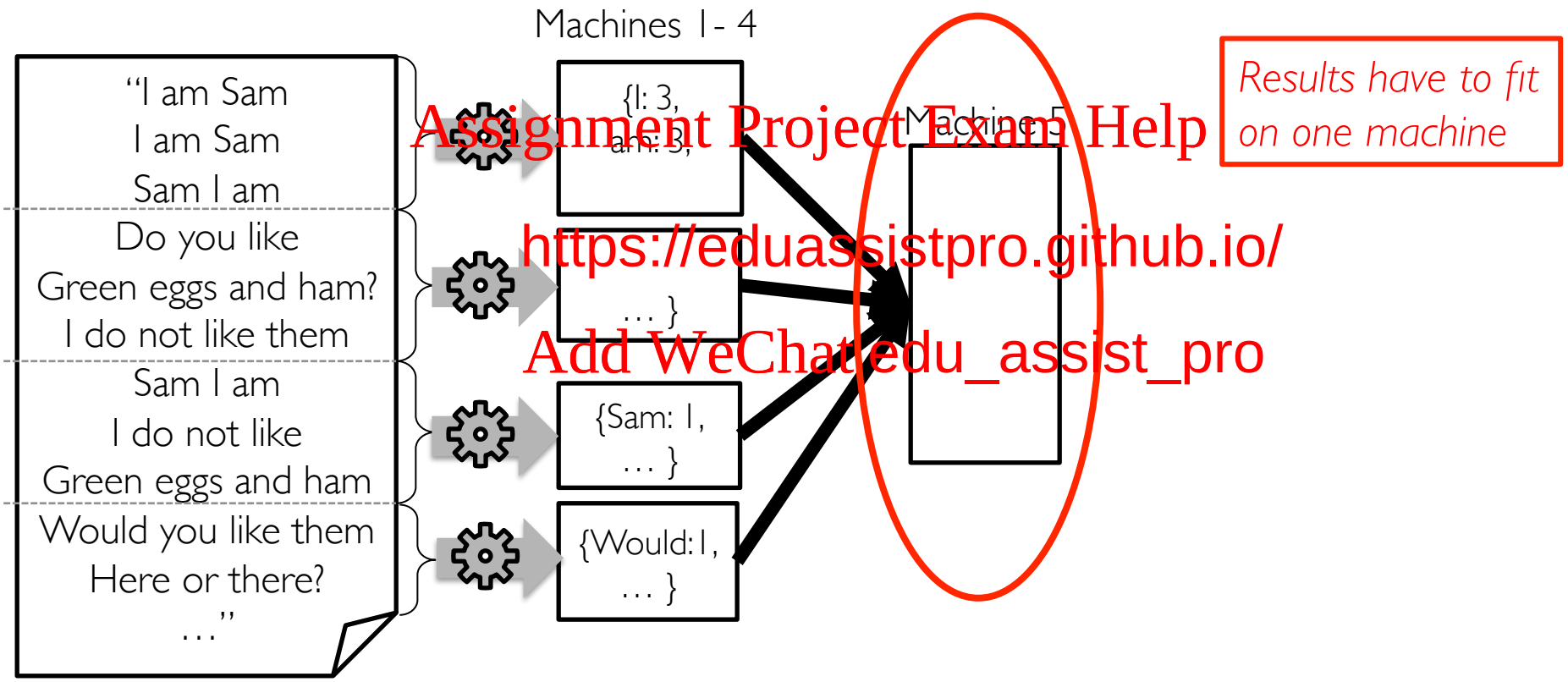


# What if the Document is Really Big?

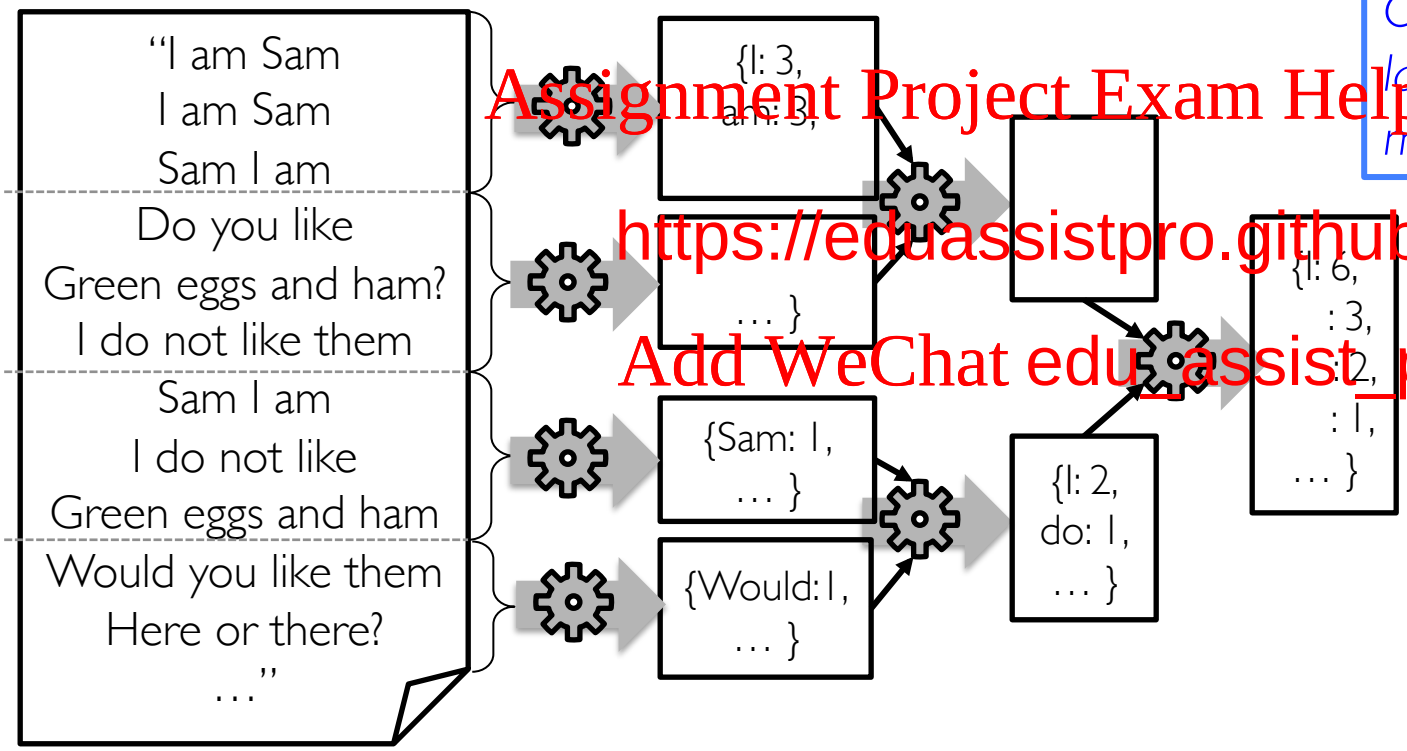
# What if the Document is Really Big?



# What if the Document is Really Big?

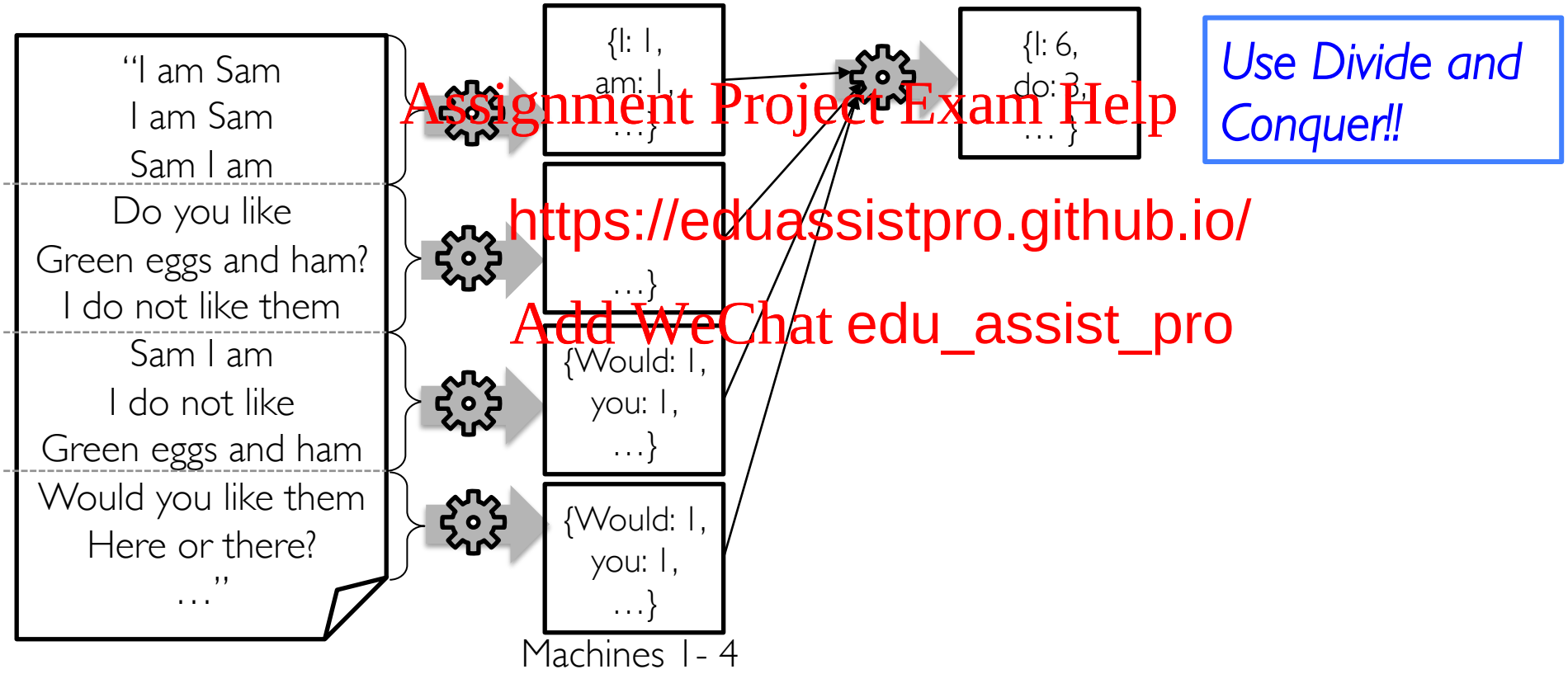


# What if the Document is Really Big?

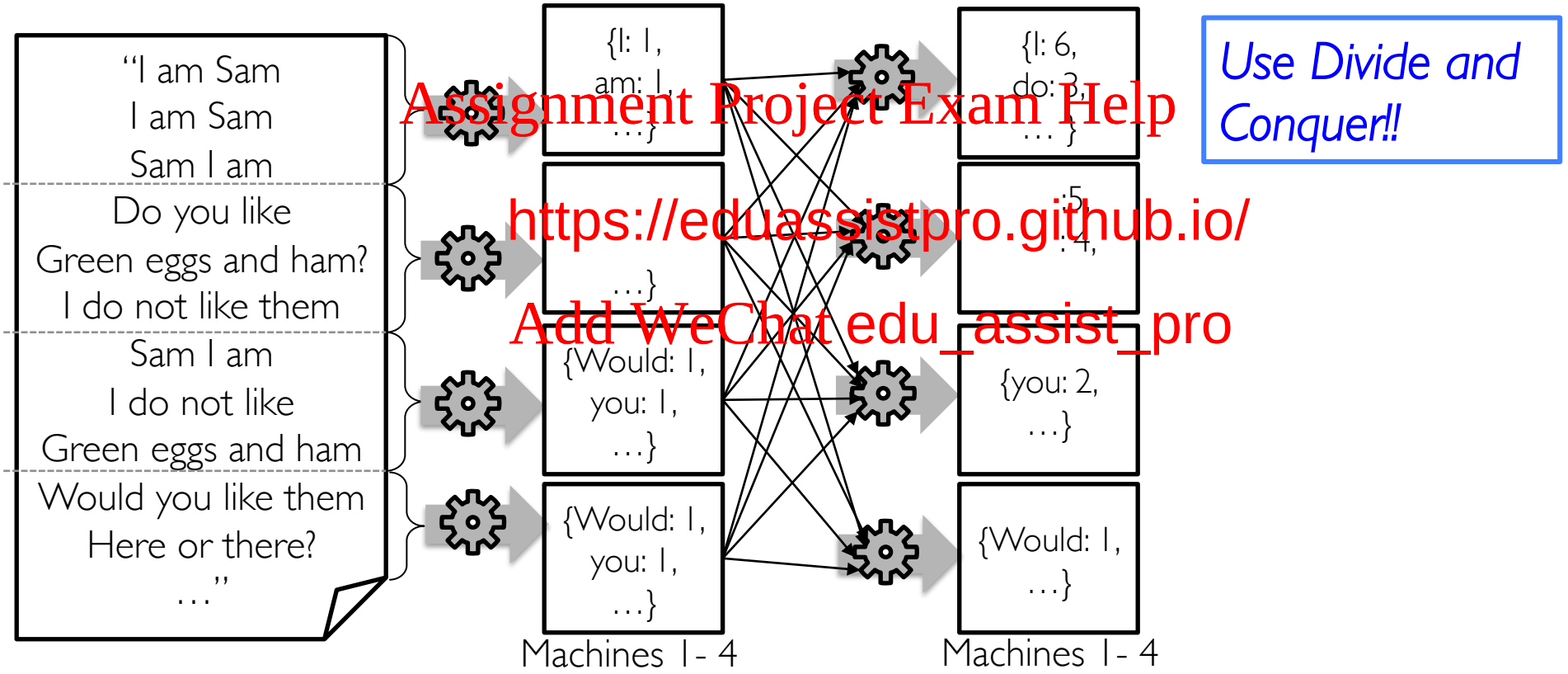



*Can add aggregation layers but results still must fit on one machine*

# What if the Document is Really Big?

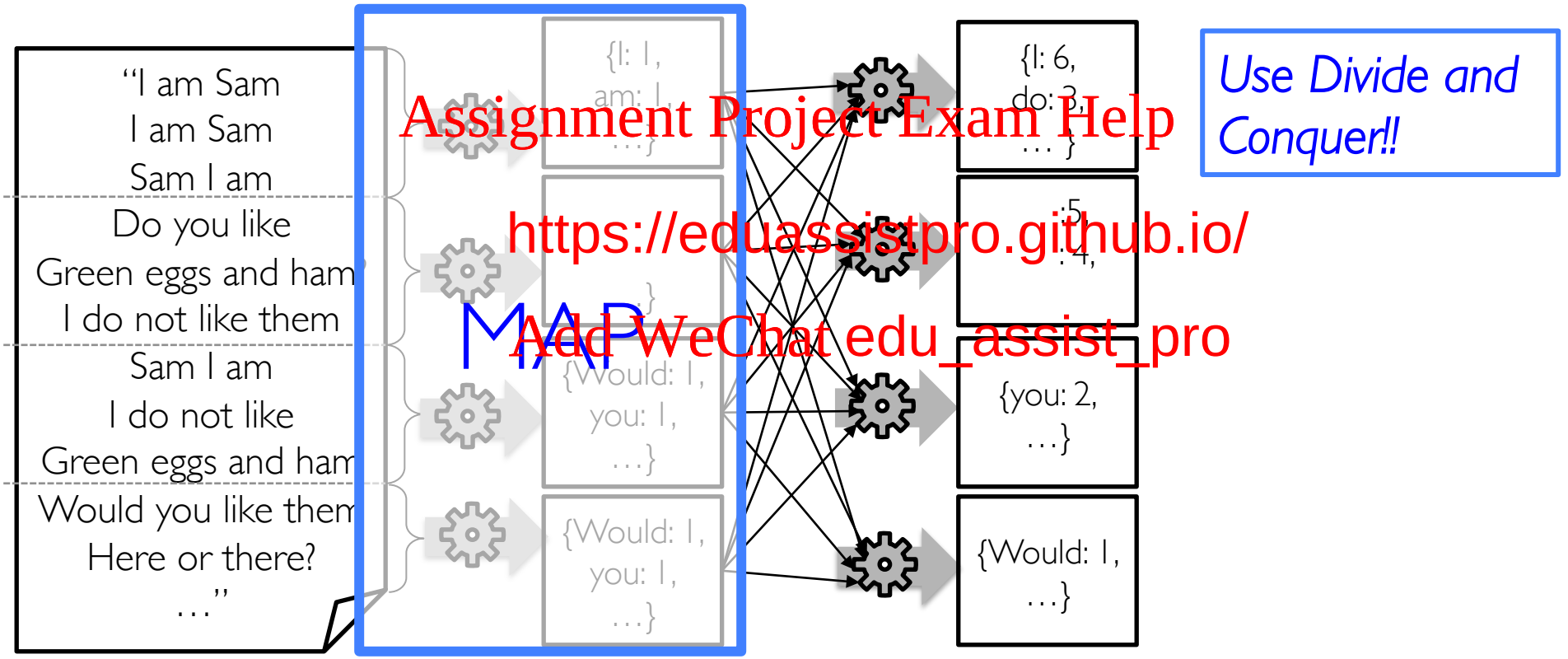


# What if the Document is Really Big?

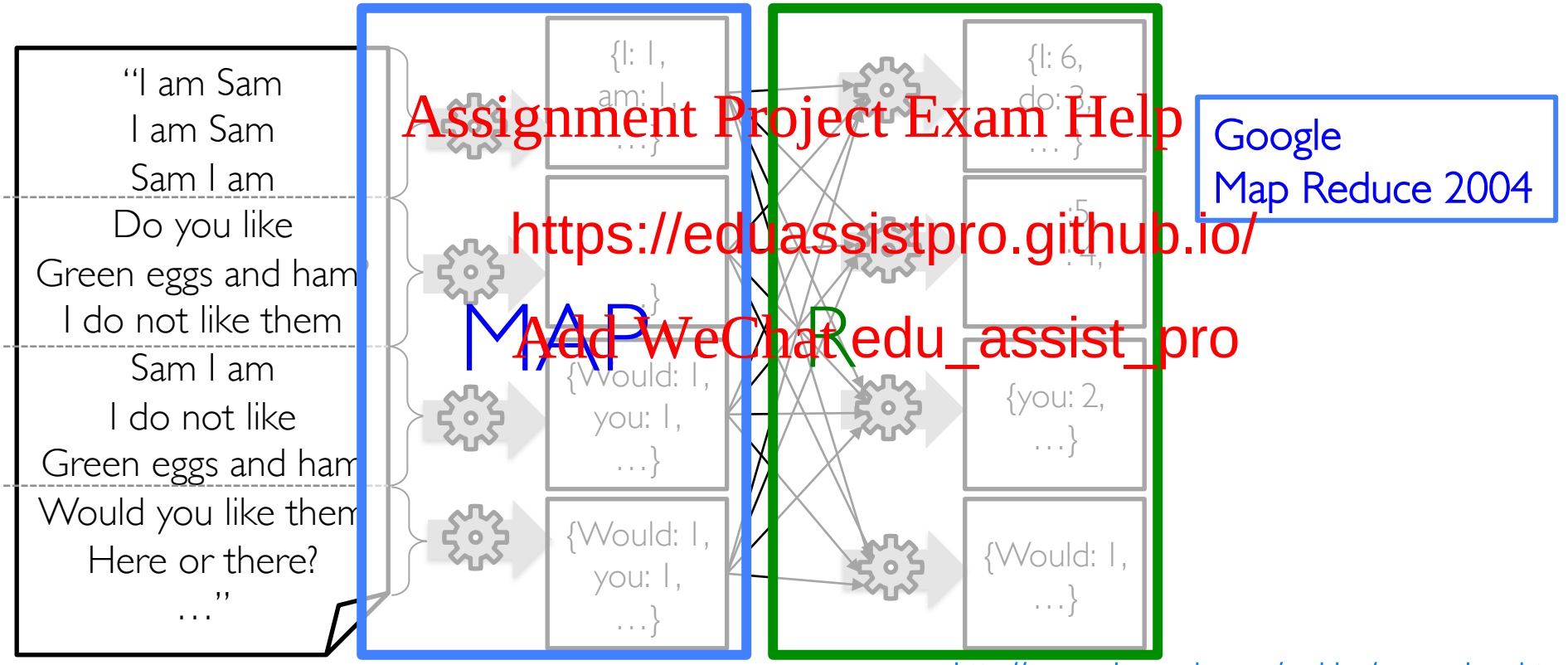



*Use Divide and Conquer!!*

# What if the Document is Really Big?



# What if the Document is Really Big?

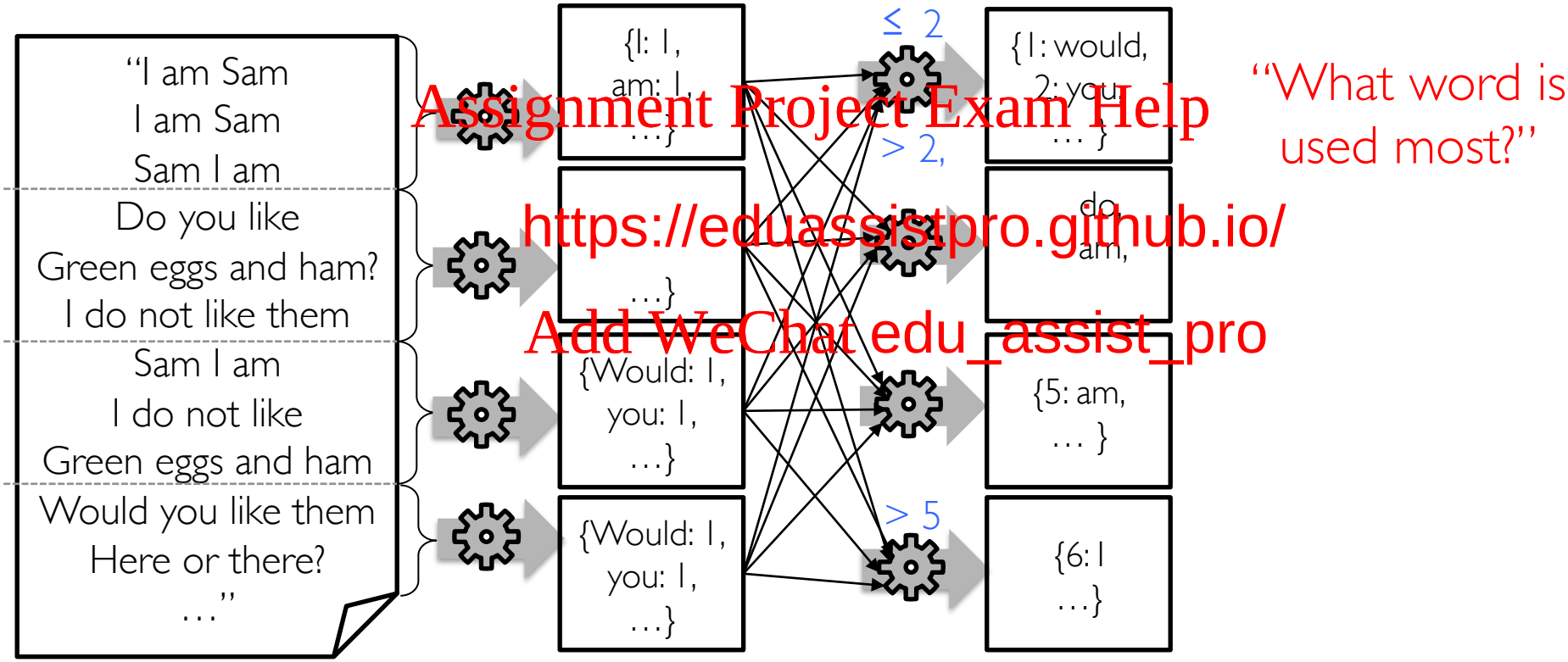





Google
Map Reduce 2004

http://research.google.com/archive/mapreduce.html

# Map Reduce for Sorting

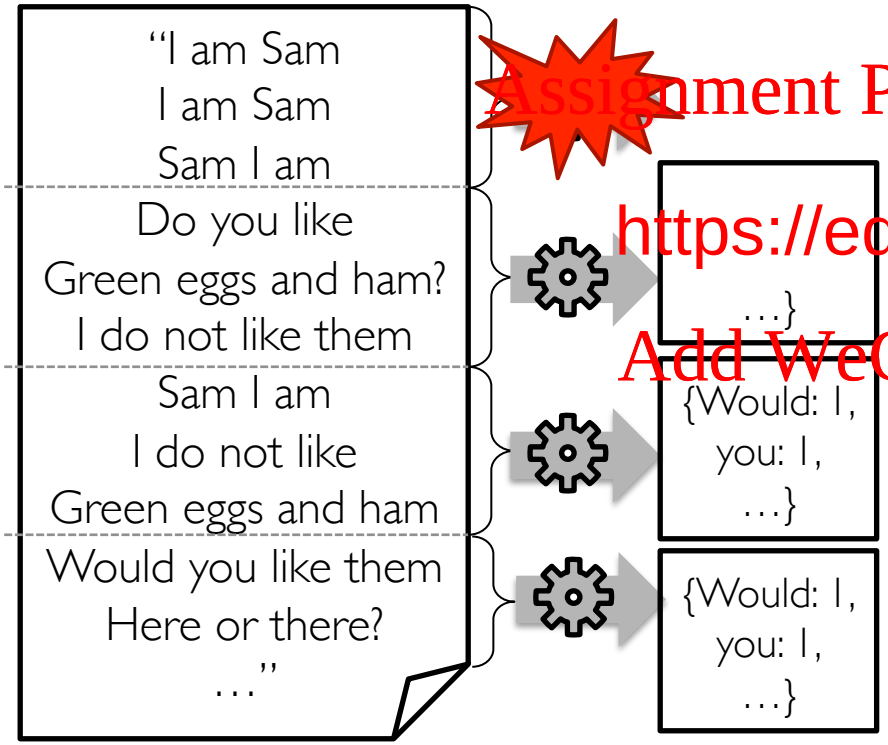



"What word is used most?"

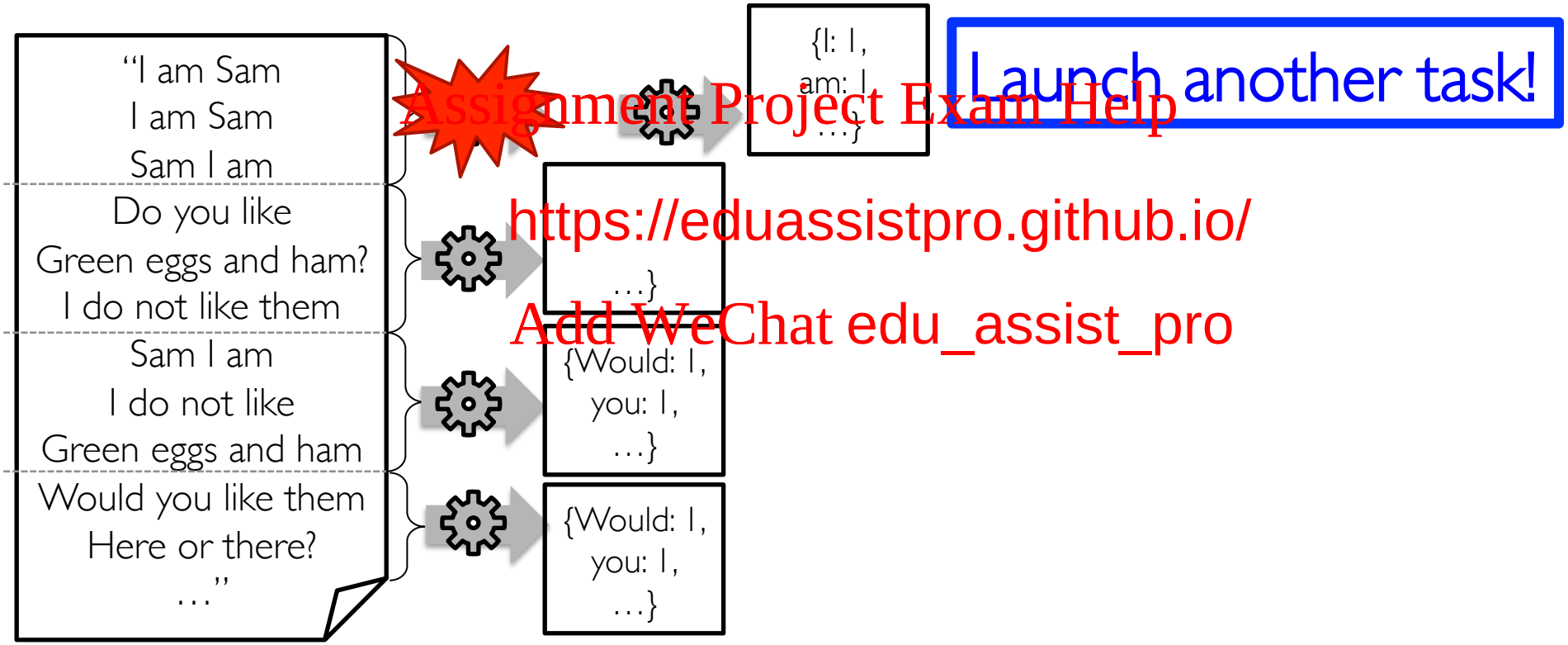
# What's Hard About Cluster Computing?

- How to divide work across machines?
  - » Must consider network, data locality
  - » Moving data may be very expensive
- How to deal wit
  - » 1 server fails every 3 years → with [illegible] es see 10 faults/day
  - » Even worse: stragglers (not failed, b [illegible] es)

# How Do We Deal with Failures?

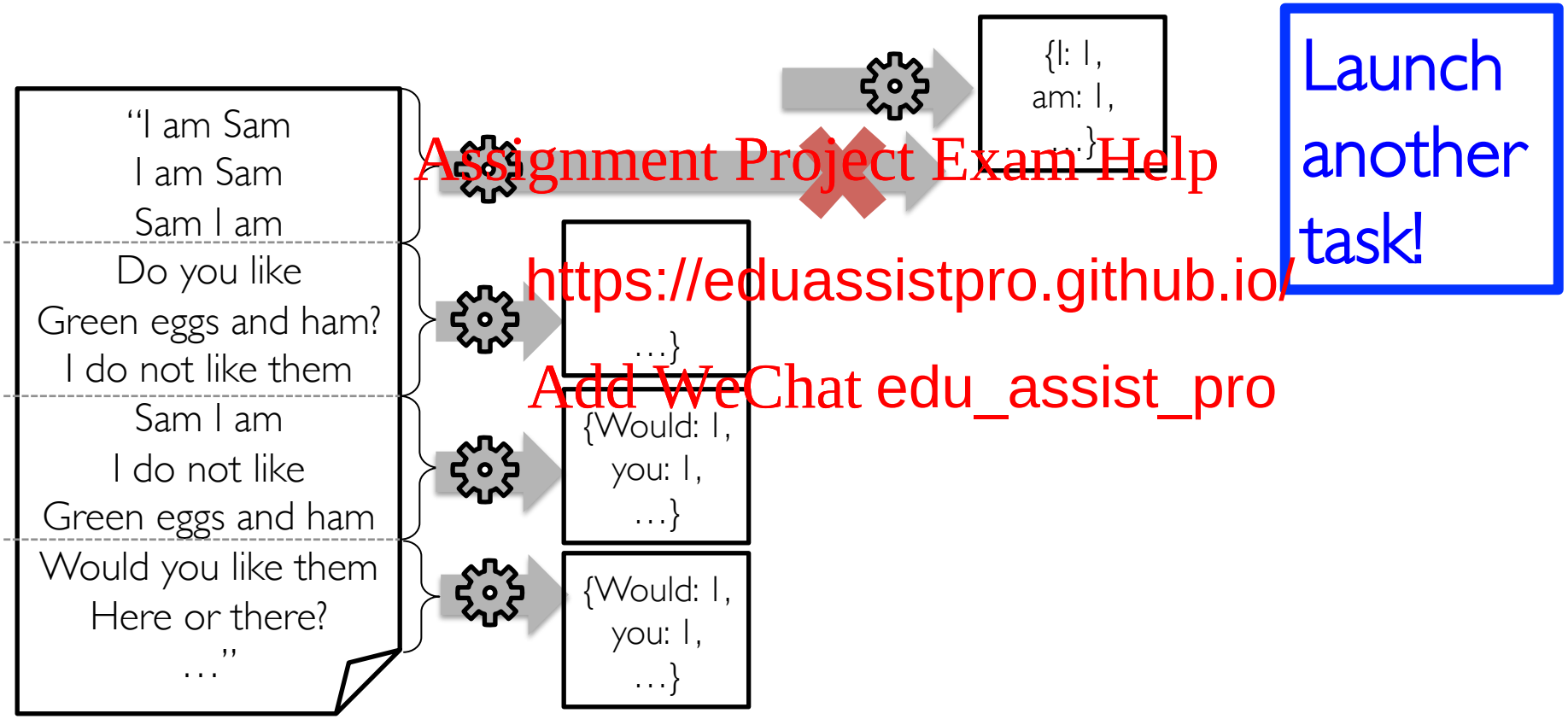


# How Do We Deal with Machine Failures?

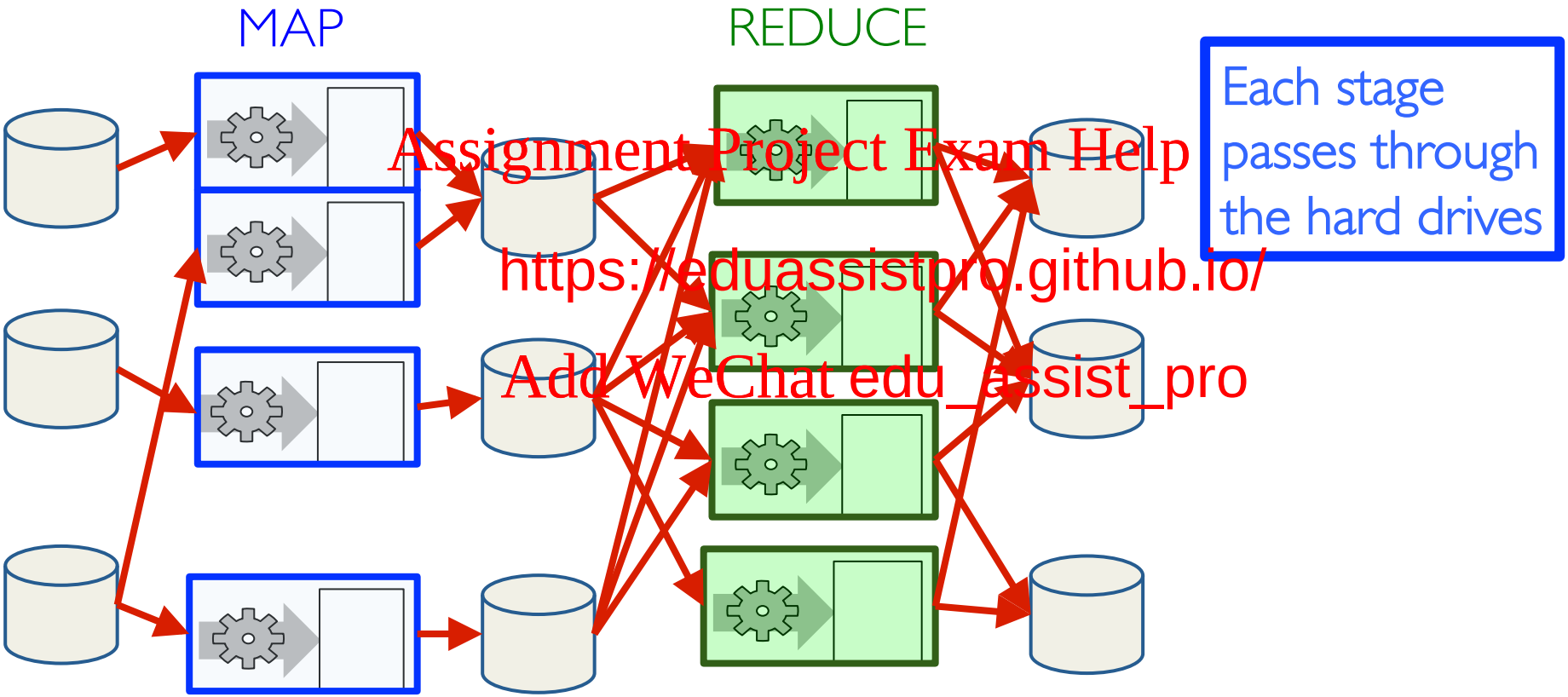


# How Do We Deal with Slow Tasks?

# How Do We Deal with Slow Tasks?

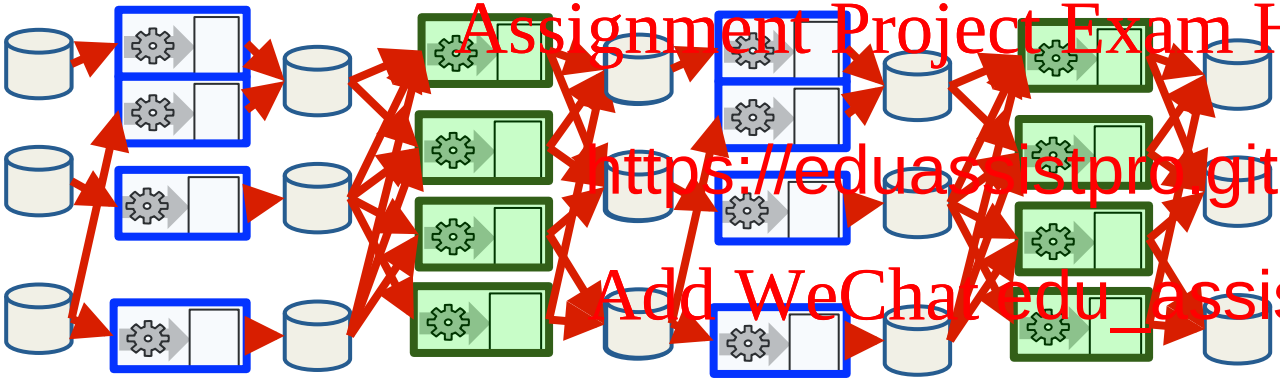


# Map Reduce: Distributed Execution



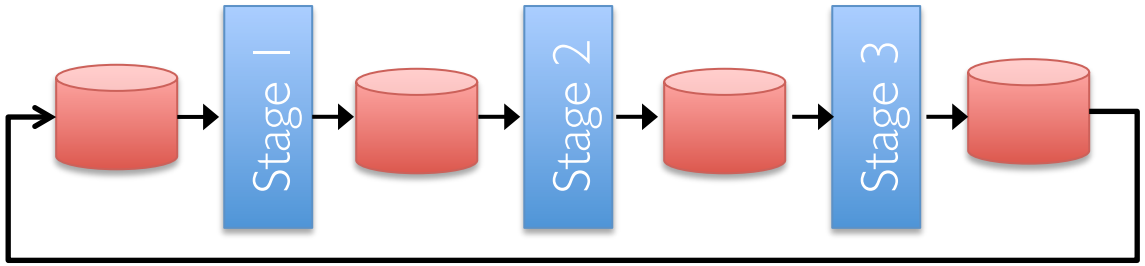
# Map Reduce: Iterative Jobs

- Iterative jobs involve a lot of disk I/O for each repetition





Disk I/O is very slow!



# Apache Spark Motivation

- Using Map Reduce for complex jobs, interactive queries and online processing involves lots of disk I/O



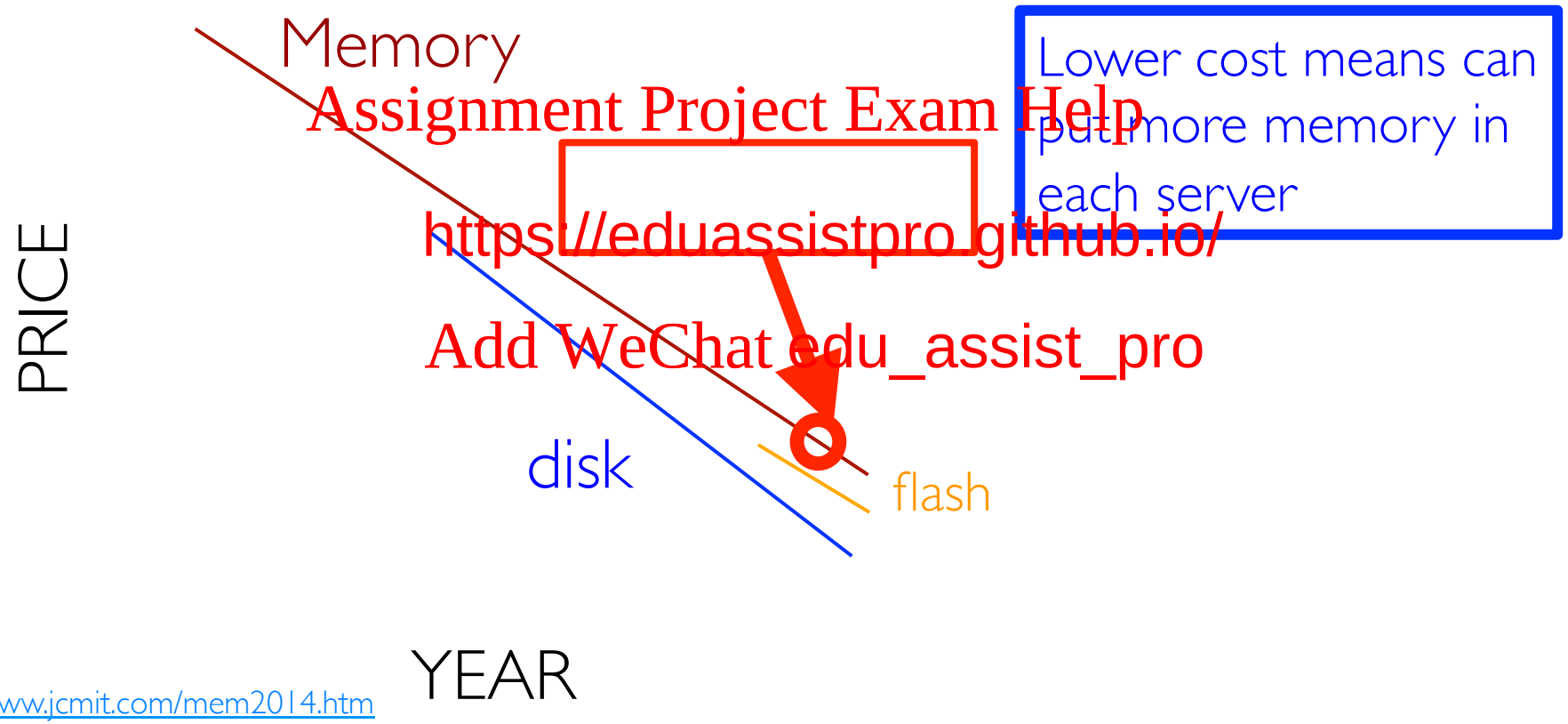

Interactive mining

Stream processing

Also, iterative jobs

Disk I/O is very slow

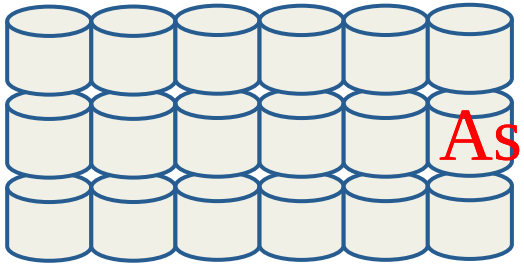
# Tech Trend: Cost of Memory





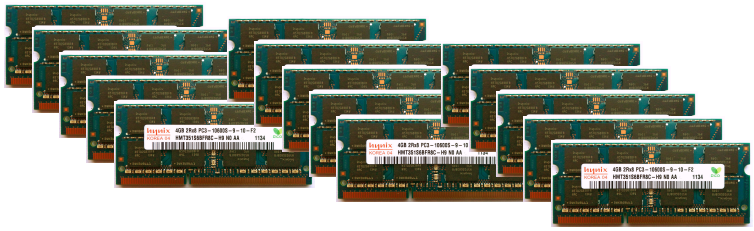

http://www.jcmit.com/mem2014.htm

# Hardware for Big Data





Lots of hard driv



… and memory!

# Opportunity

- Keep more data *in-memory*

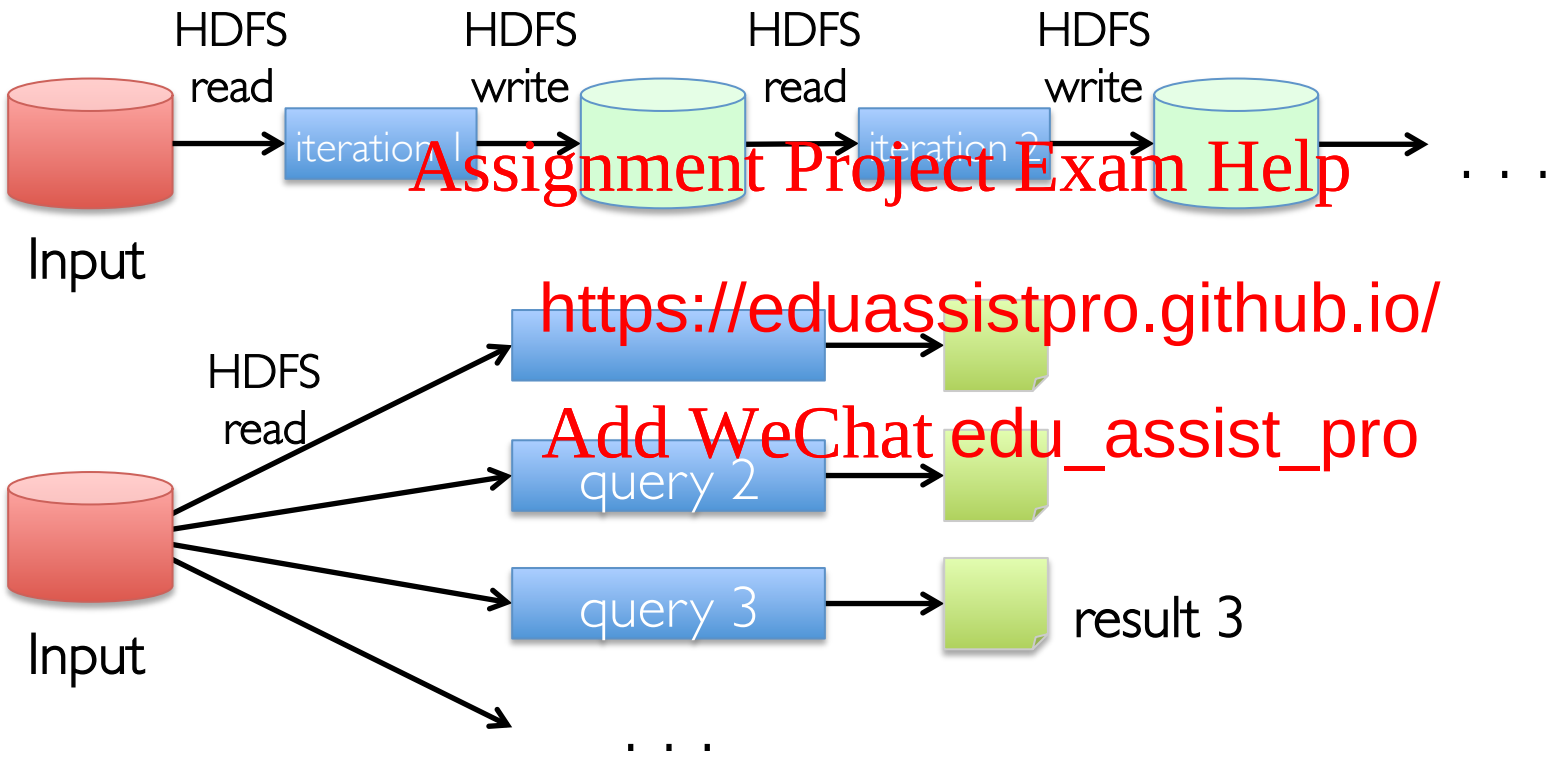


- Create new dis ine:

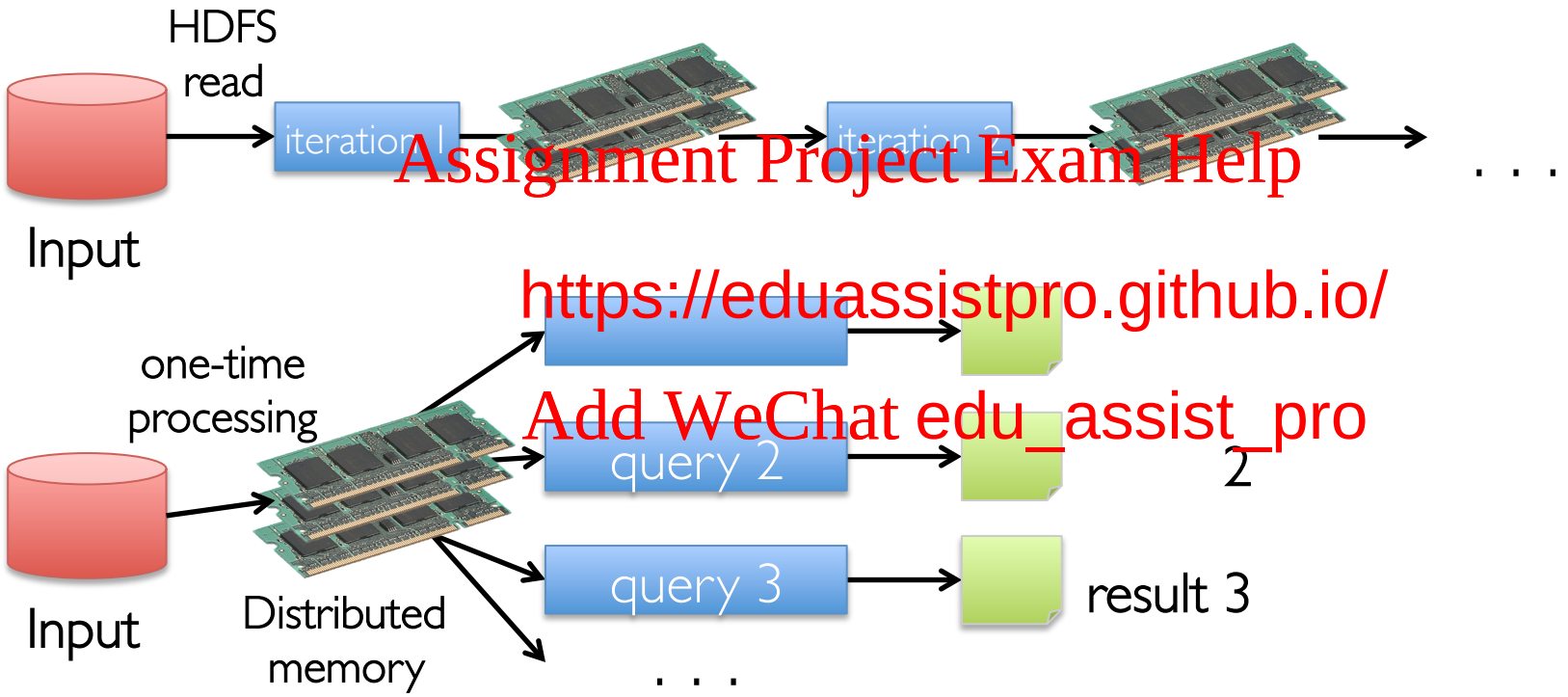



http://people.csail.mit.edu/matei/papers/2010/hotcloud_spark.pdf

# Use Memory Instead of Disk



# In-Memory Data Sharing






10-100x faster than network and disk

# Resilient Distributed Datasets (RDDs)

- Write programs in terms of operations on distributed datasets
- Partitioned collections of objects spread across a cluster, stored in memory or on di
- RDDs built and manipulated thro
  set of parallel transformations (m ) and actions (count, collect, save)
- RDDs automatically rebuilt on machine failure

# The Spark Computing Framework

- Provides programming abstraction and parallel runtime to hide complexities of fault-tolerance and slow machines
- "Here's an operation, run it [illegible] he data"
  - » I don't care where it runs (you s [illegible])
  - » In fact, feel free to run it twice on different nodes

# Spark Tools

# Spark and Map Reduce Differences

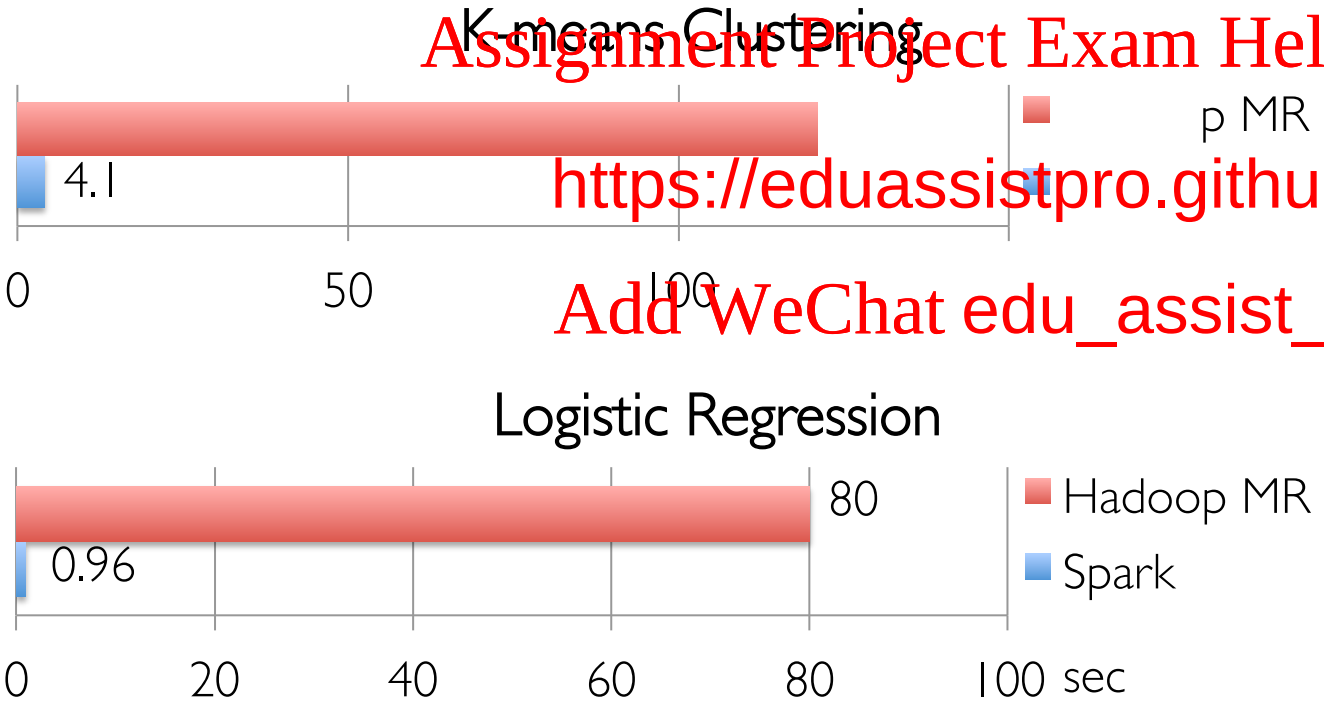
| | Hadoop Map Reduce | Spark |
|---|---|---|
| Storage | Disk o | |
| Operations | Map a Reduce | Sample, at |
| Execution model | Batch | Batch, interactive, streaming |
| Programming environments | Java | Scala, Java, R, and Python |

# Other Spark and Map Reduce Differences

- Generalized patterns
  ⇒ unified engine for many use cases
- Lazy evaluati
  ⇒ reduces wait states, bet ng
- Lower overhead for starting jobs
- Less expensive shuffles

# In-Memory Can Make a Big Difference

- Two iterative Machine Learning algorithms:



# First Public Cloud Petabyte Sort



Daytona Gray 100 TB
sort benchmark record
(tied for 1st place)

http://databricks.com/blog/2014/11/05/spark-officially-sets-a-new-record-in-large-scale-sorting.html

# Spark Expertise Tops Big Data Median Salaries



Over 800 respondents across 53 countries and 41 U.S. states

http://www.oreilly.com/data/free/2014-data-science-salary-survey.csp

# History Review

# Historical References

- circa 1979 – **Stanford**, **MIT**, **CMU**, etc.: set/list operations in LISP, Prolog, etc., for parallel processing
  http://www-formal.stanford.edu/jmc/history/lisp/lisp.htm
- circa 2004 – **Google**: *MapReduce: Simplified Data Processing on Large Clusters*
  Jeffrey Dean and Sanjay G
  http://research.google.com/
- circa 2006 – **Apache** *Hadoo* Project
  Doug Cutting
  http://research.yahoo.com/files/cutting.pdf
- circa 2008 – **Yahoo!**: web scale search indexing
  *Hadoop Summit*, HUG, etc.
  http://developer.yahoo.com/hadoop/
- circa 2009 – **Amazon AWS**: Elastic MapReduce
  Hadoop modified for EC2/S3, plus support for Hive, Pig, Cascading, etc.
  http://aws.amazon.com/elasticmapreduce/

# Spark Research Papers

- *Spark: Cluster Computing with Working Sets*
  Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica
  USENIX HotCloud (2010)
  people.csail.mit.edu/matei



- *Resilient Distributed Datasets: A Fault-Tolerant Ab*
  *In-Memory Cluster Computing*
  Matei Zaharia, Mosharaf Chowdhury, Tathagata Das,
  Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin,
  Scott Shenker, Ion Stoica
  NSDI (2012)
  usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf