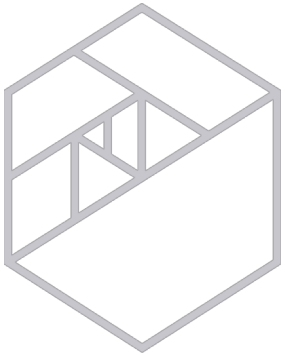


METIS

INTRODUCTION TO HADOOP

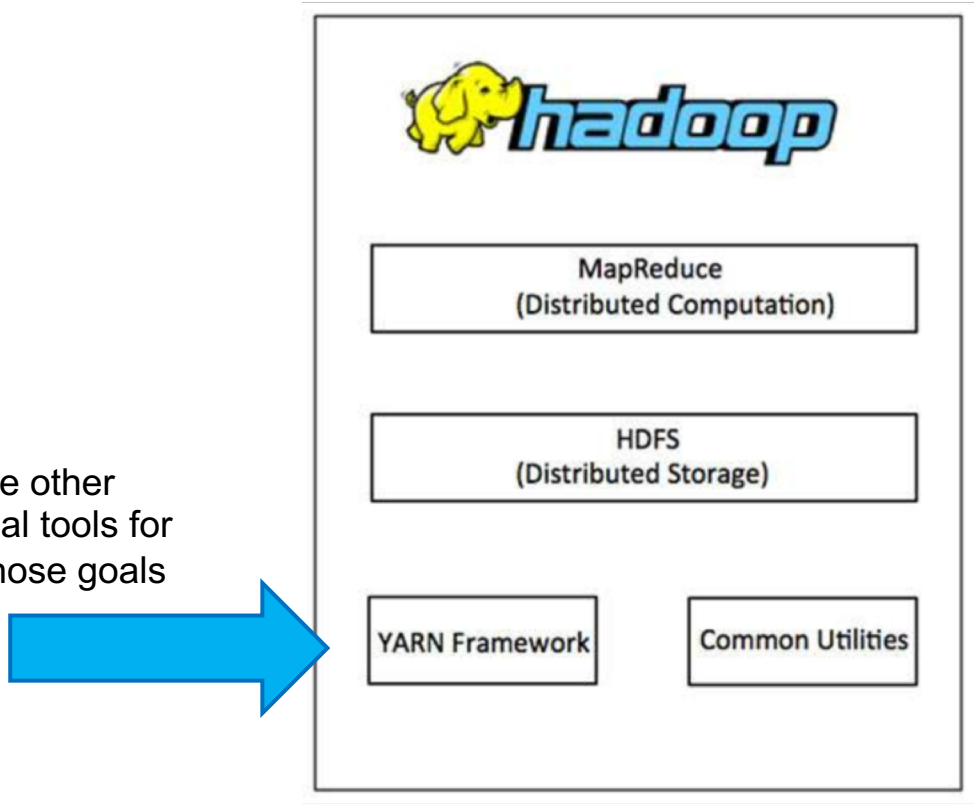
What is Hadoop?

- A system for processing big data
- A framework for storing and organizing large scale data

What is Hadoop?

- MapReduce
- HDFS: The Hadoop Distributed File System

Plus some other managerial tools for making those goals happen

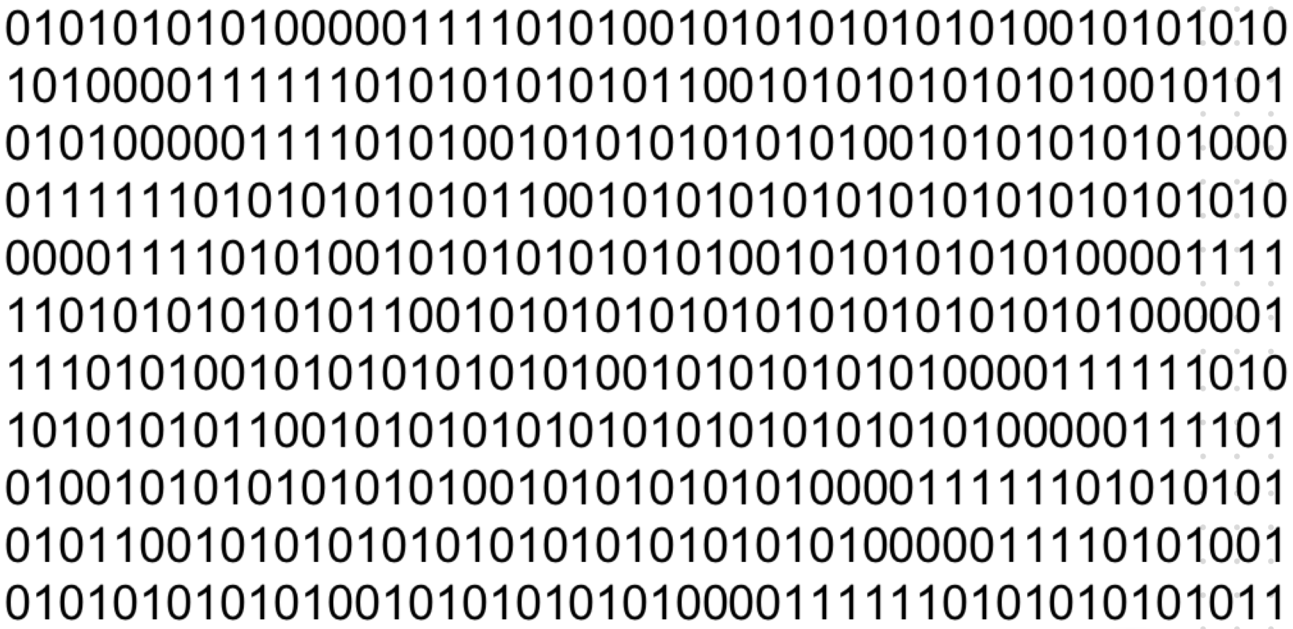


HDFS

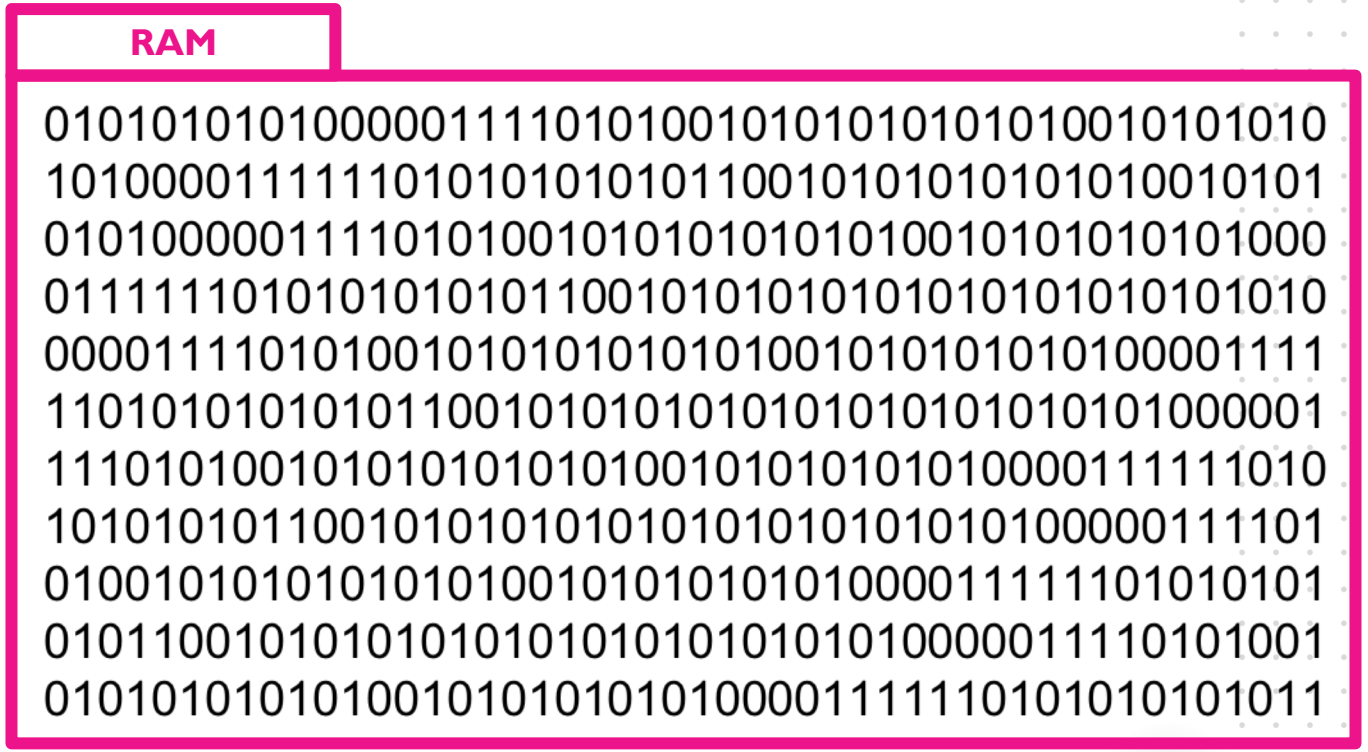
Hadoop Distributed File System

- The largest legacy from Hadoop is HDFS
- It’s a clever system for “chunking” our data up and storing it on lots of different computers for later use.
- The storage points are called “nodes,” and they are distributed across a “cluster” of computers.

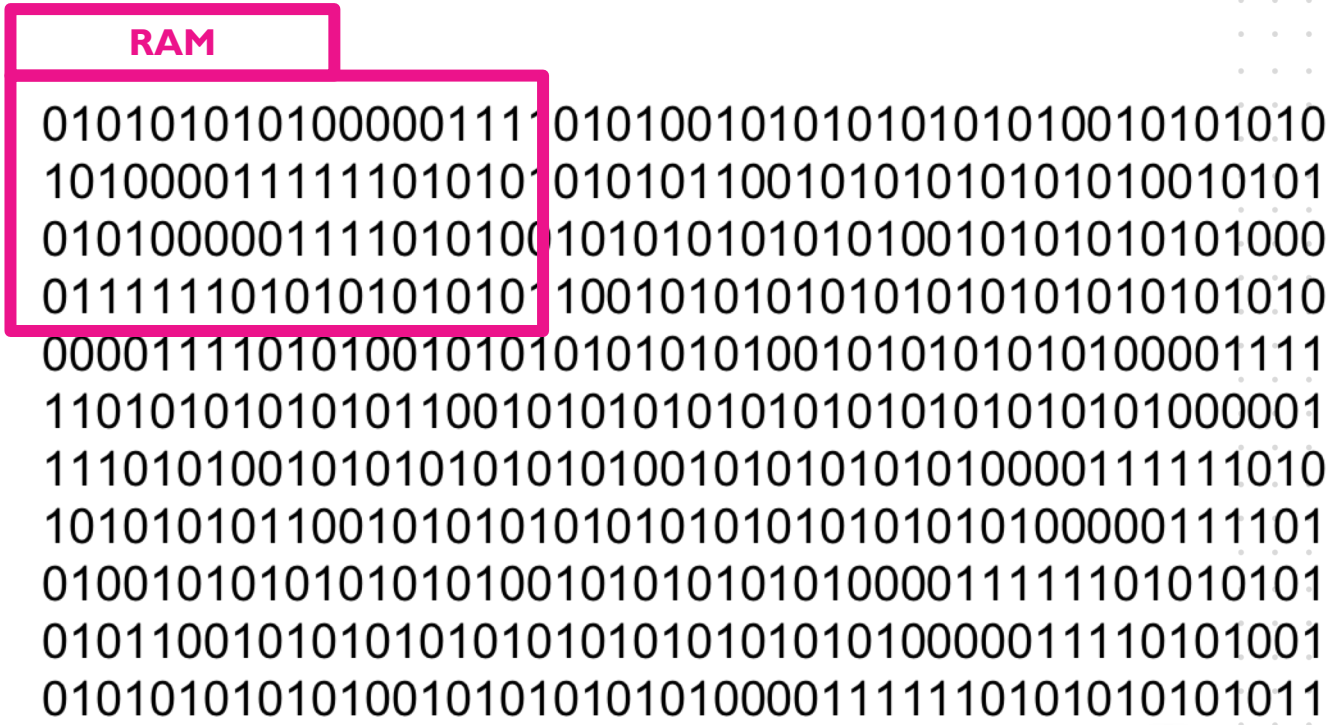
We have data. When we want to use it, we just load it up and go.



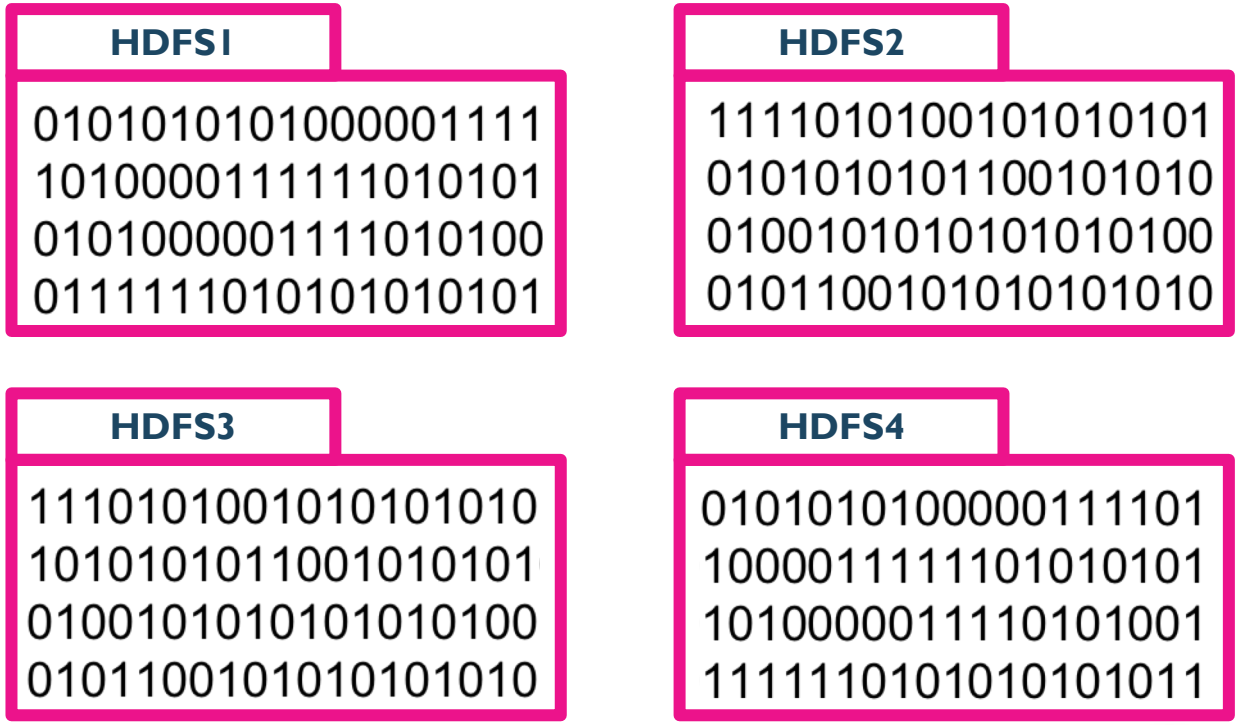
So far, we've always been in this situation.



What happens when we can’t fit everything in RAM?



We need a system that can handle all of this.



HDFS

- HDFS chunks data into smaller blocks (128 MB default), and distributes those blocks across a cluster
- It’s designed to work on “commodity” (read: old, unreliable) hardware. So it automatically creates backups of the data across the cluster.
- It does this by using two types of “nodes” in the cluster

Name Node

Data Node

Name Node

Data Node

- Controls where the data lives.
- Processes requests to add and find data on the cluster.
- Stores metadata like permissions, disk space, data access information.

Name Node

- Controls where the data lives.
- Processes requests to add and find data on the cluster.
- Stores metadata like permissions, disk space, data access information.

Data Node

- Stores data.
- Listens to the NameNode for commands and requests
- Stores block-data information (like verification pieces)

LET'S 'BUILD' A CLUSTER TO SEE IT ALL IN ACTION

I'd like a cluster please!

Node

Node

Node

Node

Node

Node

Node

I'd like a cluster please!

ON IT, BOSS.

Name Node

Node

Node

Node

Node

Node

Node

Node

HAS ANYONE CLAIMED YOU?

Name Node

Node

Node

Node

Node

Node

Node

Node

Name Node

Node

Node

Node

Node

Node

Node

Node

COOL. YOU'RE A DATANODE.

Name Node

Data Node

Node

Node

Node

Node

Node

HAS ANYONE CLAIMED YOU?

Name Node

Data Node

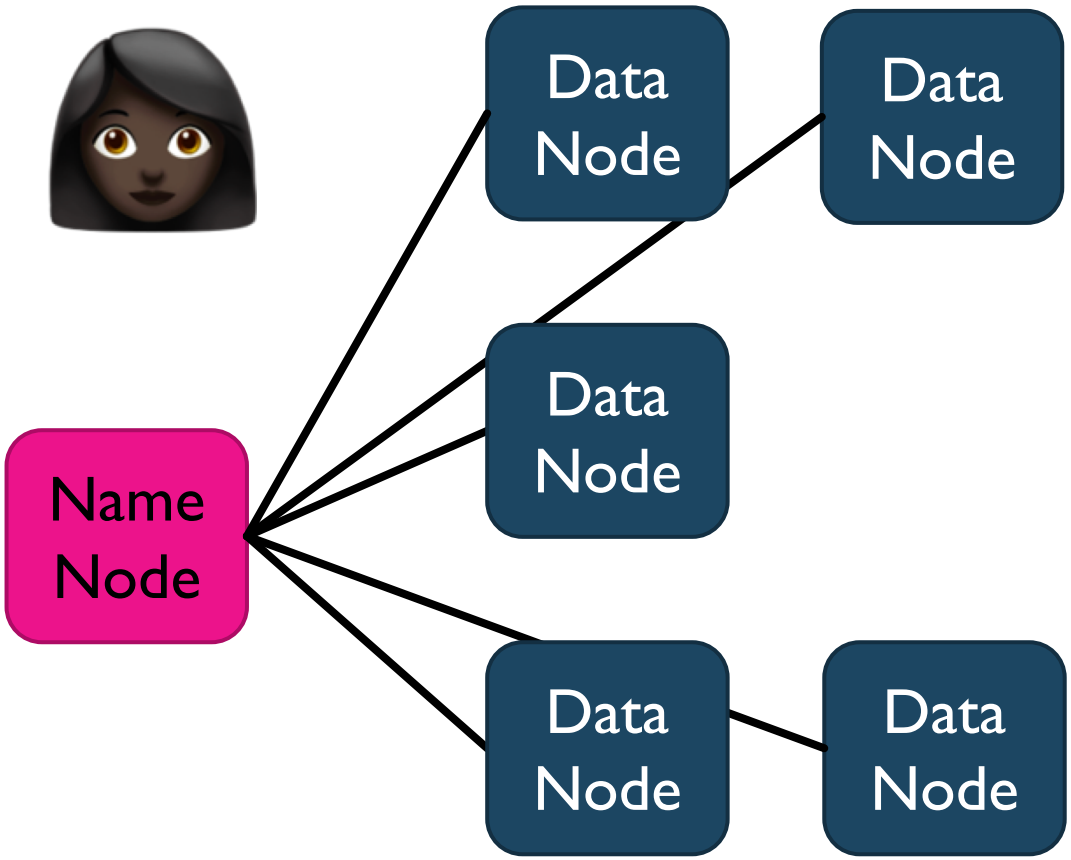
Node

Node

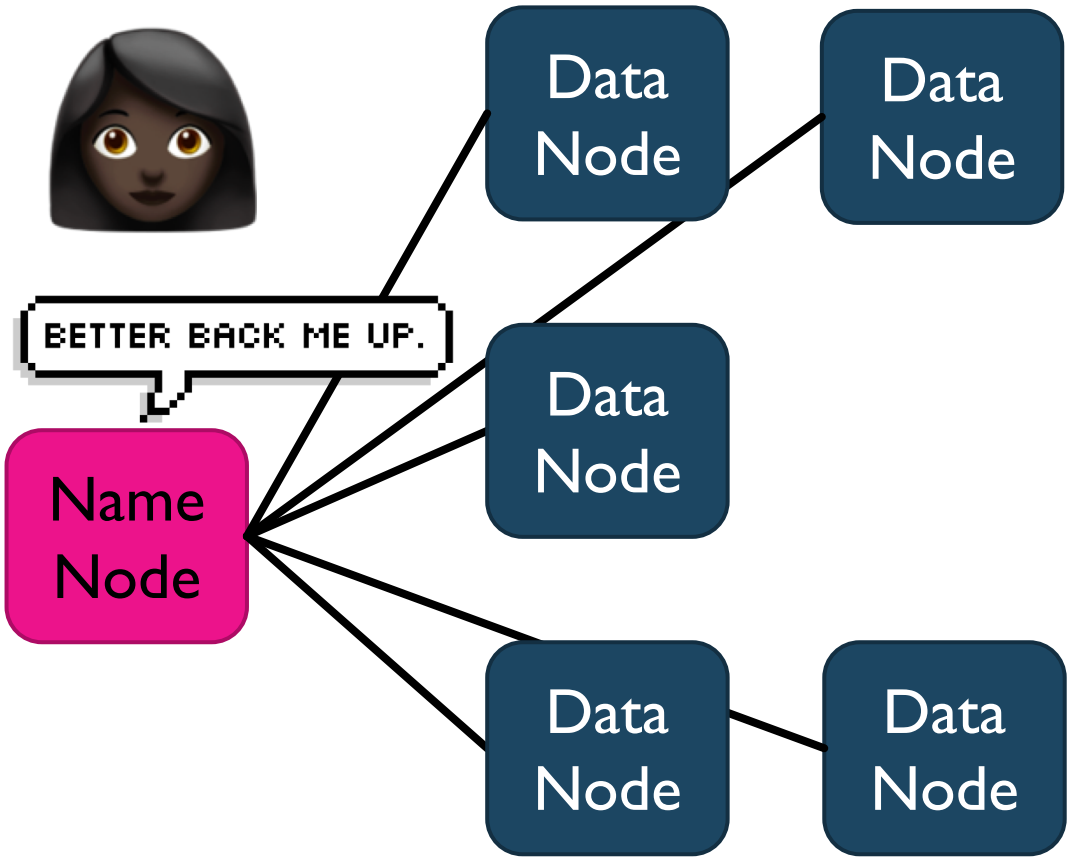
Node

Node

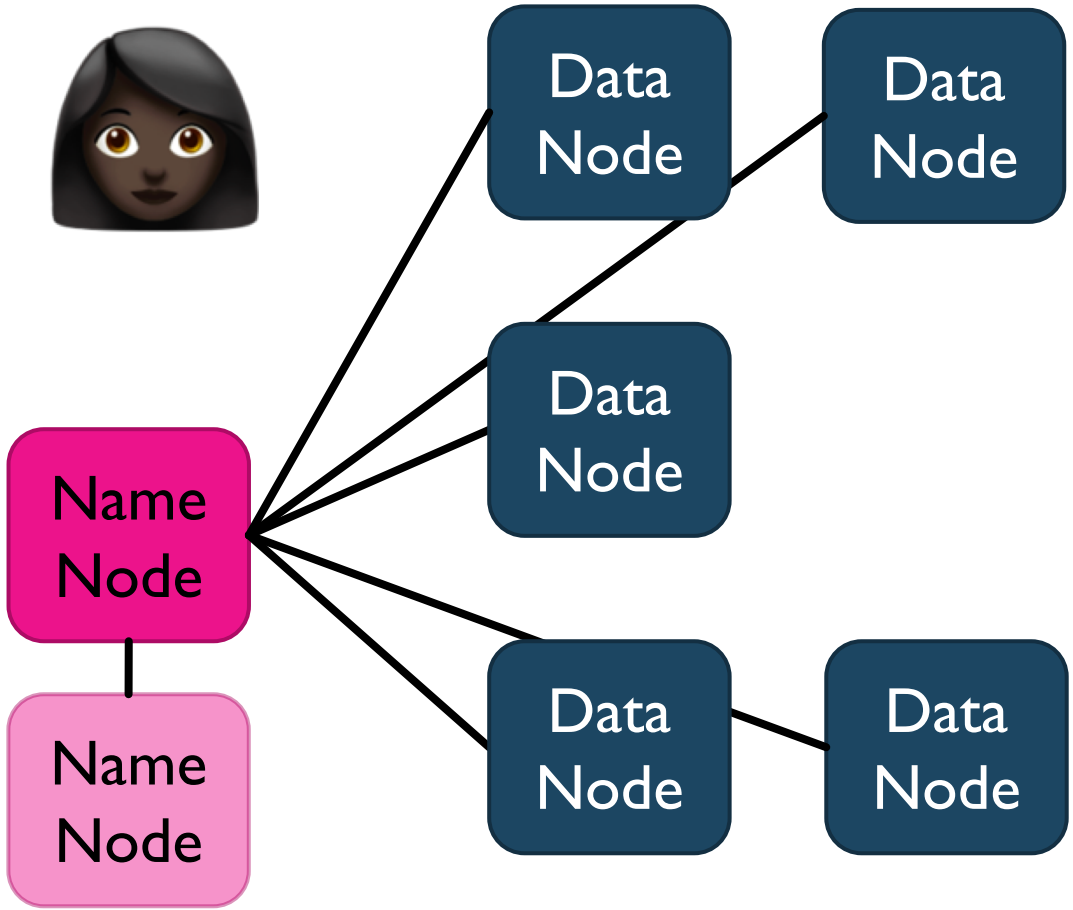
Node



Node


Data Node
Data Node
BETTER BACK ME UP.
Data Node
Name Node
Data Node
Data Node

Node


Data Node
Data Node
Data Node
Name Node
Data Node
Data Node
Name Node

Name Node

Name Node

Data Node

Data Node

Data Node

Data Node

Data Node

Data Node

Add this data to the cluster

Name Node

Name Node

Data Node

Data Node

Data Node

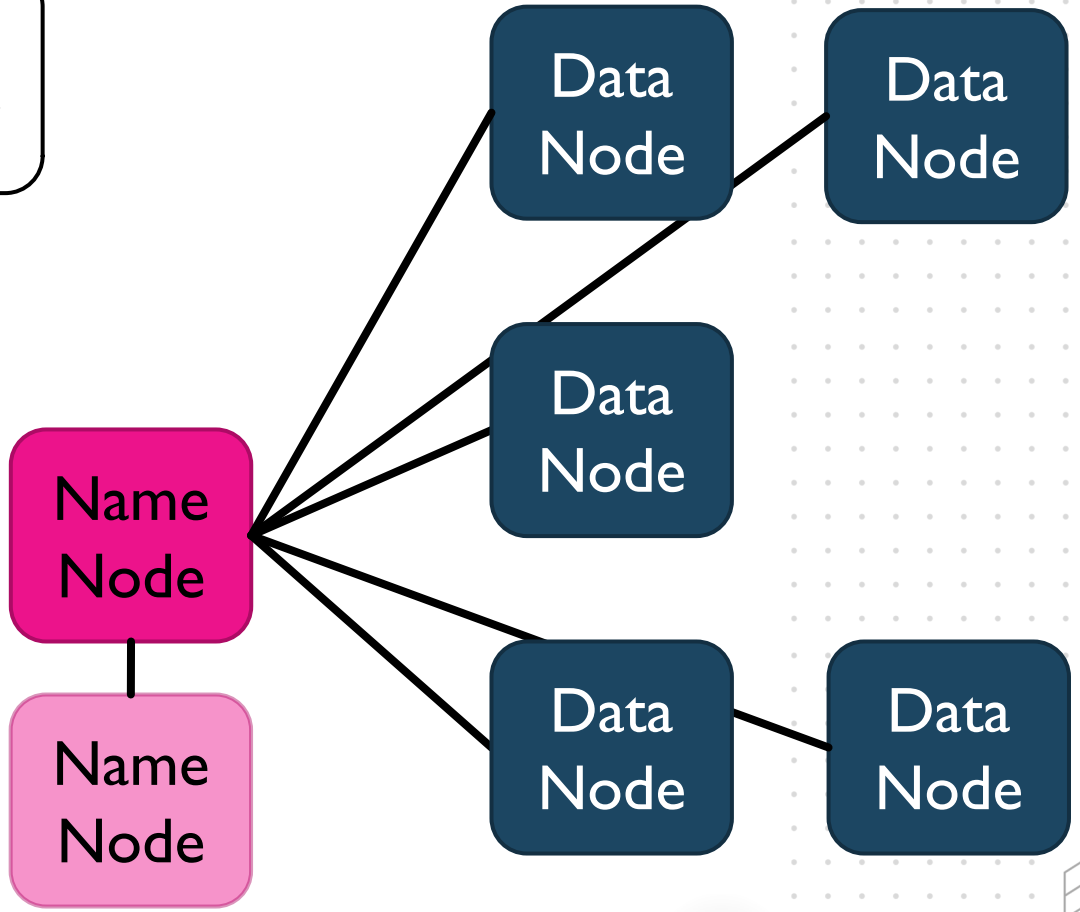
Data Node

Data Node

Data Node

Add this data to the cluster

The system chunks the data, moves those chunks to data nodes, and then makes sure the data is backed up across many nodes.



Name Node

Name Node

Data Node

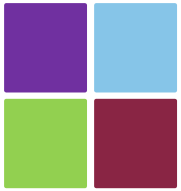
Data Node

Data Node

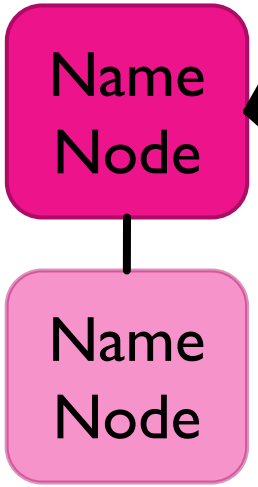
Data Node

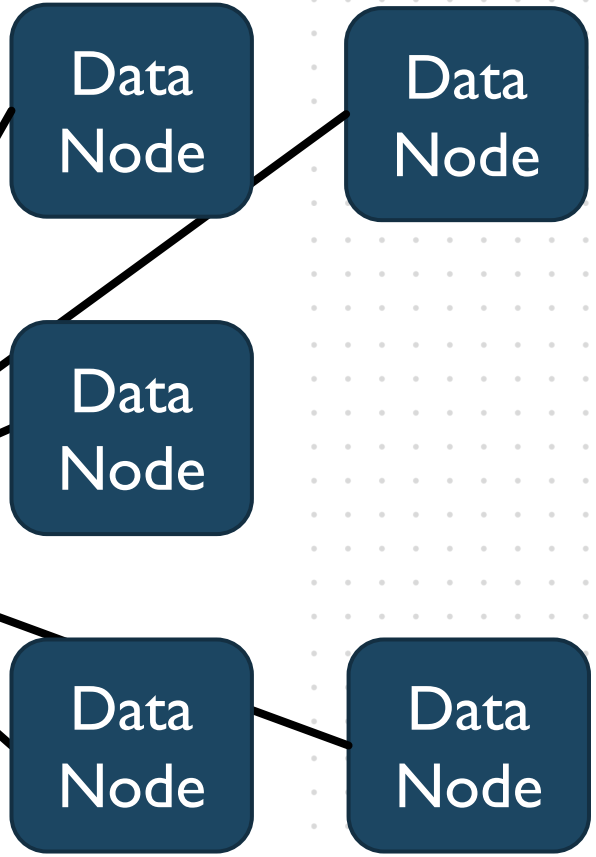
Data Node

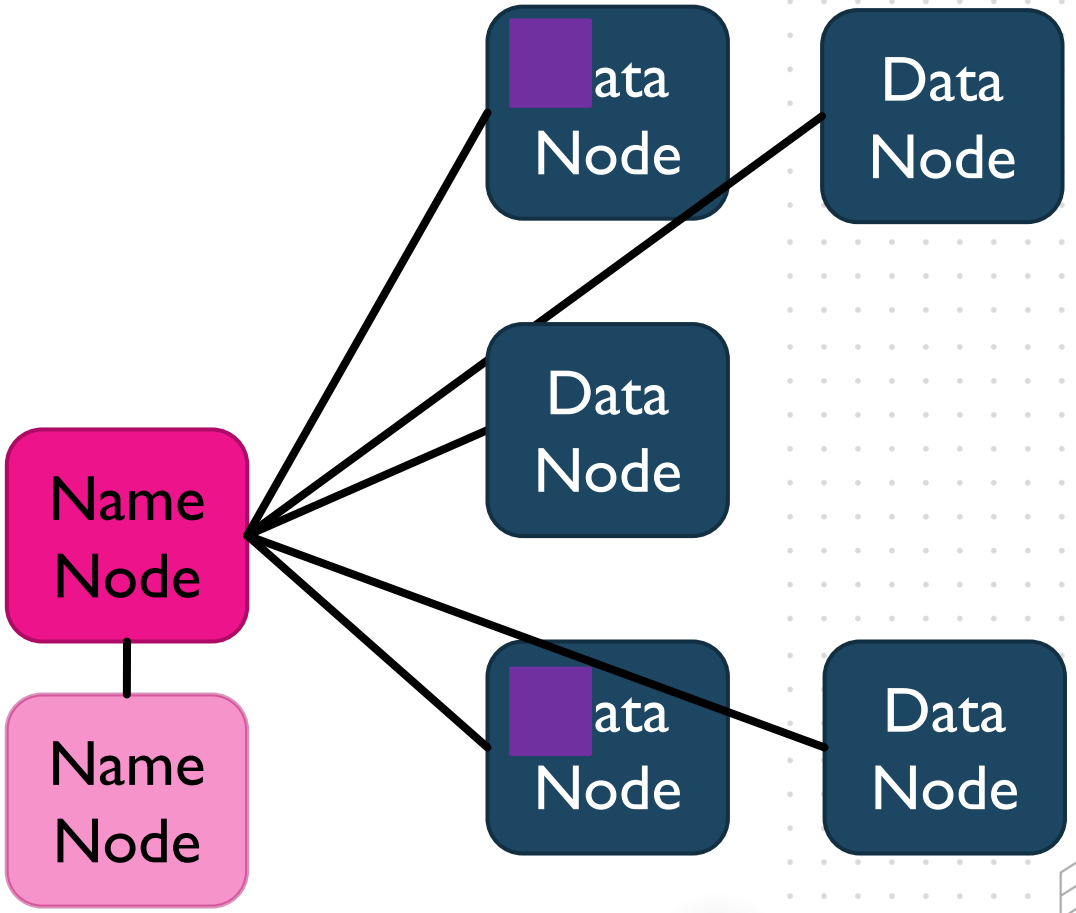
Data Node

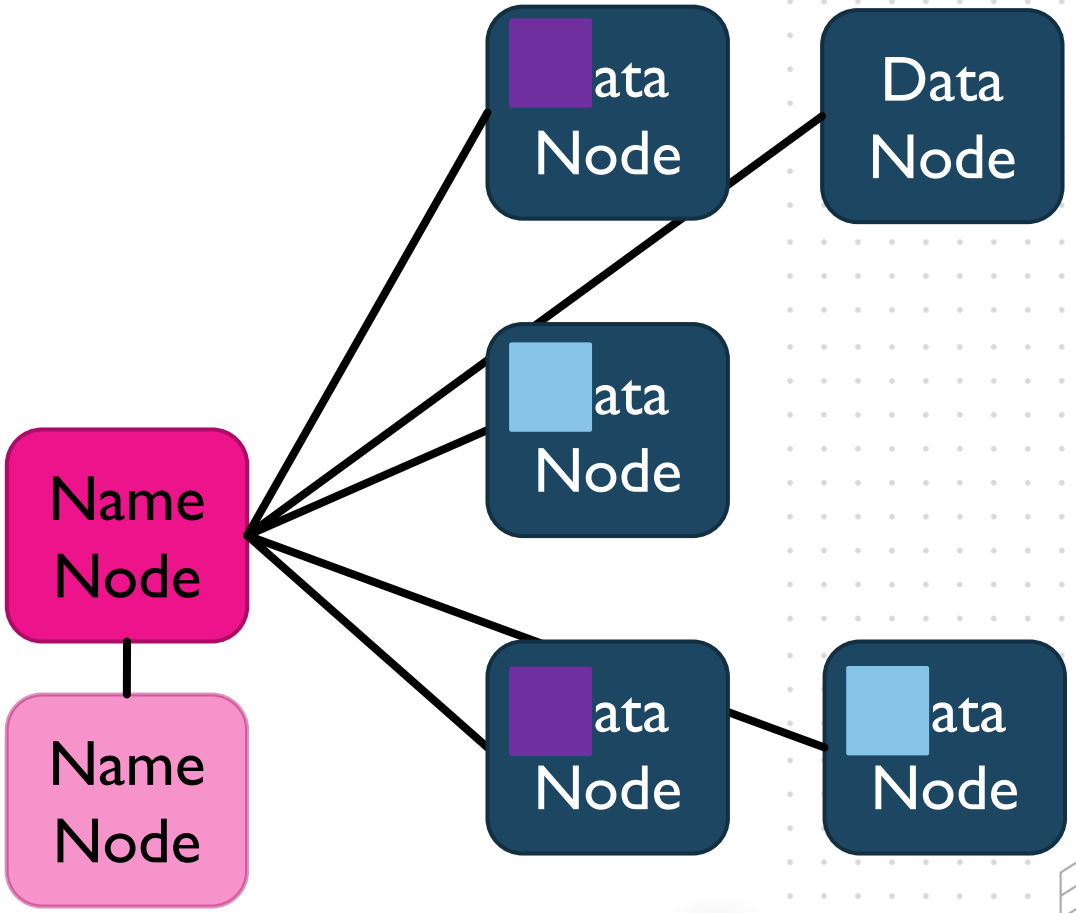


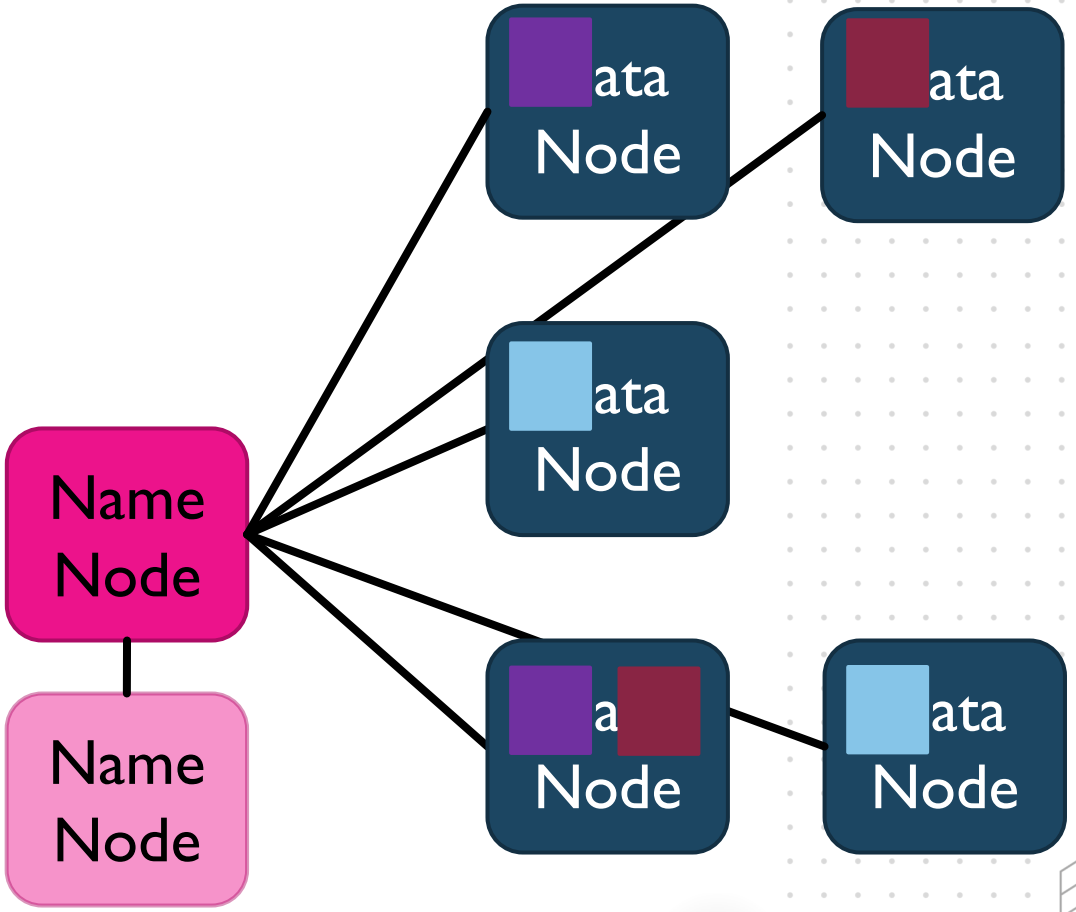

TOO BIG. I'LL BREAK IT UP.

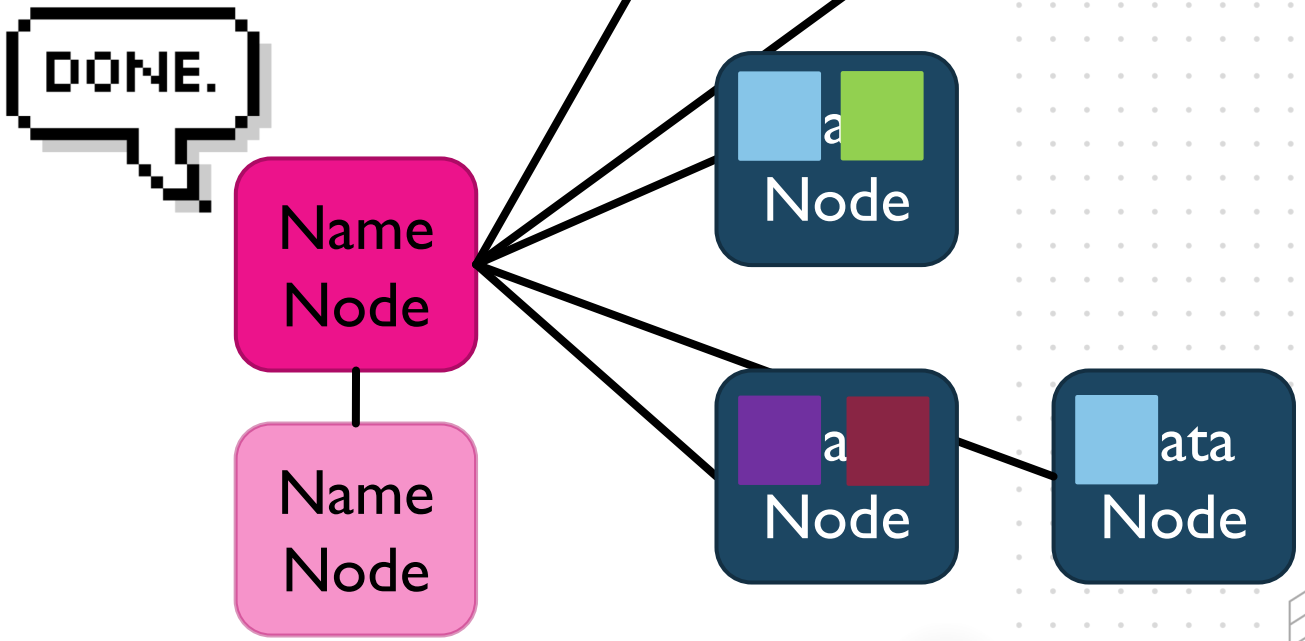

Name Node
Name Node


Data Node
Data Node
Data Node
Data Node
Data Node

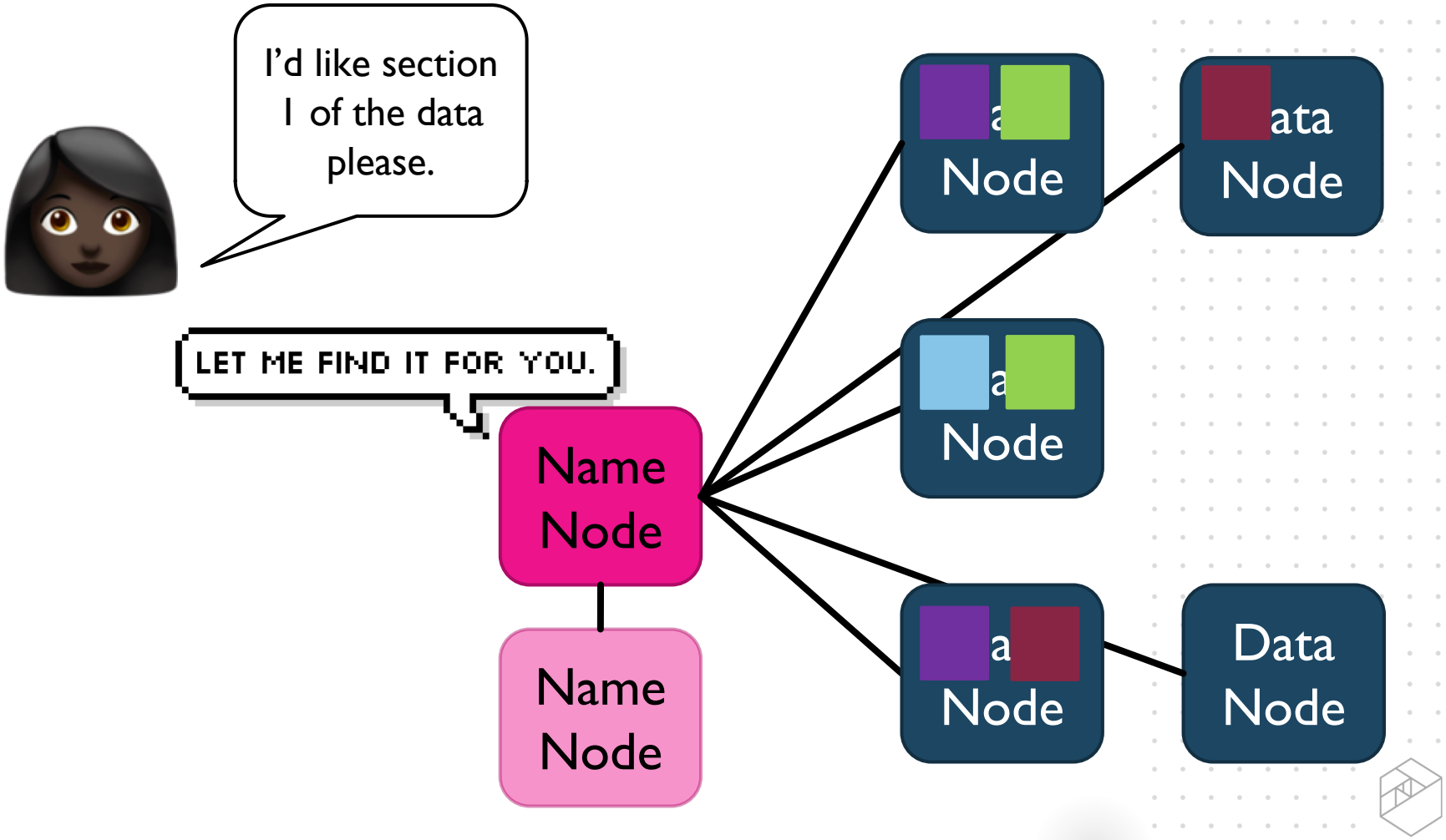

ata
Node
Data
Node
Data
Node
Name
Node
Name
Node
ata
Node
Data
Node

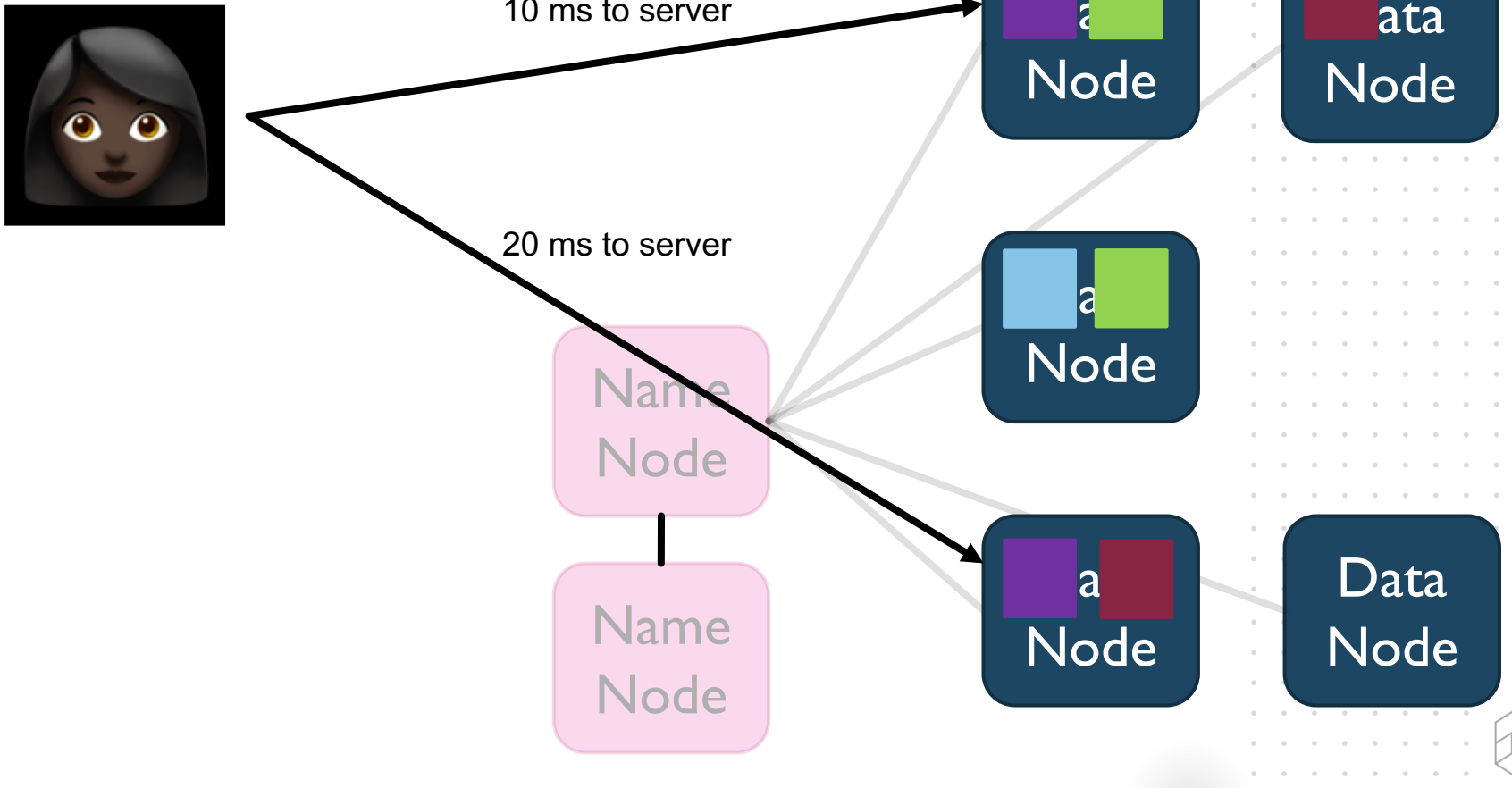

ata
Node
Data
Node
ata
Node
Name
Node
Name
Node
ata
Node
ata
Node

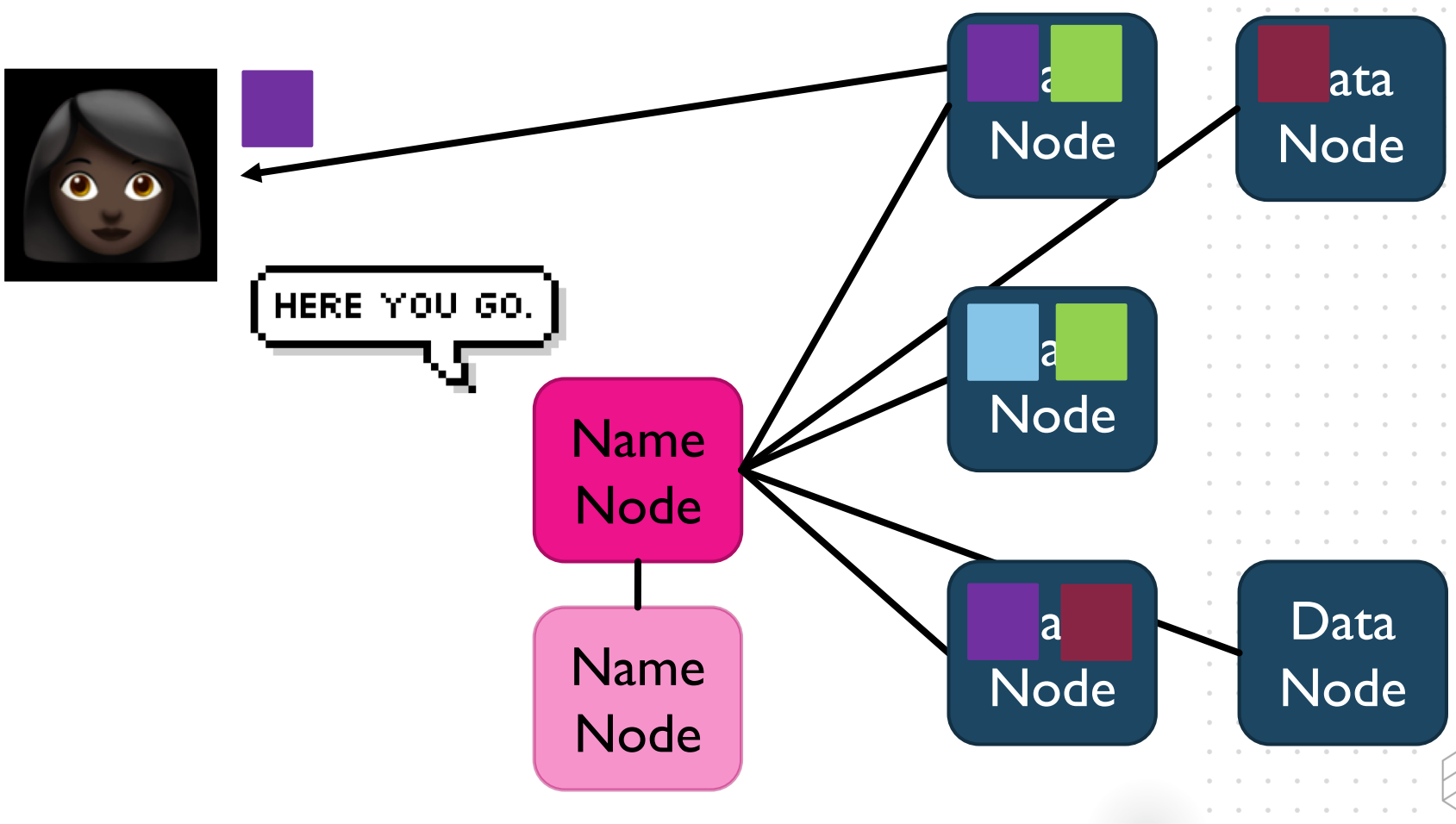

ata
Node
ata
Node
Name
Node
ata
Node
Name
Node
a
Node
ata
Node


DONE.
Name Node
Name Node
Node
ata Node
Node
Node
ata Node

WHAT IF WE WANT TO USE THE DATA?


I'd like section 1 of the data please.
LET ME FIND IT FOR YOU.
Name Node
Name Node
Node
ata Node
Node
Node
Data Node


10 ms to server
20 ms to server
Node
ata
Node
Node
Name
Node
Name
Node
Node
Data
Node

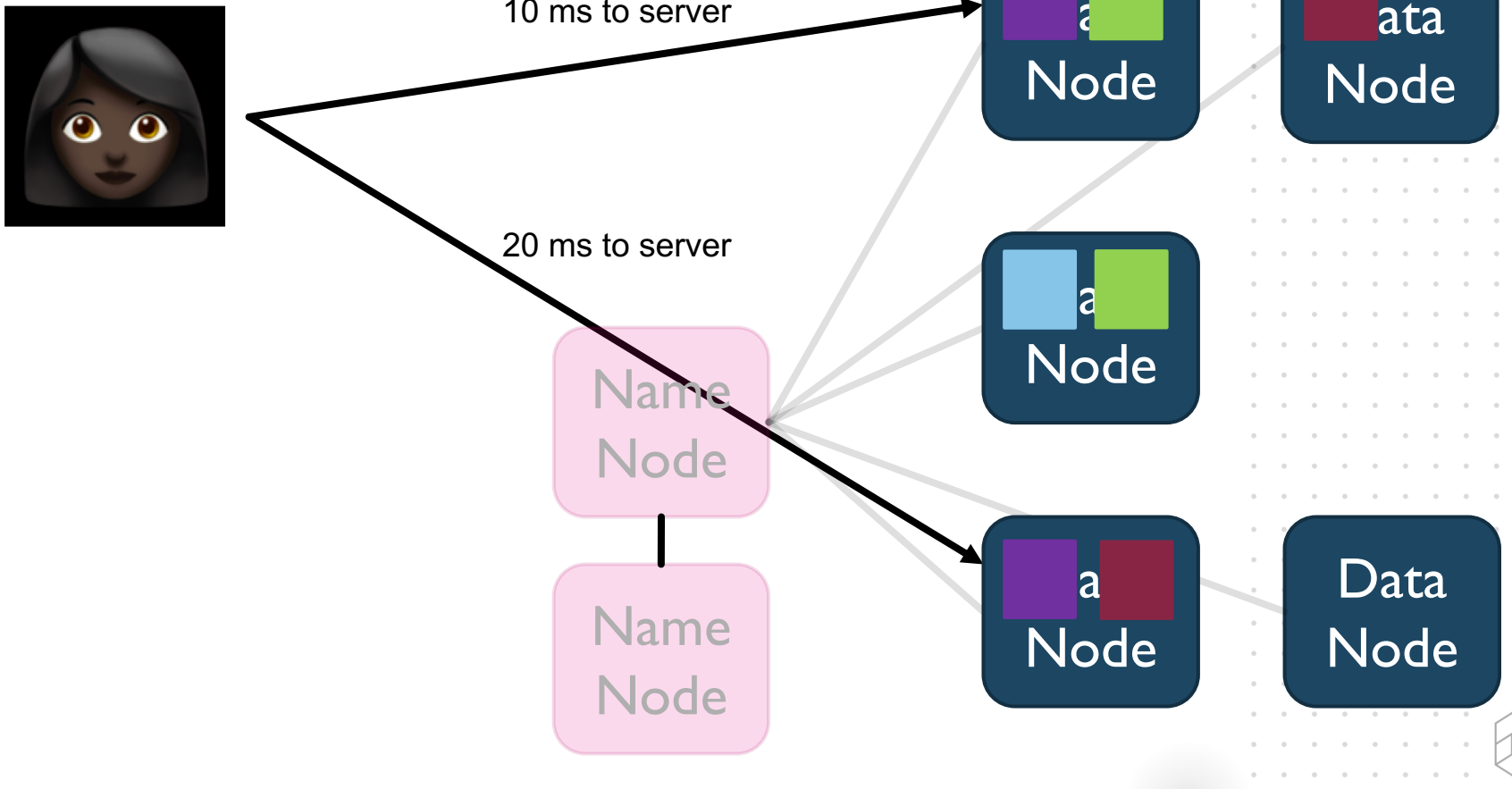

HERE YOU GO.
Name Node
Name Node
Node
ata Node
Node
Node
Data Node

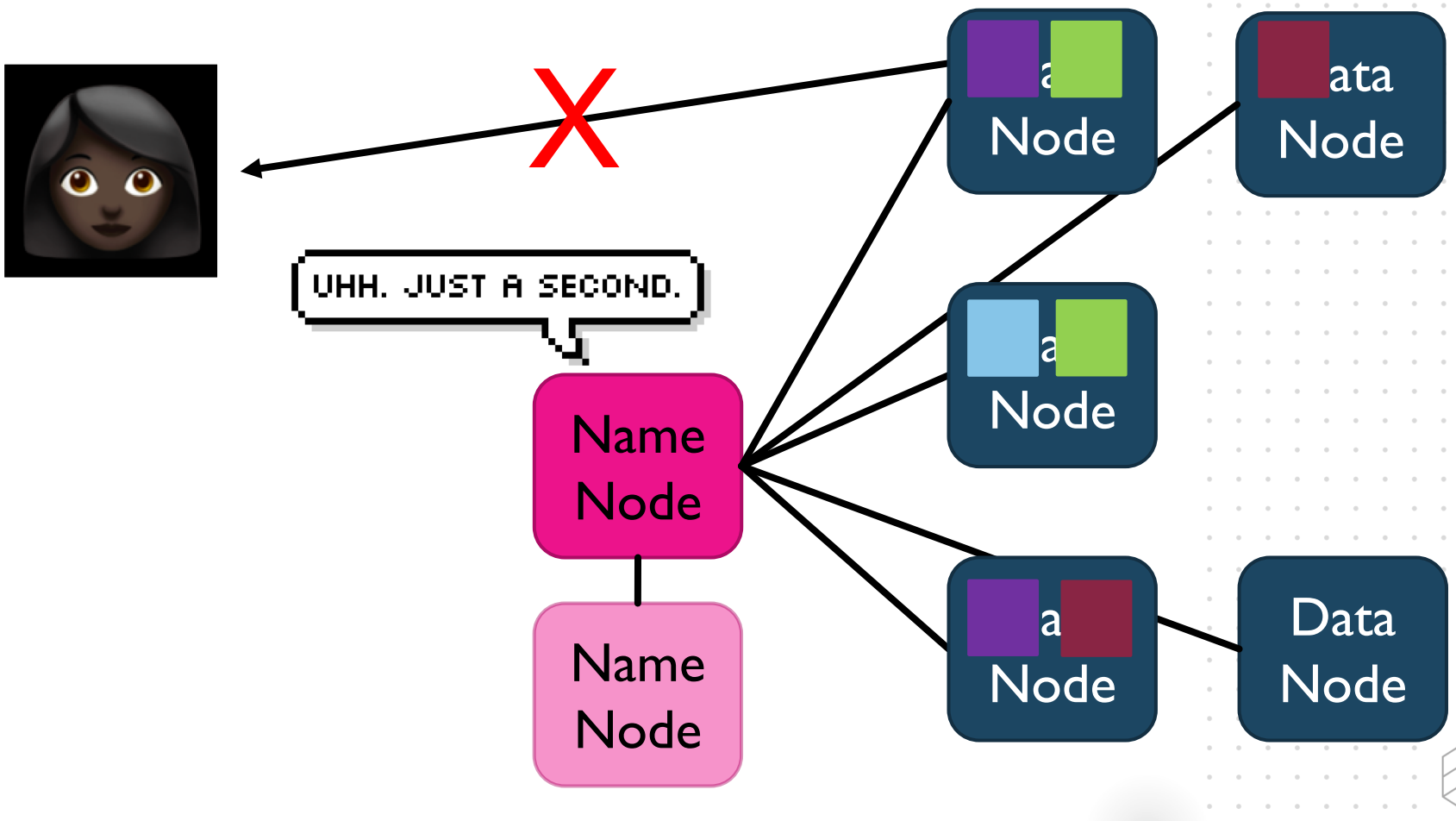
HDFS

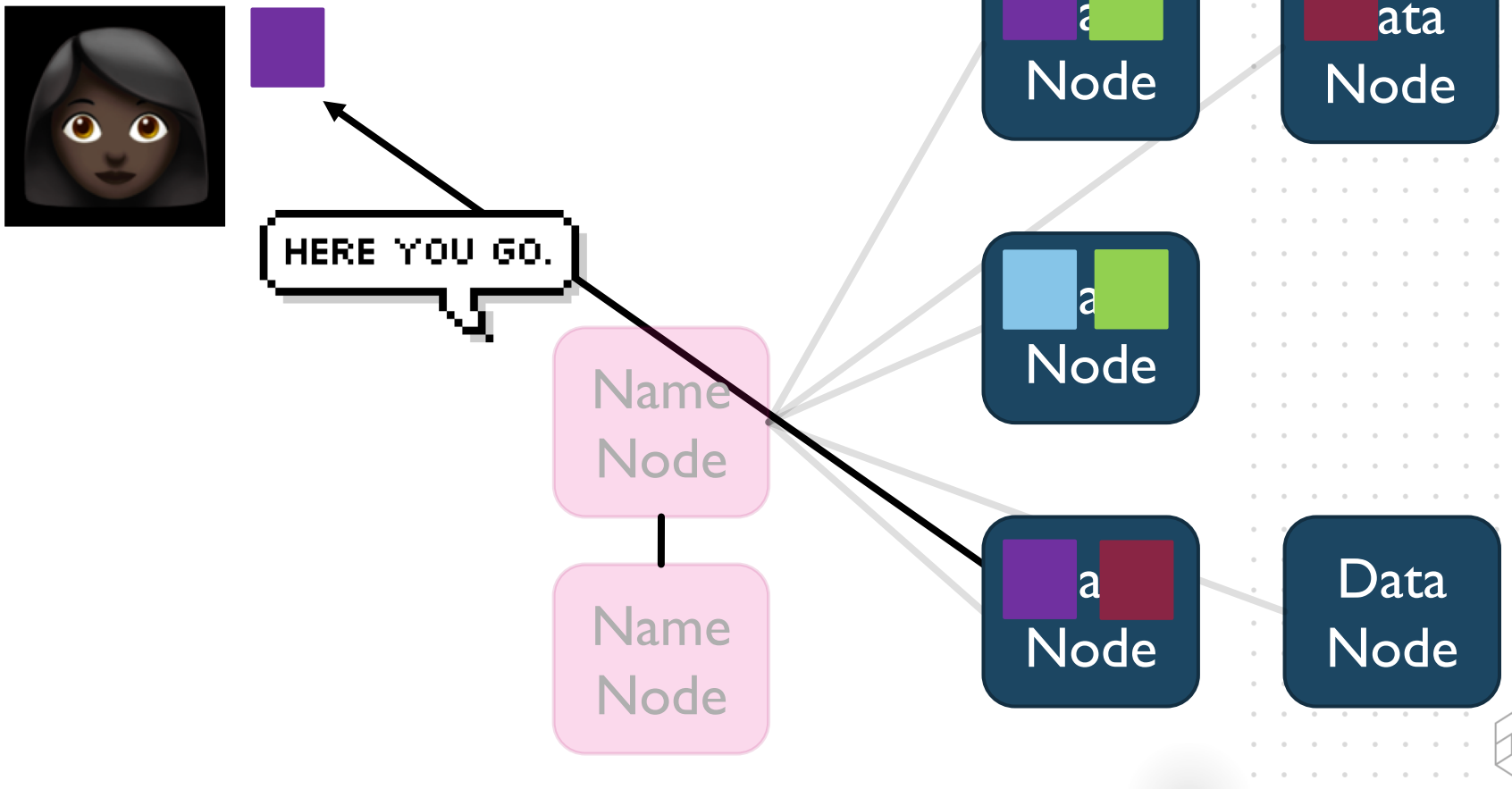
- Redundancy
- Central control for a user
- Data management through chunking

Commodity Hardware

- It’s cheaper to scale horizontally. Can always tie in more nodes from older hard drives, hard to get a 2,000,000 TB single drive.
- Allows us to salvage old systems and make a new, better system.
- Redundancy makes us fault tolerant when old hardware dies.


10 ms to server
20 ms to server
Name Node
Name Node
Node
ata Node
Node
Node
Data Node


UHH. JUST A SECOND.
Name Node
Name Node
Node
ata Node
Node
Node
Data Node


HERE YOU GO.
Name Node
Name Node
Node
ata Node
Node
Node
Data Node

HDFS I/O

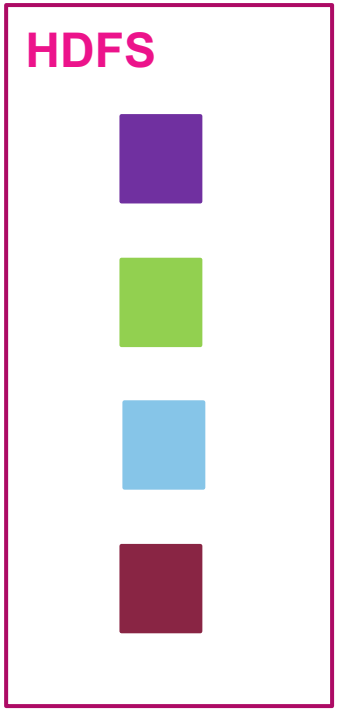
- HDFS supports:
  - Read
  - Write
  - Delete
- Once the data is on HDFS, it’s read-only, except for appending. Makes it safe from idiots using the system.

MAPREDUCE

Processing Data

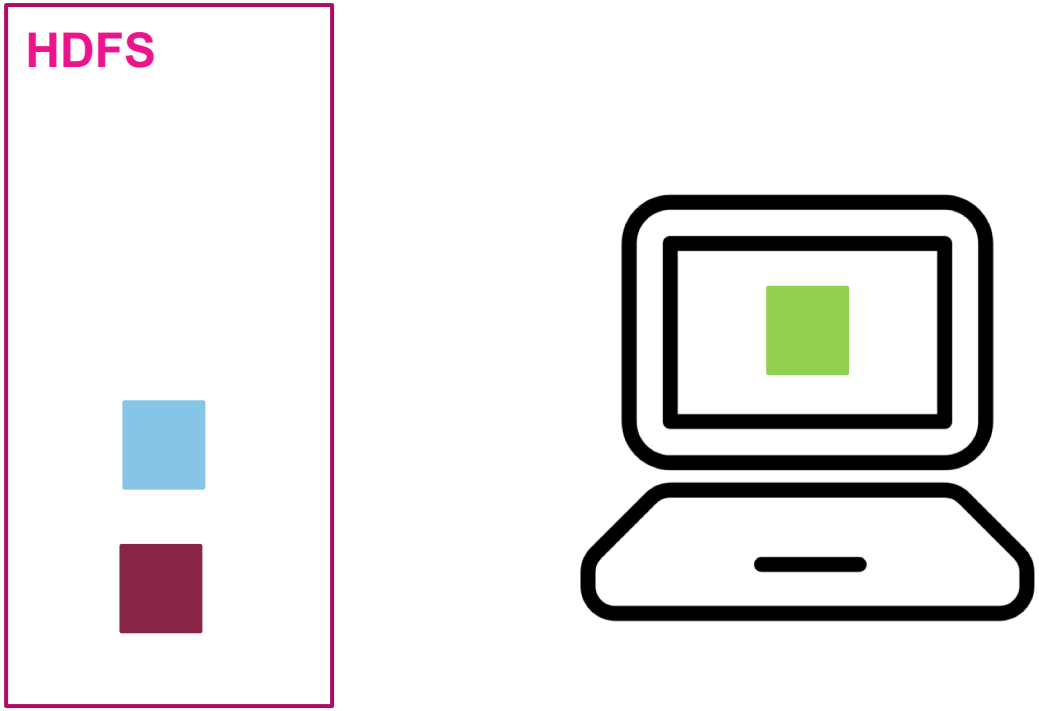
- We have two options:
  1. Pull the data from HDFS and process
  2. Make use of the cluster to process

Pull-and-Process

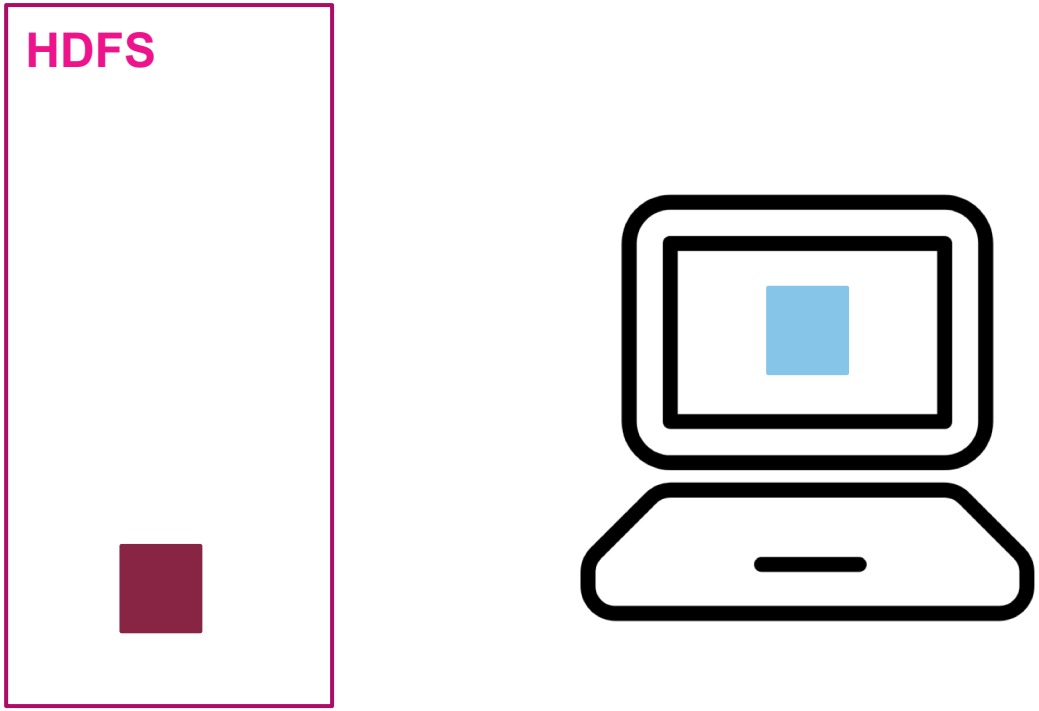


Pull-and-Process

Pull-and-Process

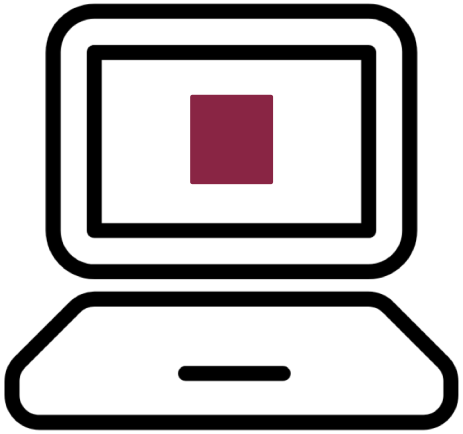


Pull-and-Process

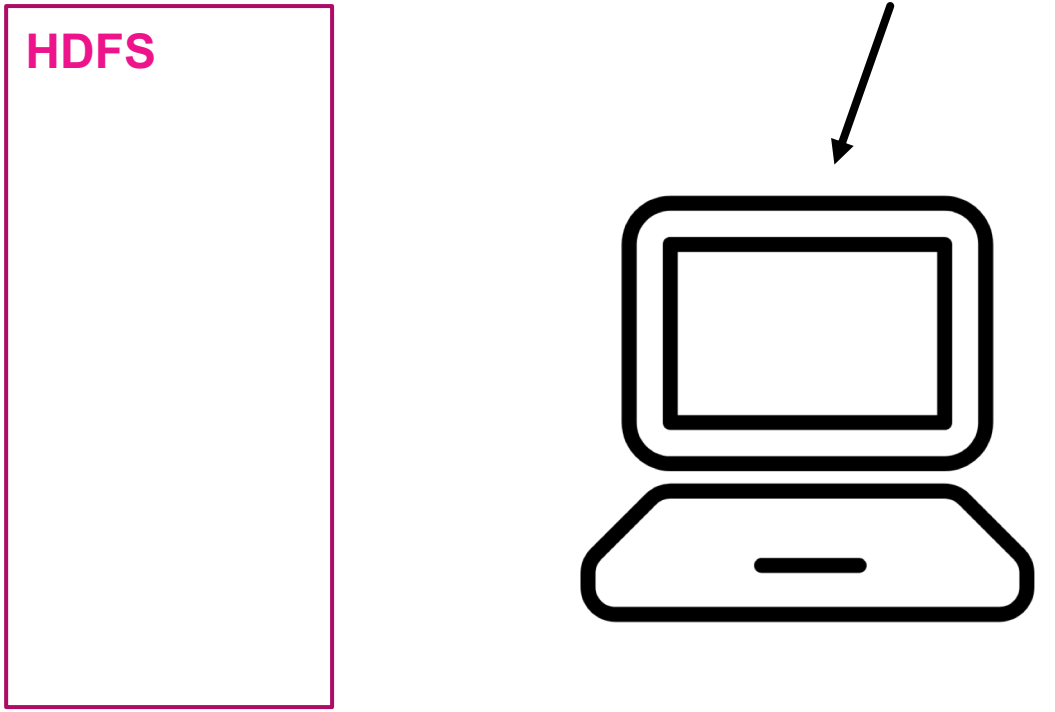


Pull-and-Process

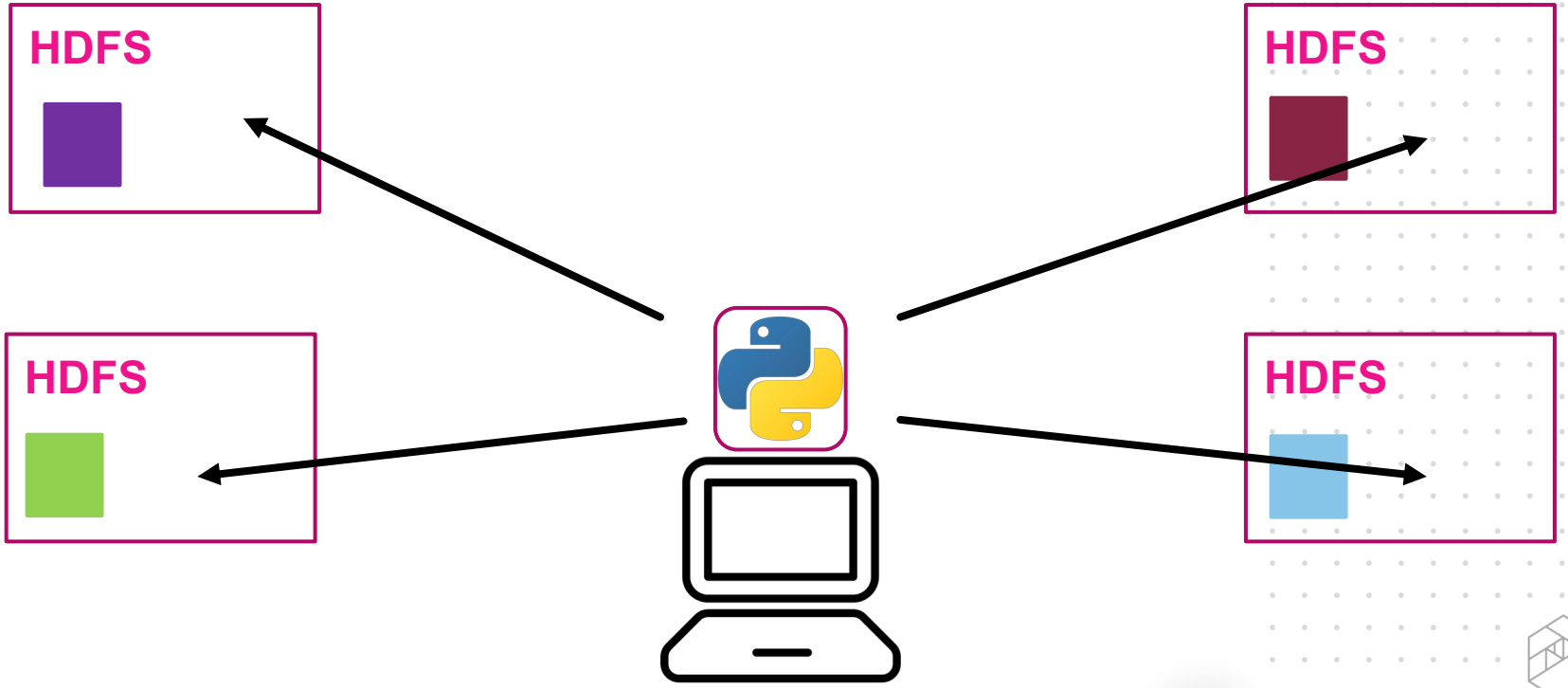
HDFS



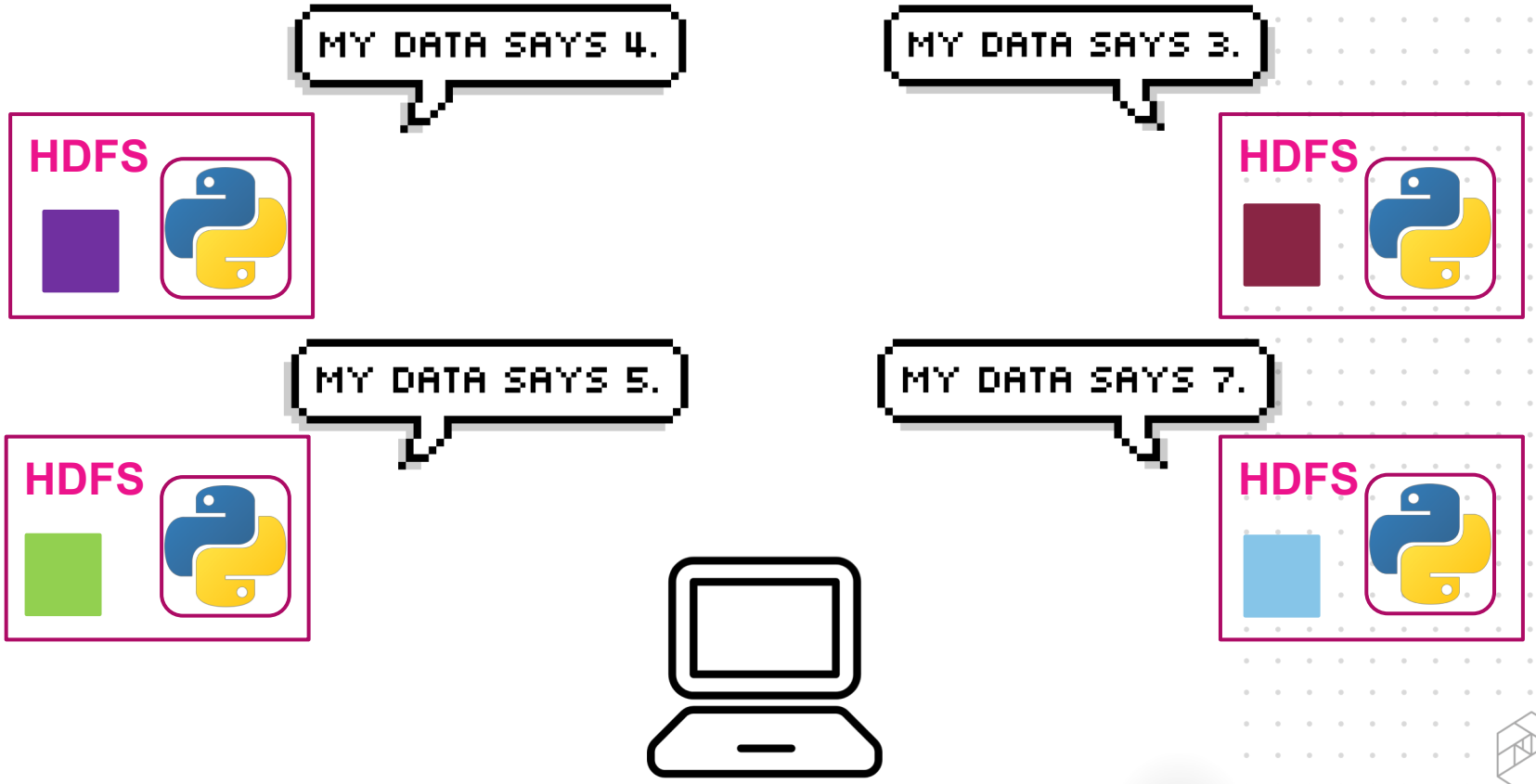
Pull-and-Process



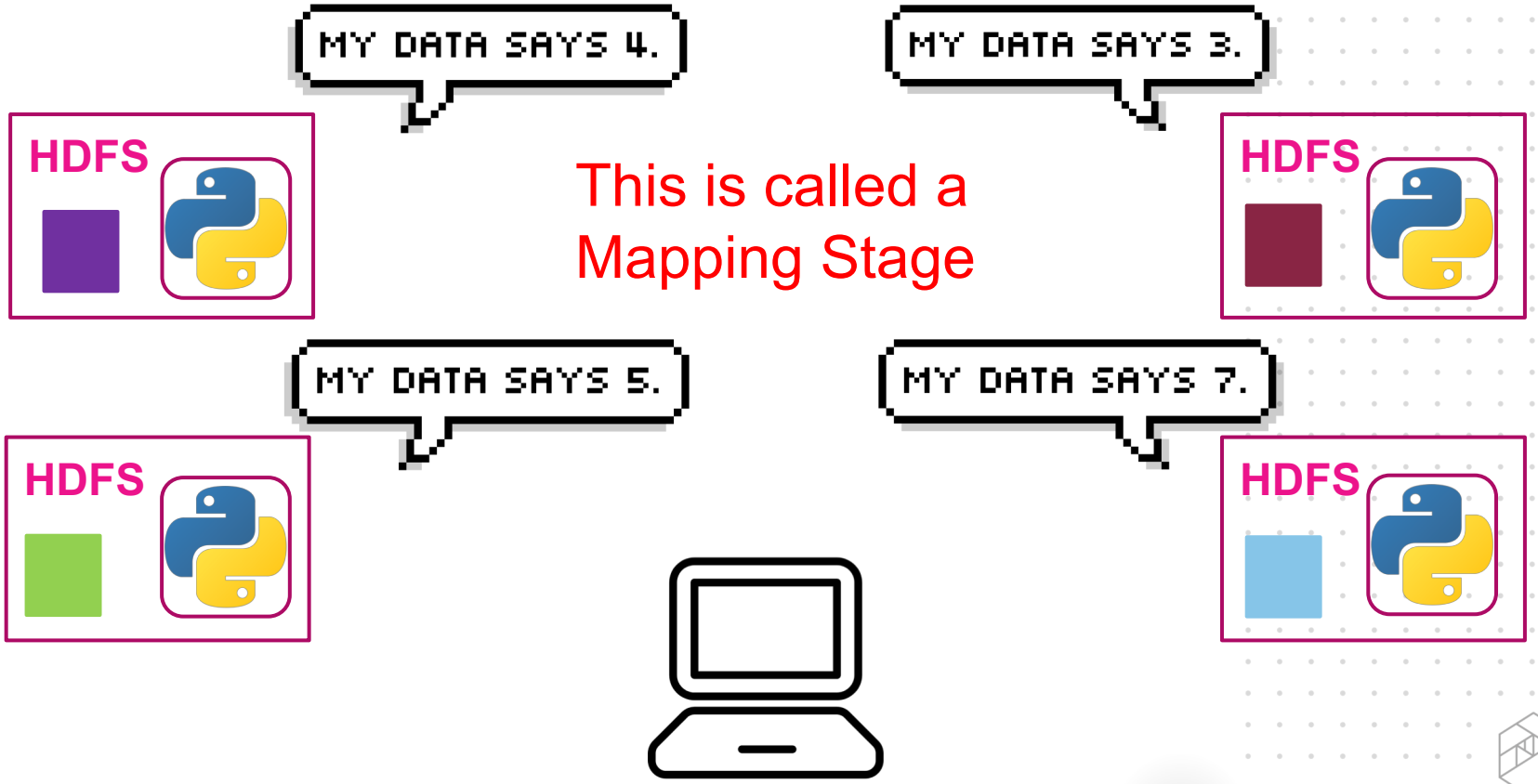
Use the Cluster



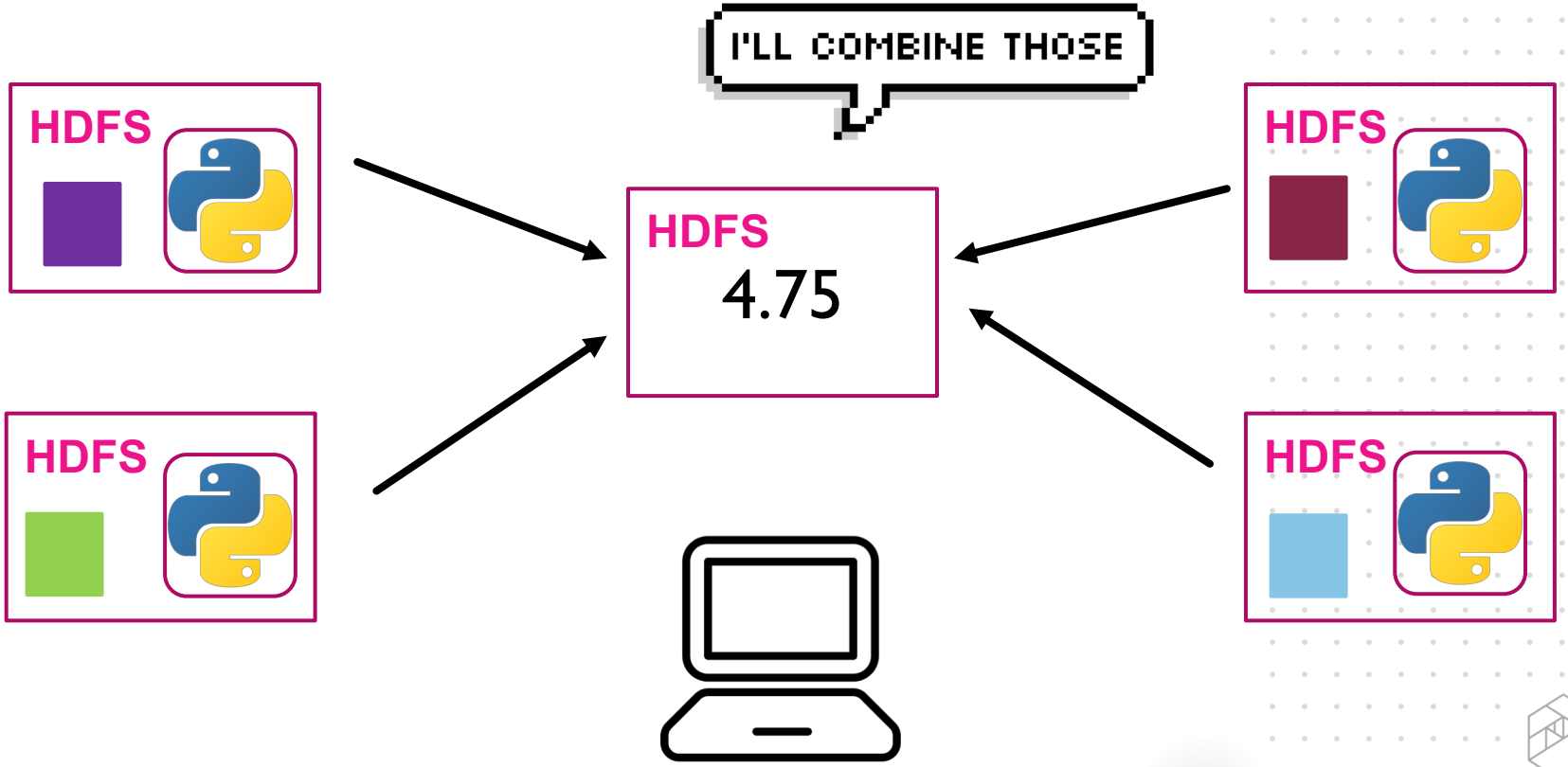
Use the Cluster



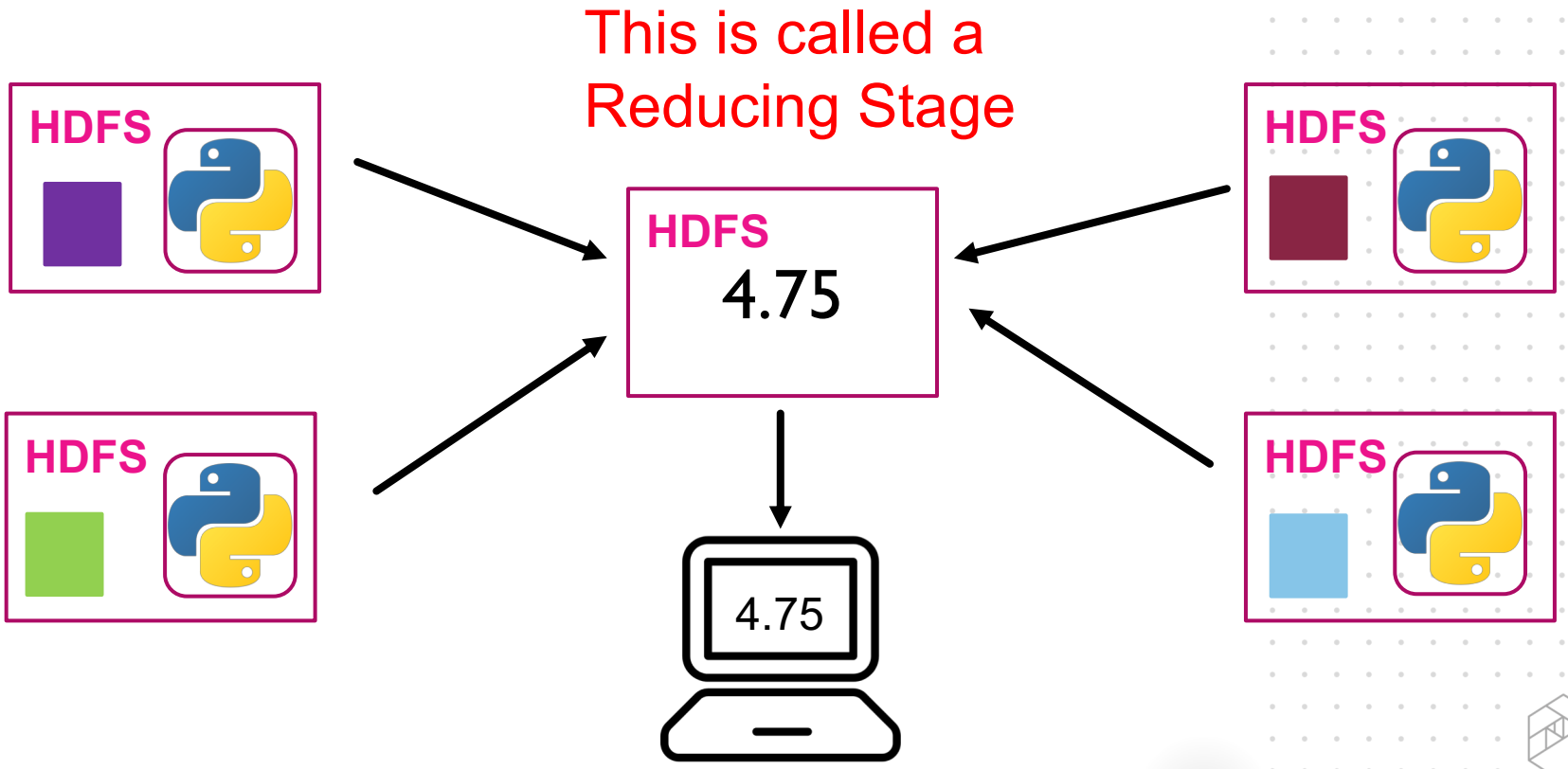
Use the Cluster



Use the Cluster



Use the Cluster



MapReduce

- The “ah-ha” of MapReduce is to push the work to the data
- Transferring scripts is easy. Transferring data is hard.
- “Convert a big problem into lots of small problems”

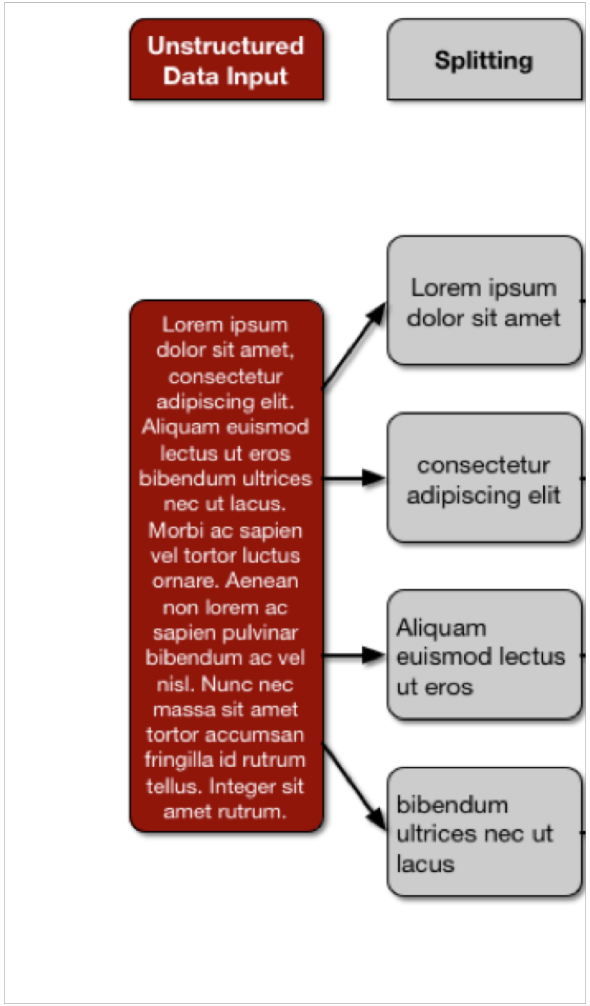
Map Phase

- Processes data on each node and outputs to a collection stream.
- Can be anything that has consistent output
- Can work on rows or blocks or raw text, etc

Reduce Phase

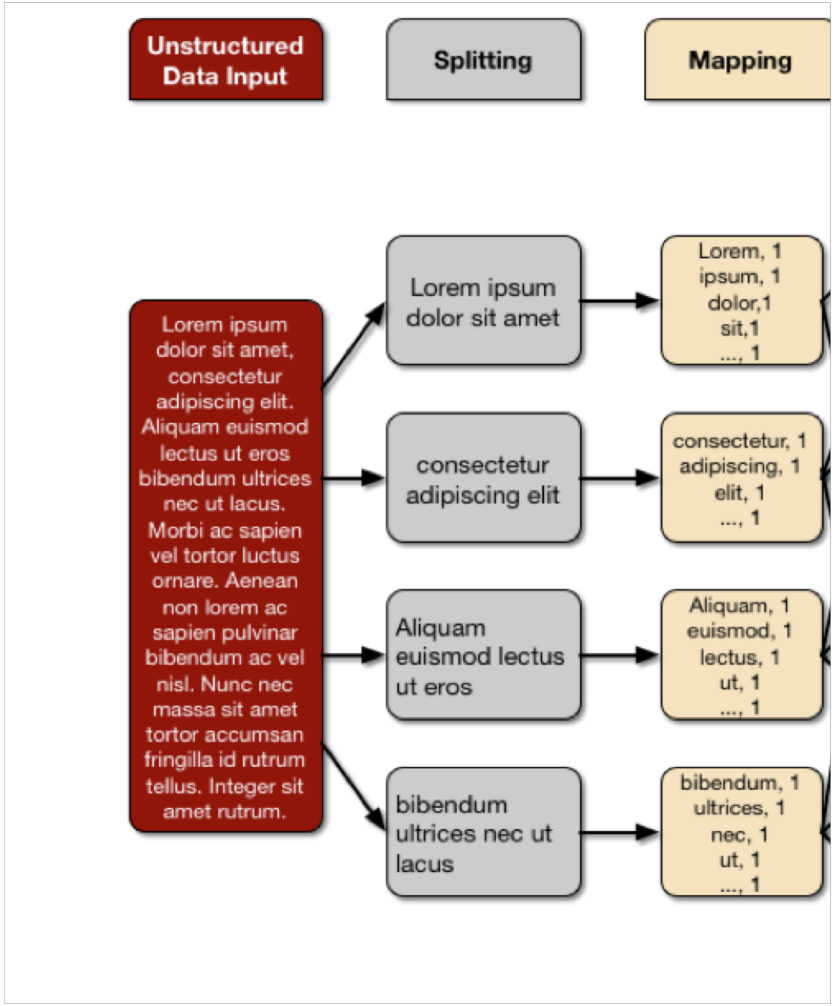
- Takes all the outputs from the Mappers and combines them into a condensed output
- Could be averaging or counting or filtering or whatever you can code
- Passes output to either next reducing phase or back to the user

USING MAPREDUCE TO COUNT WORDS



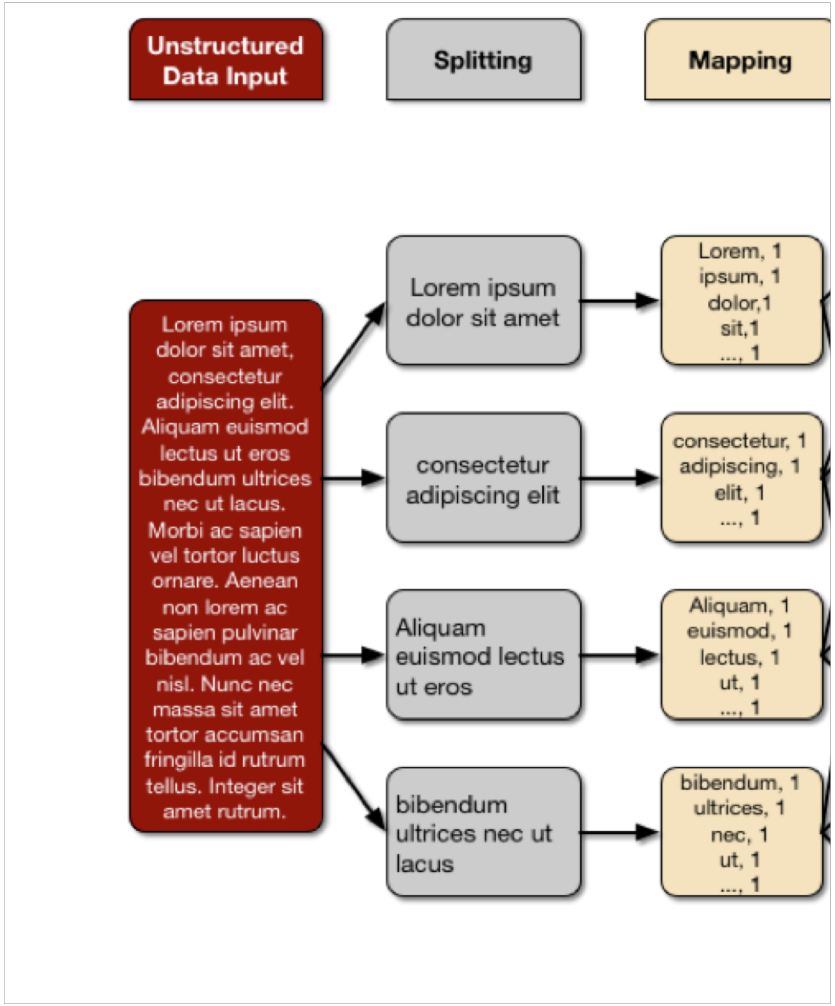
We load our data onto HDFS and have it chunked up into different DataNodes.

If we want to count the total number of occurrences for each word, how could we do it with mapping?



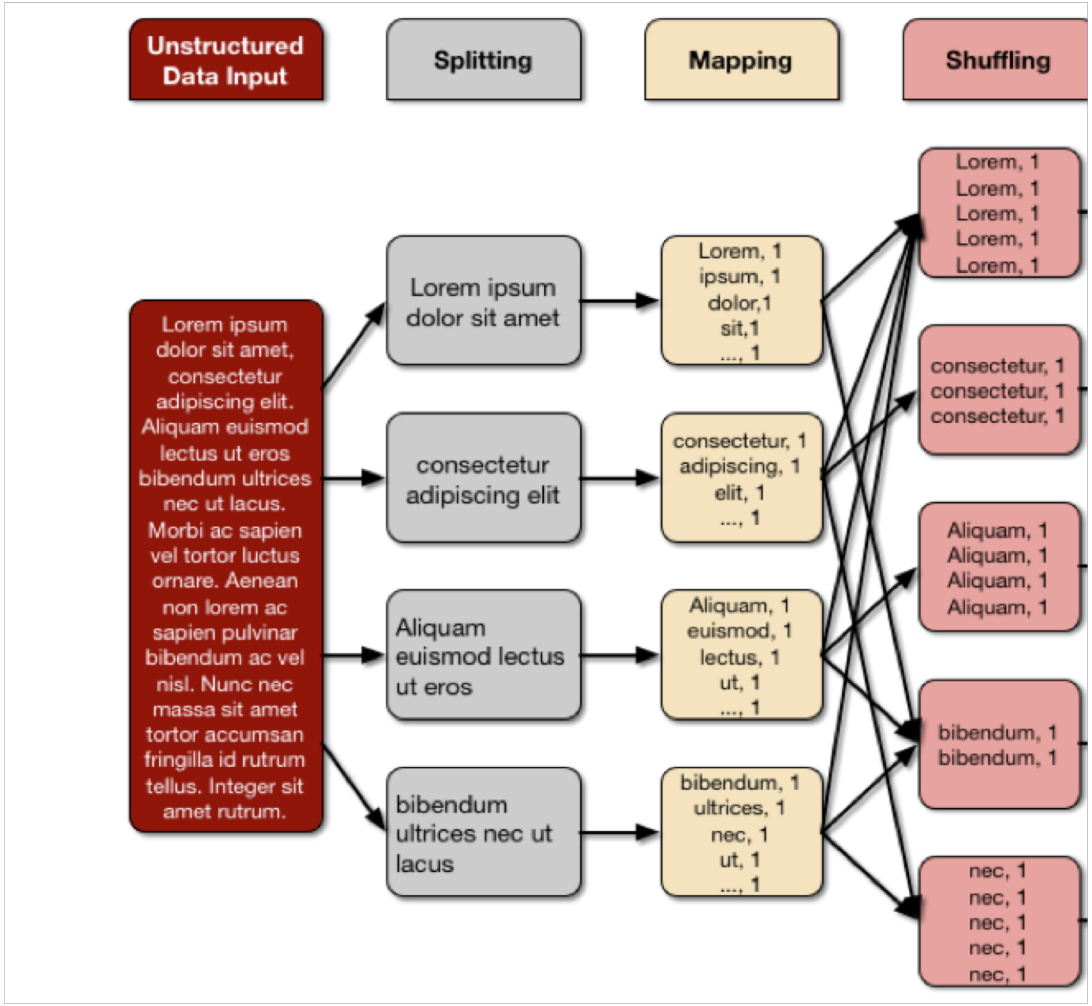
We have each a program that writes each word into a stream. It's traditional to do it in key-value pairs. So we see:

```
word, 1 (appearance)
```



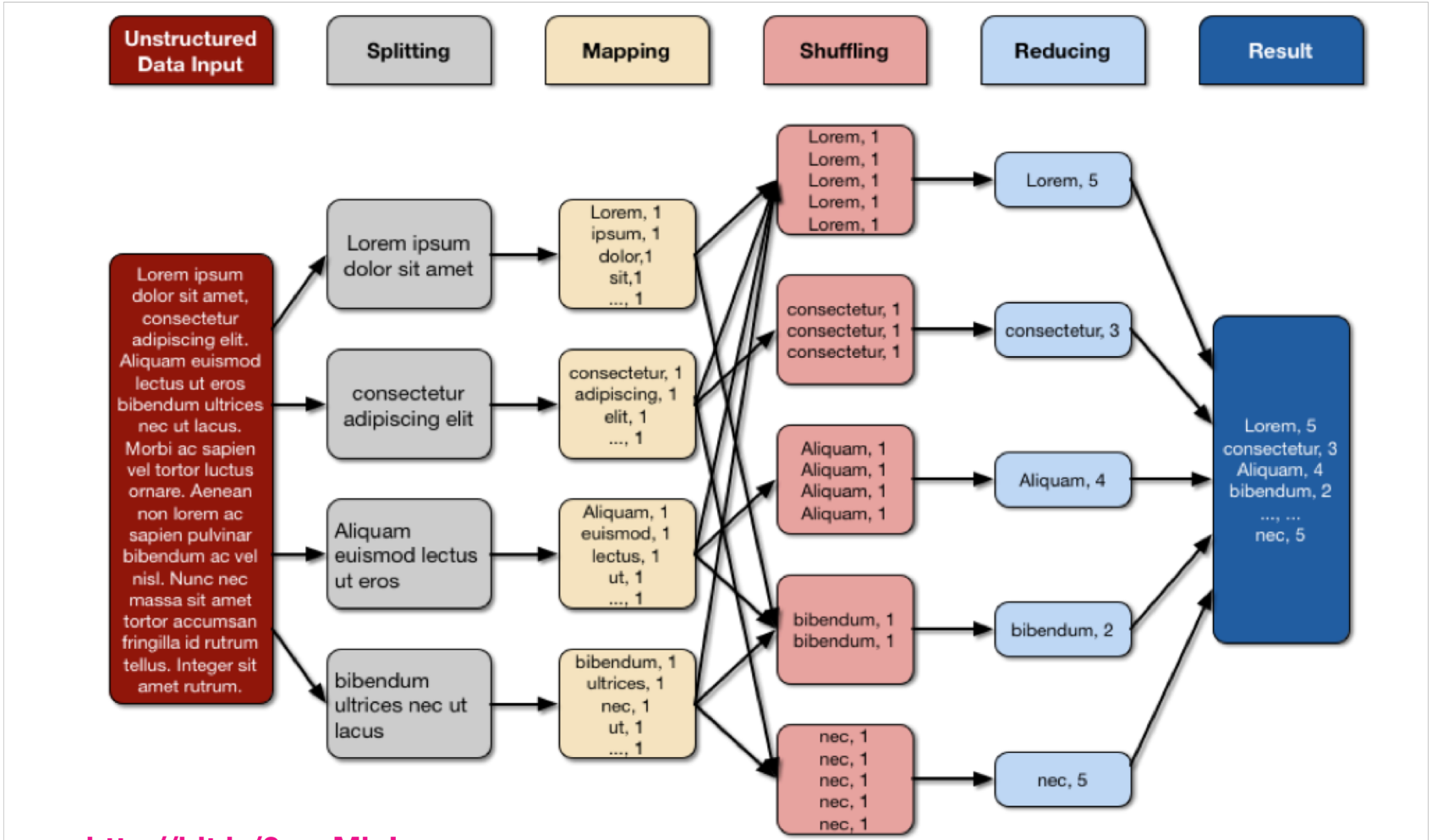
One of the keys to MapReduce is to write data only to the stream. No objects that require long-term memory, so we can't use dictionaries to count.

So how can we reduce?



Sometimes we sort the data via a “shuffle” stage to help the reducer be more efficient.

Now that the data looks like this, how could we reduce?



Source: http://bit.ly/2zueMbJ

When should I use Hadoop?

- Massive data. Must be at multiple terabytes or it simply isn't worth the overhead of maintaining a cluster.
- Hadoop isn't a "machine learning tool" it's about maintaining, pipelining, and cleaning data. We can make it do ML, but remember that it's designed for ETL.
- MapReduce reads from hard drive, so it can work, but it's slower than from RAM.

What else should I know?

- There’s a whole suite of Hadoop based tools.
- Hive allows SQL queries to become MapReduce programs automatically
- Pig is another processing library that looks a lot like SQL
- Sqoop allows you to load data from SQL databases to HDFS
- Etc, etc, etc. This is a well supported tool set and if you need something, someone has likely built it.

HIVE

- Hive auto-converts SQL queries into MapReduce programs and can run on the distributed data
- Can work with both structured and unstructured data using it’s special version of SQL called HiveQL
- Extremely robust to faults and great for exploring data
- Essentially allowed the use of Hadoop as an analytics tool

What else should I know?

- However, almost all of the Hadoop tools are now outperformed by Spark.
- So the main legacy of Hadoop is HDFS and its management tools like YARN. You don’t need to understand those deeply – but know that HDFS is special because almost all the big data tools use it (like Spark).
- Understanding Hadoop and the Map-Reduce pattern forms the basis for all future big data technologies, so take this to heart.

QUESTIONS?

Cites

- Pixel speech bubbles: https://pixelspeechbubble.com/