Lecture 2: XSEDE Jetstream, MapReduce Programming Model, and Problem Solving

COSC 526: Introduction to Data Mining

Spring 2020

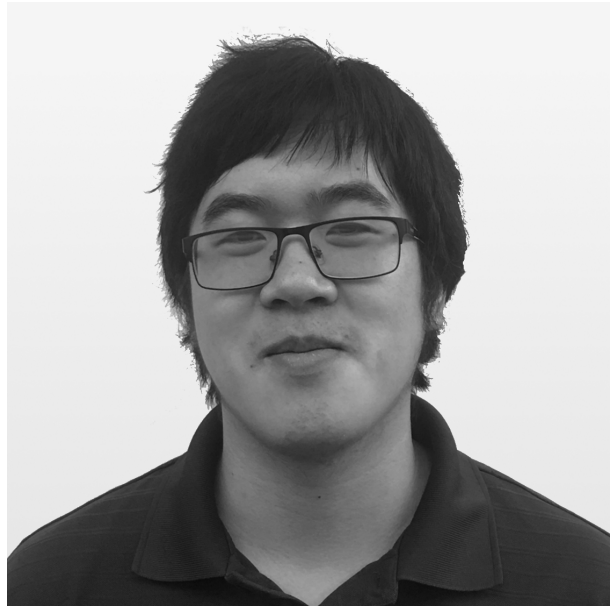
THE UNIVERSITY OF TENNESSEE KNOXVILLE

Instructors:

Michela Taufer

Danny Rorabaugh

GRA:



Nigel Tan

Topics covered today

- The XSEDE Jetstream cloud
  - Creature your XSEDE account
- The MapReduce Programming model
  - Capture high level concepts
- Problem solving
  - Learn how different solutions can have different execution times

Jetstream
-
a brief introduction

NSF Funding Areas in HPC

Traditionally concentrated on enabling petascale capability
- Blue Waters – 13.3 petaflops, 2012
- Stampede – 9.6 petaflops, 2013
- Comet – ~2.0 petaflops, 2014

Has funded research into building clouds and computer science
- CloudLab
- Chameleon

Now funding clouds to do research
- Bridges (Hybrid system)
- Jetstream

From slides by Jeremy Fischer
jetstream-cloud.org/files/Jetstream-SIAM-CSE-Presentation.pdf

What is Jetstream and why does it exist?

- NSF's first production cloud facility
- Part of the NSF eXtreme Digital (XD) program
- Provides on-demand *interactive* computing and analysis
- Enables *configurable* environments and architectures
- User-friendly, widely accessible cloud environment
- User-selectable library of preconfigured virtual machines

From slides by Jeremy Fischer
jetstream-cloud.org/files/Jetstream-SIAM-CSE-Presentation.pdf

HPC vs Cloud

Adapting to a different environment:

- No reservations, no queueing
- More interactive use and less/no batch queuing
- What? No parallel filesystem?!?
- Being your own admin – hey, we have root!
- You really can have almost any (linux) software you want**
- Constantly getting new features (https://www.openstack.org/software/project-navigator/)

** Here there be dragons…

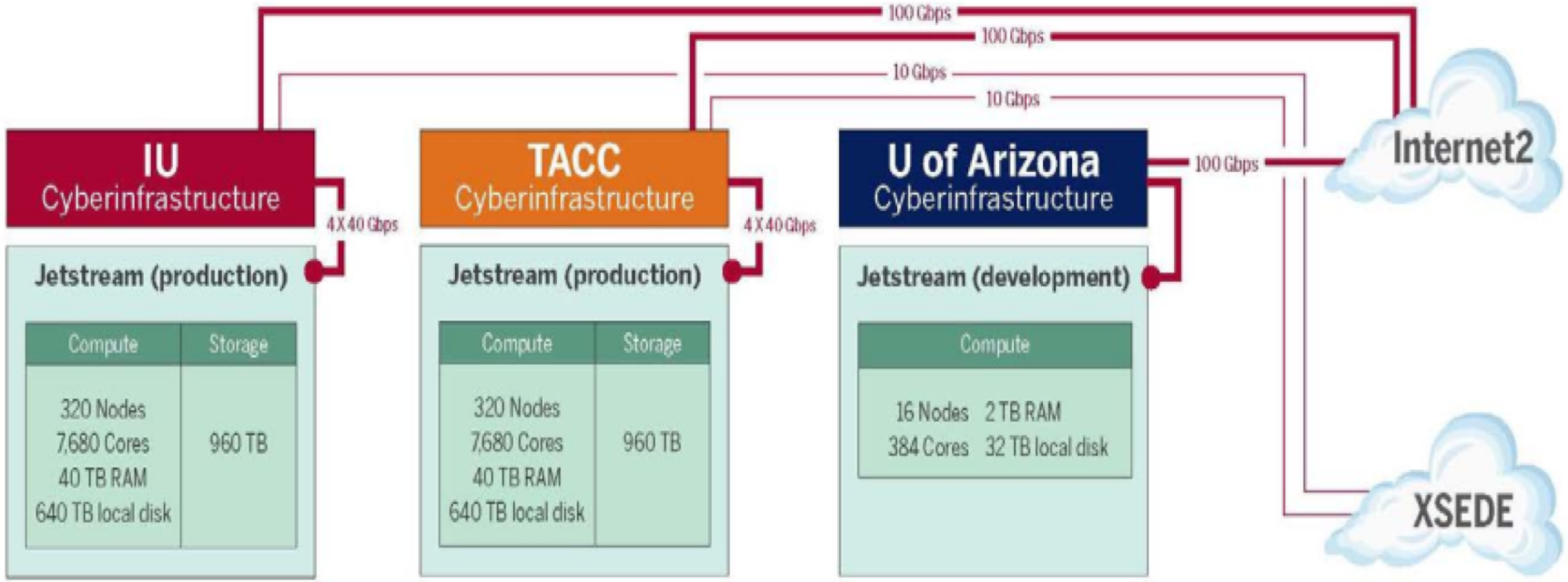
From slides by Jeremy Fischer
jetstream-cloud.org/files/Jetstream-SIAM-CSE-Presentation.pdf

Who uses Jetstream?

- The researcher needing a handful of cores (1 to 44/vCPU)
- Software creators and researchers needing to create their own customized virtual machines and workflows
- Science gateway creators using Jetstream as either the frontend or processor for scientific jobs
- STEM Educators teaching on a variety of subjects

From slides by Jeremy Fischer
jetstream-cloud.org/files/Jetstream-SIAM-CSE-Presentation.pdf

Jetstream System Overview



From slides by Jeremy Fischer
jetstream-cloud.org/files/Jetstream-SIAM-CSE-Presentation.pdf

Hardware and Instance "Flavors"

VM Host Configuration

- Dual Intel E-2680v3 "Haswell"
- 24 physical cores/node @ 2.5 GHz (Hyperthreading on)
- 128 GB RAM
- Dual 1 TB local disks
- 10GB dual uplink NIC
- Running KVM Hypervisor

Flavor	vCPUs	RAM	Storage	Per Node
m.tiny	1	2	8	46
m.small	2	4	20	23
m.medium	6	16	60	7
m.large	10	30	120	4
m.xlarge	24	60	240	2
m.xxlarge	44	120	480	1

- Short-term storage comes as part of launched instance
- Long-term storage is XSEDE-allocated
- Implemented on backend as OpenStack Volumes
- Each user gets 10 volumes up to 500GB total storage
- Piloting object storage as well after recent update

From slides by Jeremy Fischer
jetstream-cloud.org/files/Jetstream-SIAM-CSE-Presentation.pdf

Requesting access to Jetstream

- You can request startup allocations anytime. (Startups are simple!)
- You can request allocations for educational use anytime.

From slides by Jeremy Fischer
jetstream-cloud.org/files/Jetstream-SIAM-CSE-Presentation.pdf

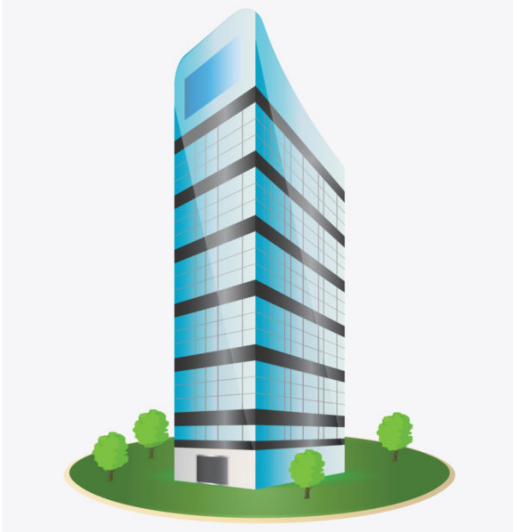
Create your XSEDE Jetstream account

- Create your account by following the directions in the file: JetstreamGuide.ipynb
- Enter your XSEDE Jetstream user name here:

  https://tinyurl.com/wwfcysm
- Your account will be active in a couple of days

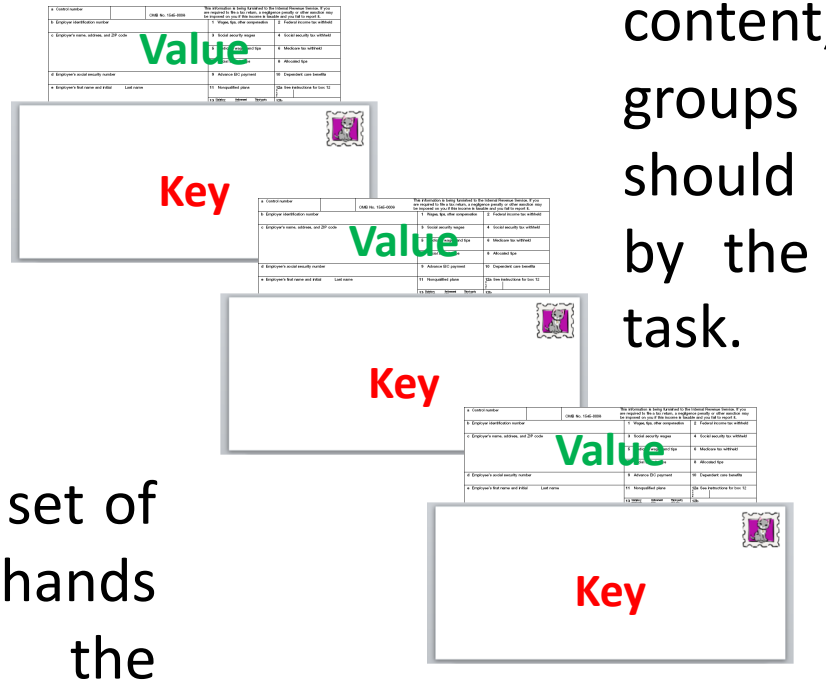
The Canonical MapReduce Programming Model

The MapReduce Workflow



Step 1: Map

The **map** task creates a set of **Key-Value** pairs and hands them off to the **sorting**/**shuffling** task.

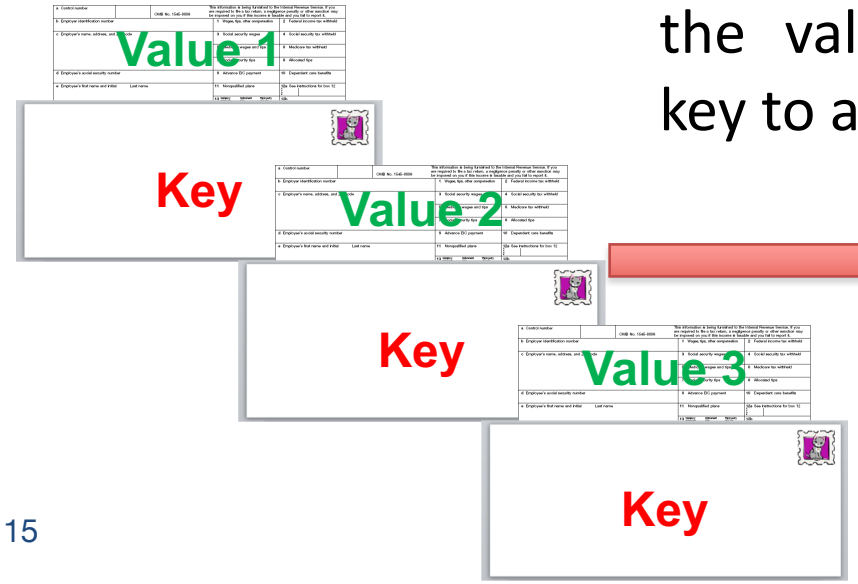


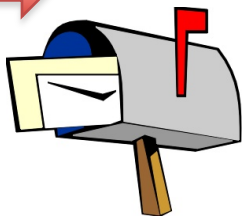
The **Value** is the content, the **Key** groups content that should be processed by the same **reduce** task.

The MapReduce Workflow

Step 2: Sort/Shuffle

The **sorting** and **shuffling** tasks collect all the **key-value** pairs and deliver all the values with the same key to a **reduce** task.



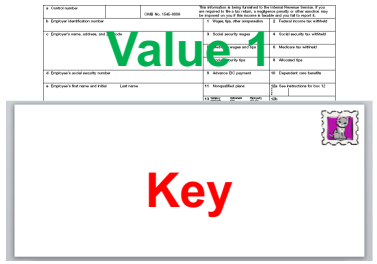


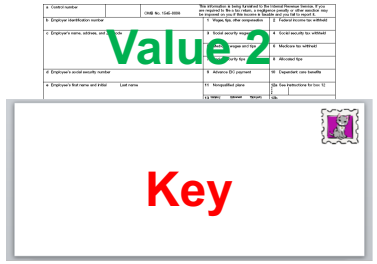
15

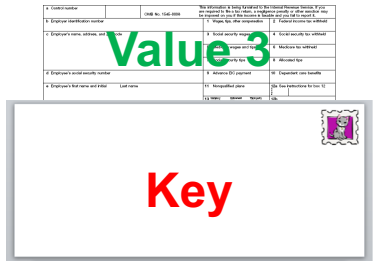
The MapReduce Workflow

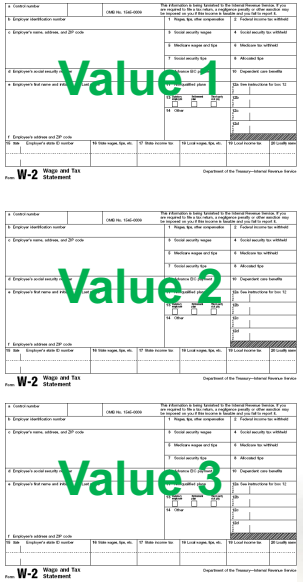
Step 3: Reduce

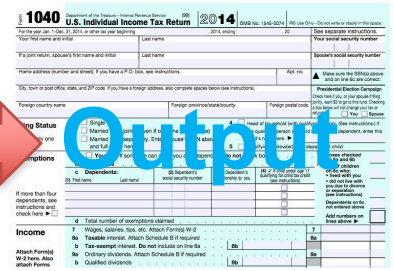
The **reduce** task receives **all** the **values** with the **same key** and combines them into the final output.



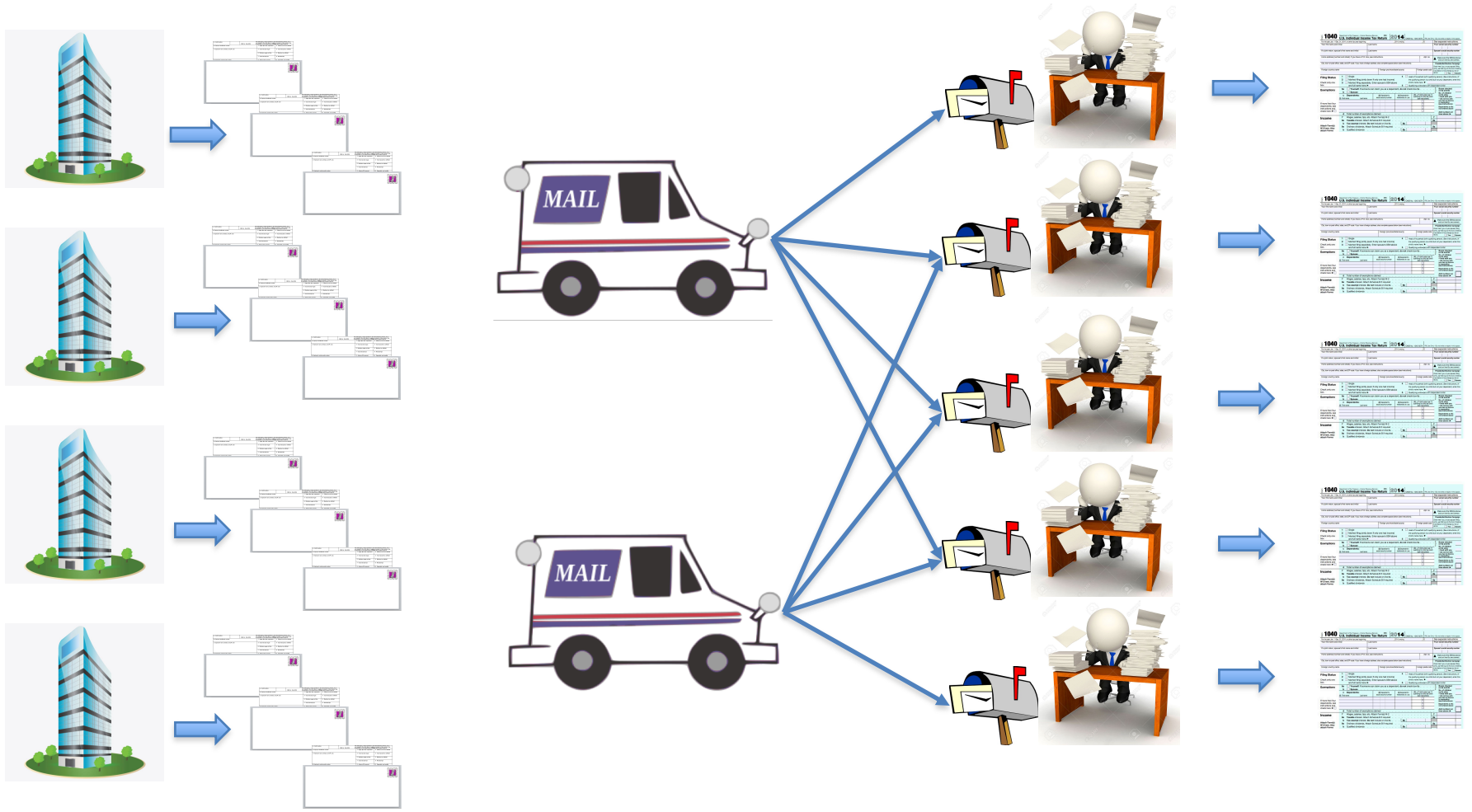








The MapReduce Workflow



Map

- The map function is implemented by the user.
- Each map task processes a single line of an input file at a time.
- Map tasks do not communicate with other map tasks.
- Map tasks communicate with reduce tasks **only** through the content of the value in the KV pair.
- A single map task may emit **any number** of KV pairs (including none).

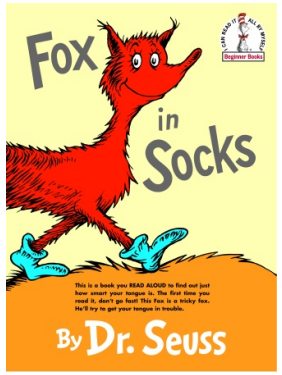
Sort/Shuffle

- The sorting and shuffling process is implemented at the framework level and is opaque to the user.

Reduce

- The reduce function is implemented by the user.
- Each reduce task receives **all** the **values** associated with a given **key**.
- Reduce tasks do not communicate with each other
- A single reduce task may emit **any number** of result values for each **key** it processes.

WordCount: an example of MapReduce



Map

Key **Value**

When tweetle beetles fight, → (When, 1), (**tweetle**, **1**), (beetles, 1), (fight, 1)

it's called a tweetle beetle battle. → (it's, 1), (called, 1), (a, 1), (tweetle, 1), (beetle, 1), (battle, 1)

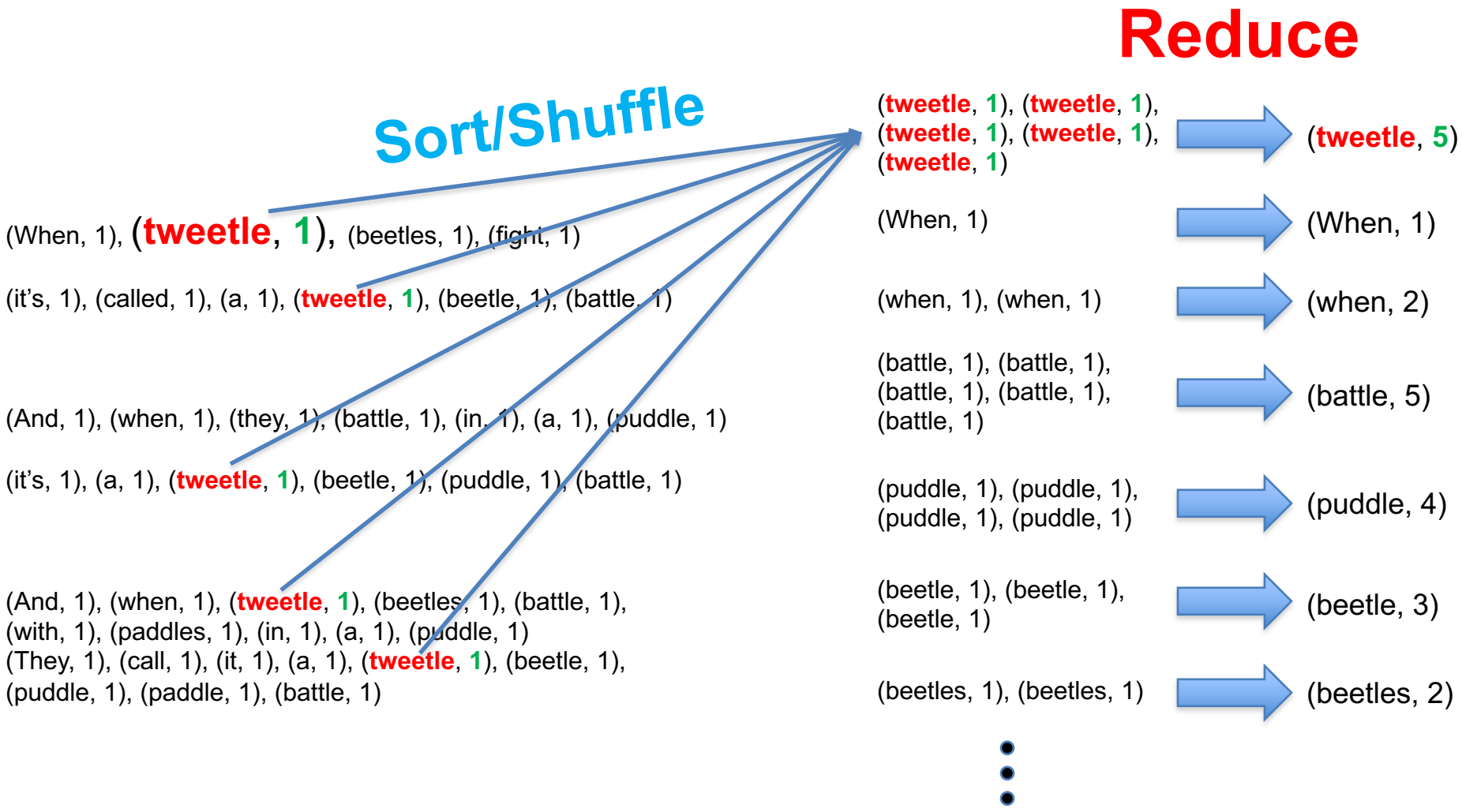
And when they battle in a puddle, → (And, 1), (when, 1), (they, 1), (battle, 1), (in, 1), (a, 1), (puddle, 1)

it's a tweetle beetle puddle battle. → (it's, 1), (a, 1), (tweetle, 1), (beetle, 1), (puddle, 1), (battle, 1)

And when tweetle beetles battle with paddles in a puddle, → (And, 1), (when, 1), (tweetle, 1), (beetles, 1), (battle, 1), (with, 1), (paddles, 1), (in, 1), (a, 1), (puddle, 1)

They call it a tweetle beetle puddle paddle battle. → (They, 1), (call, 1), (it, 1), (a, 1), (tweetle, 1), (beetle, 1), (puddle, 1), (paddle, 1), (battle, 1)

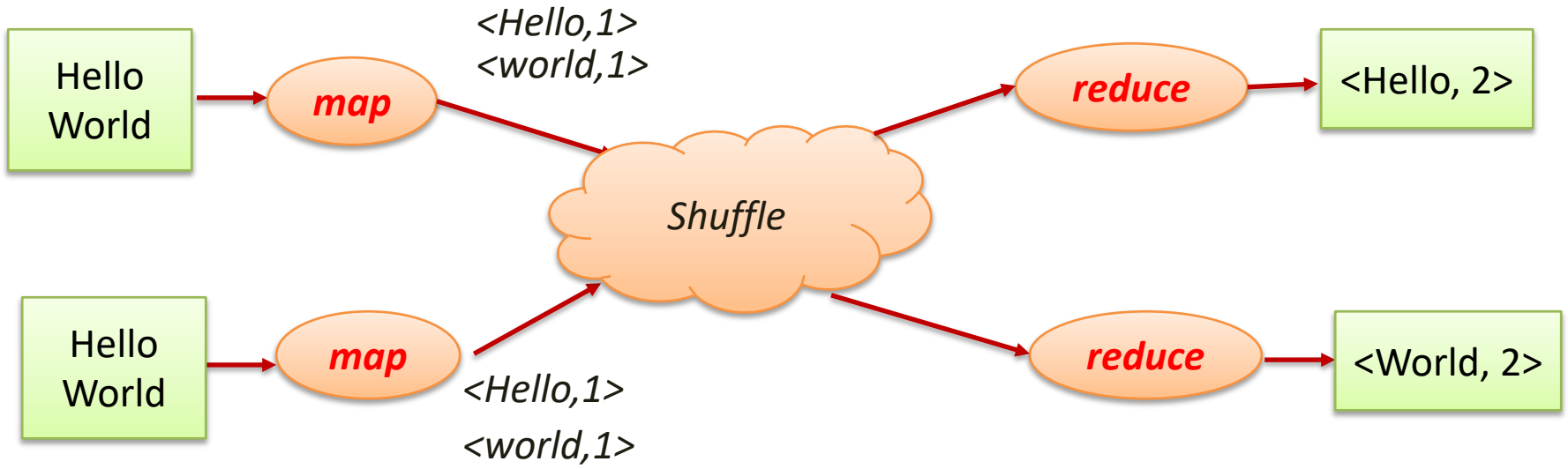
WordCount: an example of MapReduce



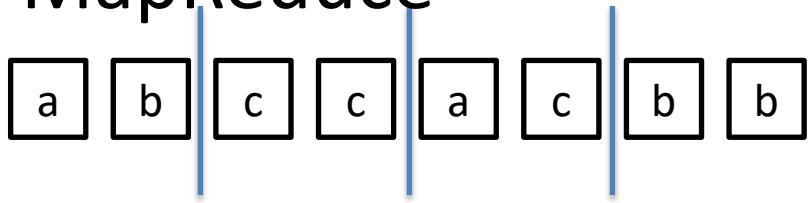
Canonical MapReduce

- MapReduce runtime handles the parallel job execution, communication, and data movement
- Users provide *map* and *reduce* functions

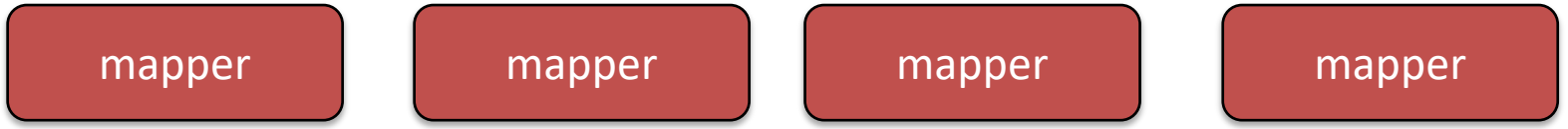
Wordcount example:



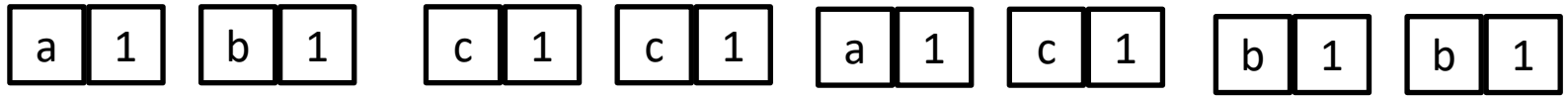
The **Canonical** MapReduce



Mappers: applied to all input data

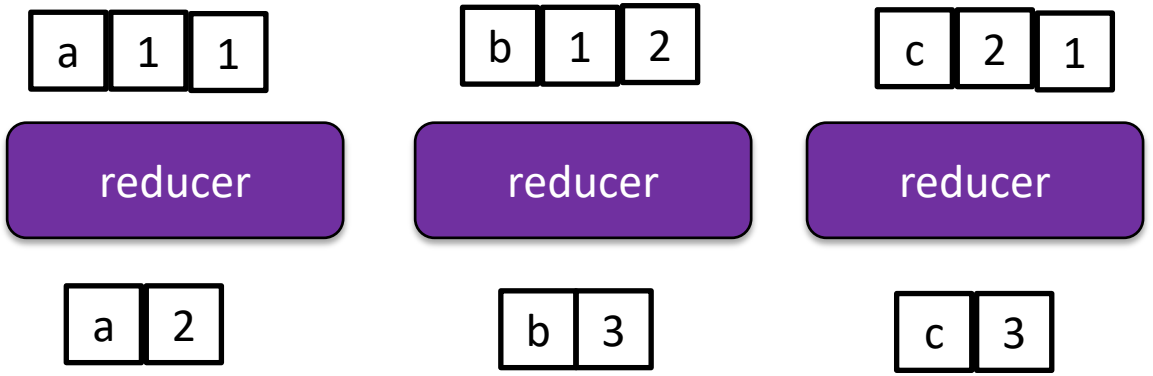


Arbitrary number of key-value pairs



Barrier: distributed sort and group by key

Reducers: applied to all values associated with the same key

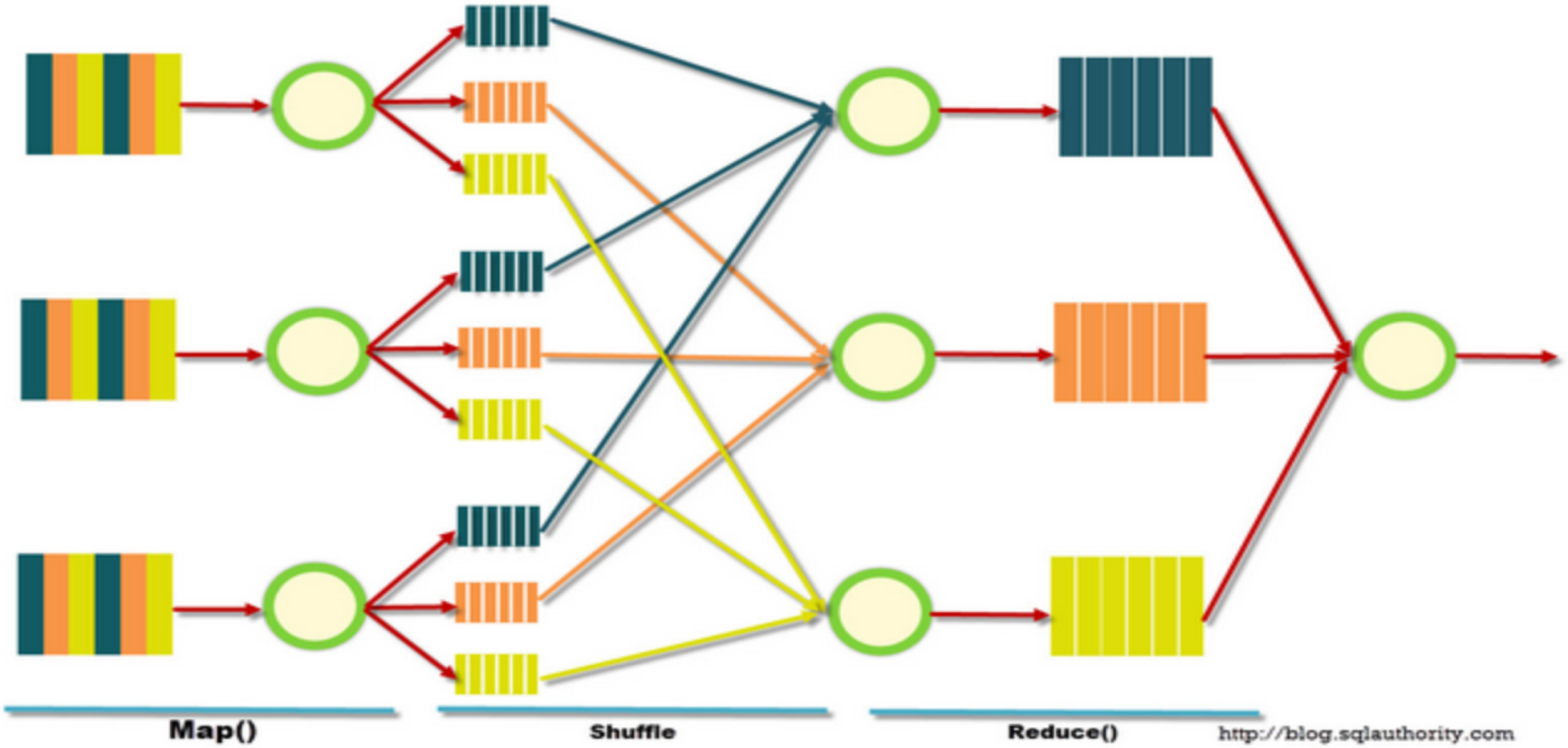


Pseudo-code: WordCount in MapReduce

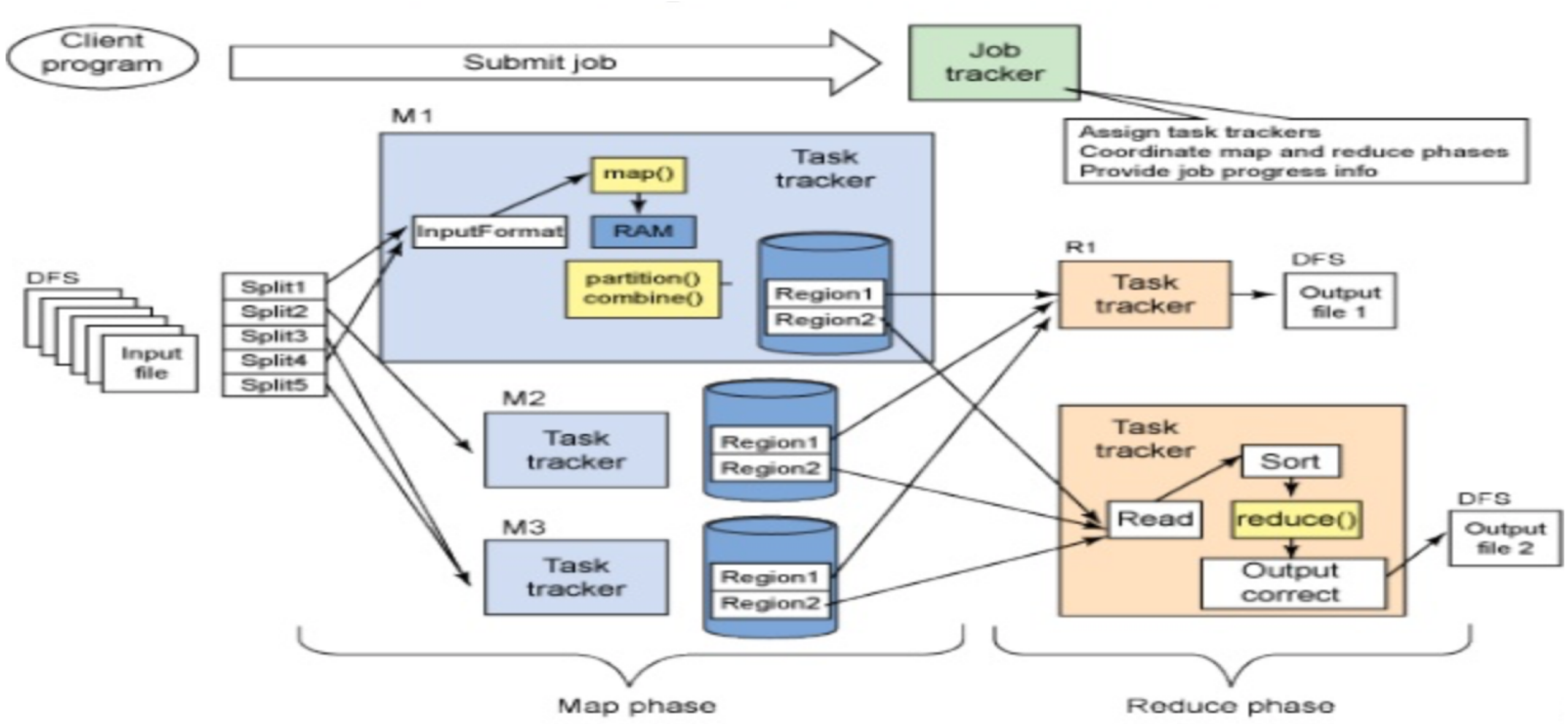
```
1: class Mapper
2:     method Map(docid a, doc d)
3:         for all term t ∈ doc d do
4:             Emit(term t, count 1)

1: class Reducer
2:     method Reduce(term t, counts [c1, c2, ...])
3:         sum ← 0
4:         for all count c ∈ counts [c1, c2, ...] do
5:             sum ← sum + c
6:         Emit(term t, count sum)
```

How MapReduce works

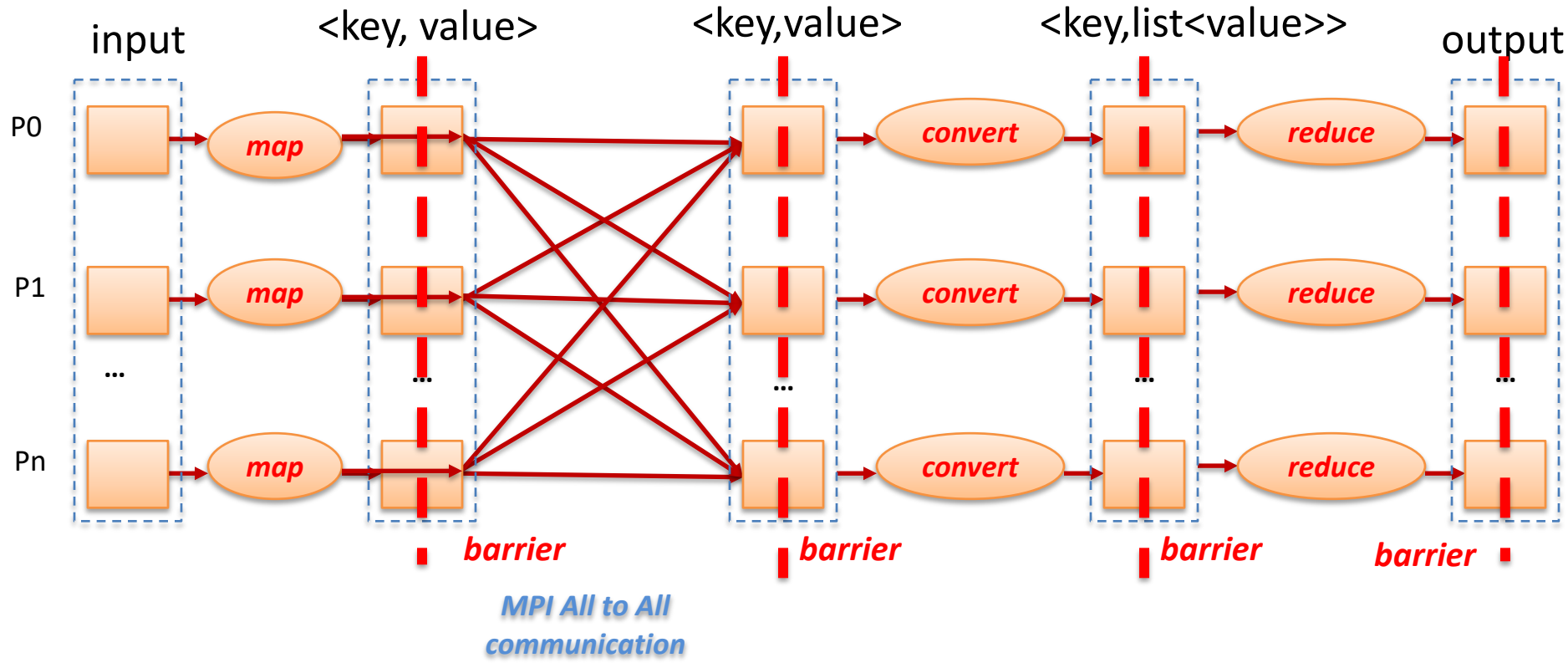


Architecture: Hadoop



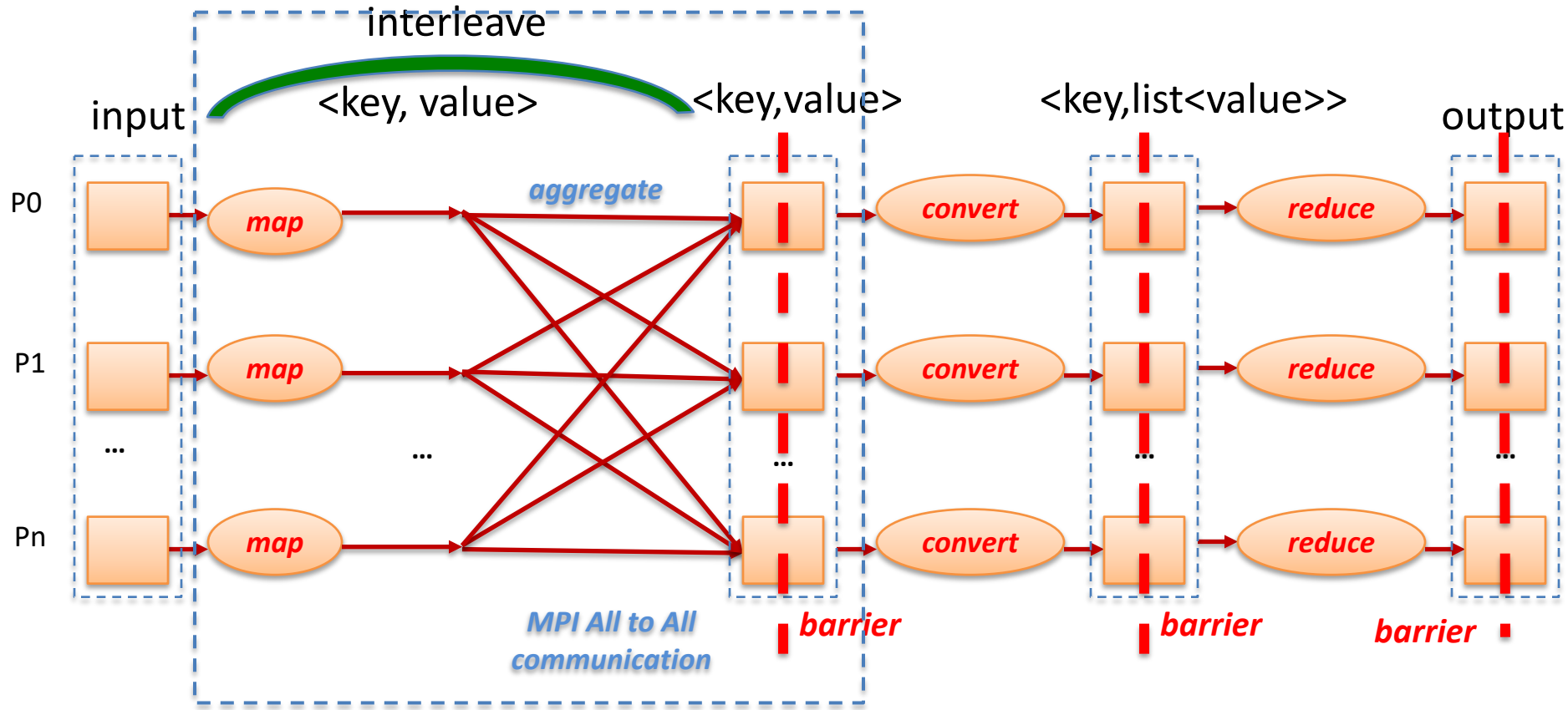
https://commons.wikimedia.org/wiki/File:Big-data-concepts-20-638.jpg

Architecture: MapReduce-MPI



MapReduce in MPI for Large-Scale Graph Algorithms, S. J. Plimpton and K. D. Devine, Parallel Computing, 37, 610-632 (2011)

Architecture: Mimir



Mimir: Memory-Efficient and Scalable MapReduce for Large Supercomputing Systems.
T. Gao, Y. Guo, B. Zhang, P. Cicotti, Y. Lu, P. Balaji, and **M. Taufer**. *IPDPS,* 2017

Building Expertise
- Assignment 2

Assignment 2

- Consolidate expertise in Github, Python, and Jupyter Notebook
- Learn to strategize on problems' solutions
  - Write a simple solution first
  - Use git commit to save significant steps
  - Use git branch for new solution efforts
  - Take performance (executing times) into account as the data analyzed grows
  - Learn that problems come with constraints and assumptions

Assignment 2 – Problem 1

(From ProjectEuler Problem 1)

If we list the natural numbers below 10 that are multiples of 3 or 5, we get: 3, 5, 6, and 9. The sum of these multiples is 23.

Write a function that finds the sum of the multiples of p or q below N.

Assumptions and Constraints (I)

- Assume that $1 \leq p < q < N$ and that each of these values are integers. Your code should be able to
  - handle values of N up to at least 100,000,000 (larger and larger data)
  - terminate in less than 1 second (constant execution time!)

Assumptions and Constraints (I)

- Things to keep in mind
  - There are two general approaches to this problem, the naïve (slower) approach and a more mathematical (faster) approach involving the inclusion-exclusion principle. To meet the performance constraints, you will have to implement the fast approach.
  - There are different approaching to measure execution times (wall-clock times). The approach below is one of them. Use what you prefer, as long as you measure wall clock time.

Assignment 2 – Problem 2

- Manipulate a list of names
  - Use the *csv module* to import a list of names
  - Score names in a list based on name “worth” and alphabetical order

Assignment 2 – Problem 2

(From ProjectEuler Problem 22.)

- The file p022_names.txt contains one line with over 5000 comma-separated names, each in all-capital letters and enclosed in quotes.
- Import the names and sort them into alphabetical order.
- Working out the alphabetical value for each name, multiply this value by its alphabetical position in the list to obtain a name score.
- Example:
  - When the list is sorted into alphabetical order, COLIN, which is worth 3 + 15 + 12 + 9 + 14 = 53, is the 938th name in the list.
  - COLIN would obtain a score of 938 * 53 = 49714.
- What is the total of the scores for all the names in the file?

Assignment 2 – Problem 3

- Find the smallest TPH number bigger than *n*
  - Triangular (T), Pentagonal (P), and Hexagonal (H) numbers are generated by given formulae
  - Write the code to find the next triangle number that is also pentagonal and hexagonal

Assignment 2 – Problem 3

- (From ProjectEuler Problem 45.)
- Triangular, Pentagonal, and Hexagonal numbers are generated by the following formulae:

Polygonal	*formula for nth term*	*sequence of terms*
Triangular	$T_n = \frac{n(n+1)}{2}$	1, 3, 6, 10, 15, ...
Pentagonal	$P_n = \frac{n(3n-1)}{2}$	1, 5, 12, 22, 35, ...
Hexagonal	$H_n = (2n - 1)\, n$	1, 6, 15, 28, 45, ...

Assignment 2 – Problem 3

- The number 1 is triangular, pentagonal, and hexagonal (TPH).
- Less obviously, $40755=T_{285}=P_{165}=H_{143}$ is also TPH
  - 40755 is the smallest TPH number bigger than 1.
- Write a function to find the smallest TPH number bigger than *n*.
- Use your function to find the smallest TPH bigger than 40755.

Things to consider

- Your choice of data structure can have a significant impact on runtime.
- Think about which operations you are doing the most and choose a data structure which minimizes the average time for this particular operation.
- Python has many built-in data structures
  - the most common data structures are lists, dictionaries, and sets, but Python also has heaps and queues.

Assignment 2 – Problem 4

From ProjectEuler Problem 87.)

- The smallest number expressible as the sum of a prime square, a prime cube, and a prime fourth power is $2^8=2^2+2^3+2^4$.
- There are exactly four numbers below 50 that can be expressed in such a way:

$$
\begin{aligned}
2^8 &= 2^2 + 2^3 + 2^4 \\
3^3 &= 3^2 + 2^3 + 2^4 \\
4^9 &= 5^2 + 2^3 + 2^4 \\
4^7 &= 2^2 + 3^3 + 2^4
\end{aligned}
$$

Assignment 2 – Problem 4

- Write code to determine the number of positive integers smaller than N that can be written as the sum of a prime square, a prime cube, and a prime fourth power.
- Your code must accept a single command line parameter
  - this time your Jupyter notebook accepts a user input N
- Your code must print a single integer
  - Example: given the input equal to 50, the output is 4

Assumptions and constraints

- For testing, you may assume that N is a positive integer and that N ≤ 50,000,000 (or larger).
- You should be able to compute the answer when N = 50,000, 000 and terminate in approximately 1 minutes.

Things to consider

- If you are having a hard time getting started, then break down the problem into smaller manageable pieces.
- Almost certainly you'll need to have a list of primes handy.
  - Can you generate a list of primes?
  - How big do your prime numbers need to be?

For the next week ….

- Get your solutions for Assignment 2 done before our next class
  - Push your solution into your own repos
- Suggested reading
  - S. Keshav **How to Read a Paper**
  - Jeffrey Dean and Sanjay Ghemawat (2004) **MapReduce: Simplified Data Processing on Large Clusters**
- Next week ….
  - More about MapReduce and its optimizations
  - More practical problems