CS 199 ACC
MapReduce

Prof. Robert J. Brunner

Ben Congdon
Tyler Kim
Bhuvan Venkatesh

MP 0

How was it?

This Week

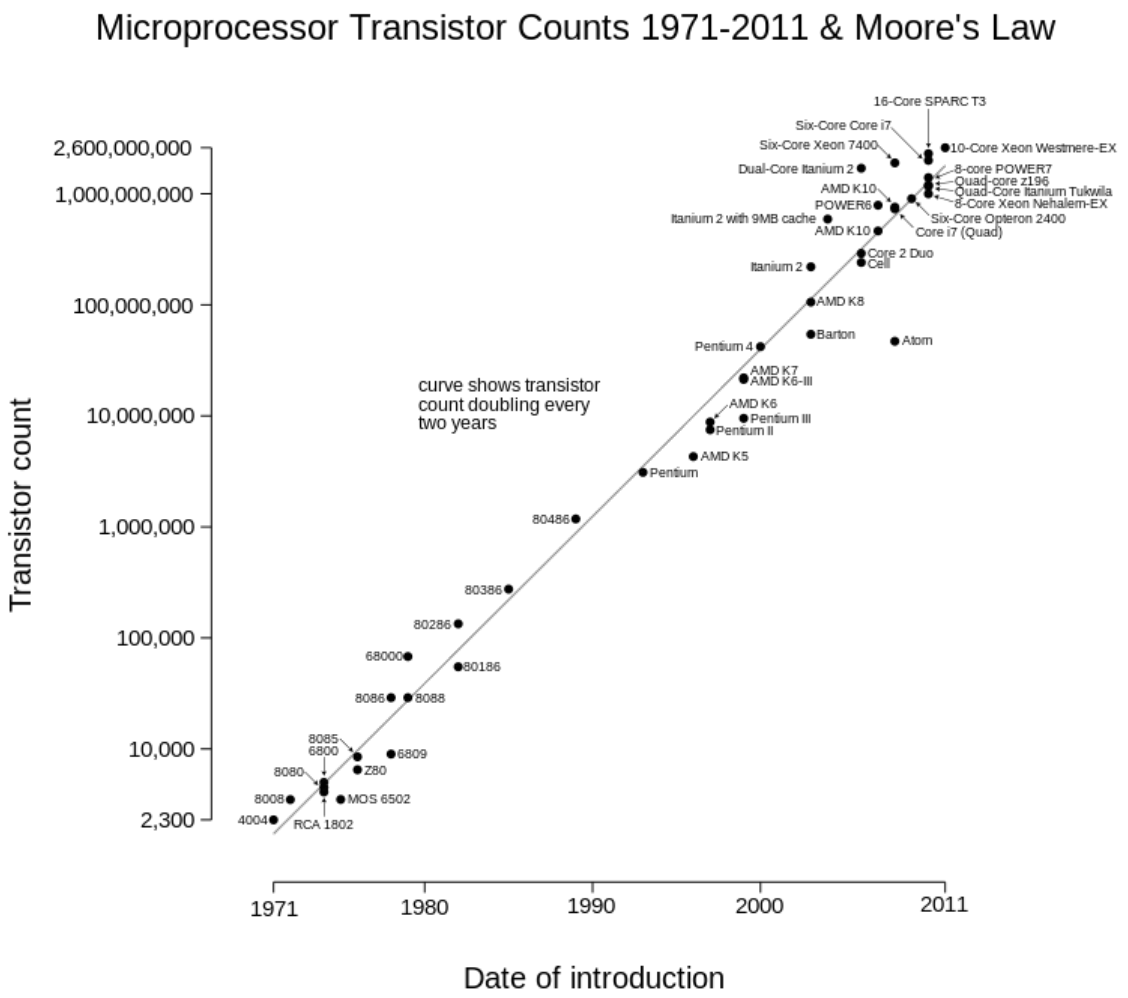
- A Vague idea for why we need MapReduce (more detail next week!)
- The MapReduce programming paradigm

What makes a computer fast?

- Processor frequency
  - Fastest commodity processor runs at 3.7 - 4.0 Ghz
- Processors are measured by how fast they can process instructions
  - Not a perfect metric
    - Simple instruction sets

Moore’s Law

- The number of transistors in a dense integrated circuit doubles approximately every two years

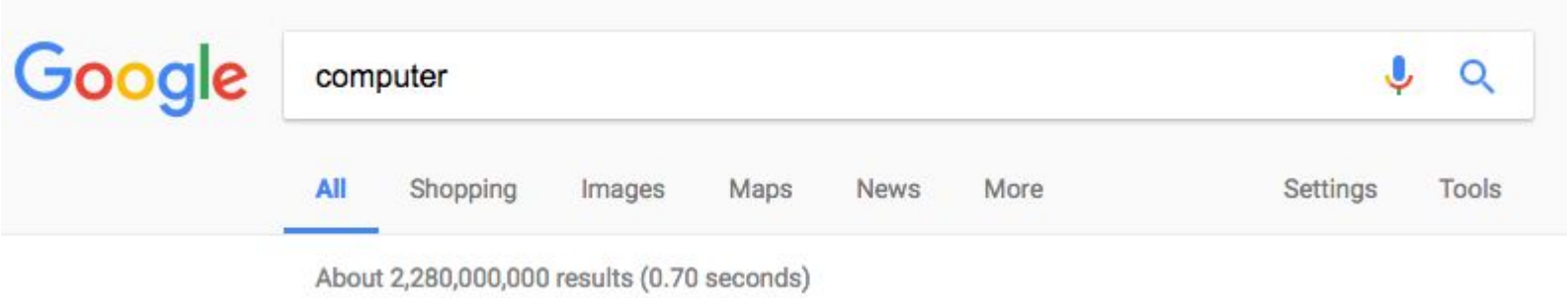


Google Search

- How fast does it take for a google search?

Google Search

- How fast does it take for a google search?



- 2.2 Billion results in less then a second
- How’d they do it?

Parallelism

If Moore’s law is slowing down how can we process more data?

- More CPU cores
- Better multithreading

Still not fast enough for Google

Distributed Systems

- What if we used more computers instead of more CPU cores?
- Let’s us process more data by just adding more computers

This Scales really really really well if done right

- This is how Google is really fast
- Also changes how we write code
    - We can no longer consider our code to only run sequentially on one computer

MapReduce

A MapReduce *job* usually splits the input data-set into independent chunks which are processed by the **map** tasks in a completely parallel manner.

The *framework* sorts the outputs of the maps, which are then input to the **reduce** tasks.

Map & Reduce

Map -- A function to process input key/value pairs to generate a set of intermediate key/value pairs. All the values corresponding to each intermediate key are grouped together and sent over to the Reduce function.

Reduce -- A function that merges all the intermediate values associated with the same intermediate key.

Code

How can we rewrite this code on multiple computers?

```
arr = range(10000000)
evens = [ ]
for i in arr:
    if i % 2 == 0:
        evens.append(i)
```

Map

How can we rewrite this code on multiple computers?

```
arr = chunks(range(10000000)) # Break arr into chunks
evens = []
index = 0
for chunk in arr:         # run each chunk on a different computer
    for i in chunk:
        if i % 2 == 0:
            evens[index].append(i)
    index += 1
# more code to recombine the lists of even numbers
```

Map

This is a common pattern that we can abstract away to something called map.

The map( ) takes a function and an array and runs the function on each element of the array

```
map(isEven, [0,1,2,3,4])
>   [True, False, True, False, True]
map(addOne, [0,1,2,3,4])
>   [1,2,3,4,5]
```

Map

```
map(isEven, [0,1,2,3,4])
>   [True, False, True, False, True]
map(addOne, [0,1,2,3,4])
>   [1,2,3,4,5]
```

- By default map only runs on one processor like normal code so there is no speedup
- But map can be rewritten to run on multiprocessors at the same time or even multiple computers
- Every map function is equivalent to a for loop

Reducing

```
reduce(lambda x, y: x+y, [1, 2, 3, 4, 5])
```

> Calculates `(((((1+2)+3)+4)+5)`

- Reduce is also a function which by default runs on one processor but can be run on multiple processors or multiple computers

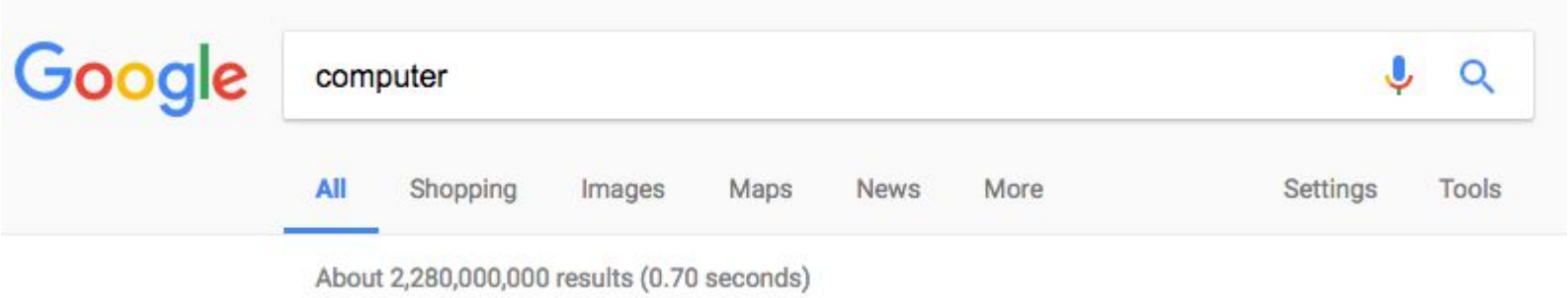
Reduce

- Map often has a partner function called reduce
- Map returns an array of results, but a lot of the time you only want one final result

```
Reduce(
    function(accumulator, currentElement),
    array
)
Results = map(isEven, [0,1,2,3,4])    ## [True, False, True, False, True]
F = lambda total, curEle: total + 1 if curEle == true else total
numEven = reduce(F, results)
numEven == 3
```

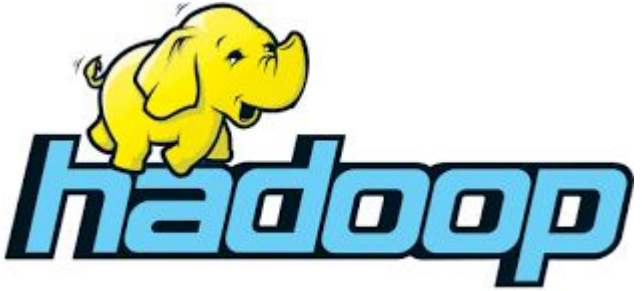
MapReduce Demo

Google Search



> 2.2 Billion results in less then a second

Next Week:



- Lets you run MapReduce on MANY computers for a single task.
- Can scales to 1000s of computers
- Processes PETABYTES (1,000,000,000,000,000) with ease
  - 1 thousand terabytes

MP 1

Due in one week (9/13) at 11:55pm

Introduces how to run MapReduce using in Python on a single machine.

> Start Early and Go to the office hours!

> Check Piazza for Q&A and Announcements

Distributing Keys for Cluster Access

Accessing NCSA Server

SSH to the cluster

- SSH Keys will be emailed to you. If you do not receive an email, send us a private post on Piazza.

- SSH from either your VM or from your local machine
- chmod 700 <your_key>
- ssh USERNAME@141.142.210.25 -i /path/to/your_key