CS 199 #1
Map Reduce

Prof. Robert J. Brunner

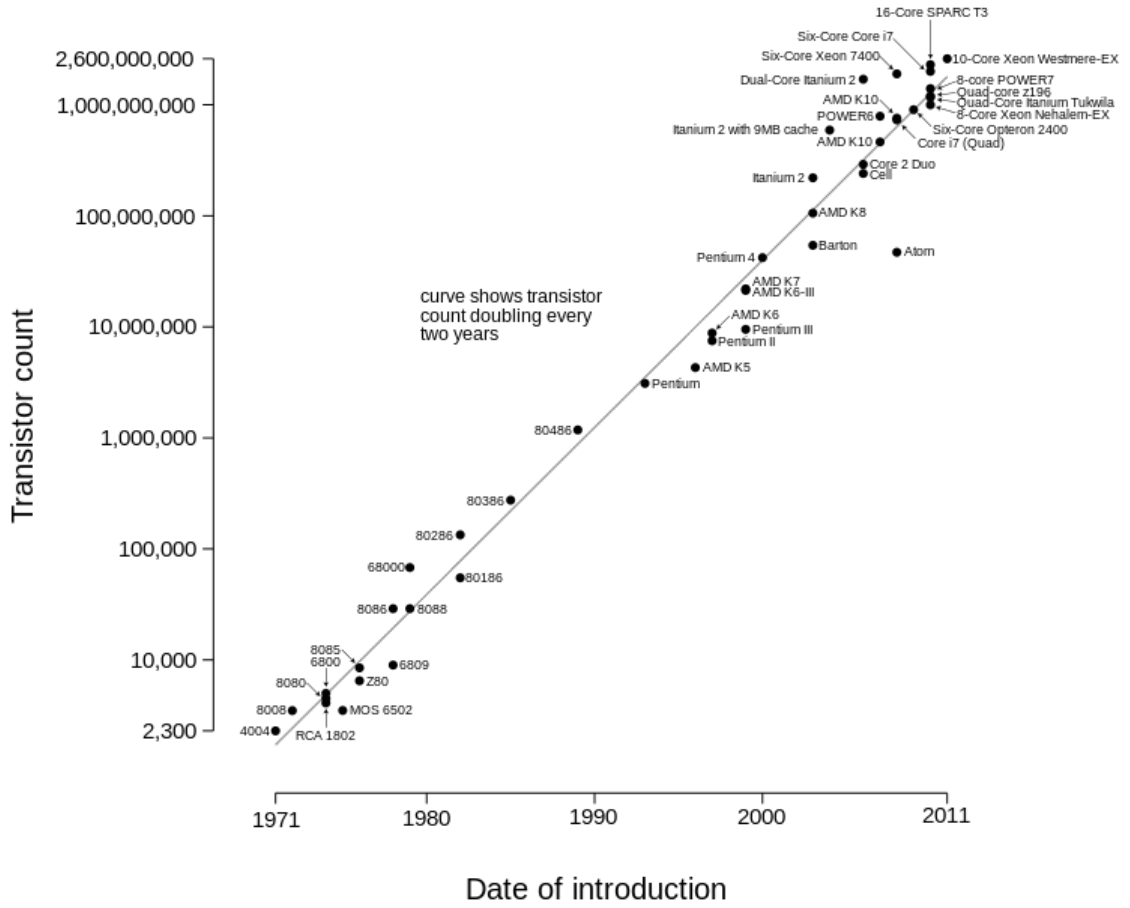
Quinn Jarrell
Tyler Kim

What makes a computer fast?

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

Moore’s Law

- Transistor count grows at an exponential rate for the same area

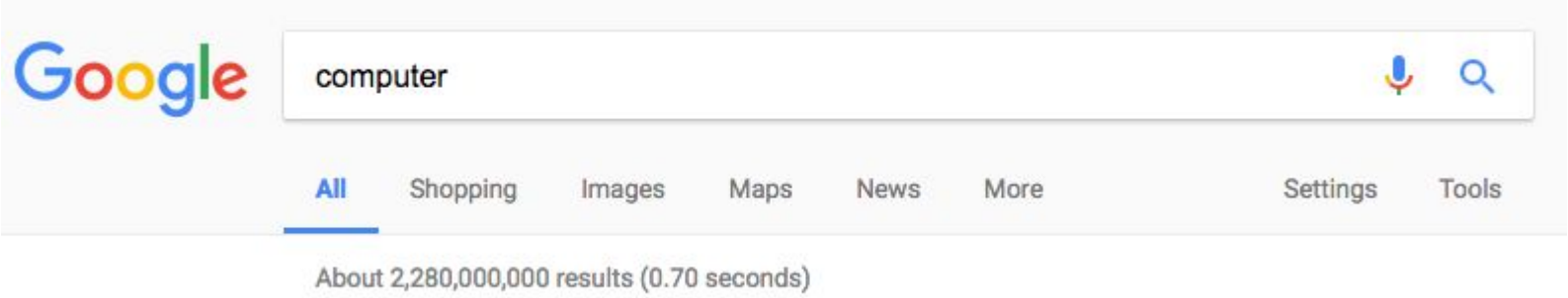


Google Search

- 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
- Scales really really really well
- This is how Google is so fast
- Changes how we write code
    - We can no longer consider our code to only run sequentially on one computer

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 function 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

- 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])
F = lambda total, curEle: total + 1 if curEle == true else total
numEven = reduce(F, results)

numEven == 3
```

More Complex Examples

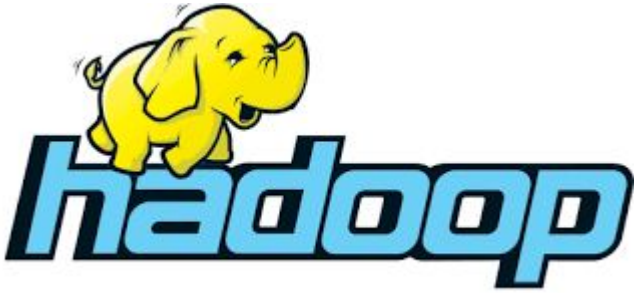
- So far our reduce has been a little too easy
- Let’s build a deduplicator

```
Data = [randints(100,100) for i in range(10)]
Def mapFunct(bunchOfInts):
        Return sum(bunchOfInts)

Def reduceFunct(uniqueSums, newSum):
        If newSum not in uniqueSums:
                uniqueSums.add(newSum)
        Return uniqueSums

reduce(reduceFunct, map(mapFunct, Data, set())) # Set is the initializer
```

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



- 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
- Covered in depth next lecture

Lab 1

- Due in one week (2/2) at 11:55pm
- Introduces how to run MapReduce using different processes
- Will be released tonight
- SSH into your VM to do it
- Submit through Moodle

Attendance:

http://bit.ly/2ndweekCS199

Shared Folders on your VM