CS 535 Large Scale Data Analysis

Amit Jain

Big Data, Big Disks, Cheap Computers

- *"In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, they didn't try to grow a larger ox. We shouldn't be trying for bigger computers, but for more systems of computers."* Rear Admiral Grace Hopper.
- *"More data usually beats better algorithms."* Anand Rajaraman.
- *"The good news is that Big Data is here. The bad news is that we are struggling to store and analyze it."* Tom White.
  - Hopefully not, with some new widely available tools!

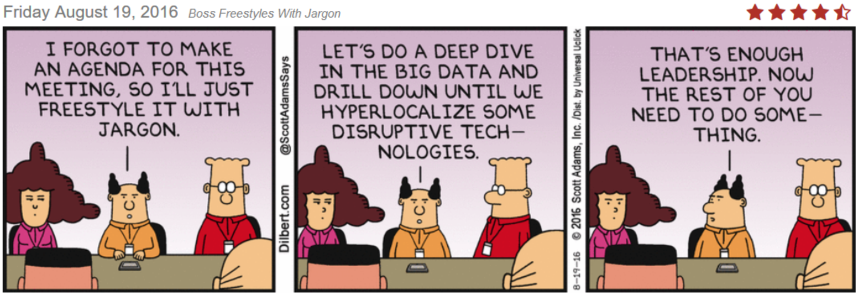
Units and Units

Unit	Value
1 KB (Kilobyte)	1024 bytes
1 MB (Megabyte)	1024 KB $\approx$ 1 million or 10^6
1 TB (Terabyte)	1024 MB $\approx$ 1 billion or 10^9
1 PB (Petabyte)	1024 TB $\approx$ 1 trillion or 10^{12}
1 EBB (Exabyte)	1024 PB $\approx$ 1 quadrillion or 10^{15}
1 ZN (Zettabyte)	1024 EB $\approx$ 1 quintillion or 10^{18}
1 YB (Yottabyte)	1024 ZB $\approx$ 1 sextillion or 10^{21}

Big Data

Big Data knows everything

Big Data


Friday August 19, 2016
Boss Freestyles With Jargon
I FORGOT TO MAKE AN AGENDA FOR THIS MEETING, SO I'LL JUST FREESTYLE IT WITH JARGON.
LET'S DO A DEEP DIVE IN THE BIG DATA AND DRILL DOWN UNTIL WE HYPERLOCALIZE SOME DISRUPTIVE TECH-NOLOGIES.
THAT'S ENOUGH LEADERSHIP. NOW THE REST OF YOU NEED TO DO SOME-THING.
@ScottAdamsSays
Dilbert.com
© 2016 Scott Adams, Inc. /Dist. by Universal Uclick
8-19-16

Big Data


DO WE HAVE ANY ACTIONABLE ANALYTICS FROM OUR BIG DATA IN THE CLOUD?
Dilbert.com DilbertCartoonist@gmail.com
YES, THE DATA SHOWS THAT MY PRODUCTIVITY PLUNGES WHENEVER YOU LEARN NEW JARGON.
1-9-13 ©2013 Scott Adams, Inc./Dist. by Universal Uclick
MAYBE IN-MEMORY COMPUTING WILL ACCEL-ERATE YOUR APPLICA-TIONS.
PLUNGE, PLUNGE, PLUNGE.

Word-count: Hello World of Big Data

Problem: Given a collection of text files, find the frequency of each word.
For example:

File1.txt	File2.txt	File3.txt
large big	data	huge
large	data data	small
big		deluge

Result:

```
large 2
big 3
data 4
small 1
deluge 1
huge 1
```

Questions: Do we want the output sorted by frequency? Sorted by word?
How would you solve this problem?

Sequential Solution

```
create empty dictionary
for f over all input files
    open file f
    while not end of file f
        read next word
        if search(word, dictionary)
            increment frequency count for word
        else
            add word to the dictionary

open output file
iterate over dictionary
    write next word to output file
```

Sequential Solution (Analysis)

- Let's say that the size of all the files together is O(n).
- We then have $O(n)$ search operations in the dictionary. The time for the search depends on how we implement the dictionary.
  - Hashtable: $O(1)$ average time
  - Height balanced search tree: $O(\lg n)$ worst-case time
- So the main loop takes $O(n)$ time on average and $O(n \lg n)$ in the worst case
- The time to output is insignificant as the size of the dictionary will be much smaller than n. Why?
- See solution in Python and Java here: wordcount

Streaming Solution

- Different approach
  - Break each file into one word per line
  - Sort all the words together
  - Count unique words and output (count, word) pairs
- Simple to do in the shell with pipes and filters. See wordcount.sh for a streaming solution in a shell script:

  ```
  cat input/* | tr ' ' '\n' | sort | uniq -c
  ```
- The data is streamed from the input files using `cat` and flows through three programs `tr`, `sort`, and `uniq` that each do some mapping and filtering (or reducing).

Analysis of Streaming Solution

- The solution:

  ```
  cat input/* | tr ' ' '\n' | sort | uniq -c
  ```

- **Input stream**: The first command `cat` outputs all the files into one stream to the next program `tr` in the pipeline. $O(n)$ time.
- **Mapping**: The `tr` command maps the words in each line into a line by itself and then streams them to the `sort` command. $O(n)$ time.
- **Sort**: The `sort` command sorts all of the data. $O(n \lg n)$ time.
- **Reduce or Filter**: After sorting, all instances of a word end up being together, which the command `uniq -c` counts and outputs. This is the reduce stage. $O(n)$ time.
- Overall runtime for the streaming solution is $O(n \lg n)$.

Large Scale Word-Count (1)

- What if the the number of files is in millions and will not fit in one server?
- What if the total size of the files is in Petabytes (or Exabytes) and will not fit in one server?
- How do you modify your solution from before? Assume that you have a cluster of n servers available with the files distributed across the servers.
- But how do we create a cluster and get the files loaded on it?

Large Scale Word-Count (2)

- What if some of the servers fail while running your program?
- What if some of the server disks fail or get corrupted while your program is running?
- What if the some system administrator reboots some of your servers for software/hardware updates without letting you know?

Frameworks/Systems for Distributed Storage and Analysis

- MapReduce: An abstract framework that helps us solve large scale data analytics problems.
- Hadoop: Provides the Hadoop Distributed File System (HDFS) and MapReduce framework on top of it. Storage and batch processing of large scale data.
- Hive and Hadoop HDFS: SQL on top of MapReduce.
- Spark : faster batch processing, streaming and real-time analysis. We can combine with Hadoop HDFS to get storage, manipulation, and analysis of large scale data!
- Storm and Heron: large real-time data flow.
- and more...

Data Scientist versus Data Engineer

Data Science tasks:

- Goal is to answer a question or discovering insights.
- Often uses interactive shells for ad-hoc analysis
- Typically uses Python, R, Matlab, and Spark

Data engineer tasks:

- Builds and maintains a production application (that may use hardened versions of the original data science work)
- Use principles of software engineering like encapsulation, object-oriented design and interface design
- They have to deal with parallelization, complexity of distributed systems, fault tolerance etc
- Typical languages would be Java (with Hadoop and/or Spark), Scala (with Spark), Python (with Spark) (at smaller scales)

Insights from Big Data

The point of large scale data analysis is meaningful insight!

We should consider two things about insights presented by analysis:

- Investigate carefully to see if it uses a significant amount of data.
- Think about each of the insights and label them Actionable, Useless (trivia), or potentially Misleading or dangerous.

For example:

`https://blogs.scientificamerican.com/guest-blog/9-bizarre-and-surprising-insights-from-data-science/`