Today

- MapReduce
  - Description
  - Examples
- Social implications of Computing
  - Therac-25 case study

Recap: Reduce

<table>
<thead>
<tr>
<th>reduce(lambda so_far, elt: so_far * elt, [1, 3, 8, 7], 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(((2 * 1) * 3) * 8) * 7</td>
</tr>
</tbody>
</table>

Map and Reduce

Many problems can be solved by mapping a function over a sequence, and then reducing the new items.

To find the sum of squares of items in a list:

```python
from functools import reduce
from operator import add

def sum_squares(list):
    return reduce(add, map(lambda num: num ** 2, list), 0)
```

1. Square each item in a list...

Map and Reduce: Example

Count the number of letters in the words of a list

```python
def count_letters(wordlist):
    return reduce(add, map(lambda word: len(word), wordlist), 0)
```

1. Find the length of each word in a list...
**Map and Reduce: Example**

*Count the number of words in a list*

```python
def count_words(wordlist):
    return reduce(____, 
                  map(______________, 
                       wordlist), 
                  0)
```

**Map and Reduce: Example**

*Count the number of words in a list*

```python
def count_words(wordlist):
    return reduce(add, 
                  map(lambda word: 1, 
                       wordlist), 
                  0)
```

**Map and Reduce: Example**

*Count the number of words in a list of even length*

```python
def count_words(wordlist):
    return reduce(____, 
                  map(____________________, 
                       wordlist), 
                  0)
```

**Map and Reduce: Example**

*Count the number of words in a list of even length*

```python
def count_words(wordlist):
    return reduce(add, 
                  map(lambda word: 1 if len(word)%2 == 0 else 0, 
                       wordlist), 
                  0)
```

**Map and Reduce: Example**

*Count the number of items in a list that satisfy a given predicate*

```python
def count_pred(pred, list):
    return reduce(____, 
                  map(____________________, 
                       list), 
                  0)
```

**Map and Reduce: Example**

*Count the number of items in a list that satisfy a given predicate*

```python
def count_pred(pred, list):
    return reduce(add, 
                  map(lambda item: 1 if pred(item) else 0, 
                       list), 
                  0)
MAP AND REDUCE: EXAMPLE

Filter out the items in a list that satisfy a given predicate

```python
def filter(pred, list):
    return \
    reduce(lambda so_far, sublist: so_far + sublist,
           map(lambda item: [item] if pred(item) \n                else [],
                 list),
           [])
```

ANNOUNCEMENTS

- You are now done with all projects and homework assignments! Congratulations! 😊
- Final lecture: You choose!
  Piazza poll will be sent out today asking for suggestions for tomorrow’s lecture.

ANNOUNCEMENTS: FINAL

- Final is Thursday, August 9.
  - Where? 1 Pimentel.
  - When? 6PM to 9PM.
  - How much? All of the material in the course, from June 18 to August 8, will be tested.
- Closed book and closed electronic devices.
- One 8.5” x 11” ‘cheat sheet’ allowed.
- No group portion.
- Post-final laser tag?

HOW MUCH DATA?

As of 2007, Google processes **20 petabytes** of data per day. Now, they probably process thousands of petabytes per day.

*How much is a petabyte?*
HOW MUCH DATA?

Assume that every 0 or 1 in a file is as large as a pixel on a screen (approx. 100 $\mu$m$^2$).

Then, a homework file, of size (approximately) 5 kilobytes, is about as big as

HOW MUCH DATA?

A music file has (approximately) 6 megabytes.
(1 megabyte = 1024 kilobytes)

HOW MUCH DATA?

One DVD-R has (approximately) 5 gigabytes.
(1 gigabyte = 1024 megabytes)

This is about as big as the total area of all pages of the paper copy of a homework.

HOW MUCH DATA?

One terabyte is 1024 gigabytes.
One petabyte is 1024 terabytes.

This is as much information as would fit in:
6 billion photos on Facebook, or around 13 years of HD video.

PARALLELISM

Idea: Have a lot of computers working at once on different chunks of the same data, independently of other chunks and computers.

One petabyte is an extremely large amount of data.
We cannot process this on one computer alone.

My computer would take around 7 years to process all of 20 petabytes exactly once, and that assumes that all it ever does is process data and never fails.

Companies like Google do not have this luxury: they need data to be processed as quickly as possible.
**MapReduce**

Framework for batch processing of Big Data

- All data is available at the outset; results are not consumed until processing completes.
- System used by programmers to build applications.
- Very large and complex data sets.

**MapReduce: History**

- Developed at Google in 2003.
- Programmers had to consider a lot of details as part of their code: what to do when a computer fails, and how to divide the work up among different computers.
- However, many operations fit neatly into a map step, followed by a reduce step.

**MapReduce**

The MapReduce framework is an abstraction: it allows programmers to code, while taking care of the underlying messy details, such as dealing with failure and distributing work across computers.

All the programmers had to do was create:
- a mapper (the function to map), and
- a reducer (the function to combine mapped results).

**MapReduce: Execution Model**

- Map Phase: Apply a mapper over inputs, and emit intermediate key-value pairs.
- Sort Phase: Group the intermediate key-value pairs by keys.
- Reduce Phase: For each intermediate key, apply a reducer to “accumulate” all values associated with that key.

**Running example**

Count all the vowels in Shakespeare’s works.

We need a mapper and a reducer.

**Idea:**

The mapper will count the number of vowels in each line, for each vowel. The reducer will take the counts for each vowel from each line and total all the counts.
**MAPREDUCE: Mapper**

```python
import sys
from ucb import main
from mr import emit

def emit_vowels(line):
    for vowel in 'aeiou':
        count = line.count(vowel)
        if count > 0:
            emit(vowel, count)

@main
def run():
    for line in sys.stdin:
        emit_vowels(line)
```

1. For every line in a file, fed through the standard input of this program...
2. Check if the line has a vowel.
3. If it does, "emit" – or send to the standard output – a key-value pair that associates the vowel with its count.

**MAPREDUCE: Reducer**

```python
import sys
from ucb import main
from mr import emit, values_by_key

@main
def run():
    for key, value_iterator in values_by_key(sys.stdin):
        emit(key, sum(value_iterator))
```

1. Iterate over the keys and associated values, fed through the standard input...
2. ... and "emit" each key to the standard output, followed by the sum of the associated values.

**Standard Input and Output**

The **standard input** is where the program receives its input from external sources, such as the user or a file.

The **standard output** is where the program will send its output, such as the screen or another file.

**MAPREDUCE: Practice**

What is a mapper and reducer that we can use to count the number of times particular words appear in a file?

**Mapper:** For every line in a file, emit...

**Reducer:** For every key,
**MAPREDUCE: PRACTICE**

What is a mapper and reducer that we can use to count the number of times particular words appear in a file?

**Mapper:** For every line in a file, emit a key-value pair that associates a “special” word with the number of times it appears in that line.

**Reducer:** For every key, sum up all the values associated with that key.

What is a mapper and reducer that we can use to count the number of lines in a file?

**Mapper:** For every line in a file, emit...  

**Reducer:** For every key, sum up all the values associated with that key.

---

**MAPREDUCE: PRACTICE**

What does the framework provide us?

- **Fault Tolerance:** A machine or hard drive might crash. The framework will automatically re-run failed tasks.
- **Speed:** Some machine might be slow because it is overloaded. The framework can run multiple copies and keep the result of the one that finishes first.

---

**MAPREDUCE**

What does the framework provide us?

- **Network locality:** Data transfer is expensive. The framework tries to schedule map tasks on the machines that hold the data to be processed.
- **Monitoring:** When will my job finish? The framework provides a web-based interface describing jobs.
**THERAC-25**

- Built by AECL (Atomic Energy of Canada Limited)
- Dual-mode machine:
  - Treat using low-energy electron beams.
  - Treat using high-energy X-ray beams.
- Turntable with items for each mode placed on it: rotates to the correct position before a beam is started up.

**What happened?**

- High-power beam activated instead of the low-power beam.
- High-power beam had approximately 100 times the intended dose of radiation.
- Three deaths from radiation poisoning.

**Why?** Software bugs and user interface issues.

- A one-byte counter frequently “overflowed”, or got too big for the memory allocated to it. If an operator provided manual input at the same time as the overflow, the interlock would fail.
- Failure only occurred during a nonstandard sequence of keystrokes: this went unnoticed for a long time.

**Why?** Software bugs and user interface issues.

- Therac-25 did *not* have the hardware interlocks that its predecessors had, to prevent the wrong beam from activating.
- The developers relied on software interlocks, which were incorrectly coded.
- The code reused software from older models, where the hardware interlocks could not report that they were triggered.
THERAC-25

Why? Software bugs and user interface issues.
- The combination of hardware and software was never tested until at the hospital.
- If the system noticed that something was wrong, it would halt the X-ray beam and display the word “MALFUNCTION”, followed by an unhelpful error code.
  The operator could override the warning and proceed anyway.
- Complaints were not believed initially.

THERAC-25

What can we learn?
- Computers are everywhere: this means that the code you write can potentially be used in a life-or-death situation.
- In a complex codebase, software bugs can be really hard to catch and fix.
- This can be ameliorated significantly with proper design and frequent testing.

CONCLUSION
- MapReduce is a framework that allows programmers to process large amounts of data in parallel, without having to concern themselves with details such as computer failure and task assignment.
- Proper design and test of software is becoming increasingly essential, especially in a world where computing is ubiquitous.