Announcements

- Project 4 is due tomorrow (8/5)
- Final Review tomorrow (8/5) from 11-12:30pm in 2050 VLSB
- Final Exam on Friday (8/12) from 5-8pm in 155 Dwinelle
- Ants composition revisions due Saturday (8/6)
- Scheme Recursive Art Contest is open! Submissions due 8/9
- Potluck II on 8/10! 5-8pm (or later) in Wozniak Lounge
  - Bring food and board games!
- Homework 10 will be due 8/9
- Homework 11 and 12 will be due 8/10 and 8/12
  - Last two of the three extra credit surveys

Roadmap

- Introduction
- Functions
- Data
- Mutability
- Objects
- Interpretation
- Paradigms
- Applications

This week (Paradigms), the goals are:
- To study examples of paradigms that are very different from what we have seen so far
- To expand our definition of what counts as programming

Big Data

Examples of Big Data

- There's a lot of data out there!
  - Facebook's daily logs: 60 Terabytes (60,000 Gigabytes)
  - 1,000 genomes project: 200 Terabytes
  - Google web index: 10+ Petabytes (10,000,000 Gigabytes!!)
- These datasets are too large to fit on a single computer
- Reading 1 Terabyte from disk: 3 hours (100 MB per second)

Examples from Anthony Joseph
Distributed Algorithms

• If data can't be stored on a single machine, then our programs can't run on a single machine
• Therefore, we need to develop distributed algorithms to distribute and coordinate work between worker machines
• Machines can communicate, but perform computations in their own isolated environment

Computers for Big Data

• Typical hardware for big data applications:
  • Consumer-grade hard disks and processors
  • Independent computers are stored in racks
  • Concerns: heat, power, monitoring, networking
  • When using many computers, some will fail!

Apache Spark

• Apache Spark is a data processing system that provides a simple interface for large data
  • Developed right here at Berkeley in 2010!
  • A Resilient Distributed Dataset (RDD) is a collection of values or key-value pairs
  • Supports common sequence operations: map, filter, reduce
  • These operations can be performed on RDDs that are partitioned across machines
  • Idea: Working with distributed data is complicated. Use abstraction to hide the fact that the data is distributed!

Apache Spark Execution Model

• An RDD is distributed in partitions to worker nodes
  • A driver program defines transformations and actions
  • Transformations: Create a new RDD from an existing RDD
  • Actions: Summarize RDD into one value (e.g. sum, take)
  • A cluster manager assigns tasks to individual worker nodes to carry them out
  • Worker nodes perform computation and communicate values to each other
  • Final results are communicated back to the driver program
The Last Words of Shakespeare

- A driver program defines transformations and actions
- A cluster manager assigns tasks to individual worker nodes
- Worker nodes perform computation and communicate values to each other

```
>> sc
>>> sc.contextSparkContext ...
>>> shakes = sc.textFile('shakespeare.txt')
>>> shakes.sortBy(lambda line: line, False)
... .take(2)
['you shall...', 'yet, a...']
```

```
What Does Apache Spark Provide?

- Fault tolerance: A machine or hard drive might crash
  * The cluster manager automatically re-runs failed tasks
- Speed: Some machine might be slow because it's overloaded
  * The cluster manager can run multiple copies of a task
    and keep the result of the one that finishes first
- Monitoring: Will my job finish before dinner??
  * The cluster manager provides a web-based interface
    describing jobs
- Abstraction!

MapReduce

```
MapReduce Applications

- An important early distributed processing system was
  MapReduce, published by Google in 2004
- Simple structure that happened to capture many common data
  processing tasks
  * Step 1: Each element in an input collection produces
    zero or more key-value pairs (map)
  * Step 2: All key-value pairs that share a key are
    aggregated together (shuffle)
  * Step 3: All the values for a key are processed as a
    sequence (reduce)
  * Early applications: indexing web pages, computing PageRank
```

```
MapReduce Evaluation Model

- Map step: Apply a mapper function to all inputs, emitting
  intermediate key-value pairs
- Reduce step: For each intermediate key, apply a reducer
  function to accumulate all values associated with that key
  * All key-value pairs with the same key are processed
    together
```

```
Google MapReduce
Is a Big Data Framework
For batch processing
mapper
```

```
```
MapReduce Evaluation Model

- Reduce step: For each intermediate key, apply a reducer function to accumulate all values associated with that key.
- All key-value pairs with the same key are processed together.

Google MapReduce
Is a Big Data framework
For batch processing

Mapper
Reducer

Summary

- Some problems are too big for one computer to solve!
- However, distributed programming comes with its own issues
- We can use abstractions (such as Apache Spark) to manage some of the complexity that is inevitable when running programs on many machines.

MapReduce on Apache Spark (demo)

Key-value pairs are just two-element Python tuples

<table>
<thead>
<tr>
<th>Call Expression</th>
<th>Data</th>
<th>fn Input</th>
<th>fn Output</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>data.flatMap(fn)</td>
<td>Values</td>
<td>One value</td>
<td>Zero or more key-value pairs</td>
<td>All key-value pairs returned by calls to fn</td>
</tr>
<tr>
<td>data.reduceByKey(fn)</td>
<td>Key-value pairs</td>
<td>Two values</td>
<td>One value</td>
<td>One key-value pair for each unique key</td>
</tr>
</tbody>
</table>