Announcements

• Recursive art contest entries due Monday 12/2 @ 11:59pm
• Guerrilla section about logic programming on Monday 12/2 1pm–3:30pm in 273 Soda
• Homework 11 due Thursday 12/5 @ 11:59pm
• No video of lecture on Friday 12/6
  • Come to class and take the final survey
  • There will be a screencast of live lecture (as always)
• Screencasts: http://www.youtube.com/view_play_list?p=-XXv-cvA_iCIEwJhyDVdyLMCiimv6Tup
Unix
Systems research enables the development of applications by defining and implementing abstractions:

- **Operating systems** provide a stable, consistent interface to unreliable, inconsistent hardware.

- **Networks** provide a simple, robust data transfer interface to constantly evolving communications infrastructure.

- **Databases** provide a declarative interface to software that stores and retrieves information efficiently.

- **Distributed systems** provide a unified interface to a cluster of multiple machines.

A unifying property of effective systems:

> Hide *complexity*, but retain *flexibility*
The Unix Operating System

Essential features of the Unix operating system (and variants):

• **Portability**: The same operating system on different hardware.
• **Multi-Tasking**: Many processes run concurrently on a machine.
• **Plain Text**: Data is stored and shared in text format.
• **Modularity**: Small tools are composed flexibly via pipes.

“We should have some ways of coupling programs like [a] garden hose – screw in another segment when it becomes necessary to massage data in another way,” Doug McIlroy in 1964.

The **standard streams** in a Unix-like operating system are similar to Python iterators.

(Demo)

```
ls *.py | cut -f 1 -d '.' | grep hw | cut -c 3- | sort -n
```
The built-in `input` function reads a line from *standard input*. The built-in `print` function writes a line to *standard output*.

(Demo)

The values `sys.stdin` and `sys.stdout` also provide access to the Unix *standard streams* as files.

A Python file is an interface that supports iteration, read, and write methods.

Using these "files" takes advantage of the operating system *standard stream* abstraction.

(Demo)
MapReduce
MapReduce is a *framework* for batch processing of Big Data.

- **Framework**: A system used by programmers to build applications.
- **Batch processing**: All the data is available at the outset, and results aren't used until processing completes.
- **Big Data**: Used to describe data sets so large that they can reveal new facts about the world, usually from statistical analysis.

The MapReduce idea:

- Data sets are too big to be analyzed by one machine.
- Using multiple machines has the same complications, regardless of the application.
- Pure functions enable an abstraction barrier between data processing logic and coordinating a distributed application.

(Demo)
**MapReduce Evaluation Model**

**Map phase:** Apply a *mapper* function to inputs, emitting *intermediate key-value pairs*.
- The *mapper* takes an iterator over inputs, such as text lines.
- The *mapper* yields zero or more *key-value pairs* per input.

**Reduce phase:** For each *intermediate key*, apply a *reducer* function to accumulate all values associated with that key.
- The *reducer* takes an iterator over *key-value pairs*.
- All pairs with a given key are consecutive.
- The *reducer* yields 0 or more values, each associated with that *intermediate key*.
MapReduce Evaluation Model

Google MapReduce
Is a Big Data framework
For batch processing

Reduce phase: For each intermediate key, apply a reducer function to accumulate all values associated with that key.

- The reducer takes an iterator over key-value pairs.
- All pairs with a given key are consecutive.
- The reducer yields 0 or more values, each associated with that intermediate key.
MapReduce Execution Model
Execution Model

A "task" is a Unix process running on a machine.
MapReduce Assumptions

**Constraints** on the *mapper* and *reducer*:

- The *mapper* must be equivalent to applying a deterministic pure function to each input independently.
- The *reducer* must be equivalent to applying a deterministic pure function to the sequence of values for each key.

**Benefits** of functional programming:

- When a program contains only pure functions, call expressions can be evaluated in any order, lazily, and in parallel.
- *Referential transparency*: a call expression can be replaced by its value (or *vice versa*) without changing the program.

In MapReduce, these functional programming ideas allow:

- Consistent results, however computation is partitioned.
- Re-computation and caching of results, as needed.
MapReduce Applications
Python Example of a MapReduce Application

The **mapper** and **reducer** are both self-contained Python programs.

*Read from *standard input* and write to *standard output*!

### Mapper

```python
#!/usr/bin/env python3
import sys
from mr import emit

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

for line in sys.stdin:
    emit_vowels(line)
```

- **Tell Unix**: This is Python 3 code
- **The emit function outputs a key and value as a line of text to standard output**
- **Mapper inputs are lines of text provided to standard input**
Python Example of a MapReduce Application

The mapper and reducer are both self-contained Python programs.

Read from standard input and write to standard output!

Reducer

```python
#!/usr/bin/env python3
import sys
from mr import emit, values_by_key

for key, value_iterator in values_by_key(sys.stdin):
    emit(key, sum(value_iterator))
```

Input: lines of text representing key-value pairs, grouped by key
Output: Iterator over (key, value_iterator) pairs that give all values for each key
MapReduce Benefits
What Does the MapReduce Framework Provide

**Fault tolerance:** A machine or hard drive might crash.
• The MapReduce framework automatically re-runs failed tasks.

**Speed:** Some machine might be slow because it's overloaded.
• The framework can run multiple copies of a task and keep the result of the one that finishes first.

**Network locality:** Data transfer is expensive.
• The framework tries to schedule map tasks on the machines that hold the data to be processed.

**Monitoring:** Will my job finish before dinner?!?
• The framework provides a web-based interface describing jobs.

(Demo)