Recitation 15: MapReduce
Plan

* Setting
* Design
* MapReduce it?
* Failures

Logistics

* Schedule your proj presentation!
* Design project updates released

What did you notice about this paper?
Notes on terminology:

MapReduce paper uses "master" for server that coordinates workers.

→ We will use "main" instead (Katrina prefers "coordinator").

→ Most open-source projects & companies have deprecated use of "master" and we will follow that convention.
The Setting

* Vast quantities of data = 2^{60} bytes
* Thousands of machines
* Used for lots of things.

Always new applications.

Problem: When your new employee shows up, how can you give them access to this data set?

⇒ When you have 2^{60} bytes of data, even simple tasks are difficult.
Idea: Give programmer a simple way to interact with the data.

The simple API is really the cleverness in this paper (IMO).

It’s just this:

\[
\text{map}(\text{key}_1, \text{val}_1) \rightarrow [(\text{key}_2, \text{val}_2), \ldots] \\
\text{reduce}(\text{key}_2, [\text{val}_2, \ldots]) \rightarrow \text{val}_2
\]

User (application developer) doesn’t worry about:

* where code runs in data center
* fault tolerance
* storing intermediate results
* stragglers
* locality
* resource consumption??
Example: Page popularity

For each URL $u$, how many pages link to $u$?

- Used in first versions of Google search

\[
\text{map}(\text{page name}, \text{page.html}) \rightarrow [(\text{cat.com}, 1), (\text{dog.com}, 1), (\text{veg.com}, 1)]
\]

- For each page, output URLs of all outgoing links on that page

\[
\text{reduce}(\text{url}, (1, 1, 2, \ldots, 1)) \rightarrow 134
\]

- Sum up the # of incoming links.

\[
(\text{dog.com}, (1, 1, 2, \ldots, 1)) \rightarrow 134
\]
Poll: MapReduce it?

* You have a copy of 2 so web pages.
  You want to find all pages written in Spanish.
  → Yes, definitely makes sense.

* You have 2²₀ images of dogs and you need to resize them all to 50%.
  → Data probably too small.
  → Might as well run on your laptop.

* You have 2²₀ labeled images of dogs & want to train on ML classifier on them.
  → Probably depends on your model.
  → Most don’t parallelize super well.

* You have a map of all roads in the U.S. and you want to find the shortest path from every city to every other city.
  → Seems messy?... problem: global computation
  → But if data is big...

* Run the web server that hosts nytimes.com.
  → Really only good for batch jobs

→ Manipulating state is also problematic (e.g. Amazon Warehouse)
How to implement MapReduce?

- As in GFS, a single main server.

Tricks:
- Minimize network use.
- Schedule compute where the data is.
Failures

What happens if worker disappears?
- Rerun? Only lose a chunk of work
- Same trick can handle slow workers

What happens if a worker gives corrupt output?
- You're on your own...

What happens if main server fails?
- Didn't cover this in eval.
- Example failure in room... everyone picks at one main versus everyone

Pr[main fails] = $f$

Pr[at least 1 server fails] = $1 - (1 - f)^n \approx 1 - e^{-nf}$

[Assume independent failures...]

Probability of a failure goes up exponentially with the # of servers?

Failures are the common case.

⇒ Good life lesson 😊
Closing thought:

- Central challenge of systems is exposing the power of computer to the application programmer (e.g. UNIX)

- Choosing the right interface is key
  - Ease of use
  - Flexibility in implementation
  - Generality

MapReduce hits a really nice sweet spot in the design space.