View Source: Design Patterns in the Wild

Map/Reduce

Ashwin J Mathew      Spring 2008
School of Information, UC Berkeley
Today

- Functional programming fun with Map/Reduce
- Map/Reduce as a distributed programming pattern, implemented in Apache Hadoop
The Problem

- **Reduce algorithm implementation complexity**
  - Apply a function to every item of a collection
  - Reduce a collection to a single value, or smaller set of values

- **Duplicated iteration code**
  - Potentially inefficient
  - Error-prone: index variables are globals in Javascript
  - Hard to refactor nested loops
Why Map/Reduce?

\[
\begin{align*}
\text{for } (i=0; i<a.length; i++) & \quad a[i] = a[i] \ ^\ 2; \\
\text{for } (i=0; i<a.length; i++) & \quad \text{sum } += a[i];
\end{align*}
\]

\[
\begin{align*}
a & = a\text{.map(function}(x) \{ \\
& \quad \text{return } x^\ 2\ \}); \\
\text{sum } & = a\text{.reduce(} \\
& \quad \text{function}(x,y) \{ \\
& \quad \quad \text{return } x+y\});
\end{align*}
\]

**Hide your loops!**
Potential

By abstracting away the very concept of looping, you can implement looping any way you want, including implementing it in a way that scales nicely with extra hardware.

- Joel Spolsky
Distributed Map/Reduce

From MapReduce: Simplified Data Processing on Large Clusters by Dean and Ghemawat
Introducing Apache Hadoop

- Originally part of the Apache Lucene search engine project
- Distributed Map/Reduce engine
- Used for distributed indexing by Apache Nutch web search engine
Benefits + Considerations

- Simpler, smaller map/reduce job implementations
- More complex, distributed map/reduce engine
- Scales based on hardware and map/reduce engine implementation
- Limited network capacity -> Locality of data
- Anticipate slow and failed workers
Yahoo’s Hadoop Clusters

- We have ~10,000 machines running Hadoop
- Our largest cluster is currently 2000 nodes
- 1 petabyte of user data (compressed, unreplicated)
- We run roughly 10,000 research jobs / week