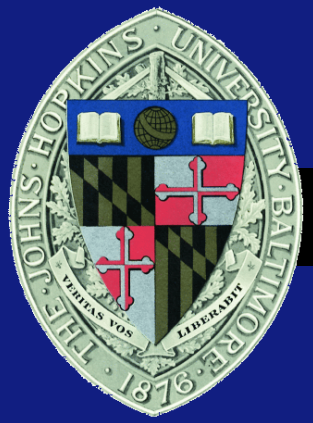
Lecture 14.1
Google's Map/Reduce

EN 600.320/420

Instructor: Randal Burns

27 March 2017



Department of Computer Science, *Johns Hopkins University*

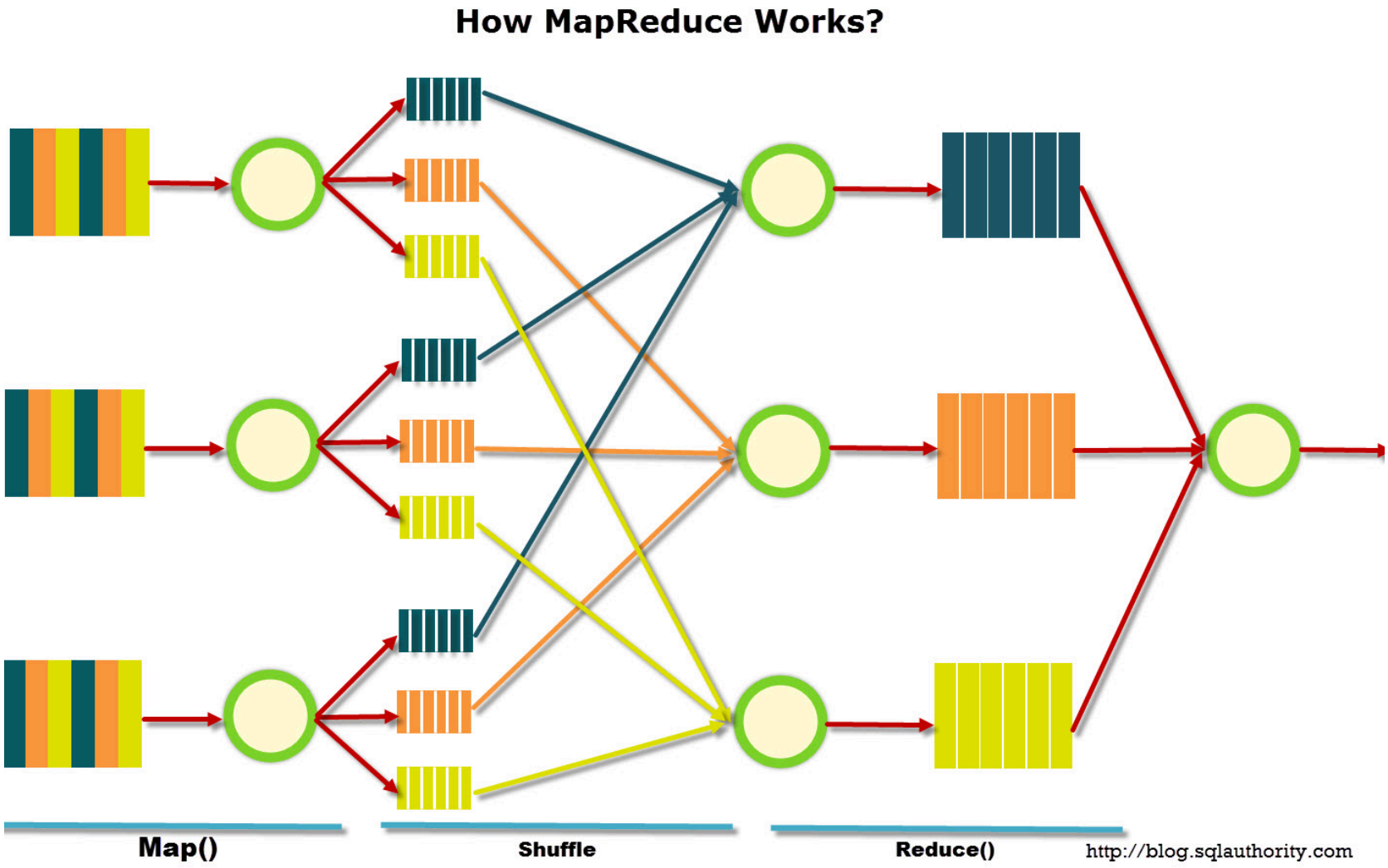
Overview

- Map/Reduce
  - A structured programming model for data parallelism
  - Geared toward data intensive applications
- The Google File System
  - Large scale parallel file system for low $$
  - Failures are the normal case
  - M/R requires such a global data service
- Open-source re-implementation
  - HADOOP: Map/Reduce
  - HDFS: Google File System
- What is it?
  - Data-parallel, distributed-memory, data-intensive

Map/Reduce Model

- Map
  - Process key/value pairs
  - Produce intermediate key/value pairs
- Reduce
  - Merge intermediate pairs on key value
- Map and reduce were LISP primitives
  - Hence the functional style
  - Actually write procedural code under functional constraints

An M/R Schematic



Example: Count all Words

- This is pseudocode
  - Need a tokenizer, etc.

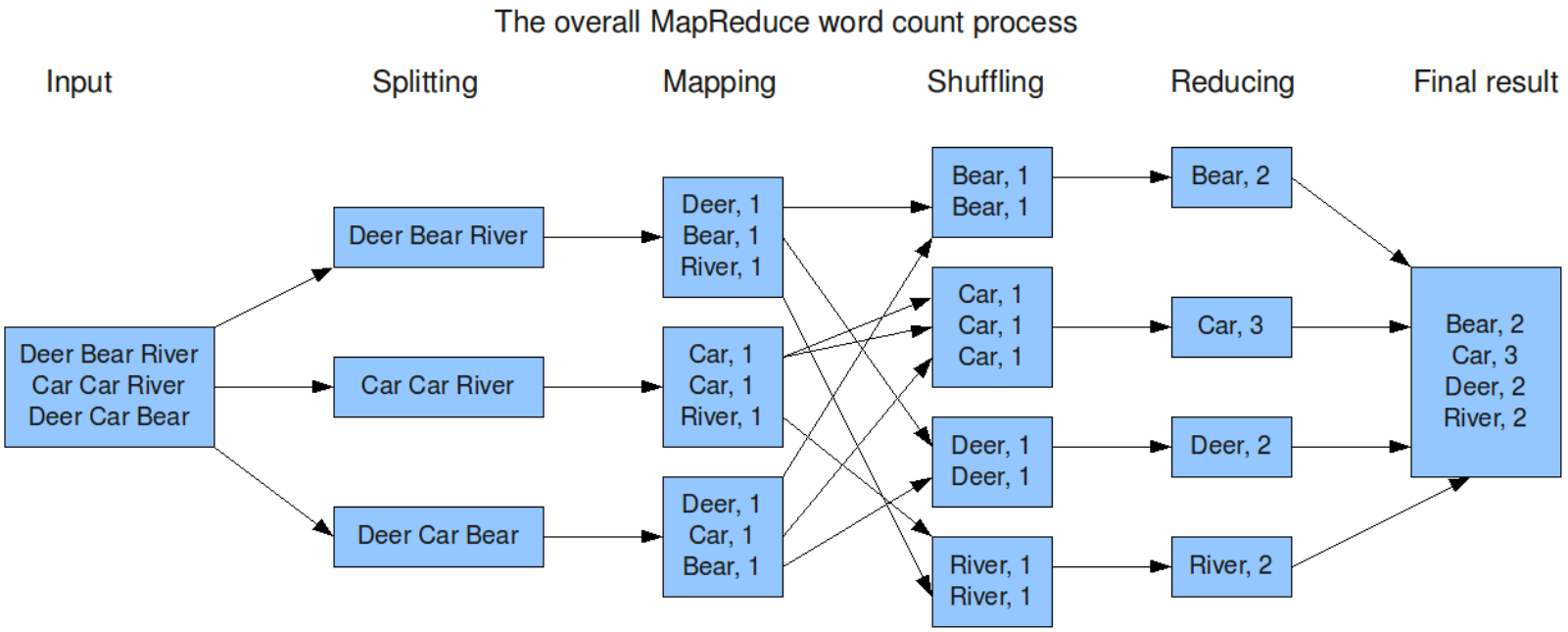
```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");
```

Example: Count all Words

```
reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

Word Count Visualized

- Splits: by files, by lines
  - Like most things, customizable within the framework
- Output is deceptive: not a file, but many files (one per reducer)



What can you do?

- Seems like a limited model
- But, many string processing problems fit naturally
- Can be used iteratively
- Examples
  - Grep
  - Web-log processing (e.g. URL hit counts)
  - Reverse Web-link graph
  - Terms vector per host
  - Build an inverted index
  - Distributed sort (naturally)