Map-Reduce

Service-Oriented Architecture

Jeremy Gibbons

©JG 2010

1 Google's Map–Reduce architecture

- 'simplified data processing on large clusters'
- massively parallel data analysis applications
- programming model and library
- distribution, fault tolerance, I/O scheduling, monitoring
- inspired by FP notions ('map' and 'reduce')
- paper by Jeffrey Dean and Sanjay Ghemawat at OSDI2004
- see also `http://labs.google.com/papers/mapreduce.html`
- more recent:
  `http://www.slideshare.net/jhammerb/mapreduce-pact06-keynote`

©JG 2010

2 Problem

- 20+ billion web pages, 20KB each: 400+TB
- disk read speed 30–35MB/s: four months to read data
- 1000 disks even to store it, but then only 3 hours to read
- however, programming is very difficult
- stuff breaks: 10,000 servers, MTBF $\approx$ 3 years each...

©JG 2010

3 Typical task structure

1. read a lot of data
2. extract something of interest from each record (‘map’)
3. shuffle and sort
4. aggregate, summarize, filter, or transform (‘reduce’)
5. write results

©JG 2010

4 Programming model

Based on key–value pairs. Programmer writes just two functions.

Map function takes input key–value to set of intermediate key–values:

```
map :: (InKey, InValue) -> List (OutKey, OutValue)
```

Reduce function combines values for each intermediate key
(usually producing just a single value):

```
reduce :: (OutKey, List OutValue) -> List OutValue
```

Sometimes also a *partition function*:

```
partition :: (OutKey, NumPartitions) -> PartitionId
```

(by default, hashing modulo number of partitions).

©JG 2010

4.1 Example: word frequencies (pseudocode)

```
type InKey = String       -- document name
type InValue = String     -- document contents
type OutKey = String      -- word
type OutValue = Integer   -- count

map (name, contents)  = [ (word, 1) | word <- contents ]
reduce (word, counts) = [ sum counts ]
```

(In fact, it's a C++ library, and all keys and values are just strings.)

©JG 2010

4.2 Many more examples

distributed grep: mapper emits matching lines, reducer is identity

term vectors: like wordcount, but discarding infrequent words

access statistics: wordcount on web server logs

inverted index: mapper emits `(Word,DocId)` pairs

reverse web-links: mapper emits `(Target,Source)` pairs for hyperlinks

distributed sort: depends on ordering properties of partition, output

Lots of *extract–transform–load* tasks work well.

Thousands of applications in Google source tree (6000+ in 2006).
Production indexing system is a sequence of 24 MR instances.

©JG 2010

5 Implementation

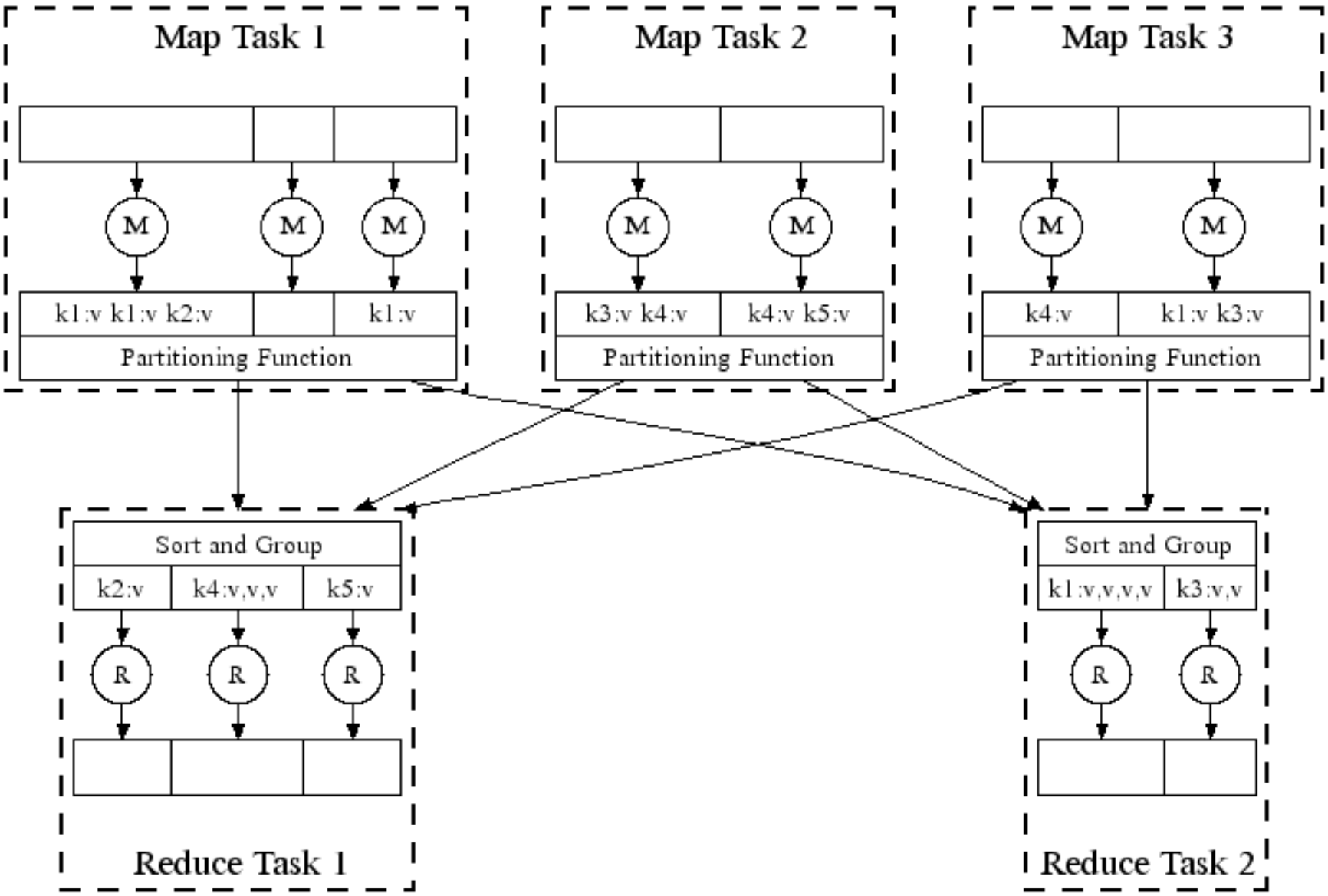
- commodity PCs (eg dual-core x86 running linux)
- commodity networking (eg 100Mb/s or 1Gb/s)
- cheap disks attached directly to computers
- custom redundant distributed file system
- cluster of hundreds or thousands of machines (eg 1800 in OSDI paper)

©JG 2010

5.1 Execution

- divide input data into M *splits*, typically about 64MB
- apply map function to each split
- arrange intermediate values into R *partitions*
- apply reduce function to each partition
- often $M = 200,000$ and $R = 5,000$ on $N = 2,000$ workers: fine-grain
- *master* process assigns tasks to workers, taking locality into account

©JG 2010


Map Task 1
Map Task 2
Map Task 3
M
k1:v k1:v k2:v
k1:v
k3:v k4:v
k4:v k5:v
k4:v
k1:v k3:v
Partitioning Function
Sort and Group
k2:v
k4:v,v,v
k5:v
k1:v,v,v,v
k3:v,v
R
Reduce Task 1
Reduce Task 2

©JG 2010

5.2 Fault tolerance

- regular polling of worker machines
- failure handled by restarting map tasks and in-progress reduce tasks
- (complete reduce tasks stored on global server)
- master failure is unlikely, because only a single machine
- soundness depends on lack of side-effects
  (and on transactional commits)

©JG 2010

5.3 Backup tasks

- straggler tasks lengthen overall computation time
- eg from bad disk (slowing reads from 30MB/s to 1MB/s)
- eg faulty initialization, disabling processor cache (100-fold slowdown)
- when nearly done, master process spawns backup executions of remaining in-progress tasks

©JG 2010

5.4 Refinements

- guaranteed ordering by key within each partition
- optional *combiner* function for output of map tasks
- side-effects allowed, if atomic and idempotent
- skipping bad records (eg for buggy user code)
- local execution mode for debugging
- master process runs internal web server for statistics

©JG 2010

6 Hadoop

- Apache open-source framework for scalable distributed computing
- MapReduce, after Google
- HDFS distributed file system
- other subprojects: Avro, Chukwa, Hive, Pig...
- `http://hadoop.apache.org/`

©JG 2010