The MapReduce Abstraction

Parallel Computing at Google

- Leverages multiple technologies to simplify large-scale parallel computations
  - Proprietary computing clusters
  - Map/Reduce software library
- Lots of other homegrown systems as well
  - Google File Sys: a distributed fault tolerant file system
  - BigTable: A distributed, fault tolerant database

Problems are Really Big

- 20+ billion web pages x 20KB = 400+ terabytes
- If computer can read 30-35 MB/sec from disk
  - Need ~4 months to read
- Takes ~1,000 hard drives to store data
- Even more to do something with the data

Use Large Computing Clusters

- Spread the work over many machines
  - With 1000 CPUs previous problem takes < 3 hours
- Still difficult to implement & manage
  - Programming effort
  - Communication & coordination
  - Recovering from machine failure
  - Status reporting
  - Debugging & optimization

Programming Implications

- Single-thread performance isn't limiting factor
  - Because problems are so large, total throughput/$ more important than peak performance
- Failure is pervasive
  - Assume device lifetime of ~3 years
  - With 10,000 devices, expect to lose 10/day
  - Software must be fault-tolerant
- Communication between computing racks is slow
  - Data locality is very important

MapReduce

- Model applicable to many large computing problems
- Hides many messy details
  - Automatic parallelization
  - Load balancing
  - Network and disk transfer optimization
  - Handling machine failures

Map & Reduce

- Map & Reduce are basic tools of functional programming
- Map – applies a function to each element of a list
- Reduce – combines all elements of a list by applying a binary function

Map

```
(define (square n)
    (* n n))

(map square '(1 2 3 4 5))
    outputs (1 4 9 16 25)
```

Reduce

```
(define (plus a b)
    (+ a b))
```

```
(reduce plus 0 '(1 4 9 16 25)) => 55
    (25+...(4+(1+0)))
```

MapReduce

- MapReduce extends Map and Reduce model to HashMaps

MapReduce

- Map
  - takes key/value pair
  - produces a new set of key/value pairs
- Reduce
  - Combines all intermediate values for a particular key
  - Produces a set of merged output values (usually just one)

MapReduce Template

- Read data
- Map
  - extract some info from each record
- Shuffle and Sort
- Reduce
  - aggregate, summarize, filter, or transform Map output
- Write the results

MapReduce Template (cont.)

- Intermediate values processed by MapReduce
  - Part of shuffle & sort step
- MapReduce groups together all values associated with the same key & passes them to Reduce

Implementing Map

- Map processes data files
  - Web logs, URLs, etc.
- Usually
  - Input key is record location
  - Input value is record

Reduce Implementation

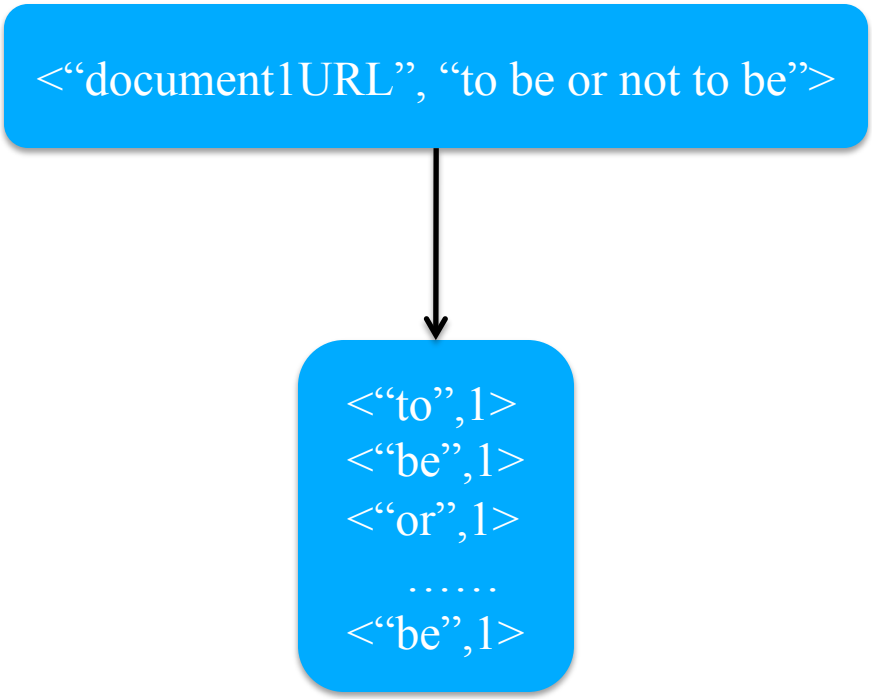
- Inputs are a key and all values for the key
- Merges values, outputting a new list of values
  - Typically 0 or 1 output values per invocation
- Intermediate values supplied to Reduce via an iterator
  - Can thus process of very large input lists

MapReduce Example: Word Count

- Inputs are documents
- Map function takes a key/value pair
  - key = document URL
  - value = document contents
- Outputs the key/value pair (*word*, “1”) for each instance of *word* in the document

MapReduce Example (cont.)

- Result of Map step



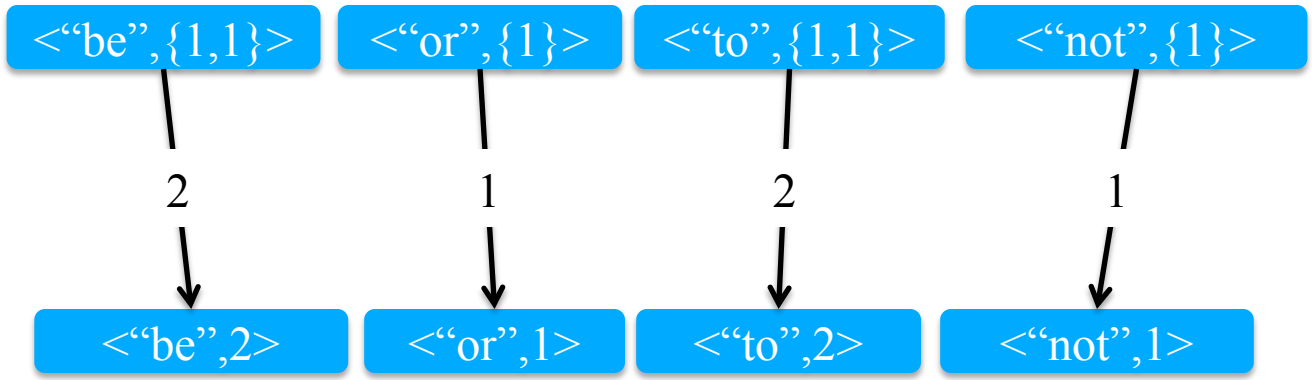
MapReduce Example (cont.)

- Shuffle & Sort gathers all pairs with the same key
- Output:

<“be”,{1,1}> <“or”,{1}> <“to”,{1,1}> <“not”,{1}>

MapReduce Example (cont.)

- Reduce function combines the values for a key
  - Computes and outputs combined value
  - Output of each Reduce call paired with key
- Output



Pseudocode

```
Map(String key, String values):
   // key: doc name
   // values: doc contents
 for each word w in values:
      EmitIntermediate(w, "1");

Reduce(String key, Iterator inValues):
  // key: a word, same for input and output
  // inValues: a list of counts
int result = 0;
for each v in inValues:
      result += ParseInt(v);
Emit(AsString(result));
```

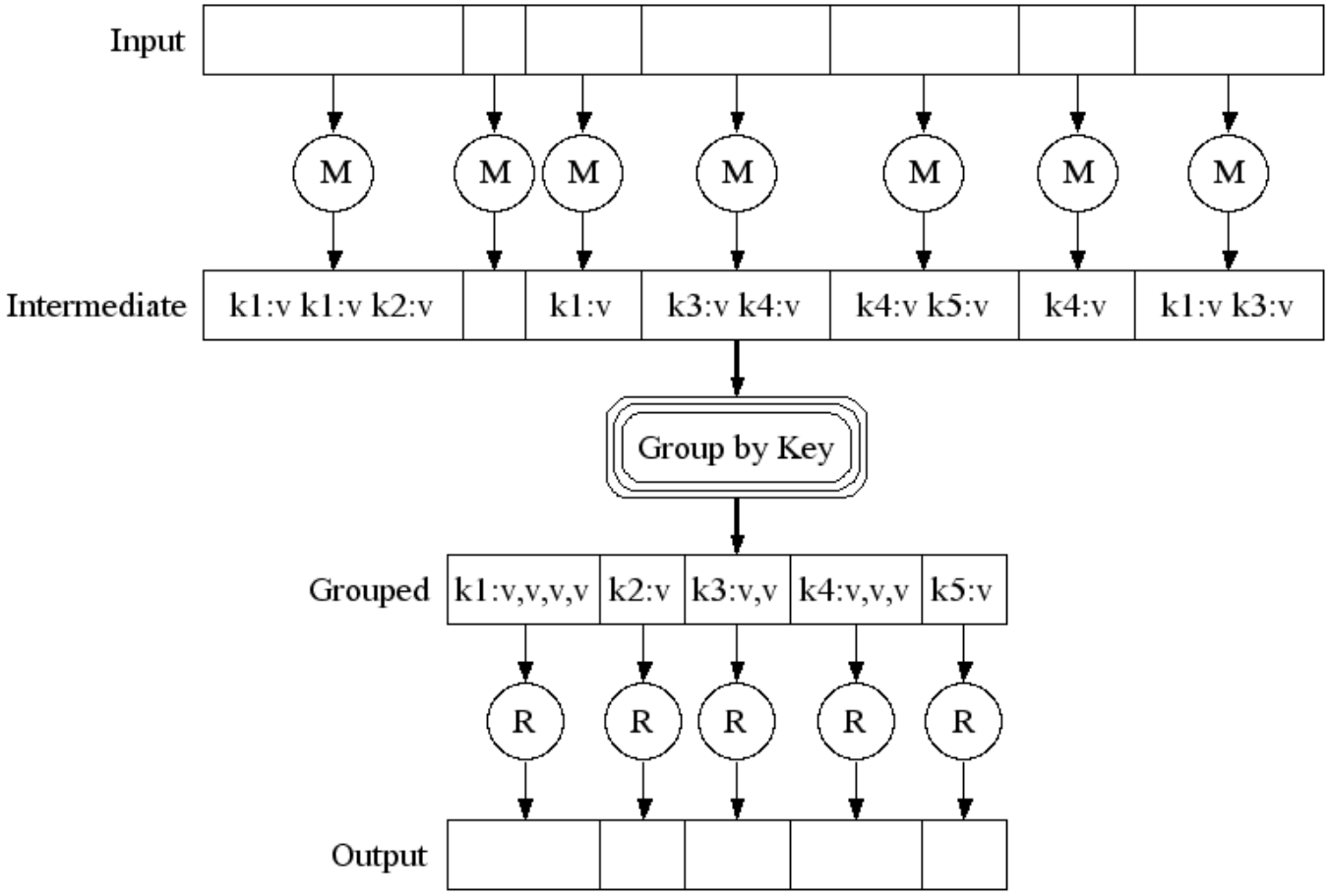
Example: Generating Language Stats

- Used in machine translation
  - need to count # of times every 5-word sequence occurs in a set of docs, storing those where count >= 4
- With MapReduce:
  - Map: emit <5-word seq, 1> from each doc
  - Reduce: sum counts, output if count >= 4

Example: Reverse Web Link Graph

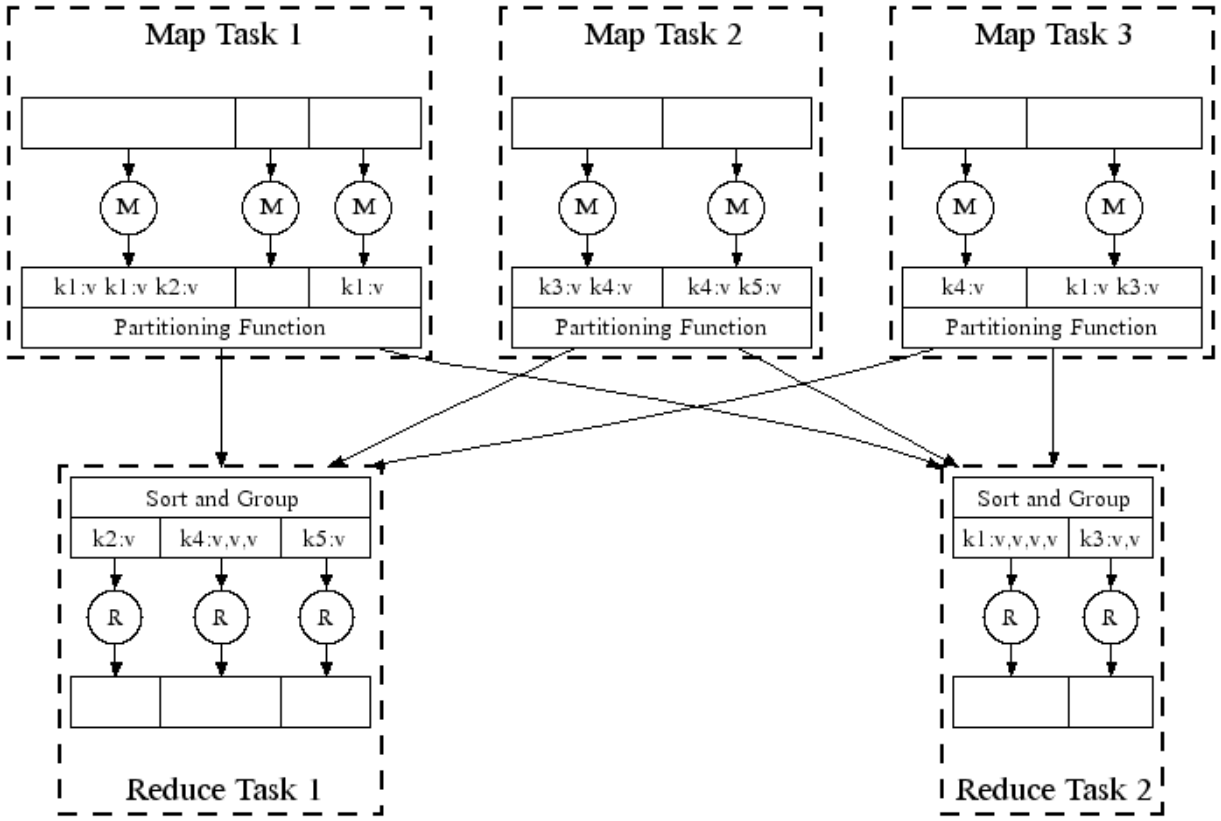
- Compute all webpages that link to a given target
- Map reads webpage named “source”
  - Outputs ⟨target, source⟩ pairs for each link to target found in source
- Reduce concatenates the list of all source URLs associated with a given target URL
  - Outputs the pair: ⟨target, list(source)⟩

Logical Execution



MapReduce: Simplified Data Processing on Large Clusters, Jeffrey Dean, Sanjay Ghemawat, OSDI'04: Sixth Symposium on Operating System Design and Implementation, 2004, pp. 137-150

Parallel Execution



MapReduce: Simplified Data Processing on Large Clusters, Jeffrey Dean, Sanjay Ghemawat, OSDI'04: Sixth Symposium on Operating System Design and Implementation, 2004, pp. 137-150

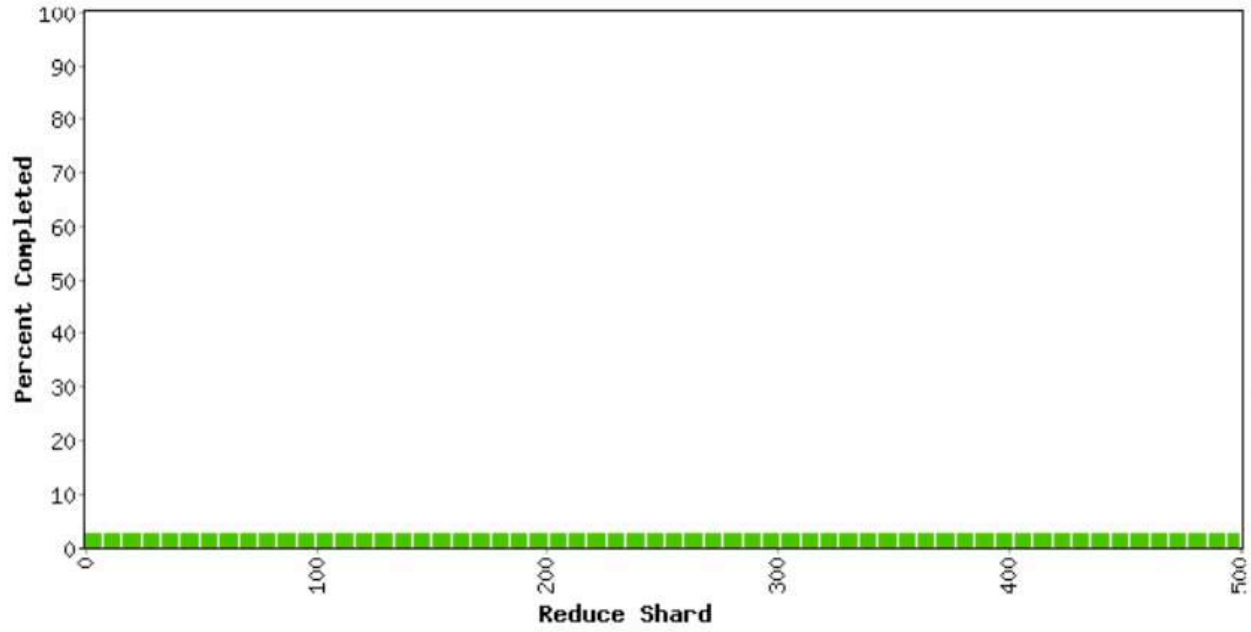
Scheduling

- **One master, many workers**
  - Input data split into *M map tasks (typically 64 MB in size)*
  - Reduce phase partitioned into *R reduce tasks*
  - Tasks are assigned to workers dynamically
  - Often: *M=200,000; R=4,000; workers=2,000*
- **Master assigns each map task to a free worker**
  - Considers locality of data to worker when assigning task
  - Worker reads task input (often from local disk)
  - Worker produces R local files containing inter. k/v pairs
- **Master assigns each reduce task to a free worker**
  - Worker reads intermediate k/v pairs from map workers
  - Worker sorts & applies user's Reduce op to produce the output

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 00 min 18 sec

323 workers; 0 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	0	323	878934.6	1314.4	717.0
Shuffle	500	0	323	717.0	0.0	0.0
Reduce	500	0	0	0.0	0.0	0.0



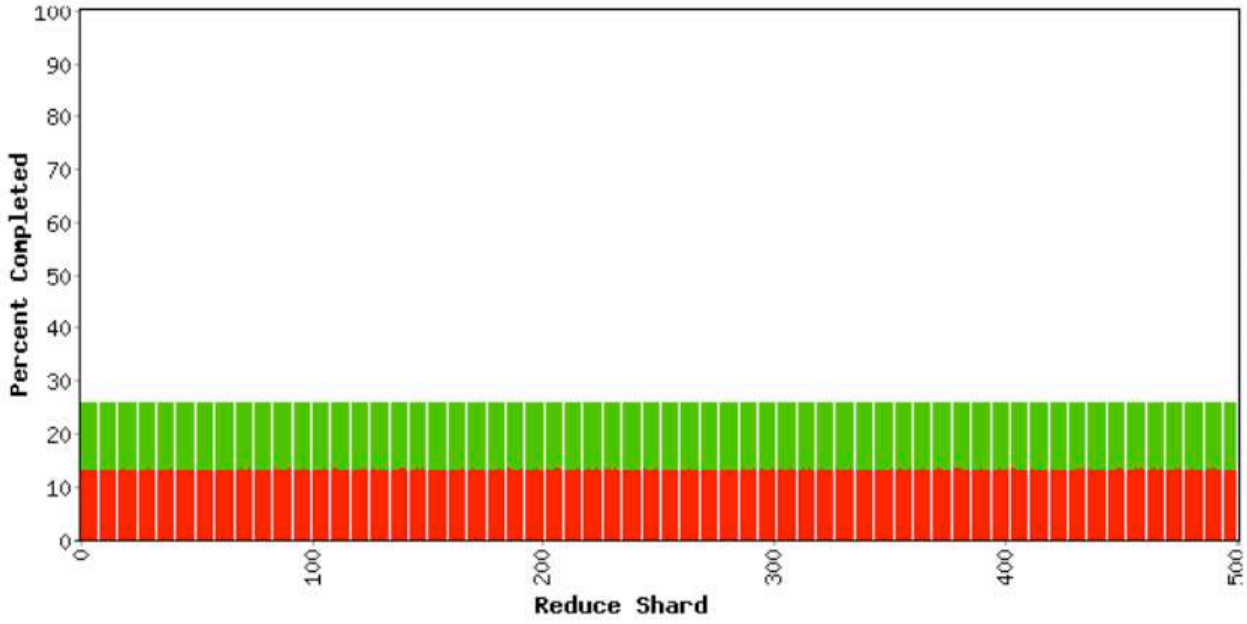
Counters

Variable	Minute
Mapped (MB/s)	72.5
Shuffle (MB/s)	0.0
Output (MB/s)	0.0
doc-index-hits	145825686
docs-indexed	506631
dups-in-index-merge	0
mr-operator-calls	508192
mr-operator-	506631

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 05 min 07 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	1857	1707	878934.6	191995.8	113936.6
Shuffle	500	0	500	113936.6	57113.7	57113.7
Reduce	500	0	0	57113.7	0.0	0.0



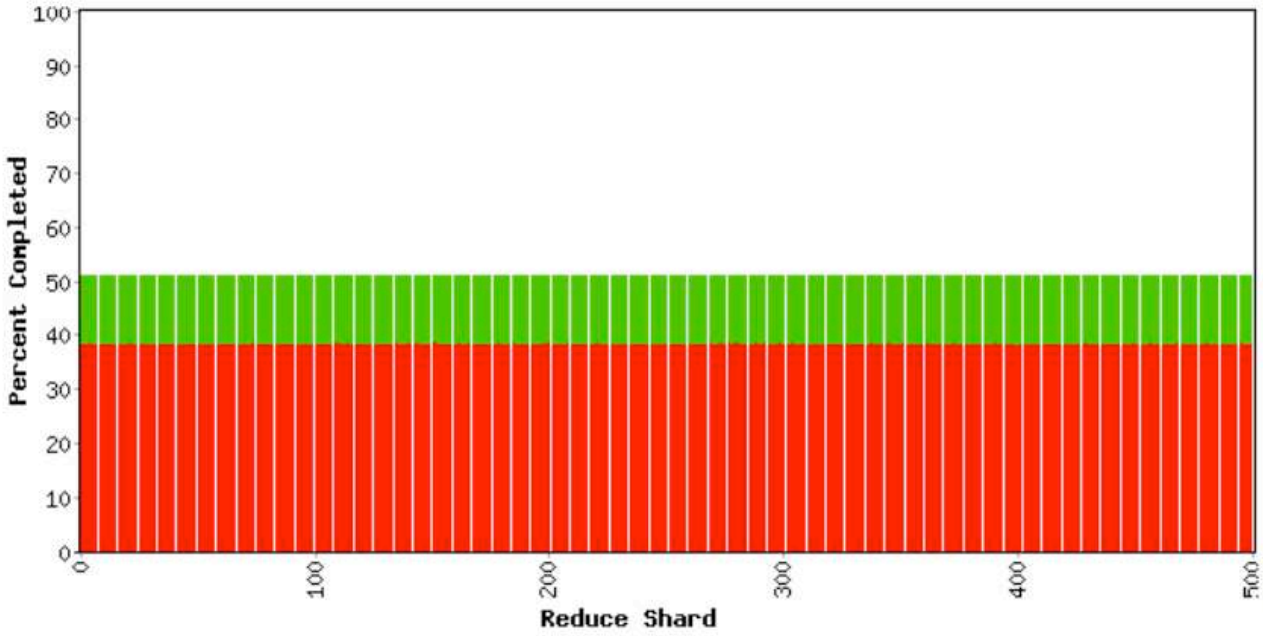
Counters

Variable	Minute
Mapped (MB/s)	699.1
Shuffle (MB/s)	349.5
Output (MB/s)	0.0
doc-index-hits	5004411944
docs-indexed	17290135
dups-in-index-merge	0
mr-operator-calls	17331371
mr-operator-outputs	17290135

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 10 min 18 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	5354	1707	878934.6	406020.1	241058.2
Shuffle	500	0	500	241058.2	196362.5	196362.5
Reduce	500	0	0	196362.5	0.0	0.0



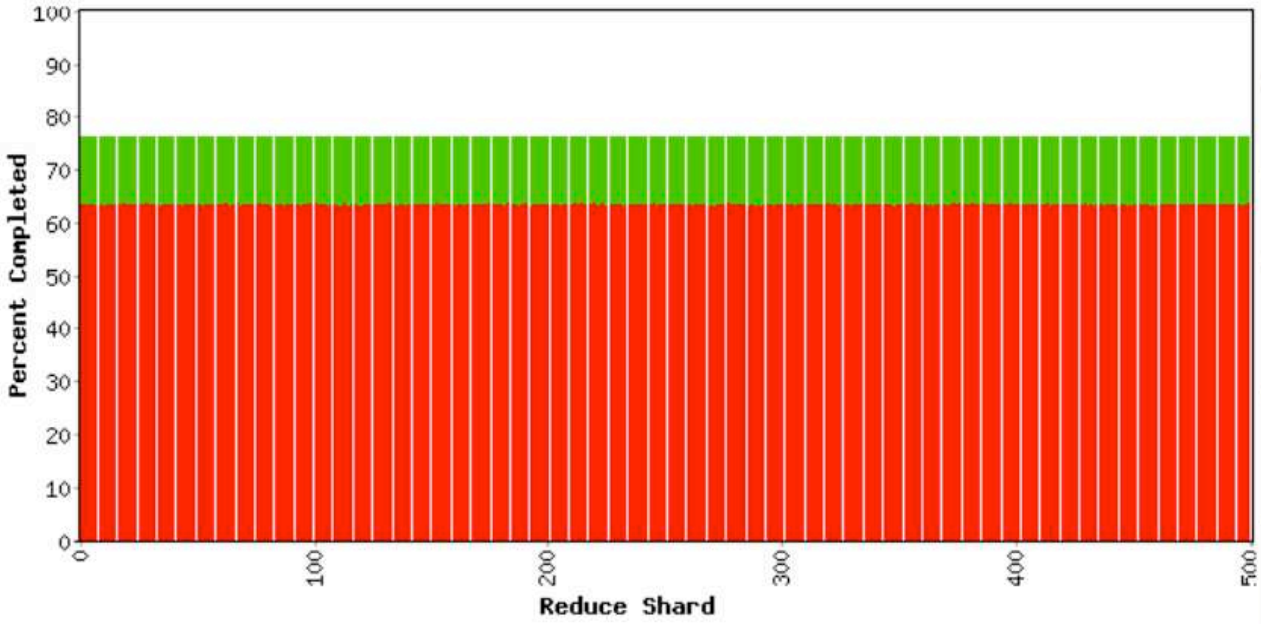
Counters

Variable	Minute
Mapped (MB/s)	704.4
Shuffle (MB/s)	371.9
Output (MB/s)	0.0
doc-index-hits	5000364228
docs-indexed	17300709
dups-in-index-merge	0
mr-operator-calls	17342493
mr-operator-outputs	17300709

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 15 min 31 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	8841	1707	878934.6	621608.5	369459.8
Shuffle	500	0	500	369459.8	326986.8	326986.8
Reduce	500	0	0	326986.8	0.0	0.0



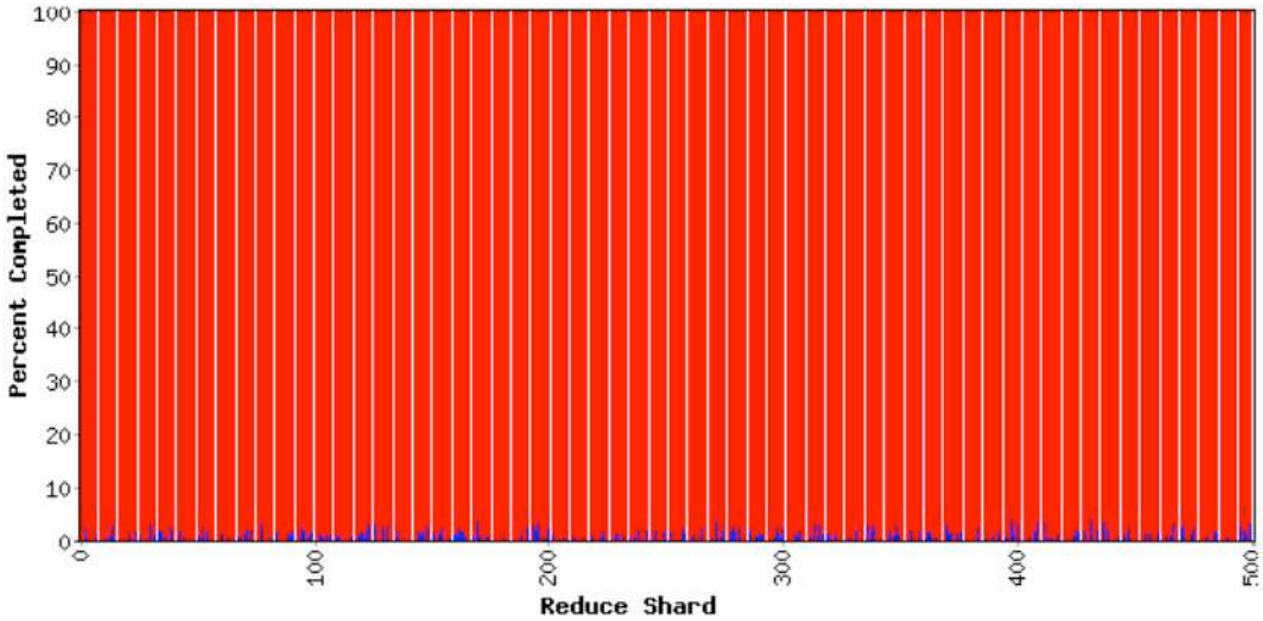
Counters

Variable	Minute
Mapped (MB/s)	706.5
Shuffle (MB/s)	419.2
Output (MB/s)	0.0
doc-index-hits	4982870667
docs-indexed	17229926
dups-in-index-merge	0
mr-operator-calls	17272056
mr-operator-outputs	17229926

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 29 min 45 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	195	305	523499.2	523389.6	523389.6
Reduce	500	0	195	523389.6	2685.2	2742.6



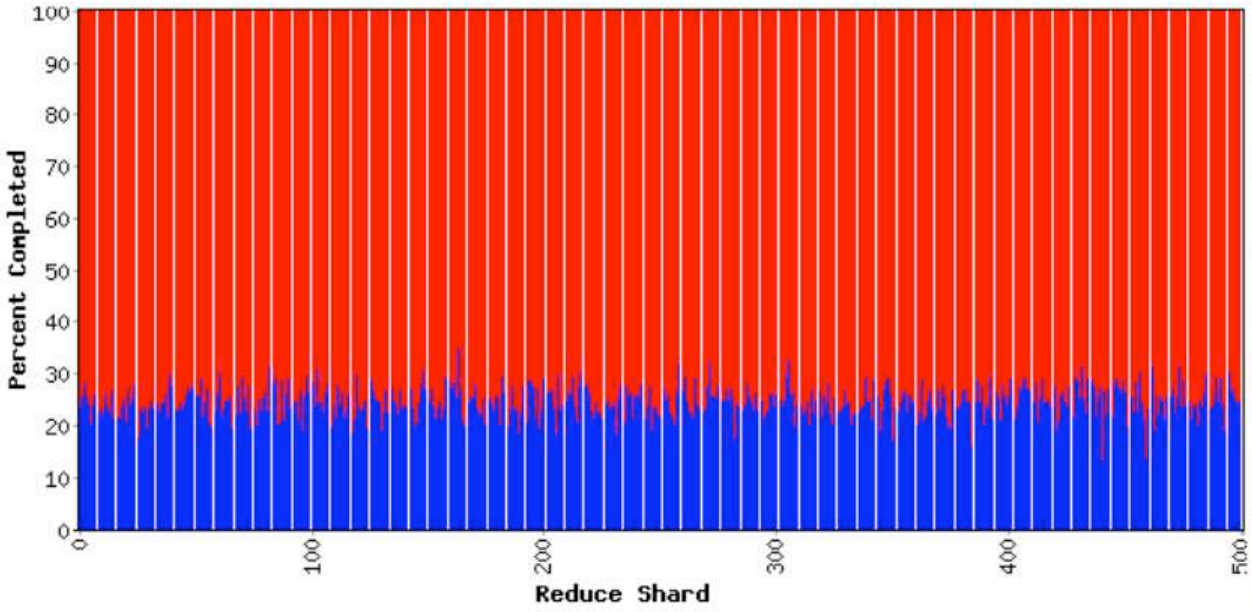
Counters

Variable	Minute	
Mapped (MB/s)	0.3	
Shuffle (MB/s)	0.5	
Output (MB/s)	45.7	
doc-index-hits	2313178	10
docs-indexed	7936	
dups-in-index-merge	0	
mr-merge-calls	1954105	
mr-merge-outputs	1954105	

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 31 min 34 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	133837.8	136929.6



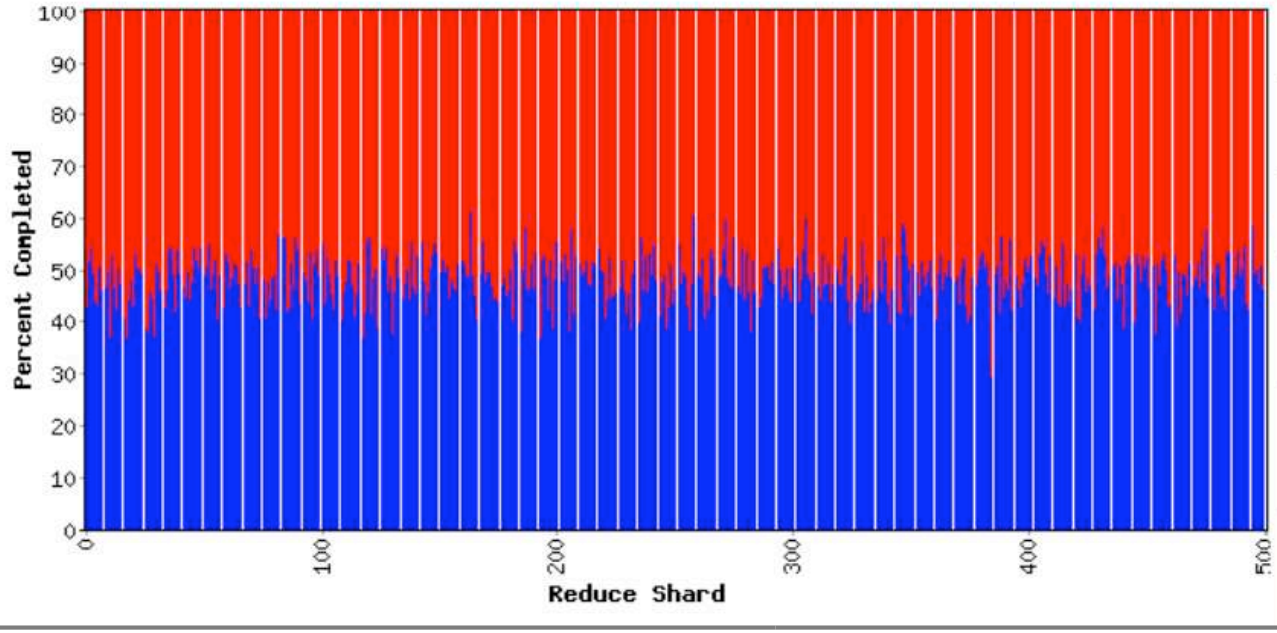
Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.1
Output (MB/s)	1238.8
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	51738599
mr-merge-outputs	51738599

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 33 min 22 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	263283.3	269351.2



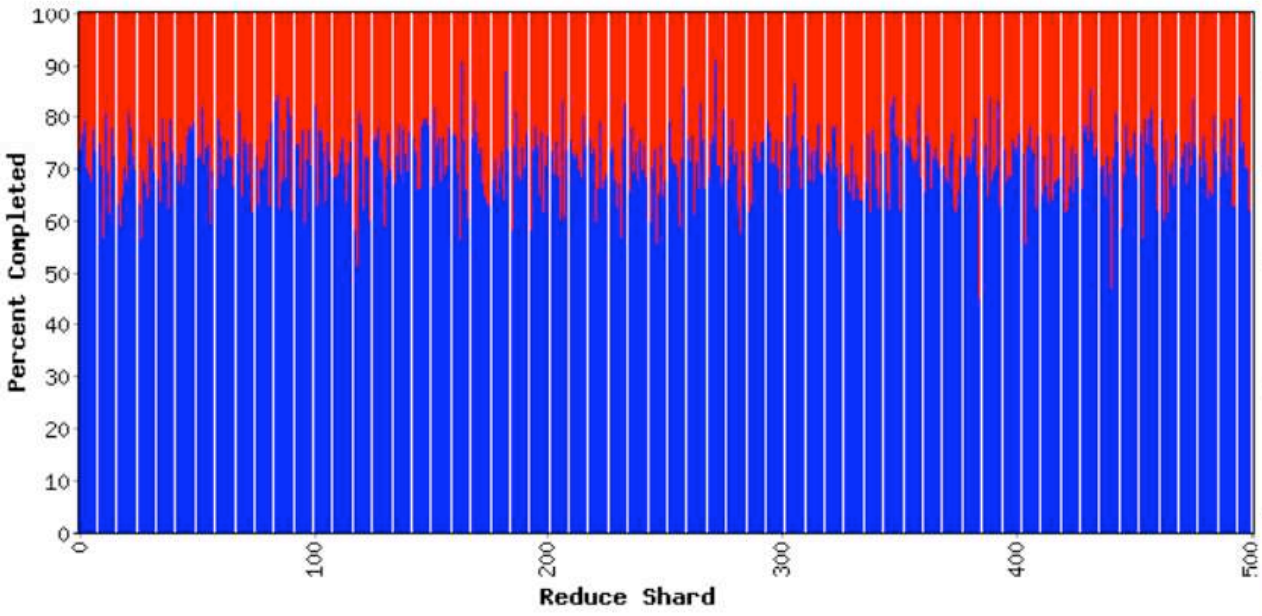
Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	1225.1
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	51842100
mr-merge-outputs	51842100

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 35 min 08 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	523499.5	523499.5
Reduce	500	0	500	523499.5	390447.6	399457.2



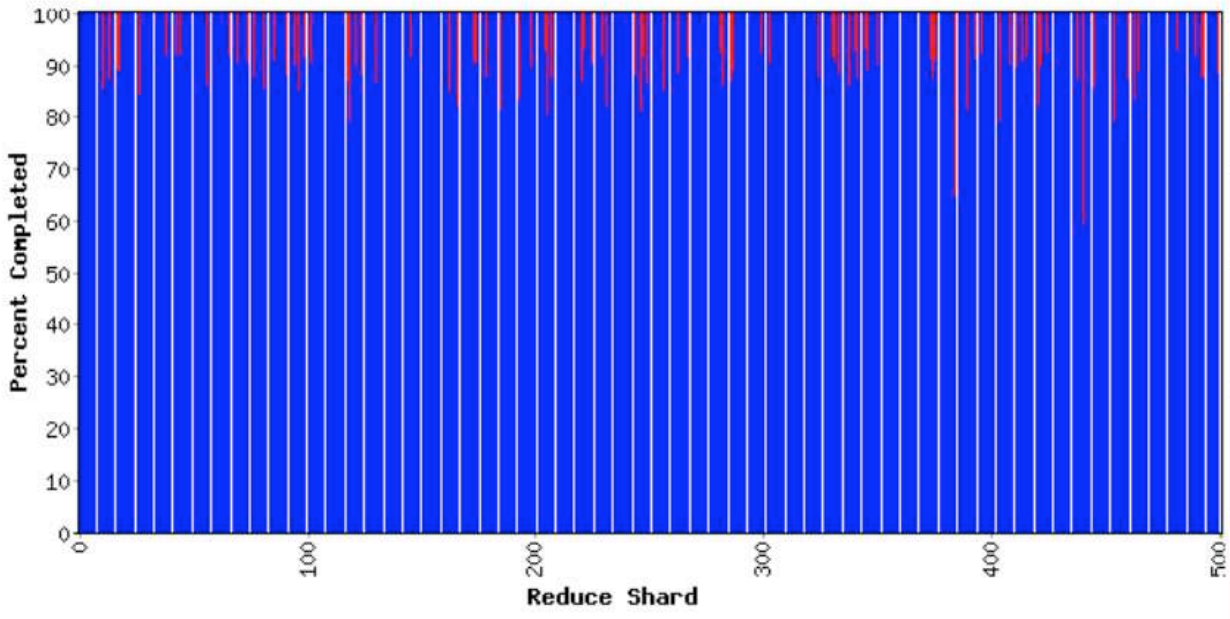
Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	1222.0
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	51640600
mr-merge-outputs	51640600

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 37 min 01 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	520468.6	520468.6
Reduce	500	406	94	520468.6	512265.2	514373.3



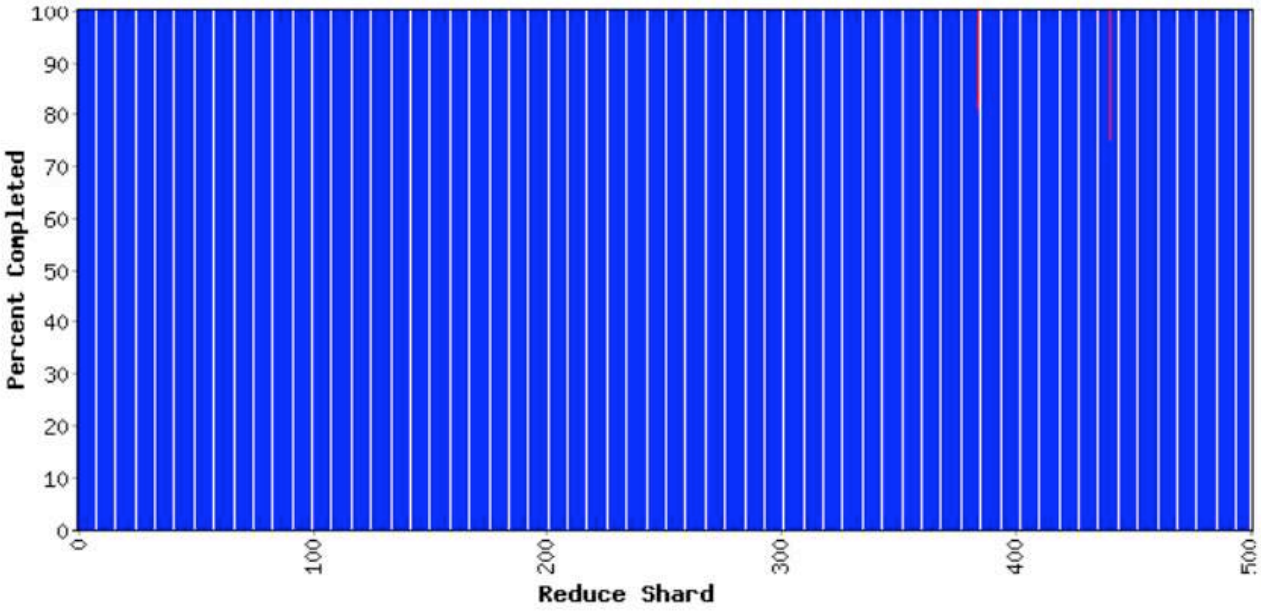
Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	849.5
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	35083350
mr-merge-outputs	35083350

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 38 min 56 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	519781.8	519781.8
Reduce	500	498	2	519781.8	519394.7	519440.7



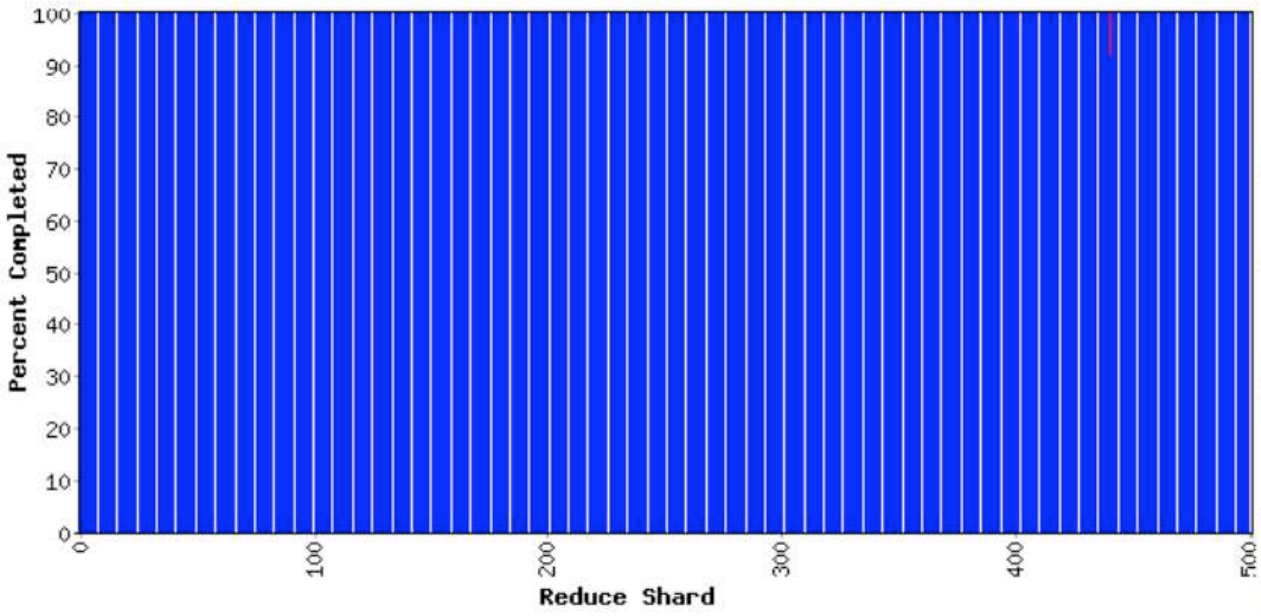
Counters

Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.0	
Output (MB/s)	9.4	
doc-index-hits	0	105
docs-indexed	0	
dups-in-index-merge	0	
mr-merge-calls	394792	
mr-merge-outputs	394792	

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 40 min 43 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	519774.3	519774.3
Reduce	500	499	1	519774.3	519735.2	519764.0



Counters

Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.0	
Output (MB/s)	1.9	
doc-index-hits	0	105
docs-indexed	0	
dups-in-index-merge	0	
mr-merge-calls	73442	
mr-merge-outputs	73442	

Fault tolerance

- Handled via re-execution
- On worker failure:
  - Detect failure via periodic heartbeats
  - Re-execute completed and in-progress map tasks
  - Re-execute in progress reduce tasks
- On master failure:
  - State is checkpointed to GFS: new master recovers & continues

Refinement: Backup Tasks

- Slow workers lengthen completion time
  - Other jobs consuming resources on machine
  - Bad disks, local network problems slow data transfer
  - Other failures
- Solution: Near end, launch duplicate tasks
  - Whoever finishes first "wins”
- Effect: Dramatically shortens job completion time

Refinement: Locality Optimization

- Master scheduling policy:
- Find location of input file blocks replicas
- Map tasks split into 64MB (== GFS block size)
- Map tasks scheduled so data is on same machine or same rack
- Effect: Thousands of machines read input at local disk speed
  - Without this, rack switches limit read rate