CS 61C: Great Ideas in Computer Architecture (Machine Structures)

Instructors:
Randy H. Katz
David A. Patterson

http://inst.eecs.Berkeley.edu/~cs61c/fa10

Agenda

- Request and Data Level Parallelism
- Administrivia
- Technology Break
- Map-Reduce Examples

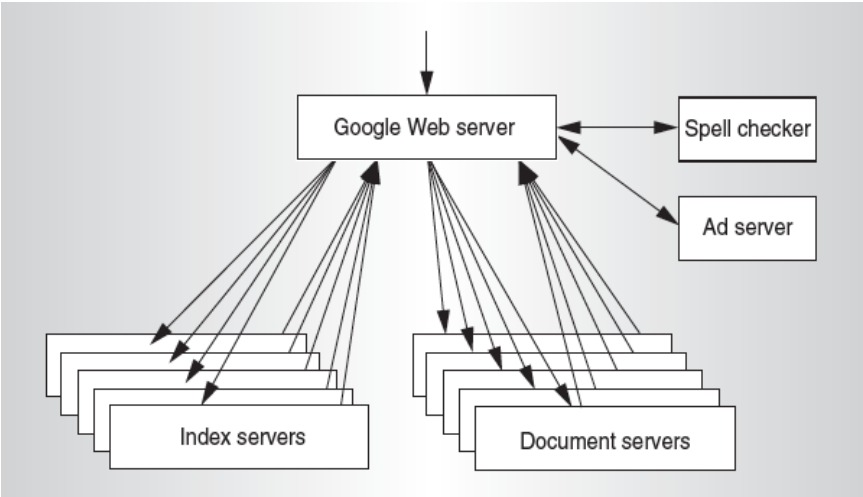
Agenda

- Request and Data Level Parallelism
- Administrivia
- Technology Break
- Map-Reduce Examples

Request-Level Parallelism

- Hundreds or thousands of requests per second
  - Not your laptop or cell-phone, but popular Internet services like Google search
  - Such requests are largely independent
    - Little read-write (aka "producer-consumer") sharing
    - Mostly involve read-only databases
    - Rarely involve read–write data sharing or synchronization across requests
- Computation easily partitioned within a request and across different requests

Google Query-Serving Architecture

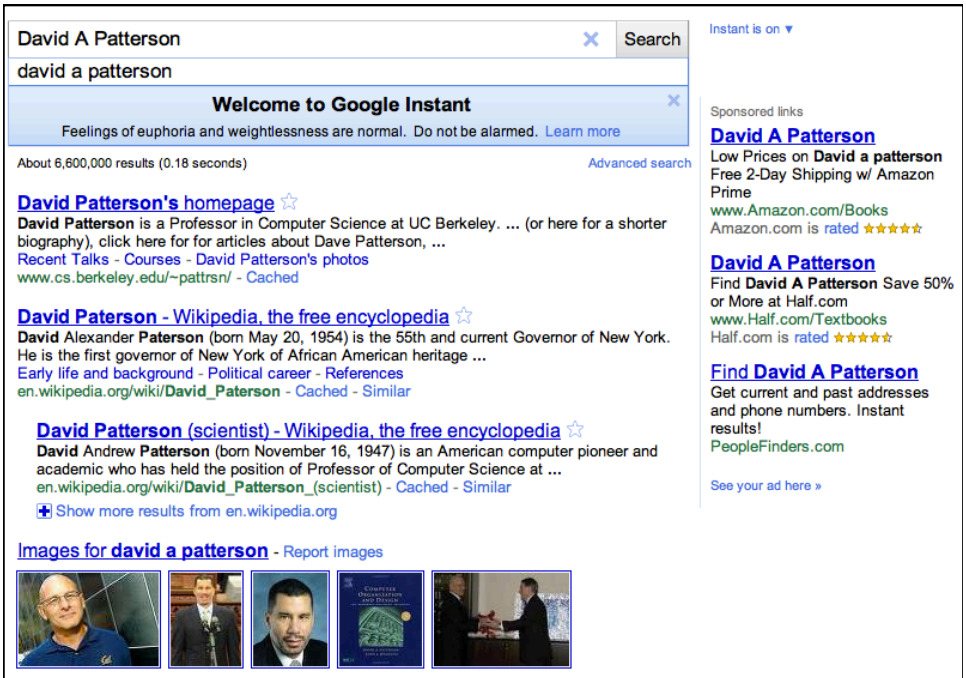


Anatomy of a Web Search

- Google "David A. Patterson"
  - Direct request to "closest" Google datacenter
  - Front-end load balancer directs request to one of many clusters within the datacenter
  - Within cluster, select one of many Google Web Servers (GWS) to handle the request and compose the response pages
  - GWS communicates with Index Servers to find documents that contain the search words, "David", "Patterson"
  - Return document list with associated relevance score

Anatomy of a Web Search

- In parallel,
  - Spell checker: "Did you mean David Paterson?"
  - Ad system: books by Patterson at Amazon.com
  - Images of David Patterson
- Use docids to access indexed documents
- Compose the page
  - Result document extracts (with keyword in context) ordered by relevance score
  - Sponsored links (along the top) and advertisements (along the sides)



Anatomy of a Web Search

- Implementation strategy
  - Randomly distribute the entries
  - Make many copies (aka replicas)
  - Load balance requests across replicas
- Redundant copies of indices and documents
  - Breaks up hot spots, e.g., "Justin Bieber"
  - Increases opportunities for request-level parallelism
  - Makes the system more tolerant of failures

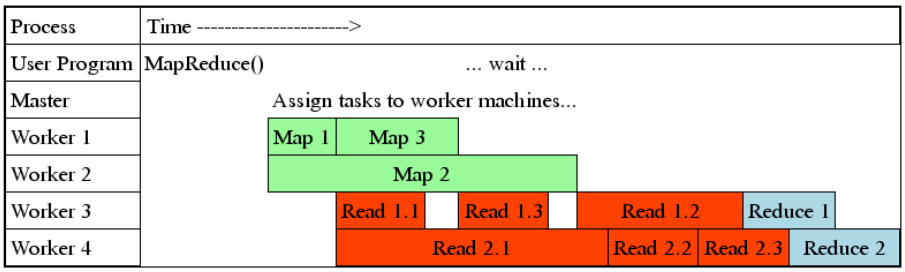
Data-Parallel "Divide and Conquer" (Map-Reduce Processing)

- Map:
  - Slice data into "shards", distribute these to workers, compute sub-problem solutions
  - `map(in_key,in_value)->list(out_key,intermediate value)`
    - Processes input key/value pair
    - Produces set of intermediate pairs
- Reduce:
  - Collect and combine sub-problem solutions
  - `reduce(out_key,list(intermediate_value))->list(out_value)`
    - Combines all intermediate values for a particular key
    - Produces a set of merged output values (usually just one)

Google Uses MR For …

- E.g.:
  - Extracting the set of outgoing links from a collection of HTML documents and aggregating by target document
  - Stitching together overlapping satellite images to remove seams and to select high-quality imagery for Google Earth
  - Generating a collection of inverted index files using a compression scheme tuned for efficient support of Google search queries
  - Processing all road segments in the world and rendering map tile images that display these segments for Google Maps
  - Fault-tolerant parallel execution of programs written in higher-level languages across a collection of input data

Map-Reduce Processing Time Line



- Master assigns map + reduce tasks to worker machines
- As soon as a map task finishes, it can be assigned a new map or reduce task
- Data shuffle begins as soon as a given Map finishes
- Reduce task begins as soon as all data shuffles finish

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

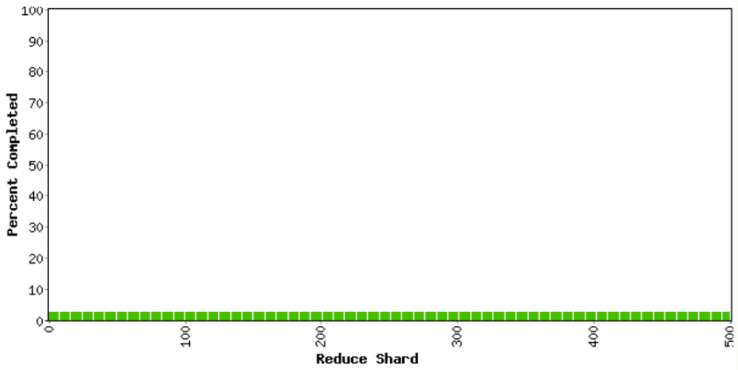
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 00 min 18 sec

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



MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

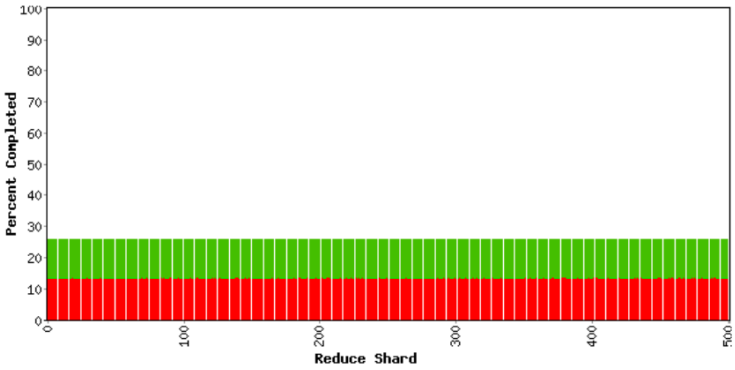
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 05 min 07 sec

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



MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

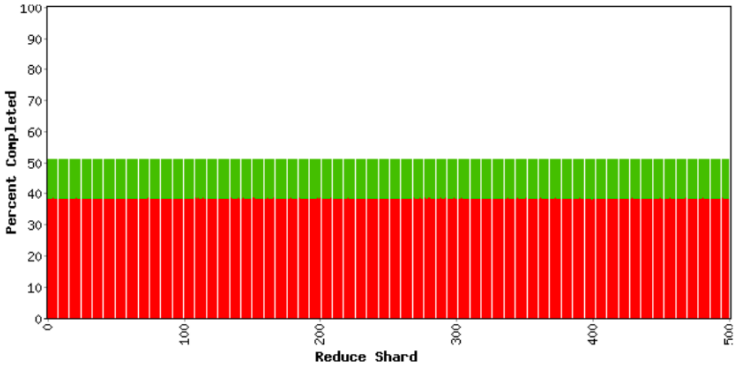
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 10 min 18 sec

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



MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

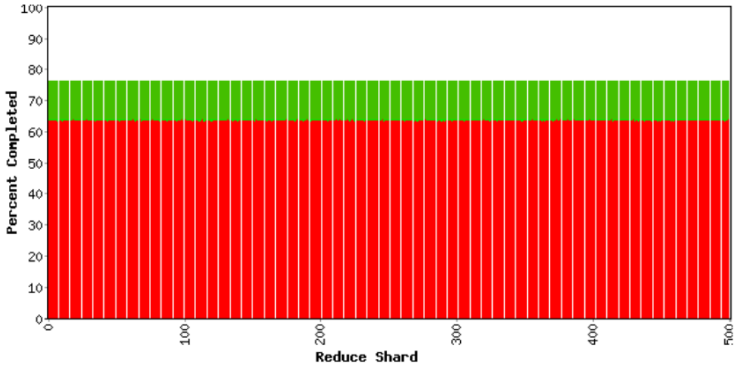
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 15 min 31 sec

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

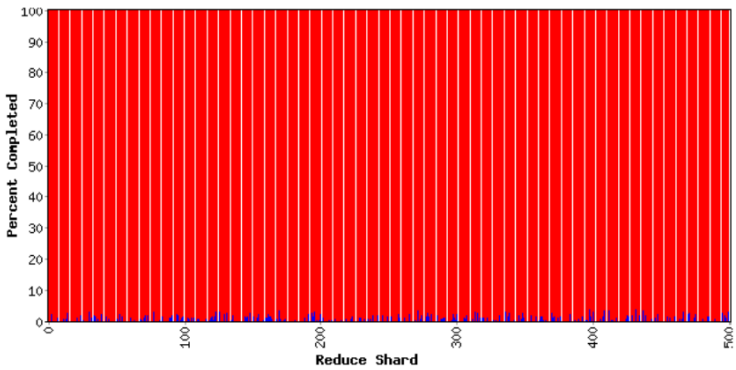


MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

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

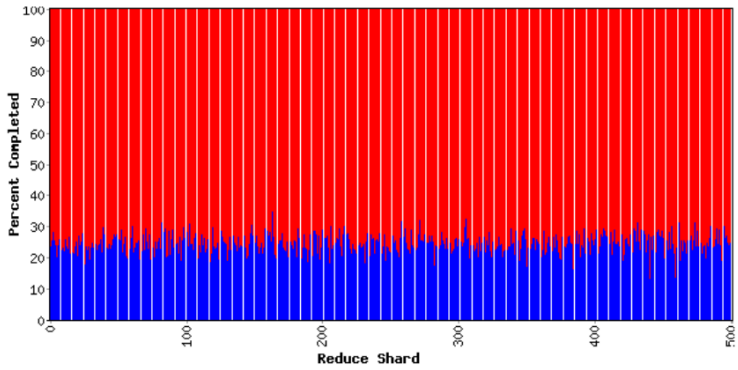
Variable	Minute	
Mapped (MB/s)	0.3	
Shuffle (MB/s)	0.5	
Output (MB/s)	45.7	
doc-index-hits	2313178	105
docs-indexed	7936	
dups-in-index-merge	0	
mr-merge-calls	1954105	
mr-merge-outputs	1954105	

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

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

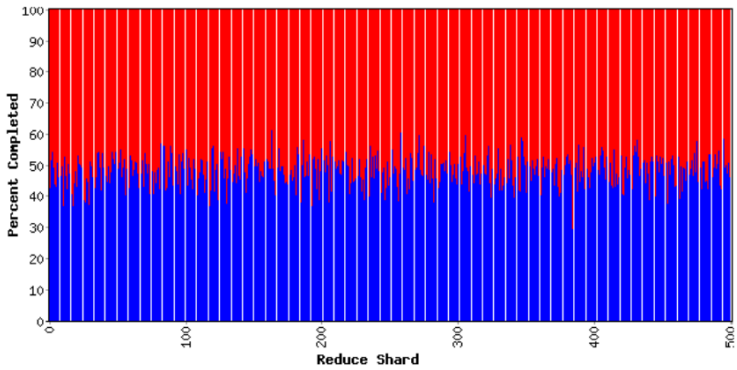
Variable	Minute	
Mapped (MB/s)	0.0	
Shuffle (MB/s)	0.1	
Output (MB/s)	1238.8	
doc-index-hits	0	10
docs-indexed	0	
dups-in-index-merge	0	
mr-merge-calls	51738599	
mr-merge-outputs	51738599	

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

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

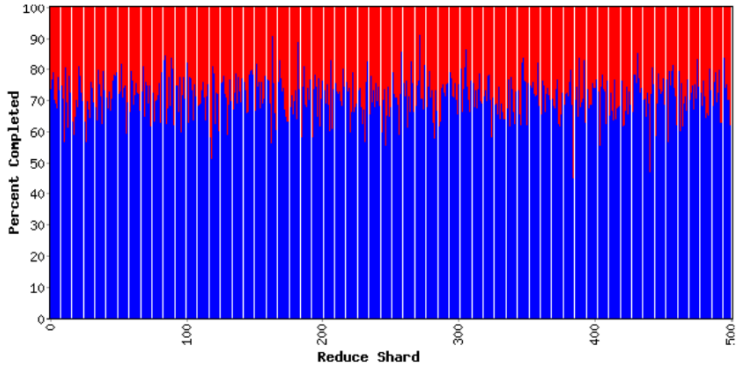
Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	1225.1
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	51842100
mr-merge-outputs	51842100

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

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

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

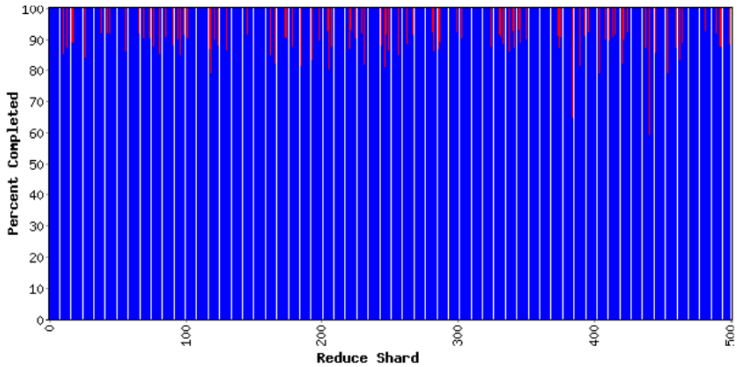
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 37 min 01 sec

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



MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

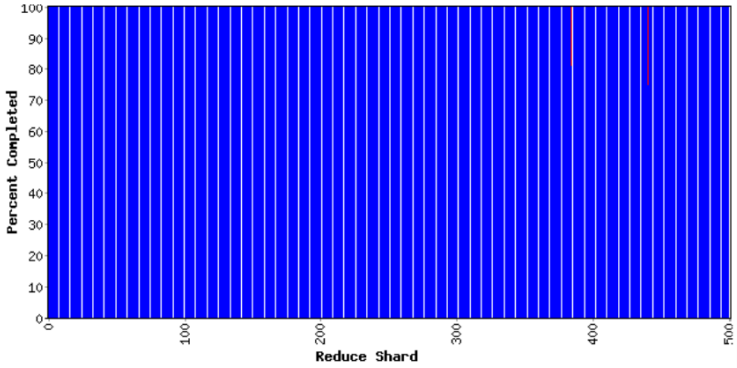
Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 38 min 56 sec

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
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	394792
mr-merge-outputs	394792

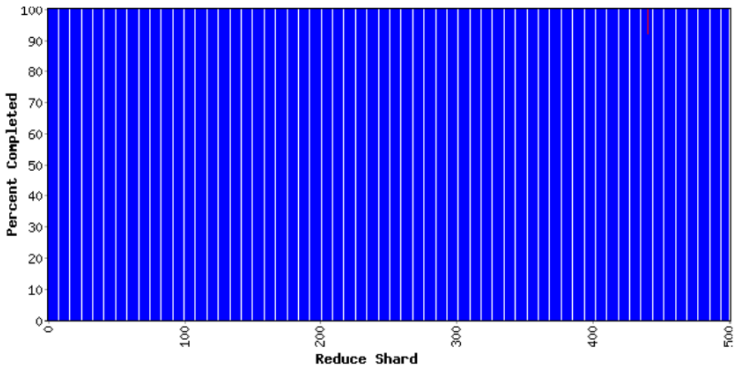


MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

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	

Agenda

- Request and Data-Level Parallelism
- Administrivia
- Technology Break
- Map-Reduce Examples

British Chip Designer Prepares for Wider Demand

By ASHLEE VANCE
Published: September 19, 2010

http://www.nytimes.com/2010/09/20/technology/20arm.html

RECOMMEND
TWITTER
E-MAIL
SEND TO PHONE
PRINT
SINGLE PAGE
REPRINTS
SHARE

CAMBRIDGE, England — Near the southeastern edge of Cambridge, where this idyllic university town gives way to fields of green, sits the headquarters of ARM Holdings. Neither the modest three-building campus nor its surroundings evoke notions of a thriving hotbed of computing.

Enlarge This Image

An ARM chip,and its cousins, may soon be embedded in things beyond nearly every cellphone.

Enlarge This Image

Heidi Schumann for The New York Times

Warren East is chief of ARM Holdings, the quiet licenser of ubiquitous chips.

But ARM, which designs the low-power chips that go into just about every cellphone sold today, commands a prime position when it comes to one of the next major technological revolutions. This is the so-called Internet of Things, when all sorts of everyday objects will have tiny chips placed inside them and gain the ability to process information and talk to the Web.

In this post-PC era, some analysts say, Intel's familiar jingle — bummmm, bum, bum, bum, bum — will fade as the central soundtrack of computing. Instead, people will hear nothing, or rather the understated silence that has accompanied ARM's rise as one of the most important technology companies.

ARM bases its business on licensing chip designs to companies like Apple, Samsung and Qualcomm, which often tweak them to suit their

Agenda

- Data Level Parallelism
- Administrivia
- Technology Break
- Map-Reduce Examples

Agenda

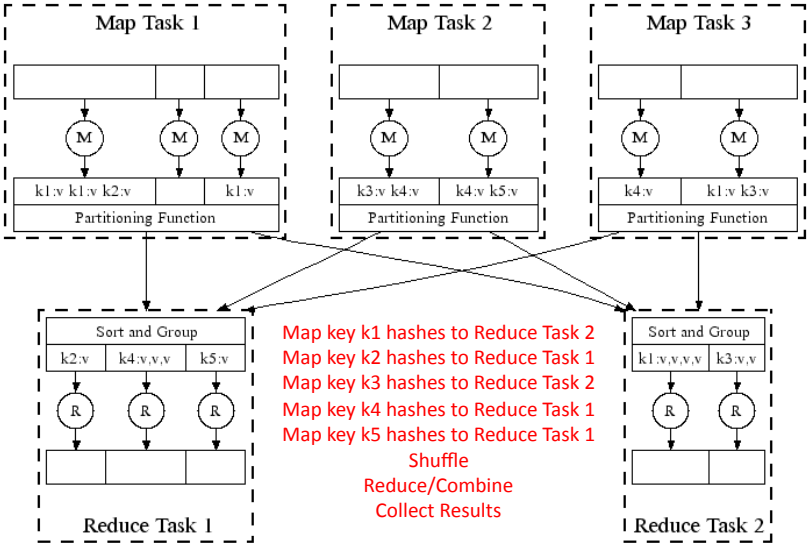
- Data Level Parallelism
- Administrivia
- Technology Break
- Map-Reduce Examples

Map-Reduce Processing Example: Count Word Occurrences

- Pseudo Code:

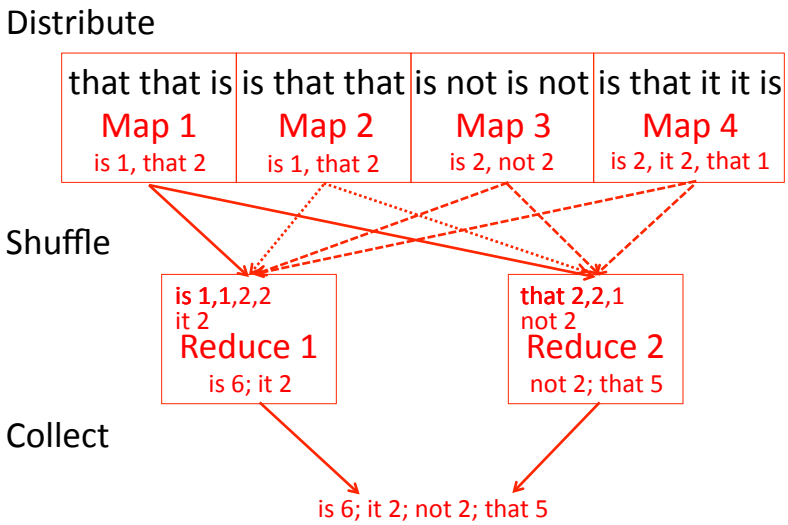
```
map(String input_key, String input_value):
  // input_key: document name
  // input_value: document contents
  for each word w in input_value:
    EmitIntermediate(w, "1");


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


Map-Reduce Processing: Execution



Map key k1 hashes to Reduce Task 2
Map key k2 hashes to Reduce Task 1
Map key k3 hashes to Reduce Task 2
Map key k4 hashes to Reduce Task 1
Map key k5 hashes to Reduce Task 1
Shuffle
Reduce/Combine
Collect Results

Another Example: Word Index (How Often Does a Word Appear?)



Map-Reduce Failure Handling

- On worker failure:
  - Detect failure via periodic heartbeats
  - Re-execute completed and in-progress map tasks
  - Re-execute in progress reduce tasks
  - Task completion committed through master
- Master failure:
  - Could handle, but don't yet (master failure unlikely)
- Robust: lost 1600 of 1800 machines once, but finished fine

Map-Reduce Redundant Execution

- Slow workers significantly lengthen completion time
  - Other jobs consuming resources on machine
  - Bad disks with soft errors transfer data very slowly
  - Weird things: processor caches disabled (!!)
- Solution: Near end of phase, spawn backup copies of tasks
  - Whichever one finishes first "wins"
- Effect: Dramatically shortens job completion time

Map-Reduce Locality Optimization

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

Summary

- Request-Level Parallelism
  - High request volume, each largely independent of the other
  - Use replication for better request throughput and availability
- Map-Reduce Data Parallelism
  - Divide large data set into pieces for independent parallel processing
  - Combine and process intermediate results to obtain final result