

In the news
SILICONVALLEY.COM

Server farms sprouting in Silicon Valley

pcarey@mercurynews.com
Posted: 10/06/2011 03:00:00 PM PDT

"The cloud isn't this big fluffy area up in the sky," said Rich Miller, who edits Data Center Knowledge, a publication that follows the data center industry. "It's buildings filled with servers and generators."

".... you'd be hard-pressed to go three miles here without passing some type of data center. "They're all over the place."

10/7/11 1

CS 61C: Great Ideas in Computer Architecture (Machine Structures)
Lecture 18 – Data Parallelism 1

Instructors:
Michael Franklin
Dan Garcia

<http://inst.eecs.Berkeley.edu/~cs61c/fa11>

10/7/11 2

Review

- Great Ideas
 1. Layers of Representation/Interpretation
 2. Moore's Law
 3. Principle of Locality/Memory Hierarchy
 4. **Parallelism**
 5. Performance Measurement and Improvement
 6. Dependability via Redundancy
- Post PC Era: Parallel processing, smart phones to WSC
- WSC SW must cope with failures, varying load, varying HW latency bandwidth
- WSC HW sensitive to cost, energy efficiency

10/7/11 3

New-School Machine Structures (It's a bit more complicated!)

Today's Lecture Software Hardware

- **Parallel Requests**
Assigned to computer
e.g., Search "Garcia"
- **Parallel Threads**
Assigned to core
e.g., Lookup, Ads
- **Parallel Instructions**
>1 instruction @ one time
e.g., 5 pipelined instructions
- **Parallel Data**
>1 data item @ one time
e.g., Add of 4 pairs of words
- **Hardware descriptions**
All gates @ one time

Warehouse Scale Computer Smart Phone

Harness Parallelism & Achieve High Performance

Computer
Core ... Core
Memory (Cache)
Input/Output
Core
Instruction Unit(s) Functional Unit(s)
Main Memory
Logic Gates

10/7/11 4

Agenda

- Request and Data Level Parallelism
- MapReduce
- MapReduce Examples

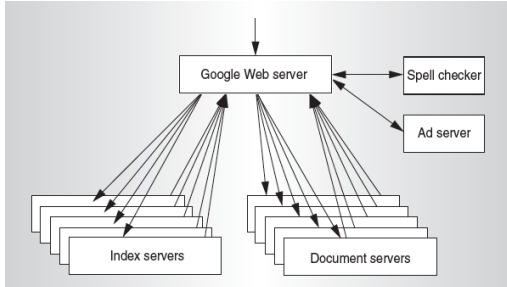
10/7/11 5

Request-Level Parallelism (RLP)

- Hundreds or thousands of requests per second
 - Not your laptop or cell-phone, but popular Internet services like web search, social networking,...
 - Such requests are largely independent
 - Often involve read-mostly databases
 - Rarely involve strict read-write data sharing or synchronization across requests
- Computation easily partitioned within a request and across different requests

10/7/11 6

Google Query-Serving Architecture



10/7/11

7

Anatomy of a Web Search

- Google “Dan Garcia”

10/7/11

8

9

Anatomy of a Web Search (1 of 3)

- Google “Dan Garcia”
 - Direct request to “closest” Google Warehouse Scale Computer
 - Front-end load balancer directs request to one of many arrays (cluster of servers) within WSC
 - Within array, 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, “Dan”, “Garcia”, uses location of search as well
 - Return document list with associated relevance score

10/7/11

10

Anatomy of a Web Search (2 of 3)

- In parallel,
 - Ad system: run ad auction for bidders on search terms
 - Get Images of various Dan Garcias
- Use docids (document IDs) 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)

10/7/11

11

Anatomy of a Web Search (3 of 3)

- Implementation strategy
 - Randomly distribute the entries
 - Make many copies of data (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

10/7/11

12

Data-Level Parallelism (DLP)

- 2 kinds
 1. Lots of **data in memory** that can be operated on in parallel (e.g., adding together 2 arrays)
 2. Lots of **data on many disks** that can be operated on in parallel (e.g., searching for documents)
- Future lecture and 3rd project does Data Level Parallelism (DLP) in memory
- Today's lecture and 2nd project does DLP across many servers and disks using MapReduce

10/7/11

13

What is MapReduce?

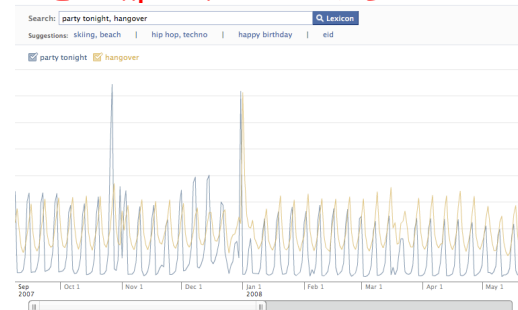
- Simple data-parallel programming model designed for scalability and fault-tolerance
- Pioneered by Google
 - Processes >25 petabytes of data per day
- Popularized by open-source Hadoop project
 - Used at Yahoo!, Facebook, Amazon, ...



What is MapReduce used for?

- At Google:
 - Index construction for Google Search
 - Article clustering for Google News
 - Statistical machine translation
 - For computing multi-layer street maps
- At Yahoo!:
 - “Web map” powering Yahoo! Search
 - Spam detection for Yahoo! Mail
- At Facebook:
 - Data mining
 - Ad optimization
 - Spam detection

Example: Facebook Lexicon



www.facebook.com/lexicon (no longer available)

MapReduce Design Goals

1. **Scalability to large data volumes:**
 - 1000's of machines, 10,000's of disks
2. **Cost-efficiency:**
 - Commodity machines (cheap, but unreliable)
 - Commodity network
 - Automatic fault-tolerance (fewer administrators)
 - Easy to use (fewer programmers)

Jeffrey Dean and Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” *Communications of the ACM*, Jan 2008.

MapReduce Solution

- Apply Map function to user supplied record of key/value pairs
 - Compute set of intermediate key/value pairs
- Apply Reduce operation to all values that share same key in order to combine derived data properly
- User supplies Map and Reduce operations in functional model
 - so can parallelize,
 - can use re-execution for fault tolerance

10/7/11

18

Data-Parallel "Divide and Conquer" (MapReduce Processing)

- Map:**
 - Slice data into "shards" or "splits", 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)
- Fun to use:** focus on problem, let MapReduce library deal with messy details

10/7/11 19

Typical Hadoop Cluster

- 40 nodes/rack, 1000-4000 nodes in cluster
- 1 Gbps bandwidth within rack, 8 Gbps out of rack
- Node specs (Yahoo terasort):
8 x 2GHz cores, 8 GB RAM, 4 disks (= 4 TB?)

http://www.infoq.com/news/2008/09/hadoop-architecture

10/7/11 20

MapReduce Execution

2000 servers =>
≈ 200,000 Map Tasks,
≈ 5,000 Reduce tasks

10/7/11 21

MapReduce Processing Example: Count Word Occurrences

```

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"); // Produce count of words

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); // get integer from key-value
  Emit(AsString(result));
    
```

10/7/11 22

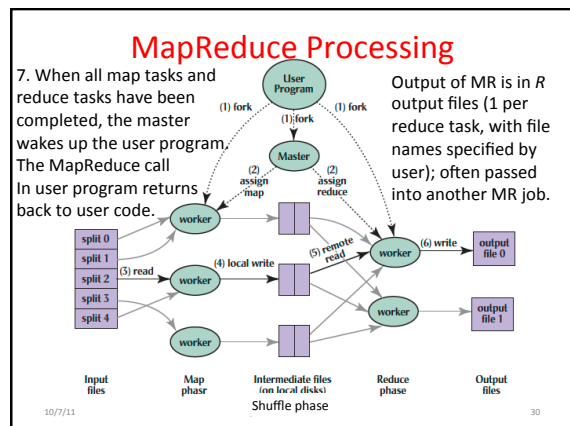
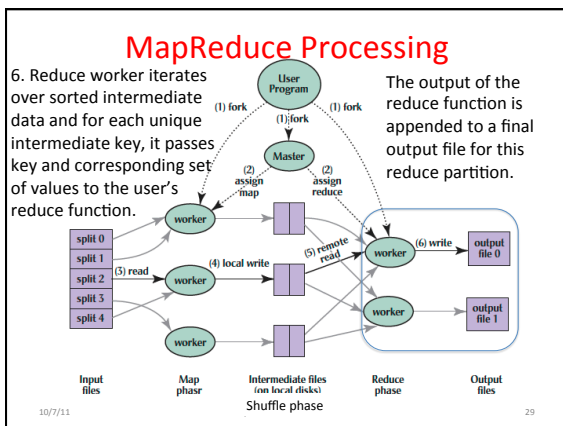
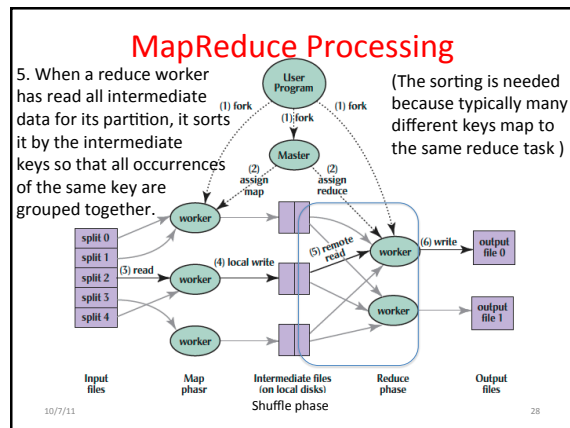
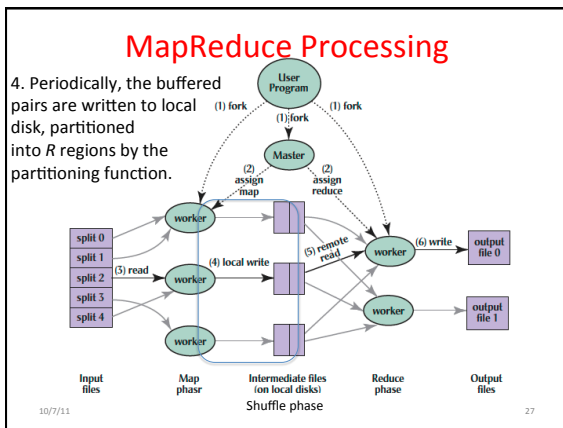
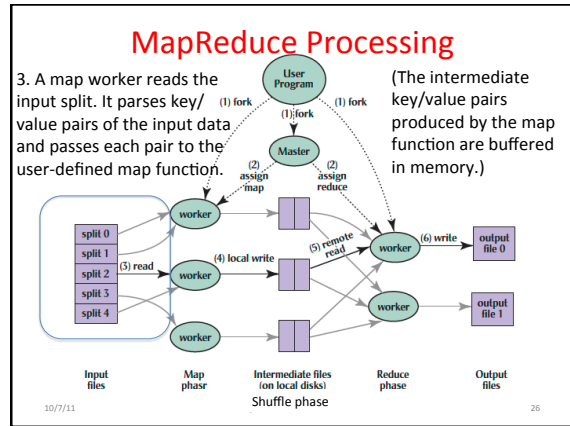
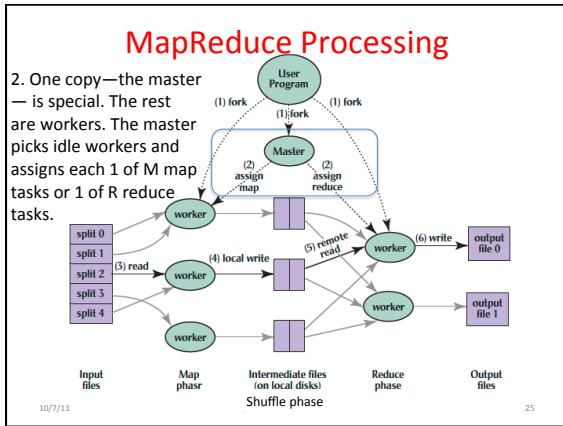
MapReduce Processing

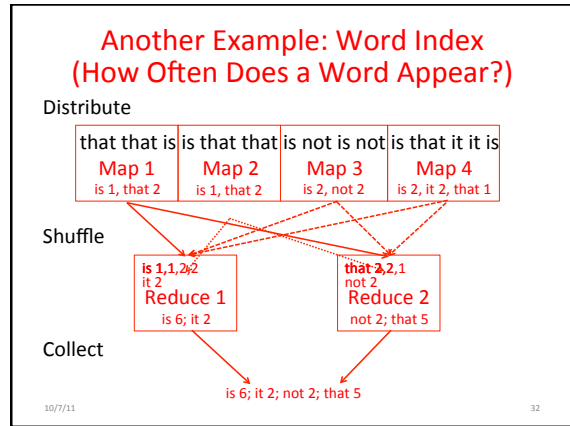
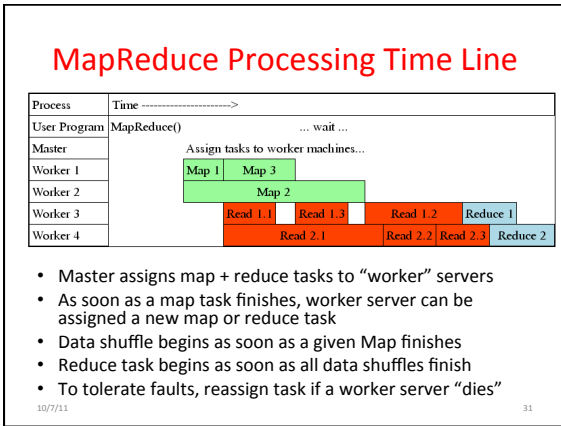
1. MR 1st splits the input files into M "splits" then starts many copies of program on servers

10/7/11 23

MapReduce Processing

10/7/11 24





- ### MapReduce 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:
 - Protocols exist to handle (master failure unlikely)
 - Robust: lost 1600 of 1800 machines once, but finished fine (story from Google?)
- 10/7/11 33

- ### MapReduce 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
 - 3% more resources, large tasks 30% faster
- 10/7/11 34

- ### Summary
- Request-Level Parallelism
 - High request volume, each largely independent of other
 - Use replication for better request throughput, availability
 - MapReduce Data Parallelism
 - Divide large data set into pieces for independent parallel processing
 - Combine and process intermediate results to obtain final result
- 10/7/11 35