Lecture 24: Parallel Databases

Wednesday, November 24, 2010

Overview

- Parallel architectures and operators: Ch. 20.1
- Map-reduce: Ch. 20.2
- Semijoin reductions, full reducers: Ch. 20.4
  – We covered this a few lectures ago

Parallel v.s. Distributed Databases

- Parallel database system:
  – Improve performance through parallel implementation
- Distributed database system:
  – Data is stored across several sites, each site managed by a DBMS capable of running independently

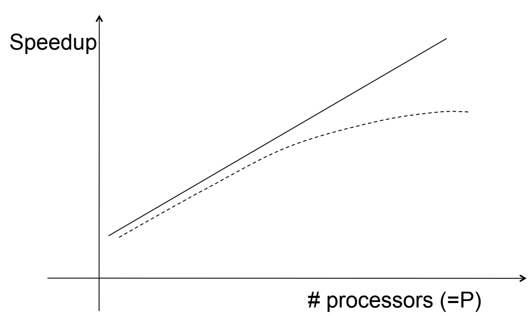
Parallel DBMSs

- Goal
  – Improve performance by executing multiple operations in parallel
- Key benefit
  – Cheaper to scale than relying on a single increasingly more powerful processor
- Key challenge
  – Ensure overhead and contention do not kill performance

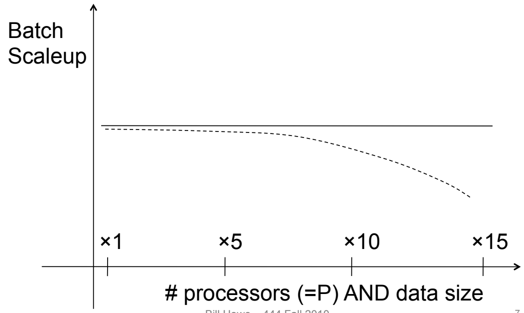
Performance Metrics for Parallel DBMSs

- Speedup
  – More processors → higher speed
  – Individual queries should run faster
  – Should do more transactions per second (TPS)
  – Fixed problem size *overall*, vary # of processors ("strong scaling")
- Scaleup
  – More processors → can process more data
  – Fixed problem size *per processor*, vary # of processors ("weak scaling")
  – Batch scaleup
    - Same query on larger input data should take the same time
  – Transaction scaleup
    - N-times as many TPS on N-times larger database
    - But each transaction typically remains small

Linear v.s. Non-linear Speedup



Linear v.s. Non-linear Scaleup



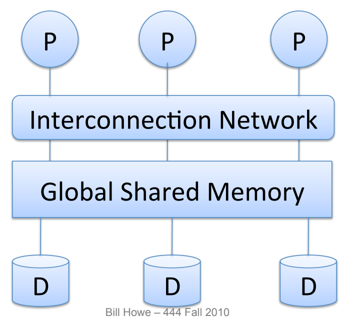
Challenges to Linear Speedup and Scaleup

- Startup cost
  - Cost of starting an operation on many processors
- Interference
  - Contention for resources between processors
- Skew
  - Slowest processor becomes the bottleneck

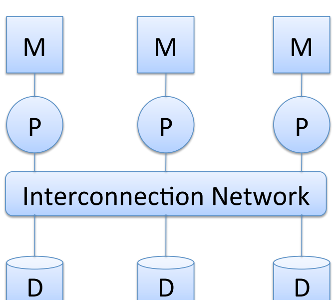
Architectures for Parallel Databases

- Shared memory
- Shared disk
- Shared nothing

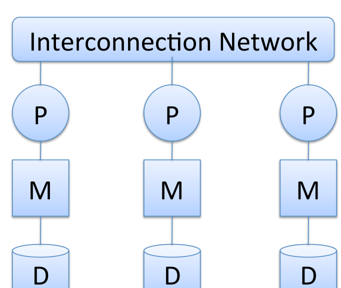
Shared Memory



Shared Disk



Shared Nothing



Shared Nothing

- Most scalable architecture
  - Minimizes interference by minimizing resource sharing
  - Can use commodity hardware
- Also most difficult to program and manage
- Processor = server = node
  - “Processor” != core
- P = number of nodes

We will focus on shared nothing

Question

- What exactly can we parallelize in a parallel DB ?

Taxonomy for Parallel Query Evaluation

- Inter-query parallelism
  - Each query runs on one processor
- Inter-operator parallelism
  - A query runs on multiple processors
  - An operator runs on one processor
- Intra-operator parallelism
  - An operator runs on multiple processors

Horizontal Data Partitioning

- Relation R split into P chunks R_0, …, R_{P-1}, stored at the P nodes
- Round robin: tuple t_i to chunk (i mod P)
- Hash based partitioning on attribute A:
  - Tuple t to chunk h(t.A) mod P
- Range based partitioning on attribute A:
  - Tuple t to chunk i if $v_{i-1} < t.A < v_i$

Horizontal Data Partitioning

- All three choices are just special cases:
  - For each tuple, compute *bin = f(t)*
  - Different properties of the function *f* determine hash vs. range vs. round robin vs. anything

Parallel Selection

Compute $\sigma_{A=v}(R)$, or $\sigma_{v1<A<v2}(R)$

- On a conventional database: cost = B(R)
- Q: What is the cost on a parallel database with P processors ?
  - Round robin
  - Hash partitioned
  - Range partitioned

Parallel Selection

- Q: What is the cost on a parallel database with P processors ?
- A: B(R) / P in all cases
- However, different processors do the work:
  - Round robin: all servers do the work
  - Hash: one server for $\sigma_{A=v}(R)$, all for $\sigma_{v1<A<v2}(R)$
  - Range: one server only

Data Partitioning Revisited

What are the pros and cons ?

- Round robin
  - Good load balance but always needs to read all the data
- Hash based partitioning
  - Good load balance but works only for equality predicates and full scans
- Range based partitioning
  - Works well for range predicates but can suffer from data skew

Parallel Group By: $\gamma_{A,\ sum(B)}(R)$

- Step 1: server i partitions chunk R_i using a hash function h(t.A) mod P: R_{i0}, R_{i1}, …, $R_{i,P-1}$
- Step 2: server i sends partition R_{ij} to serve j
- Step 3: server j computes $\gamma_{A,\ sum(B)}$ on R_{0j}, R_{1j}, …, $R_{P-1,j}$

Cost of Parallel Group By

Recall conventional cost = 3B(R)

- Cost of Step 1: B(R)/P I/O operations
- Cost of Step 2: (P-1)/P B(R) blocks are sent
  - Network costs assumed to be much lower than I/O
- Cost of Step 3: 2 B(R)/P
  - Why ?
  - When can we reduce it to 0 ?

Total = 3B(R) / P + communication costs

Parallel Group By: $\gamma_{A,\ sum(B)}(R)$

- Can we do better?
- Sum?
- Count?
- Avg?
- Max?
- Median?

Parallel Group By: $\gamma_{A,\ sum(B)}(R)$

- Sum(B) = Sum(B_0) + Sum(B_1) + … + Sum(B_n)
- Count(B) = Count(B_0) + Count(B_1) + … + Count(B_n)
- Max(B) = Max(Max(B_0) + Max(B_1) + … + Max(B_n))

distributive

- Avg(B) = Sum(B) / Count(B)

algebraic

- Median(B) =

holistic

Parallel Join: R $\bowtie_{A=B}$ S

- Step 1
  - For all servers in [0,k], server i partitions chunk R_i using a hash function h(t.A) mod P: $R_{i0}, R_{i1}, \ldots, R_{i,P-1}$
  - For all servers in [k+1,P], server j partitions chunk S_j using a hash function h(t.A) mod P: $S_{j0}, S_{j1}, \ldots, R_{j,P-1}$
- Step 2:
  - Server i sends partition R_{iu} to server u
  - Server j sends partition S_{ju} to server u
- Steps 3: Server u computes the join of R_{iu} with S_{ju}

Cost of Parallel Join

- Step 1: (B(R) + B(S))/P
- Step 2: 0
  - (P-1)/P (B(R) + B(S)) blocks are sent, but we assume network costs to be << disk I/O costs
- Step 3:
  - 0 if smaller table fits in main memory: B(S)/p <=M
  - 4(B(R)+B(S))/P otherwise

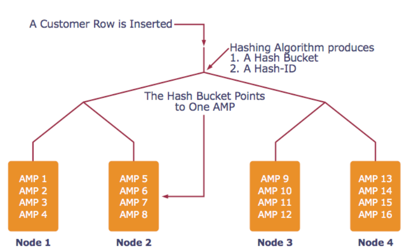
Parallel Dataflow Implementation

- Use relational operators unchanged
- Add special split and merge operators
  - Handle data routing, buffering, and flow control
- Example: exchange operator
  - Inserted between consecutive operators in the query plan
  - Can act as either a producer or consumer
  - Producer pulls data from operator and sends to n consumers
    - Producer acts as driver for operators below it in query plan
  - Consumer buffers input data from n producers and makes it available to operator through getNext interface

Shared Nothing Parallel Databases

- Teradata
- Greenplum
- Netezza
- Aster Data Systems
- ~~Datallegro~~ **Microsoft**
- Vertica
- MonetDB **Commercialized as Vectorwise**

Example System: Teradata

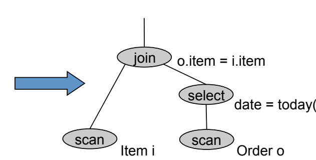


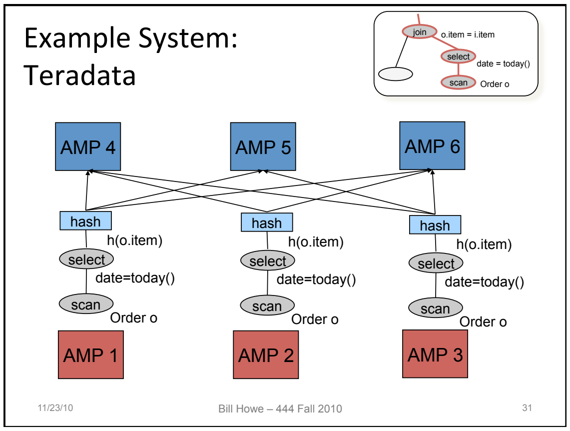
AMP = unit of parallelism

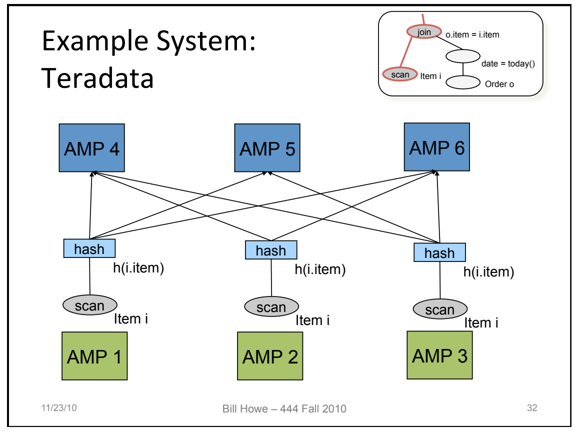
Example System: Teradata

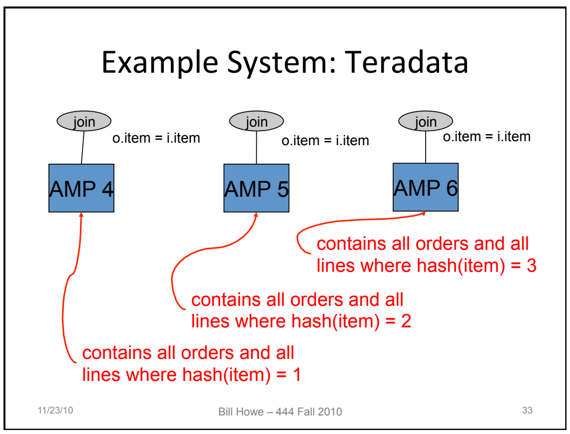
Find all orders from today, along with the items ordered

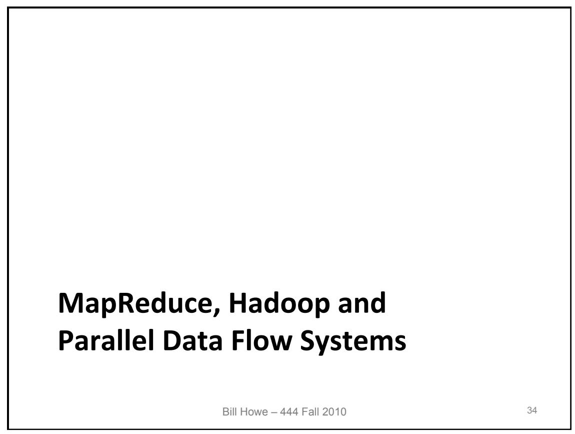
```
SELECT *
  FROM Orders o, Lines i
 WHERE o.item = i.item
   AND o.date = today()
```

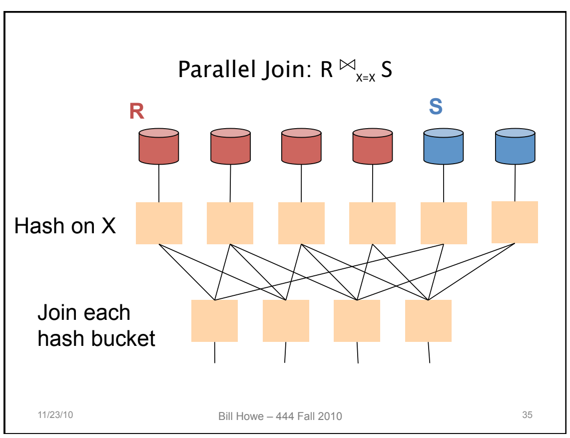


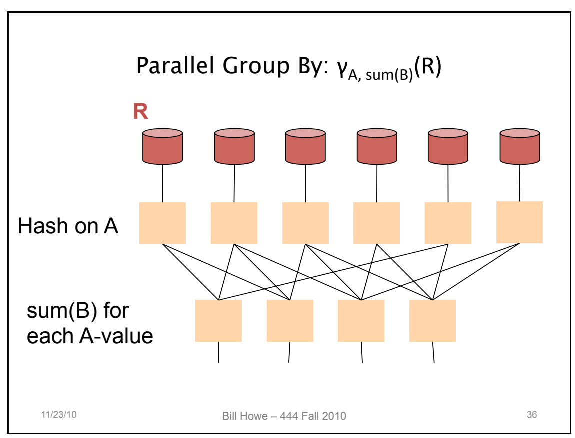

Example System:
Teradata
join
o.item = i.item
select
date = today()
scan
Order o
AMP 4
AMP 5
AMP 6
hash
h(o.item)
select
date=today()
scan
Order o
AMP 1
AMP 2
AMP 3
11/23/10
Bill Howe – 444 Fall 2010
31


Example System:
Teradata
join
o.item = i.item
date = today()
scan
Item i
Order o
AMP 4
AMP 5
AMP 6
hash
h(i.item)
scan
Item i
AMP 1
AMP 2
AMP 3
11/23/10
Bill Howe – 444 Fall 2010
32

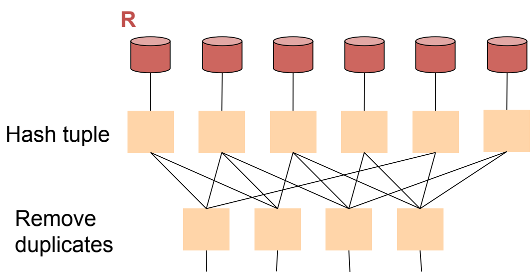

Example System: Teradata
join
o.item = i.item
AMP 4
AMP 5
AMP 6
contains all orders and all
lines where hash(item) = 3
contains all orders and all
lines where hash(item) = 2
contains all orders and all
lines where hash(item) = 1
11/23/10
Bill Howe – 444 Fall 2010
33


MapReduce, Hadoop and
Parallel Data Flow Systems
Bill Howe – 444 Fall 2010
34

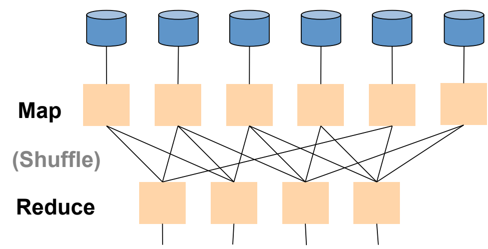

Parallel Join: R ⋈x=x S
R
S
Hash on X
Join each
hash bucket
11/23/10
Bill Howe – 444 Fall 2010
35


Parallel Group By: γA, sum(B)(R)
R
Hash on A
sum(B) for
each A-value
11/23/10
Bill Howe – 444 Fall 2010
36

Parallel Dupe-elim: δ(R)



Your favorite distributed algorithm…



MapReduce Programming Model

- Input & Output: each a set of key/value pairs
- Programmer specifies two functions:

map (in_key, in_value) -> list(out_key, intermediate_value)

  - Processes input key/value pair
  - Produces set of intermediate pairs

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)

Inspired by primitives from functional programming languages such as Lisp, Scheme, and Haskell

slide source: Google, Inc.

Example: What does this do?

```
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: word
  // output_values: ????
  int result = 0;
  for each v in intermediate_values:
     result += v;
  Emit(result);
```

slide source: Google, Inc.

Example: Document Processing

Abridged Declaration of Independence

A Declaration By the Representatives of the United States of America, in General Congress Assembled.
When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change.
We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying it's foundation on such principles and organizing it's power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.

Example: Word length histogram

Abridged Declaration of Independence

A Declaration By the Representatives of the United States of America, in General Congress Assembled.
When in the course of human events it becomes necessary for a people to advance from that subordination in which they have hitherto remained, and to assume among powers of the earth the equal and independent station to which the laws of nature and of nature's god entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the change.
We hold these truths to be self-evident; that all men are created equal and independent; that from that equal creation they derive rights inherent and inalienable, among which are the preservation of life, and liberty, and the pursuit of happiness; that to secure these ends, governments are instituted among men, deriving their just power from the consent of the governed; that whenever any form of government shall become destructive of these ends, it is the right of the people to alter or to abolish it, and to institute new government, laying it's foundation on such principles and organizing it's power in such form, as to them shall seem most likely to effect their safety and happiness. Prudence indeed will dictate that governments long established should not be changed for light and transient causes: and accordingly all experience hath shewn that mankind are more disposed to suffer while evils are sufferable, than to right themselves by abolishing the forms to which they are accustomed. But when a long train of abuses and usurpations, begun at a distinguished period, and pursuing invariably the same object, evinces a design to reduce them to arbitrary power, it is their right, it is their duty, to throw off such government and to provide new guards for future security. Such has been the patient sufferings of the colonies; and such is now the necessity which constrains them to expunge their former systems of government. the history of his present majesty is a history of unremitting injuries and usurpations, among which no one fact stands single or solitary to contradict the uniform tenor of the rest, all of which have in direct object the establishment of an absolute tyranny over these states. To prove this, let facts be submitted to a candid world, for the truth of which we pledge a faith yet unsullied by falsehood.

How many "big", "medium", and "small" words are used?

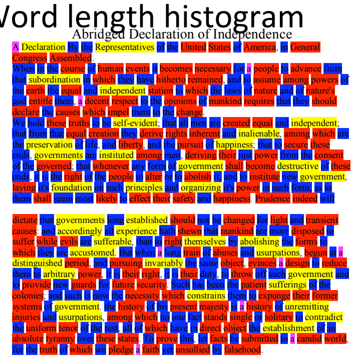
Example: Word length histogram

Big = Yellow = 10+ letters

Medium = Red = 5..9 letters

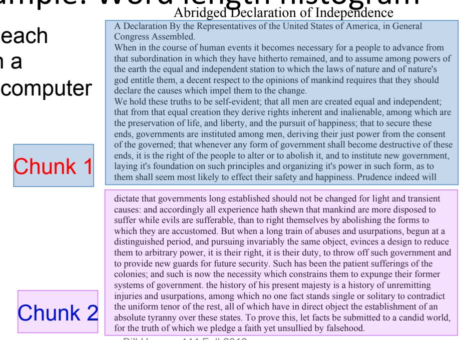
Small = Blue = 2..4 letters

Tiny = Pink = 1 letter

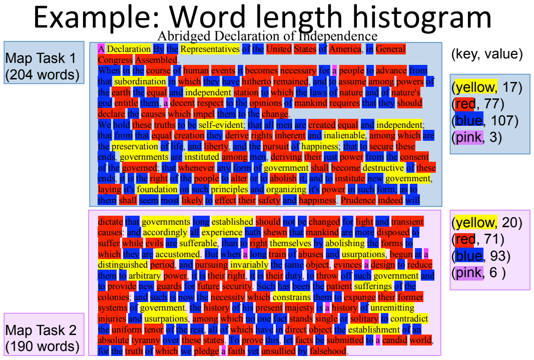


Example: Word length histogram

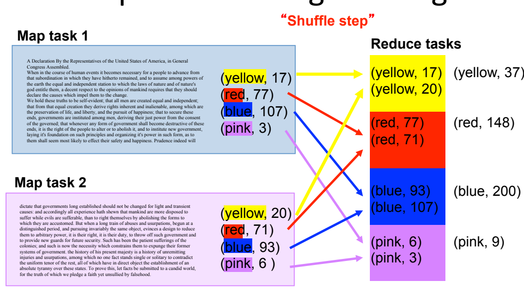
Process each chunk on a different computer



Example: Word length histogram



Example: Word length histogram



Map Reduce

- Google: [Dean 2004]
- Open source implementation: Hadoop

- Map-reduce = high-level programming model and implementation for large-scale parallel data processing

MapReduce Programming Model

- Input & Output: each a set of key/value pairs
- Programmer specifies two functions:

map (in_key, in_value) -> list(out_key, intermediate_value)

- Processes input key/value pair
- Produces set of intermediate pairs

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)

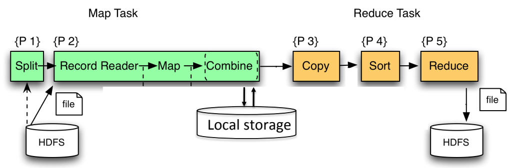
Inspired by primitives from functional programming languages such as Lisp, Scheme, and Haskell

slide source: Google, Inc.

Implementation

- There is one master node
- Master partitions input file into *M splits*, by key
- Master assigns *workers* (=servers) to the *M map tasks*, keeps track of their progress
- Workers write their output to local disk, partition into *R regions*
- Master assigns workers to the *R reduce tasks*
- Reduce workers read regions from the map workers' local disks

MR Phases



Interesting Implementation Details

- Worker failure:
  - Master pings workers periodically,
  - If down then reassigns its splits *to all other* workers → good load balance
- Choice of M and R:
  - Larger is better for load balancing
  - Limitation: master needs O(M×R) memory

Interesting Implementation Details

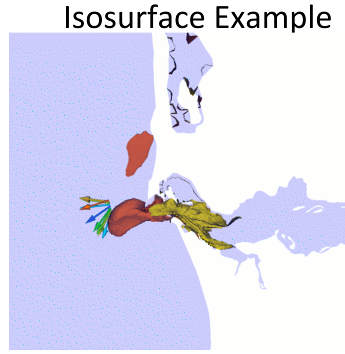
Backup tasks:

- *Straggler* = a machine that takes unusually long time to complete one of the last tasks. Eg:
  - Bad disk forces frequent correctable errors (30MB/s → 1MB/s)
  - The cluster scheduler has scheduled other tasks on that machine
- Stragglers are a main reason for slowdown
- Solution: *pre-emptive backup execution of the last few remaining in-progress tasks*

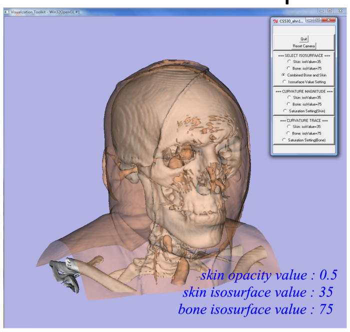
Map-Reduce Summary

- Hides scheduling and parallelization details
- However, very limited queries
  - Difficult to write more complex tasks
  - Need multiple map-reduce operations
- Solution:
  - Use MapReduce as a runtime for higher level languages
  - Pig (Yahoo!, now apache project): RA-like operators
  - Hive (Facebook, now apache project): SQL
  - Scope (MS): SQL ! But proprietary…
  - DryadLINQ (MS): LINQ ! But also proprietary…

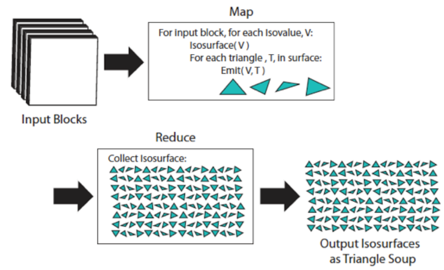
Isosurface Example



Isosurface Example

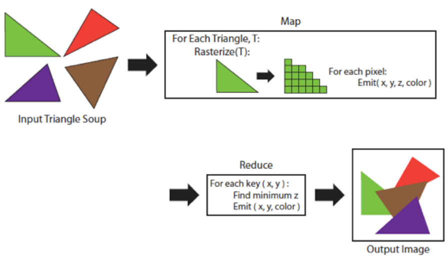


Example: Isosurface Extraction



Bronson et al. Vis 2010 (submitted)

Example: Rendering



Bronson et al. Vis 2010 (submitted)

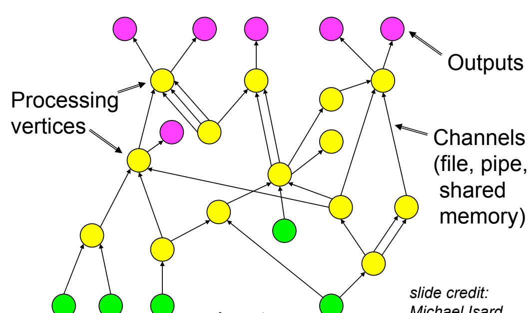
Why is MapReduce Successful?

- Easy
  - Democratization of parallel computing
  - Just two **serial** functions
  - Time to first query: a few hours (contrast with parallel DB…)
- Flexible
  - Schema-free, "In situ" processing
  - "First, load your data into the database…"
  - "First, convert your images to bitmaps…"
  - "First, encode your 3D mesh as triangle soup…"
- Fault-tolerance

What's wrong with MapReduce?

- Literally Map then Reduce and that's it…
  - Realistic jobs have multiple steps
- What else?

Realistic Job = Directed Acyclic Graph



slide credit: Michael Isard, MSR

MapReduce Contemporaries

- Dryad (Microsoft)
  - Relational Algebra
- Pig (Yahoo)
  - Near Relational Algebra over MapReduce
- HIVE (Facebook)
  - SQL over MapReduce
- Cascading
  - Relational Algebra
- Clustera
  - U of Wisconsin
- Hbase
  - Indexing on HDFS

MapReduce vs RDBMS

- RDBMS
  - Declarative query languages
  - Schemas
  - Logical Data Independence
  - Indexing
  - Algebraic Optimization
  - Caching/Materialized Views
  - *ACID/Transactions*
- MapReduce
  - High Scalability
  - Fault-tolerance
  - "One-person deployment"

DryadLINQ, Pig, HIVE

HIVE, Pig

Hbase

Pig, (Dryad, HIVE)

	Data Model	Prog. Model	Services
GPL	*	*	Typing (maybe)
Workflow	*	dataflow	typing, provenance, scheduling, caching, task parallelism, reuse
Relational Algebra	Relations	Select, Project, Join, Aggregate, …	optimization, physical data independence, data parallelism, indexing
MapReduce	[(key,value)]	Map, Reduce	massive data parallelism, fault tolerance
MS Dryad	IQueryable, IEnumerable	RA + Apply + Partitioning	typing, massive data parallelism, fault tolerance
MPI	Arrays/ Matrices	70+ ops	data parallelism, full control