Introduction to Database Systems
CSE 414

Lecture 28
Parallel Databases Wrap-up
Announcements

• Homework 8 (last) due on Friday night
  – Help each other out with configuration funnies

• Final exam Monday, 2:30
  – Review Sunday afternoon, 2:00
A Challenge

• Have P servers (say P=27 or P=1000)
• How do we compute this query?

\[ Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x) \]
A Challenge

• Have P servers (say P=27 or P=1000)
• How do we compute this query?
  \[ Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x) \]

• This computes all “triangles”.
• E.g. let \texttt{Follows}(x,y) be all pairs of Twitter users s.t. x follows y. Let \texttt{R=S=T=Follows}. Then \texttt{Q} computes all triples of people that follow each other.
A Challenge

• Have P servers (say P=27 or P=1000)
• How do we compute this query?
  \[ Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x) \]
• Step 1:
  – Each server sends \( R(x,y) \) to server \( h(y) \mod P \)
  – Each server sends \( S(y,z) \) to server \( h(y) \mod P \)
A Challenge

• Have P servers (say P=27 or P=1000)
• How do we compute this query?
  \[ Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x) \]

• Step 1:
  – Each server sends \( R(x,y) \) to server \( h(y) \) mod P
  – Each server sends \( S(y,z) \) to server \( h(y) \) mod P

• Step 2:
  – Each server computes \( R \bowtie S \) locally
  – Each server sends \([R(x,y) \bowtie S(y,z)]\) to \( h(x) \) mod P
  – Each server sends \( T(z,x) \) to \( h(x) \) mod P
A Challenge

• Have P servers (say P=27 or P=1000)
• How do we compute this query? 
\[ Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x) \]
• Step 1:
  – Each server sends \( R(x,y) \) to server \( h(y) \mod P \)
  – Each server sends \( S(y,z) \) to server \( h(y) \mod P \)
• Step 2:
  – Each server computes \( R \bowtie S \) locally
  – Each server sends \([R(x,y) \bowtie S(y,z)]\) to \( h(x) \mod P \)
  – Each server sends \( T(z,x) \) to \( h(x) \mod P \)
• Final output:
  – Each server computes locally and outputs \( R \bowtie S \bowtie T \)
A Challenge

• Have $P$ servers (say $P=27$ or $P=1000$)
• How do we compute this query in one step?
  \[ Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x) \]
A Challenge

• Have P servers (say P=27 or P=1000)
• How do we compute this query in one step?
  \[ Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x) \]
• Organize the P servers into a cube with side \( P^{\frac{1}{3}} \)
  – Thus, each server is uniquely identified by \((i,j,k)\), \(i,j,k \leq P^{\frac{1}{3}}\)
A Challenge

- Have P servers (say P=27 or P=1000)
- How do we compute this query in one step?
  \[ Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x) \]
- Organize the P servers into a cube with side \( P^{\frac{1}{3}} \)
  - Thus, each server is uniquely identified by \((i,j,k), i,j,k \leq P^{\frac{1}{3}}\)
- Step 1:
  - Each server sends \( R(x,y) \) to all servers \((h(x),h(y),\ast)\)
  - Each server sends \( S(y,z) \) to all servers \((\ast,h(y),h(z))\)
  - Each server sends \( T(x,z) \) to all servers \((h(x),\ast,h(z))\)
A Challenge

• Have P servers (say P=27 or P=1000)
• How do we compute this query **in one step?**
  \[ Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x) \]
• Organize the P servers into a cube with side \( P^{\frac{1}{3}} \)
  – Thus, each server is uniquely identified by \((i,j,k), i,j,k \leq P^{\frac{1}{3}}\)
• **Step 1:**
  – Each server sends \( R(x,y) \) to all servers \((h(x), h(y), *)\)
  – Each server sends \( S(y,z) \) to all servers \((*, h(y), h(z))\)
  – Each server sends \( T(x,z) \) to all servers \((h(x), *, h(z))\)
• **Final output:**
  – Each server \((i,j,k)\) computes the query \( R(x,y), S(y,z), T(z,x) \) locally
A Challenge

• Have P servers (say P=27 or P=1000)
• How do we compute this query in one step?
  \[ Q(x,y,z) = R(x,y) \bowtie S(y,z) \bowtie T(z,x) \]
• Organize the P servers into a cube with side \( P^{\frac{1}{3}} \)
  – Thus, each server is uniquely identified by \((i,j,k)\), \(i,j,k \leq P^{\frac{1}{3}}\)
• Step 1:
  – Each server sends \( R(x,y) \) to all servers \((h(x),h(y),\ast)\)
  – Each server sends \( S(y,z) \) to all servers \((\ast,h(y),h(z))\)
  – Each server sends \( T(x,z) \) to all servers \((h(x),\ast,h(z))\)
• Final output:
  – Each server \((i,j,k)\) computes the query \( R(x,y),S(y,z),T(z,x) \) locally
• Analysis: each tuple \( R(x,y) \) is replicated at most \( P^{\frac{1}{3}} \) times
Parallel DBs v.s. MapReduce

Parallel DB

• Plusses
  – Efficient binary format
  – Indexes, physical tuning
  – Cost-based optimization

• Minuses
  – Difficult to import data
  – Lots of baggage: logging, transactions

MapReduce

• Minuses
  – Lots of time spent parsing!
  – Text files
  – “Optimizers is between your eyes and your keyboard”

• Plusses
  – Any data
  – Lightweight, easy to speedup
  – Arguably more scalable
Example: Parallel DBMS vs. MR
1a. Parallel DBMS

R(a,b) is horizontally partitioned across N = 3 machines.

Each machine locally stores approximately 1/N of the tuples in R.

The tuples are randomly organized across machines (i.e., R is block partitioned across machines).

Show a RA plan for this query and how it will be executed across the N = 3 machines.
Pick an efficient plan that leverages the parallelism as much as possible.

SELECT a, max(b) as topb
FROM R
WHERE a > 0
GROUP BY a
R(a, b)

SELECT a, max(b) as topb
FROM R
WHERE a > 0
GROUP BY a
If more than one relation on a machine, then “scan S”, “scan R” etc

R(a, b)

```
SELECT a, max(b) as topb
FROM R
WHERE a > 0
GROUP BY a
```
SELECT a, max(b) as topb
FROM R
WHERE a > 0
GROUP BY a
SELECT a, max(b) as topb
FROM R
WHERE a > 0
GROUP BY a
SELECT a, max(b) as topb
FROM R
WHERE a > 0
GROUP BY a
SELECT a, max(b) as topb
FROM R WHERE a > 0 GROUP BY a

R(a, b)
R(a, b)

SELECT a, max(b) as topb
FROM R WHERE a > 0 GROUP BY a
1b. Map Reduce

Explain how the query will be executed in MapReduce (not PIG)

```
SELECT a, max(b) as topb
FROM R
WHERE a > 0
GROUP BY a
```

Specify the computation performed in the map and the reduce functions
Map

• Each map task
  – Scans a block of R
  – Calls the map function for each tuple
  – The map function applies the selection predicate to the tuple
  – For each tuple satisfying the selection, it outputs a record with key = a and value = b

• When each map task scans multiple relations, it needs to output something like
  key = a and value = (‘R’, b)
  which has the relation name ‘R’
Shuffle

- The MapReduce engine reshuffles the output of the map phase and groups it on the intermediate key, i.e. the attribute a

```sql
SELECT a, max(b) as topb
FROM R
WHERE a > 0
GROUP BY a
```
Reduce

- Each reduce task
  - computes the aggregate value \( \text{max}(b) = \text{topb} \) for each group (i.e. \( a \)) assigned to it (by calling the reduce function)
  - outputs the final results: \((a, \text{topb})\)

- A local combiner can be used to compute local max before data gets reshuffled (in the map tasks)

- Multiple aggregates can be output by the reduce phase like \( \text{key} = a \) and \( \text{value} = (\text{sum}(b), \text{min}(b)) \) etc.

- Sometimes a second (third etc) level of Map-Reduce phase might be needed
1c. Benefit of hash-partitioning

- What would change if we hash-partitioned R on R.a before executing this query
  - For parallel DBMS
  - For MapReduce
SELECT a, max(b) as topb
FROM R WHERE a > 0 GROUP BY a
1c. Benefit of hash-partitioning

- For parallel DBMS
  - It would avoid the data re-shuffling phase
  - It would compute the aggregates locally

SELECT a, max(b) as topb FROM R WHERE a > 0 GROUP BY a
Hash-partition on a for R(a, b)

SELECT a, max(b) as topb
FROM R WHERE a > 0 GROUP BY a
1c. Benefit of hash-partitioning

• **For MapReduce**
  – Logically, MR won’t know that the data is hash-partitioned
  – MR treats map and reduce functions as black-boxes and
don’t perform any optimizations on them

• But, if a local combiner is used
  – Saves communication cost:
    • fewer tuples will be emitted by the map tasks
  – Saves computation cost in the reducers:
    • the reducers would not have to do anything (if one map task/node) or less computation (multiple map tasks/node)

```
SELECT a, max(b) as topb
FROM R WHERE a > GROUP BY a
```