Introduction to Data Management
CSE 344

Lecture 23: Parallel DBMSs
Announcements

• WQ7 due tonight
• HW7 due on Wednesday
Final Exam

• Thursday 3/16, 2:30-4:20pm
  – Location: Here!

• You can bring two letter-sized sheets of notes
  – You can write on both sides
  – You can type / handwriting / print etc

• Exam will be comprehensive
  – Includes all lectures, readings, sections, HWs, WQs

• Final review session this Saturday 3/11
  – EEB 105, 1-2pm
Welcome to the 2nd half of 344

• Relational data model
  – Instance
  – Schema
  – Query languages
    • SQL, RA, RC, Datalog

• Query processing
  – Logical & physical plans
  – Indexes
  – Cost estimation
  – Query optimization

• Non-relational data model

• Conceptual design
  – E/R diagrams
  – Converting to SQL
  – Normalization

• Transactions
  – ACID
  – Transaction Implementation
  – Writing DB applications

• Parallel query processing
  – MapReduce
  – Spark
Today

• Architecture of parallel DBMSs

• Distributing data to multiple machines

• Executing relational query operators in parallel

• Alternative data models for parallel DBMSs
Why compute in parallel?

• Multi-cores:
  – Most processors have multiple cores
  – This trend will increase in the future

• Big data: too large to fit in main memory
  – Distributed query processing on 100x-1000x servers
  – Widely available now using cloud services
Performance Metrics for Parallel DBMSs

Nodes = processors, computers

• **Speedup:**
  – More nodes, same data $\Rightarrow$ higher speed

• **Scaleup:**
  – More nodes, more data $\Rightarrow$ same speed
Linear v.s. Non-linear Speedup
Linear v.s. Non-linear Scaleup

Batch Scaleup

# nodes (=P) AND data size

Ideal

×1 ×5 ×10 ×15
Why Sub-linear Speedup and Scaleup?

• **Startup cost**
  – Cost of starting an operation on many nodes

• **Interference**
  – Contention for resources between nodes

• **Skew**
  – Slowest node becomes the bottleneck
Architectures for Parallel Databases

- Shared memory
- Shared disk
- Shared nothing
Shared Memory

- Nodes share both RAM and disk
- Dozens to hundreds of processors

Example: SQL Server runs on a single machine and can leverage many threads to speed up a query
- check your HW3 query plans

- Easy to use and program
- Expensive to scale
  - last remaining cash cows in the hardware industry
**Shared Disk**

- All nodes access the same disks
- Found in the largest "single-box" (non-cluster) multiprocessors

Example: Oracle

- No need to worry about shared memory
- Hard to scale: existing deployments typically have fewer than 10 machines
Shared Nothing

- Cluster of commodity machines on high-speed network
- Called "clusters" or "blade servers"
- Each machine has its own memory and disk: lowest contention.

Example: Google

Because all machines today have many cores and many disks, shared-nothing systems typically run many "nodes" on a single physical machine.

- Easy to maintain and scale
- Most difficult to administer and tune.

We discuss only Shared Nothing in class
Parallel Data Processing @ 1990
Approaches to Parallel Query Evaluation

- **Inter-query parallelism**
  - Transaction per node
  - Good for transactional workloads

- **Inter-operator parallelism**
  - Operator per node
  - Good for analytical workloads

- **Intra-operator parallelism**
  - Operator on multiple nodes
  - Good for both?

We study only intra-operator parallelism: most scalable
Single Node Query Processing (Review)

Given relations $R(A,B)$ and $S(B, C)$, **no indexes**:

- **Selection**: $\sigma_{A=123}(R)$
  - Scan file $R$, select records with $A=123$

- **Group-by**: $\gamma_{A,\text{sum}(B)}(R)$
  - Scan file $R$, insert into a hash table using $A$ as key
  - When a new key is equal to an existing one, add $B$ to the value

- **Join**: $R \bowtie S$
  - Scan file $S$, insert into a hash table using $B$ as key
  - Scan file $R$, probe the hash table using $B$
Distributed Query Processing

- Data is horizontally partitioned on many servers
- Operators may require data reshuffling
- First let’s discuss how to distribute data across multiple nodes / servers
Horizontal Data Partitioning

Data:

<table>
<thead>
<tr>
<th>K</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
</table>
| ... | ... | ...

Servers:

1  2  ...  P
Horizontal Data Partitioning

Data:

<table>
<thead>
<tr>
<th>K</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Servers:

1

<table>
<thead>
<tr>
<th>K</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2

<table>
<thead>
<tr>
<th>K</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... ... ...

P

<table>
<thead>
<tr>
<th>K</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Which tuples go to what server?
Horizontal Data Partitioning

- **Block Partition:**
  - Partition tuples arbitrarily s.t. \( \text{size}(R_1) \approx \ldots \approx \text{size}(R_P) \)

- **Hash partitioned on attribute A:**
  - Tuple \( t \) goes to chunk \( i \), where \( i = h(t.A) \mod P + 1 \)
  - Recall: calling hash fn’s is free in this class

- **Range partitioned on attribute A:**
  - Partition the range of A into \( -\infty = v_0 < v_1 < \ldots < v_P = \infty \)
  - Tuple \( t \) goes to chunk \( i \), if \( v_{i-1} < t.A < v_i \)
Uniform Data v.s. Skewed Data

Let $R(K,A,B,C)$; which of the following partition methods may result in skewed partitions?

- Block partition
- Hash-partition
  - On the key $K$
  - On the attribute $A$

E.g. when all records have the same value of the attribute $A$, then all records end up in the same partition.

Assuming good hash function:

Uniform

May be skewed

Keep this in mind in the next few slides.
Parallel Execution of RA Operators: Grouping

Data: \( R(K,A,B,C) \)

Query: \( \gamma_{A,\text{sum}(C)}(R) \)

How to compute group by if:

- \( R \) is hash-partitioned on \( A \)?
- \( R \) is block-partitioned?
- \( R \) is hash-partitioned on \( K \)?
Parallel Execution of RA Operators: Grouping

Data: $R(K, A, B, C)$
Query: $\gamma_{A, \text{sum}(C)}(R)$

- $R$ is block-partitioned or hash-partitioned on $K$

### Diagram

- Reshuffle $R$ on attribute $A$
- Run grouping on reshuffled partitions

### Partitions

- $R_1$, $R_2$, $\ldots$, $R_P$
- $R_1'$, $R_2'$, $\ldots$, $R_P'$

- $A = 1$
- $A = 20$
- $A = 100$
Speedup and Scaleup

• Consider:
  – Query: $\gamma_{A,\text{sum}(C)}(R)$
  – Runtime: only consider I/O costs

• If we double the number of nodes $P$, what is the new running time?
  – Half (each server holds $\frac{1}{2}$ as many chunks)

• If we double both $P$ and the size of $R$, what is the new running time?
  – Same (each server holds the same # of chunks)
Parallel Execution of RA Operators: Partitioned Hash-Join

- **Data**: \( R(K_1, A, B), S(K_2, B, C) \)
- **Query**: \( R(K_1, A, B) \bowtie S(K_2, B, C) \)
  - Initially, both \( R \) and \( S \) are partitioned on \( K_1 \) and \( K_2 \)

Each server computes the join locally

R'\(_1\), S'\(_1\)  R'\(_2\), S'\(_2\)  ...  R'\(_P\), S'\(_P\)

Reshuffle \( R \) on \( R.B \) and \( S \) on \( S.B \)
Data: $R(K_1, A, B), S(K_2, B, C)$
Query: $R(K_1, A, B) \bowtie S(K_2, B, C)$

**Parallel Join Illustration**

<table>
<thead>
<tr>
<th>$R_1$</th>
<th>$S_1$</th>
<th>$R_2$</th>
<th>$S_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_1$</td>
<td>$B$</td>
<td>$K_1$</td>
<td>$B$</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>4</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$M_1$</th>
<th>$M_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_1$</td>
<td>$B$</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>$K_2$</td>
<td>$B$</td>
</tr>
<tr>
<td>101</td>
<td>50</td>
</tr>
<tr>
<td>102</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$R_1'$</th>
<th>$S_1'$</th>
<th>$R_2'$</th>
<th>$S_2'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_1$</td>
<td>$B$</td>
<td>$K_1$</td>
<td>$B$</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>2</td>
<td>201</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$M_1$</th>
<th>$M_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_2$</td>
<td>$B$</td>
</tr>
<tr>
<td>201</td>
<td>20</td>
</tr>
<tr>
<td>101</td>
<td>50</td>
</tr>
<tr>
<td>102</td>
<td>50</td>
</tr>
<tr>
<td>202</td>
<td>50</td>
</tr>
</tbody>
</table>