Introduction to Database Systems
CSE 414

Lecture 16: Query Evaluation
Announcements

• HW5 + WQ5 due tomorrow

• Midterm this Friday in class!
  – Review session this Wednesday evening
  – See course website

• HW6 will be released later this week
  – Due on Friday 5/11
  – No WQ6 (yet)!
Class Overview

- Unit 1: Intro
- Unit 2: Relational Data Models and Query Languages
- Unit 3: Non-relational data
- Unit 4: RDMBS internals and parallel query processing
- Unit 5: DBMS usability, conceptual design
- Unit 6: Transactions
- Unit 7: Advanced topics
From Logical RA Plans to Physical Plans
Query Evaluation Steps Review

- Parse Query
- Generate Logical Plan
- Select Physical Plan
- Query Execution

SQL query

Logical plan (RA)

Physical plan
Logical vs Physical Plans

• Logical plans:
  – Created by the parser from the input SQL text
  – Expressed as a relational algebra tree
  – Each SQL query has many possible logical plans

• Physical plans:
  – Goal is to choose an efficient implementation for each operator in the RA tree
  – Each logical plan has many possible physical plans
Review: Relational Algebra

Relational algebra expression is also called the “logical query plan”

SELECT sname
FROM Supplier x, Supply y
WHERE x.sid = y.sid
    and y.pno = 2
    and x.scity = 'Seattle'
    and x.sstate = 'WA'
Physical Query Plan 1

A physical query plan is a logical query plan annotated with physical implementation details.

\[
\begin{align*}
\text{SELECT} & \quad \text{name} \\
\text{FROM} & \quad \text{Supplier} \ x, \ \text{Supply} \ y \\
\text{WHERE} & \quad x.\text{sid} = y.\text{sid} \\
& \quad \text{and} \ y.\text{pno} = 2 \\
& \quad \text{and} \ x.\text{scity} = 'Seattle' \\
& \quad \text{and} \ x.\text{sstate} = 'WA'
\end{align*}
\]

Supplier \ (\text{sid}, \ \text{sname}, \ \text{scity}, \ \text{sstate})

Supply \ (\text{sid}, \ \text{pno}, \ \text{quantity})
Physical Query Plan 2

\[
\begin{align*}
    \text{SELECT } & \text{sname} \\
    \text{FROM } & \text{Supplier x, Supply y} \\
    \text{WHERE } & x.\text{sid} = y.\text{sid} \\
    & \text{and } y.\text{pno} = 2 \\
    & \text{and } x.\text{scity} = 'Seattle' \\
    & \text{and } x.\text{sstate} = 'WA'
\end{align*}
\]

Supplier\((\text{sid}, \text{sname}, \text{scity}, \text{sstate})\)

Supply\((\text{sid}, \text{pno}, \text{quantity})\)

(On the fly) \hspace{2cm} \pi_{\text{sname}}

(On the fly) \hspace{2cm} \sigma_{\text{scity}= 'Seattle' \text{ and sstate}= 'WA' \text{ and pno}=2}

(Hash join) \hspace{2cm} \text{sid} = \text{sid}

Same logical query plan
Different physical plan
Physical Query Plan 3

(a) $\sigma_{\text{scity}=\text{Seattle \ 'WA'}}$

(b) $\sigma_{\text{pno}=2}$  (Scan & write to T2)

(c) $\sigma_{\text{sid}=\text{sid}}$

(d) $\pi_{\text{sname}}$

Different but equivalent logical query plan; different physical plan

SELECT sname
FROM Supplier $x$, Supply $y$
WHERE $x$.sid = $y$.sid
    and $y$.pno = 2
    and $x$.scity = 'Seattle'
    and $x$.sstate = 'WA'

CSE 414 - Spring 2018
Query Optimization Problem

• For each SQL query… many logical plans

• For each logical plan… many physical plans

• Choosing the best one among them is the goal of query optimization

• More on this later in the quarter
Distributed query processing
Why compute in parallel?

• Multi-cores:
  – Most processors have multiple cores
  – This trend will likely 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
  – Recall HW3 and motivation for NoSQL!
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

Speedup

Ideal

# nodes (=P)

×1  ×5  ×10  ×15
Linear v.s. Non-linear Scaleup

Batch Scaleup

# nodes (=P) AND data size

Ideal

CSE 414 - Spring 2018
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
Approaches to Parallel Query Evaluation

- **Inter-query parallelism**
  - One query per node
  - Good for transactional (OLTP) workloads

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

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

We study only intra-operator parallelism: most scalable
Parallel Data Processing in the 20th Century
Let’s parallelize RDBMS

- 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>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Servers:

1 2 . . . P
Horizontal Data Partitioning

Data:

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

Servers:

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

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

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

Which tuples go to what server?
Recall: 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

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

Data: R(K,A,B,C)
Query: γ_{A,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 $K$

Reshuffle $R$ on attribute $A$

Run grouping on reshuffled partitions
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)

But only if the data is without skew!
Skewed Data

- $R(K, A, B, C)$
- Informally: we say that the data is skewed if one server holds much more data than the average
- E.g., we hash-partition on $A$, and some value of $A$ occurs very many times (“Justin Bieber”)
- Then the server holding that value will be skewed
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.
Parallel Join Illustration

Data: $R(K_1,A,B)$, $S(K_2,B,C)$
Query: $R(K_1,A,B) \bowtie S(K_2,B,C)$
Data: R(A, B), S(C, D)
Query: R(A,B) \bowtie_{B=C} S(C,D)

Broadcast Join

Reshuffle R on R.B

Broadcast S

Why would you want to do this?

R_1

R_2

\ldots

R_P

R'_1, S

R'_2, S

\ldots

R'_P, S

S
Putting it Together: Example Parallel Query Plan

Find all orders from today, along with the items ordered

```
SELECT *
FROM Order o, Line i
WHERE o.item = i.item
AND o.date = today()
```
Example Parallel Query Plan

Order(\textit{oid}, \textit{item}, \textit{date}), Line(\textit{item}, \ldots)
Example Parallel Query Plan

Order(oid, item, date), Line(item, …)
Order(oid, item, date), Line(item, ...)

Example Parallel Query Plan

Node 1
join
o.item = i.item

Node 2
join
o.item = i.item

Node 3
join
o.item = i.item

contains all orders and all lines where hash(item) = 1
contains all orders and all lines where hash(item) = 2
contains all orders and all lines where hash(item) = 3