CS 347
Parallel and Distributed Data Processing
Spring 2016

Notes 4: Query Optimization
Query Optimization

Cost estimation
Strategies for exploring plans
Cost Estimation

Based on estimating **result sizes**

Like in centralized databases
Cost Estimation

But # of IOs may not be the best metric

E.g., transmission time may dominate
Cost Estimation

Another reason why # of IOs is not enough: **parallelism**

**Plan A**
- 100 IOs

**Plan B**
- site 1: 50 IOs
- site 2: 70 IOs
- site 3: 50 IOs
Cost Estimation

Cost metrics

*E.g., IOs, bytes transmitted, $, ...*

Additive

Response time metric

Not additive

Need scheduling and dependency info

Skew is important
Cost Estimation

Also take into account

Start up cost
Data distribution cost/time
Resource contention (for memory, disk, network)
Cost of assembling results
Cost Estimation

Response time example

- site 1
- site 2
- site 3
- site 4

- start up
- distribution
- searching + sending results
- final processing
Search Strategies

Exhaustive (with pruning)
Hill climbing (greedy)
Query separation
Exhaustive Search

Consider *all* query plans (given a set of techniques for operators)
Prune some plans
Heuristics
Exhaustive Search

Example

\[ R \bowtie A S \bowtie B T \]

\[ |R| > |S| > |T| \]

Prune because cross-product not necessary

Prune because larger relation first
Search Strategies

In generating plans, keep goal in mind

E.g., if goal is parallelism (in system with fast network)
→ Consider partitioning relations first

E.g., if goal is reduction of network traffic
→ Consider semi-joins
Hill Climbing

Better plans

1

x ---- initial plan

Worse plans
Hill Climbing

Better plans

Worse plans

initial plan
Hill Climbing

Example

<table>
<thead>
<tr>
<th>relation</th>
<th>site</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>S</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>V</td>
<td>4</td>
<td>40</td>
</tr>
</tbody>
</table>

tuple size = 1

Goal: minimize data transmission
Hill Climbing

Initial plan
Send relations to one site

What site do we send all relations to?

To site 1: cost = 20 + 30 + 40 = 90
To site 2: cost = 10 + 30 + 40 = 80
To site 3: cost = 10 + 20 + 40 = 70
To site 4: cost = 10 + 20 + 30 = 60 ✔
Hill Climbing

$P_0$

R (1 → 4)
S (2 → 4)
T (3 → 4)

Compute $R \bowtie S \bowtie T \bowtie V$ at site 4
Hill Climbing

Local search
Consider sending each relation to neighbor
Hill Climbing

Assume

\[
\begin{align*}
\text{size } R \bowtie S &= 20 \\
\text{size } S \bowtie T &= 5 \\
\text{size } T \bowtie V &= 1
\end{align*}
\]

Option A

\[
\begin{array}{c}
\text{cost } = 30 \\
\text{cost } = 30
\end{array}
\]

No savings
Hill Climbing

Option B

cost = 30

Worse off
Hill Climbing

Option C

cost = 50

Win

cost = 35
Hill Climbing

Option D

![Diagram of hill climbing process with nodes and edges labeled with costs. The diagram shows a path from 2 to 3 with a cost of 50, and another path from 2 to 3 with a cost of 25, indicating a bigger win.](image)
Hill Climbing

$P_1$

$S \ (2 \to \ 3)$

$\alpha = S \Join T$

$R \ (1 \to \ 4)$

$T \ (3 \to \ 4)$

Compute answer at site 4
Hill Climbing

Repeat local search
Treat $\alpha = S \bowtie T$ as relation
Hill Climbing

Hill climbing may miss best plan
E.g., best plan could be

\[ P_{\text{best}} \]

\[ T (3 \rightarrow 4) \]
\[ \beta = T \bowtie V \]
\[ \beta (4 \rightarrow 2) \]
\[ \beta' = \beta \bowtie S \]
\[ \beta' (2 \rightarrow 1) \]
\[ \beta'' = \beta' \bowtie R \]
\[ \beta'' (1 \rightarrow 4) \text{ (optional)} \]

Compute answer
Hill Climbing

Hill climbing may miss best plan

E.g., best plan could be

$$P_{\text{best}}$$

- $$T \ (3 \rightarrow 4) = 30$$
- $$\beta = T \bowtie V$$
- $$\beta \ (4 \rightarrow 2) = 1$$
- $$\beta' = \beta \bowtie S$$
- $$\beta' \ (2 \rightarrow 1) = 1$$
- $$\beta'' = \beta' \bowtie R$$
- $$\beta'' \ (1 \rightarrow 4) \ (\text{optional}) = 1$$

Costs could be low because $$\beta$$ is very selective

Compute answer = 33 total
Search Strategies

Exhaustive (with pruning) ✔
Hill climbing (greedy) ✔
Query separation
Query Separation

Separate query into 2 or more steps
Optimize each step independently
Query Separation

Example
Simple queries technique
Query Separation

1. Compute
   \[ R' = \prod_A [ \sigma_{c2} R ] \]
   \[ S' = \prod_A [ \sigma_{c3} S ] \]

2. Compute
   \[ J = R' \bowtie S' \]

3. Compute answer
   \[ \sigma_{c1} \{ [ J \bowtie \sigma_{c2} R ] \bowtie [ J \bowtie \sigma_{c3} S ] \} \]
Query Separation

1. Compute
   \[ R' = \prod_A [ \sigma_{c2} R ] \]
   \[ S' = \prod_A [ \sigma_{c3} S ] \]

2. Compute
   \[ J = R' \bowtie S' \]

3. Compute answer
   \[ \sigma_{c1} \{ [ J \bowtie \sigma_{c2} R ] \bowtie [ J \bowtie \sigma_{c3} S ] \} \]

Compute the \( A \) values in the answer first

Get tuples from sites matching \( A \) and compute answer next
Query Separation

Simple query

Relations have a single attribute

Output has a single attribute

E.g., $J = R' \bowtie S'$
Query Separation

Idea

1. **Decompose** query
   - Local processing
   - Simple query (or queries)
   - Final processing

2. **Optimize** simple query
Query Separation

Philosophy

Hard part is distributed join

→ Do this part with only keys; get the rest of the data later

Simpler to optimize simple queries
Summary

Cost estimation

Optimization strategies
- Exhaustive (with pruning)
- Hill climbing (greedy)
- Query separation
Words of Wisdom

*Optimization is like chess playing*

May have to make sacrifices for later gains

Move data, partition relations, build indexes