Cluster-Based Join Algorithms

Basic Join Algorithm
More Efficient Joins Via Replication
Optimization of Multiway Joins
Data-Volume Cost

- *Data-volume cost* = sum of sizes of inputs to all tasks of an algorithm.
- Assumes transport between compute nodes is significant.
- Also assumes computation by a task is linear in input size or negligible compared with communication.
Why Not Count Output Size?

- Outputs of one task are inputs to at least one other, or are algorithm output.
- Algorithm outputs tend to be small because users can’t make use of too much information.
Joins

The Natural Join
Joining by Map-Reduce
3-Way Joins
Natural Join of Relations

**Given**: a collection of relations, each with attributes labeling their columns.

**Find**: Those tuples over all the attributes such that when restricted to the attributes of any relation $R$, that tuple is in $R$. 
**Example: Natural Join**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

The join:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
Joining by Map-Reduce

- Suppose we want to compute \( R \) \((A,B) \) JOIN \( S(B,C)\), using \( k \) Reduce tasks.
  - I.e., find tuples with matching B-values.
- \( R \) and \( S \) are each stored in a chunked file.
Joining by Map-Reduce – (2)

- Use a hash function $h$ from B-values to $k$ buckets.
  - Bucket = Reduce task.
- The Map tasks take chunks from R and S, and send:
  - Tuple $R(a,b)$ to Reduce task $h(b)$.
    - Key = $b$ value = $R(a,b)$.
  - Tuple $S(b,c)$ to Reduce task $h(b)$.
    - Key = $b$; value = $S(b,c)$.
Joining by Map-Reduce – (3)

Map tasks send
R(a,b) if h(b) = i

Reduce task i

Map tasks send
S(b,c) if h(b) = i

All (a,b,c) such that h(b) = i, and (a,b) is in R, and (b,c) is in S.
Joining by Map-Reduce – (4)

- **Key point:** If R(a,b) joins with S(b,c), then both tuples are sent to Reduce task h(b).
- Thus, their join (a,b,c) will be produced there and shipped to the output file.
3-Way Join

- Consider a chain of three relations:
  \[ R(A, B) \text{ JOIN } S(B, C) \text{ JOIN } T(C, D) \]
- **Example**: R, S, and T are “friends” relations.
- We could join any two by the 2-way map-reduce algorithm shown, then join the third with the resulting relation.
- But intermediate joins are large.
3-Way Join – (2)

- An alternative is to divide the work among $k = m^2$ Reduce tasks.
- Hash both B and C to $m$ values.
- A Reduce task corresponds to a hashed B-value and a hashed C-value.
3-Way Join – (3)

◆ Each S-tuple S(b,c) is sent to one Reduce task: (h(b), h(c)).
◆ But each tuple R(a,b) must be sent to $m$ Reduce tasks (h(b), x).
◆ And each tuple T(c,d) must be sent to $m$ Reduce tasks (y, h(c)).
Example: $m = 4; k = 16$. 

\[
\begin{array}{c|cccc}
  & 0 & 1 & 2 & 3 \\
\hline
h(b)=0 & & & & \\
\hline
h(b)=1 & & & & \\
\hline
h(b)=2 & & & & \\
\hline
h(b)=3 & & & & \\
\end{array}
\]
3-Way Join – (4)

Thus, any joining tuples R(a,b), S(b,c), and T(c,d) will be joined at the Reduce task \((h(b), h(c))\).

Data-volume cost: \(s + mr + mt\).

Convention: Lower-case letter is the size of the relation whose name is the corresponding upper-case letter.

Example: \(r\) is the size of R.
Comparison of Methods

◆ Suppose for simplicity that:
  ▶ Relations R, S, and T have the same size \( r \).
  ▶ The probability of two tuples joining is \( p \).

◆ The 3-way join has cost \( r(2m+1) \).

◆ Two two-way joins have a cost of:
  ▶ 3r to read the relations, plus
  ▶ \( pr^2 \) to read the join of the first two.
  ▶ **Total** = \( r(3+pr) \).
Comparison – (2)

◆ 3-way beats 2-way if $2m+1 < 3+pr$.

◆ $pr$ is the multiplicity of each join.
  ▪ Thus, the 3-way chain-join is useful when the multiplicity is high.

◆ Example: relations are “friends”; $pr$ is about 300. $m^2 = k$ can be 20,000.

◆ Example: relations are Web links; $pr$ is about 15. $m^2 = k$ can be 64.
Optimization of Multway Joins

Share Variables and Their Optimization

Special Case: Star Joins
Some Questions

♦ When we discussed the 3-way chain-join $R(A, B) \ JOIN \ S(B, C) \ JOIN \ T(C,D)$, we used attributes B and C for the map-key (index for the Reduce tasks).

♦ Why not include A and/or D?

♦ Why use the same number of buckets for B and C?
Share Variables

- For the general problem, we use a share variable for each attribute.
  - The number of buckets into which values of that attribute are hashed.

- Convention: The share variable for an attribute is the corresponding lowercase letter.
  - Example: the share variable for attribute A is always a.
The product of all the share variables is $k$, the number of Reduce tasks.

The data-volume cost for a multiway join is the sum of: the size of each relation times the product of the share variables for the attributes that do not appear in the schema of that relation.
Example: Minimizing Cost

- Consider the cyclic join $R(A, B) \ JOIN S(B, C) \ JOIN T(A, C)$
- Cost function is $rc + sa + tb$.
- Construct the Lagrangean:
  $$rc + sa + tb - \lambda(abc - k)$$
- Take derivative wrt each share variable, then multiply by that variable.
  - Result is 0 at minimum.
Example – Continued

- $d/da$ of $rc + sa + tb - \lambda(abc - k)$ is $s - \lambda bc$.
- Multiply by $a$ and set to 0: $sa - \lambda abc = 0$.
- **Note**: $abc = k$: $sa = \lambda k$.
- Similarly, $d/db$ and $d/dc$ give: $sa = tb = rc = \lambda k$.
- **Solution**: $a = (krt / s^2)^{1/3}; b = (krs / t^2)^{1/3}; c = (kst / r^2)^{1/3}$.
- **Cost**: $rc + sa + tb = 3(krst)^{1/3}$. 
Dominated Attributes

- Certain attributes can’t be in the map-key.
- A *dominates* B if every relation of the join with B also has A.
- Example:

  \[
  R(A,B,C) \JOIN S(A,B,D) \JOIN T(A,E) \JOIN U(C,E)
  \]

  Every relation with B also has A.
Example – (2)

R(A,B,C) JOIN S(A,B,D) JOIN T(A,E) JOIN U(C,E)

- Cost expression:
  \[ rde + sce + tbcd + uabd \]

- Since \( b \) appears wherever \( a \) does, if there were a minimum-cost solution with \( b > 1 \), we could replace \( b \) by 1 and \( a \) by \( ab \), and the cost would lower.
Dominated Attributes – Continued

This rule explains why, in the discussion of the chain join

\[ R(A, B) \text{ JOIN } S(B, C) \text{ JOIN } T(C, D) \]

we did not give dominated attributes A and D a share.
Solving the General Case

Unfortunately, there are more complex cases than dominated attributes, where the equations derived from the Lagrangean imply a positive sum of several terms = 0.

We can fix, generalizing dominated attributes, but we have to branch on which attribute needs to be eliminated from the map-key.
Solving – (2)

◆ Solutions not in integers:
   ▶ Drop an attribute with a share < 1 from the map-key and re-solve.
   ▶ Round other nonintegers, and treat $k$ as a suggestion, since the product of the integers may not be $k$. 
Special Case: Star Joins

- A *star join* combines a huge *fact table* $F(A_1,A_2,...,A_n)$ with large but smaller *dimension tables* $D_1(A_1,B_1), D_2(A_2,B_2), ..., D_n(A_n,B_n)$.
  - There may be other attributes not shown, each belonging to only one relation.
- **Example**: Facts = sales; dimensions tell about buyer, product, etc.
Star-Join Pattern
Star Joins – (2)

- Map-key = the A’s.
  - B’s are dominated.
- Solution: \( \frac{d_i}{a_i} = \lambda k \) for all \( i \).
  - That is, the shares are proportional to the dimension-table sizes.
Cool Application of Star Join Result

- Fact/dimension tables are often used for analytics.
- **Aster Data approach**: partition *(shard)* fact table among compute nodes permanently; replicate needed pieces of dimension tables.
  - They use a data-dependent way to cluster facts so replication of dimension tables is minimized.
Example: Shard by Product

But all Customer tuples for customers who bought shirts, or *might* buy shirts go here.
Our solution lets you partition the fact table to $k$ nodes in a data-independent way.

- Avoids the need to reshard the fact table.

- Replication of tuples in the dimension tables is minimized.
Summary

1. Multiway joins can be computed by replicating tuples and distributing them to many compute nodes.
2. Minimizing data-volume cost requires us to solve a nonlinear optimization.
3. Multiway beats 2-way joins for star queries and queries on high-fanout graphs.
4. Exact solution for star queries.