DATA516/CSED516
Scalable Data Systems and Algorithms

Lecture 7
Advanced Distributed Query Processing
Announcements

• Today lecture:
  – Part 1: guest lecturer Mingxi Wu, Tigergraph
  – Part 2: finish discussing distributed queries
• Reading assignment postponed for next week; you can update if you submitted
• HW4 = 3 mini homeworks + 1 theory to be posted tomorrow
• Next Tuesday: last regular lecture
• Dec. 1\textsuperscript{st}: 1-on-1 discussion of your projects
• Dec. 8\textsuperscript{th}: project presentations
Review: Distributed Join

Two algorithms for distributed join

• Hash-partition join

• Broadcast join

This lecture: how to compute general queries \textit{without} one join at a time
The Load

• We know the sizes of the input tables: $|R|, |S|, |T|, \ldots$
  – Sometimes they are all equal, then we denote this with $N$

• We run an algorithm on $p$ servers

The load of the algorithm, $L$, is the largest number of tuples received by any server
Example: Hash Join

\[ \text{Join}(x,y,z) = R(x,y) \land S(y,z) \]

**Round 1:** each server
- Hash partition \( R(x,y) \) and \( S(y,z) \) by \( y \)

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>f</td>
<td></td>
</tr>
</tbody>
</table>

**Server 1**

\[ |R|=|S|=N \]

...
Example: Hash Join

\[
\text{Join}(x,y,z) = R(x,y) \land S(y,z)
\]

Round 1: each server
- Hash partition \(R(x,y)\) and \(S(y,z)\) by \(y\)

\[
\begin{array}{|c|c|}
\hline
R & S \\
\hline
x & y & y & z \\
\hline
a & e & e & m \\
a & f & e & n \\
b & f & f & m \\
c & f & f & k \\
\hline
\end{array}
\]

\(|R| = |S| = N\)

\(O\left(\frac{N}{p}\right)\)
Example: Hash Join

Join(x,y,z) = R(x,y) ∧ S(y,z)

Round 1: each server
- Hash partition R(x,y) and S(y,z) by y

Output: each server u:
- local join R_u(x,y) ⋈ S_u(y,z)
Example: Hash Join

\[ \text{Join}(x, y, z) = R(x, y) \land S(y, z) \]

<table>
<thead>
<tr>
<th>R</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>a</td>
<td>e</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>f</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>f</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>f</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>e</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>n</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>m</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>k</td>
</tr>
</tbody>
</table>

Round 1: each server
• Hash partition \( R(x, y) \) and \( S(y, z) \) by \( y \)

\[ |R| = |S| = N \]

\[ R_1(x, y) \bowtie S_1(y, z) \]

\[ \text{Server } 1 \]

\[ \text{Server } p \]

\[ \text{R}_p(x, y) \bowtie \text{S}_p(y, z) \]

\[ L = O(N/p) \]

Output: each server \( u \):
• local join \( R_u(x, y) \bowtie S_u(y, z) \)
Example: Hash Join

\( \text{Join}(x, y, z) = R(x, y) \land S(y, z) \)

**Round 1**: each server
- Hash partition \( R(x, y) \) and \( S(y, z) \) by \( y \)

**Output**: each server \( u \):
- local join \( R_u(x, y) \bowtie S_u(y, z) \)

**Assuming no skew**

\( L = O(N/p) \) w.h.p.
Example: Hash Join

\[ \text{Join}(x,y,z) = \text{R}(x,y) \land \text{S}(y,z) \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>e</td>
<td>e</td>
</tr>
<tr>
<td>a</td>
<td>f</td>
<td>e</td>
</tr>
<tr>
<td>b</td>
<td>f</td>
<td>m</td>
</tr>
<tr>
<td>c</td>
<td>f</td>
<td>k</td>
</tr>
</tbody>
</table>

**Round 1:** each server
- Hash partition \( \text{R}(x,y) \) and \( \text{S}(y,z) \) by \( y \)

**Output:** each server \( u \):
- local join \( \text{R}_u(x,y) \bowtie \text{S}_u(y,z) \)

\( \big| \text{R} \big| = \big| \text{S} \big| = N \)

\( \text{L} = O(N/p) \) w.h.p.

Assuming no skew

Skew threshold: \( N/p \) or lower
Broadcast Join

\[ \text{Join}(x,y,z) = R(x,y) \land S(y,z) \]

\[ \begin{array}{|c|c|} \hline \text{R} & \text{S} \\ \hline x & y & y & z \\ \hline a & e & e & m \\ a & f & f & k \\ b & f \\ c & f \\ \hline \end{array} \]

**Round 1**: each server
- Broadcast \( S(y,z) \) to all servers

**Output**: each server
- local join \( R_u(x,y) \bowtie S(y,z) \)

\[ \begin{aligned} |R| &= N_1 \gg |S| = N_2 \\
L &= O(N_1/p + N_2) \\
\text{Skew no problem} \end{aligned} \]
The Triangles Query

\[ Q(x,y,z) = R(x,y) \land S(y,z) \land T(z,x) \]

Round 1: \[ \text{Temp}(x,y,z) = R(x,y) \land S(y,z) \]
Round 2: \[ Q(x,y,z) = \text{Temp}(x,y,z) \land T(z,x) \]

Problem: \[ |\text{Temp}| \gg N \]

\[ |R| = |S| = |T| = N \text{ tuples} \]
The Triangles Query

\[ Q(x,y,z) = R(x,y) \land S(y,z) \land T(z,x) \]

Algorithm in one round!

- [Afrati’10] Shares Algo (MapReduce)
- [Beame’13,’14] HyperCube Algo (MPC)

\(|R| = |S| = |T| = N\) tuples
Triangles in One Round

• Place servers in a cube $p = p^{1/3} \times p^{1/3} \times p^{1/3}$

• Each server identified by $(i,j,k)$

• Choose 3 random, independent hash functions:
  
  $h_1 : \text{Dom} \rightarrow [p^{1/3}]$
  
  $h_2 : \text{Dom} \rightarrow [p^{1/3}]$
  
  $h_3 : \text{Dom} \rightarrow [p^{1/3}]$

$Q(x,y,z) = R(x,y) \land S(y,z) \land T(z,x)$

$|R| = |S| = |T| = N$ tuples
Triangles in One Round

\[ Q(x,y,z) = R(x,y) \land S(y,z) \land T(z,x) \]

| \[|R| = |S| = |T| = N \text{ tuples} \] |

**Round 1:**
- Send \( R(x,y) \) to all servers \((h_1(x), h_2(y), *)\)
- Send \( S(y,z) \) to all servers \((*, h_2(y), h_3(z))\)
- Send \( T(z,x) \) to all servers \((h_1(x), *, h_3(z))\)

**Output:**
compute locally \( R(x,y) \land S(y,z) \land T(z,x) \)
Communication Cost

Theorem  HyperCube has load $L = O(N/p^{2/3})$ w.h.p., on any input database without skew.

Skew threshold: $N/p^{1/3}$ or lower

This load is optimal, even for data without skew

$Q(x,y,z) = R(x,y) \land S(y,z) \land T(z,x)$

$|R| = |S| = |T| = N$ tuples
HyperCube Algorithm

• In general, we have a multi-join query.
• There are $k$ join variables: $x_1, x_2, \ldots, x_k$
HyperCube Algorithm

• In general, we have a multi-join query.
• There are $k$ join variables: $x_1, x_2, \ldots, x_k$
• Organize the servers into a $k$-dimensional hypercube: $p = p_1 \cdot p_2 \cdots p_k$
HyperCube Algorithm

• In general, we have a multi-join query.
• There are $k$ join variables: $x_1, x_2, \ldots, x_k$
• Organize the servers into a $k$-dimensional hypercube: $p = p_1 \cdot p_2 \cdots p_k$
• Hash partition each relation $R(x_{i_1}, x_{i_2}, \ldots)$ to the hyperplane $p_{i_1} \times p_{i_2} \times \cdots$
HyperCube Algorithm

• In general, we have a multi-join query.
• There are $k$ join variables: $x_1, x_2, \ldots, x_k$
• Organize the servers into a $k$-dimensional hypercube: $p = p_1 \cdot p_2 \cdots p_k$
• Hash partition each relation $R(x_{i_1}, x_{i_2}, \ldots)$ to the hyperplane $p_{i_1} \times p_{i_2} \times \cdots$
• Broadcast along the other dimension
HyperCube Algorithm

• In general, we have a multi-join query.
• There are $k$ join variables: $x_1, x_2, \ldots, x_k$
• Organize the servers into a $k$-dimensional hypercube: $p = p_1 \cdot p_2 \cdots p_k$
• Hash partition each relation $R(x_{i_1}, x_{i_2}, \ldots)$ to the hyperplane $p_{i_1} \times p_{i_2} \times \cdots$
• Broadcast along the other dimension

Main challenge: compute the shares $p_1, p_2, \ldots, p_k$ to minimize the load $L$
Example: Join

\[ Q(x, y, z) = R(x, y) \land S(y, z) \]

- Hash join: \( p_1 = 1, p_2 = p, p_3 = 1 \)

- Broadcast join: \( p_1 = 1, p_2 = 1, p_3 = p \)

Which relation is broadcast?
Computing the Shares

• The secret to computing the shares lies in understanding a very simple query: the cartesian product of two, or more relations
Cartesian Product

An important special case: \( Q = R \times S \)

- In our notation: \( Q(x, y) = R(x) \land S(y) \)
- Assume: \( |R| = N_1, |S| = N_2 \)

- Algorithm:
  - Choose shares such that \( p = p_1 \cdot p_2 \)
  - Distribute \( R(x) \) to row \( h_1(x) \)
  - Distribute \( S(y) \) to column \( h_2(y) \)
**Cartesian Product**

\[ |R| = N_1, \ |S| = N_2 \]

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Y</td>
</tr>
<tr>
<td>a</td>
<td>m</td>
</tr>
<tr>
<td>b</td>
<td>n</td>
</tr>
<tr>
<td>c</td>
<td>p</td>
</tr>
<tr>
<td>d</td>
<td>q</td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
</tr>
</tbody>
</table>

\[ R(a) \rightarrow \]

\[ S(m) \rightarrow \]

\[ R \times S = N \]

\[ \uparrow \]

\[ \exists \subseteq S \]
Cartesian Product

$|R| = N_1, \; |S| = N_2$

Problem: minimize $L = \frac{N_1}{p_1} + \frac{N_2}{p_2}$ such that $p = p_1 \cdot p_2$
Cartesian Product

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Y</td>
</tr>
<tr>
<td>a</td>
<td>m</td>
</tr>
<tr>
<td>b</td>
<td>n</td>
</tr>
<tr>
<td>c</td>
<td>p</td>
</tr>
<tr>
<td>d</td>
<td>q</td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

\[ |R| = N_1, \ |S| = N_2 \]

Problem: minimize \( L = \frac{N_1}{P_1} + \frac{N_2}{P_2} \) such that \( P = P_1 \cdot P_2 \)

Solution: \( L = \frac{N_1}{P_1} + \frac{N_2}{P_2} \geq 2 \sqrt{\frac{N_1 N_2}{P_1 P_2}} = 2 \sqrt{\frac{N_1 N_2}{P}} \)
|R| = N_1, |S| = N_2

Problem: minimize \( L = \frac{N_1}{p_1} + \frac{N_2}{p_2} \) such that \( p = p_1 \cdot p_2 \)

Solution: \( L = \frac{N_1}{p_1} + \frac{N_2}{p_2} \geq 2 \sqrt{\frac{N_1 N_2}{p_1 p_2}} = 2 \sqrt{\frac{N_1 N_2}{p}} \) -- THIS is the optimal load \( L_{opt} \)
Carolesian Product

\[ |R| = N_1, \ |S| = N_2 \]

Problem: minimize \[ L = \frac{N_1}{p_1} + \frac{N_2}{p_2} \] such that \[ p = p_1 \cdot p_2 \]

Solution: \[ L = \frac{N_1}{p_1} + \frac{N_2}{p_2} \geq 2 \sqrt{\frac{N_1 N_2}{p_1 p_2}} = 2 \sqrt{\frac{N_1 N_2}{p}} \]

\( \text{THIS is the optimal load } L_{\text{opt}} \)

From here we can compute the shares: \[ \frac{N_1}{p_1} = \sqrt{\frac{N_1 N_2}{p}} \text{ so } p_1 = \cdots \]
Discussion

• Special case: when $N_1 = N_2 = N$ then:

\[ L_{opt} = \frac{N}{\sqrt{p}} \quad \text{and} \quad p_1 = p_2 = \sqrt{p} \]

• "Virtual servers" don’t work:
  – Let $p=100$, hence $L_{opt}=N/10$
  – Suppose we use $p_{virtual}=40000$: $L_{opt,virt}=N/200$
  – Each real server must simulate 400 virtual
  – Real load is $L_{real}=N/200*400=2N$

• Reason: $\frac{N}{\sqrt{p}}$ means “sub-linear speedup”
General Cartesian Product

\[ Q = R_1 \times R_2 \times \cdots \times R_c \]

- Assume: \(|R_1| = N_1, \ldots, |R_c| = N_c\)

Solution: \( L = \frac{N_1}{p_1} + \cdots + \frac{N_c}{p_c} \geq c \left( \frac{N_1 \cdots N_c}{p_1 \cdots p_c} \right)^{\frac{1}{c}} = c \left( \frac{N_1 \cdots N_c}{p} \right)^{\frac{1}{c}} \)
Edge Packing

\[ Q(x_1,\ldots,x_k) = R_1(\text{vars}_1) \land \cdots \land R_m(\text{vars}_m) \]

An edge packing is a subset of relations \( R_{i_1}, R_{i_2}, \ldots, R_{i_c} \) that do not share variables.

**Fact.** For any edge packing of size \( c \), the load of any 1-round algorithm is:

\[ L \geq c \left( \frac{N_{i_1} \cdots N_{i_c}}{p} \right)^{\frac{1}{c}} \]
Edge Packing

\[ Q(x_1, \ldots, x_k) = R_1(vars_1) \land \cdots \land R_m(vars_m) \]

An edge packing is a subset of relations \( R_{i_1}, R_{i_2}, \ldots, R_{i_c} \) that do not share variables.

**Fact.** For any edge packing of size \( c \), the load of any 1-round algorithm is:

\[
L \geq c \left( \frac{N_{i_1} \cdots N_{i_c}}{p} \right)^{\frac{1}{c}}
\]

Proof (in class)

By example, for \( Q(x, y, z, u) = R(x, y) \land S(y, z) \land T(z, u) \land K(u, x) \)

- Consider packing \( R(x, y), T(z, u) \). Claim: the algorithm **must** compute \( R \times T \)
Edge Packing

\[ Q(x_1, \ldots, x_k) = R_1(var s_1) \wedge \cdots \wedge R_m(var s_m) \]

An edge packing is a subset of relations \( R_{i_1}, R_{i_2}, \ldots, R_{i_c} \) that do not share variables

**Fact.** For any edge packing of size \( c \), the load of any 1-round algorithm is:

\[ L \geq c \left( \frac{N_{i_1} \cdots N_{i_c}}{p} \right)^{\frac{1}{c}} \]

Proof (in class)

By example, for \( Q(x, y, z, u) = R(x, y) \wedge S(y, z) \wedge T(z, u) \wedge K(u, x) \)

- Consider packing \( R(x, y), T(z, u) \). Claim: the algorithm **must** compute \( R \times T \)
- Assume not; then two tuples \( R(a, b), T(c, d) \) do not meet at any server.
Edge Packing

\[ Q(x_1, \ldots, x_k) = R_1(\text{vars}_1) \land \cdots \land R_m(\text{vars}_m) \]

An edge packing is a subset of relations \( R_{i_1}, R_{i_2}, \ldots, R_{i_c} \) that do not share variables

**Fact.** For any edge packing of size \( c \), the load of any 1-round algorithm is:

\[
L \geq c \left( \frac{N_{i_1} \cdots N_{i_c}}{p} \right)^{\frac{1}{c}}
\]

Proof (in class)
By example, for \( Q(x, y, z, u) = R(x, y) \land S(y, z) \land T(z, u) \land K(u, x) \)

- Consider packing \( R(x, y), T(z, u) \). Claim: the algorithm must compute \( R \times T \)
- Assume not; then two tuples \( R(a, b), T(c, d) \) do not meet at any server.
- “Add” tuples \( S(b, c), K(d, a) \) to the input, at some server that doesn’t have \( R(a, b), T(c, d) \).
Edge Packing

\[ Q(x_1, ..., x_k) = R_1(var s_1) \land \cdots \land R_m(var s_m) \]

An **edge packing** is a subset of relations \( R_{i_1}, R_{i_2}, ..., R_{i_c} \) that do not share variables

**Fact.** For any edge packing of size \( c \), the load of any 1-round algorithm is:

\[ L \geq c \left( \frac{N_{i_1} \cdots N_{i_c}}{p} \right)^{\frac{1}{c}} \]

Proof (in class)

By example, for \( Q(x, y, z, u) = R(x, y) \land S(y, z) \land T(z, u) \land K(u, x) \)

- Consider packing \( R(x, y), T(z, u) \). Claim: the algorithm **must** compute \( R \times T \)
- Assume not; then two tuples \( R(a, b), T(c, d) \) do not meet at any server.
- “Add” tuples \( S(b, c), K(d, a) \) to the input, at some server that doesn’t have \( R(a, b), T(c, d) \).
- The tuples \( R(a, b), T(c, d) \) still do not meet (why?), hence algorithm is incorrect
Fractional Edge Packing

\[ Q(x_1, \ldots, x_k) = R_1(var s_1) \land \cdots \land R_m(var s_m) \]

A *fractional edge packing* is a set of weights \( w_1, \ldots, w_m \) such that, for every variable, the sum of weights that contain it is \( \leq 1 \).
Fractional Edge Packing

\[ Q(x_1, \ldots, x_k) = R_1(\text{var} s_1) \land \cdots \land R_m(\text{var} s_m) \]

A \textit{fractional edge packing} is a set of weights \( w_1, \ldots, w_m \) such that, for every variable, the sum of weights that contain it is \( \leq 1 \).

\textbf{Theorem.} For any fractional edge packing, the load of any 1-round algorithm is:

\[ L \geq \left( \frac{N_1^{w_1} \cdots N_m^{w_m}}{w_1 + \cdots + w_m} \right)^{\frac{1}{p}} \]
Fractional Edge Packing

\[ Q(x_1, \ldots, x_k) = R_1(var s_1) \land \cdots \land R_m(var s_m) \]

A **fractional edge packing** is a set of weights \( w_1, \ldots, w_m \) such that, for every variable, the sum of weights that contain it is \( \leq 1 \).

**Theorem.** For any fractional edge packing, the load of any 1-round algorithm is:

\[
L \geq \left( \frac{N_1^{w_1} \cdots N_m^{w_m}}{w_1 + \cdots + w_m} \right) \frac{1}{p}
\]

Moreover, there exists shares for which the HyperCube algorithm has a load:

\[
L_{opt} = O \left( \max_{w_1, \ldots, w_m} \left( \frac{N_1^{w_1} \cdots N_m^{w_m}}{w_1 + \cdots + w_m} \right) \frac{1}{p} \right)
\]
Fractional Edge Packing

\[ Q(x_1, \ldots, x_k) = R_1(var s_1) \land \cdots \land R_m(var s_m) \]

A fractional edge packing is a set of weights \( w_1, \ldots, w_m \) such that, for every variable, the sum of weights that contain it is \( \leq 1 \).

**Theorem.** For any fractional edge packing, the load of any 1-round algorithm is:

\[
L \geq \frac{1}{\left( \frac{N_1^{w_1} \cdots N_m^{w_m}}{p^{w_1+\cdots+w_m}} \right)}
\]

Moreover, there exists shares for which the HyperCube algorithm has a load:

\[
L_{opt} = O \left( \max_{w_1, \ldots, w_m} \left( \frac{N_1^{w_1} \cdots N_m^{w_m}}{p^{w_1+\cdots+w_m}} \right) \right)
\]

The formula gives us \( L_{opt} \) up to some small constant factor (which we ignore). Once you know \( L_{opt} \) you can usually compute the optimal shares for HyperCube.
Discussion

\[ L_{opt} = O\left( \max_{w_1, \ldots, w_m} \left( \frac{N_1^{w_1} \ldots N_m^{w_m}}{p} \right) \right) \]

• We want the \textit{minimal load}, yet the formula above asks us to compute a \textit{max};

• The reason is that the formula is only a lower bound; it happens that the max has a matching algorithm (the proof is non-trivial)
Example: Join

\[ Q(x, y, z) = R(x, y) \land S(y, z) \]

\[ L = \left( \frac{N_1^{w_1} \cdot N_2^{w_2}}{p} \right)^{\frac{1}{w_1+w_2}} \]

- Fractional edge packing: 1,0: \[ L = \frac{N_1}{p} \]
- Fractional edge packing: 0,1: \[ L = \frac{N_2}{p} \]
- Assume \( N_1 \geq N_2 \). We obtain the shares:

\[ \frac{N_1}{p_1 p_2} = \frac{N_1}{p} \quad \text{and} \quad \frac{N_2}{p_2 p_3} = \frac{N_1}{p} \]  

\[ p_1 = \frac{N_1}{N_2}, \quad p_2 = p \frac{N_2}{N_1}, \quad p_3 = 1 \]

Discuss connection to hash-, broadcast-join
Example

\[ Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x) \]

When \( N_1 = N_2 = N_3 = N \), then the optimal load is \( L_{opt} = O\left(\frac{N}{p^{2/3}}\right) \)

What if their sizes are different?
Example

\[ Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x) \]

When \( N_1 = N_2 = N_3 = N \), then the optimal load is \( L_{opt} = O\left(\frac{N}{p^{2/3}}\right) \)

What if their sizes are different?

<table>
<thead>
<tr>
<th>Fractional edge packing</th>
<th>( \left(\frac{\begin{array}{c} N_1^{w_1} \ N_2^{w_2} \ N_3^{w_3} \end{array}}{p} \right) \cdot \frac{1}{w_1+w_2+w_3} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{1}{2}, \frac{1}{2}, \frac{1}{2} )</td>
<td>( \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} )</td>
</tr>
<tr>
<td>1,0,0</td>
<td></td>
</tr>
<tr>
<td>0,1,0</td>
<td></td>
</tr>
<tr>
<td>0,0,1</td>
<td></td>
</tr>
<tr>
<td>0,0,0</td>
<td></td>
</tr>
</tbody>
</table>
Example

\[ Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x) \]

When \( N_1 = N_2 = N_3 = N \), then the optimal load is \( L_{opt} = O(N/p^{2/3}) \)

What if their sizes are different?

| Fractional edge packing \( w_1, w_2, w_3 \) | \[
\left( \frac{N_1^{w_1} \cdot N_2^{w_2} \cdot N_3^{w_3}}{p} \right)^{1/(w_1+w_2+w_3)}
\] |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{1}{2}, \frac{1}{2}, \frac{1}{2} )</td>
<td>( \frac{(N_1 \cdot N_2 \cdot N_3)^{1/3}}{p^{2/3}} )</td>
</tr>
<tr>
<td>( 1,0,0 )</td>
<td></td>
</tr>
<tr>
<td>( 0,1,0 )</td>
<td></td>
</tr>
<tr>
<td>( 0,0,1 )</td>
<td></td>
</tr>
<tr>
<td>( 0,0,0 )</td>
<td></td>
</tr>
</tbody>
</table>
Example

$$Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x)$$

When $$N_1 = N_2 = N_3 = N$$, then the optimal load is $$L_{opt} = O(N/p^{2/3})$$

What if their sizes are different?

<table>
<thead>
<tr>
<th>Fractional edge packing $$w_1, w_2, w_3$$</th>
<th>[ \left( \frac{N_1^{w_1} \cdot N_2^{w_2} \cdot N_3^{w_3}}{p} \right)^{1/(w_1+w_2+w_3)} ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$\frac{1}{2}, \frac{1}{2}, \frac{1}{2}$$</td>
<td>$$\left( \frac{N_1 \cdot N_2 \cdot N_3}{p^{2/3}} \right)^{1/3}$$</td>
</tr>
<tr>
<td>$$1,0,0$$</td>
<td>$$\frac{N_1}{p}$$</td>
</tr>
<tr>
<td>$$0,1,0$$</td>
<td></td>
</tr>
<tr>
<td>$$0,0,1$$</td>
<td></td>
</tr>
<tr>
<td>$$0,0,0$$</td>
<td></td>
</tr>
</tbody>
</table>
Example

\[ Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x) \]

When \( N_1 = N_2 = N_3 = N \), then the optimal load is \( L_{opt} = O\left(\frac{N}{p^{2/3}}\right) \)

What if their sizes are different?

<table>
<thead>
<tr>
<th>Fractional edge packing ( w_1, w_2, w_3 )</th>
<th>( \left( \frac{N_1^{w_1} \cdot N_2^{w_2} \cdot N_3^{w_3}}{p} \right)^{1/(w_1+w_2+w_3)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{1}{2}, \frac{1}{2}, \frac{1}{2} )</td>
<td>( \left( \frac{N_1 \cdot N_2 \cdot N_3}{1} \right)^{1/3} )</td>
</tr>
<tr>
<td>( 1,0,0 )</td>
<td>( \frac{N_1}{p} )</td>
</tr>
<tr>
<td>( 0,1,0 )</td>
<td>( \frac{N_2}{p} )</td>
</tr>
<tr>
<td>( 0,0,1 )</td>
<td>( \frac{N_3}{p} )</td>
</tr>
<tr>
<td>( 0,0,0 )</td>
<td></td>
</tr>
</tbody>
</table>
Example

\[ Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x) \]

When \[ N_1 = N_2 = N_3 = N \], then the optimal load is \[ L_{opt} = O\left(\frac{N}{p^{2/3}}\right) \]

What if their sizes are different?

<table>
<thead>
<tr>
<th>Fractional edge packing [ w_1, w_2, w_3 ]</th>
<th>[ \frac{1}{w_1+w_2+w_3} \left( \frac{N_1^{w_1} \cdot N_2^{w_2} \cdot N_3^{w_3}}{p} \right) ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \frac{1}{2}, \frac{1}{2}, \frac{1}{2} ]</td>
<td>[ \left( N_1 \cdot N_2 \cdot N_3 \right)^{1/3} ]</td>
</tr>
<tr>
<td>[ 1,0,0 ]</td>
<td>[ \frac{N_1}{p} ]</td>
</tr>
<tr>
<td>[ 0,1,0 ]</td>
<td>[ \frac{N_2}{p} ]</td>
</tr>
<tr>
<td>[ 0,0,1 ]</td>
<td>[ \frac{N_3}{p} ]</td>
</tr>
<tr>
<td>[ 0,0,0 ]</td>
<td>[ 0 \text{ (why?)} ]</td>
</tr>
</tbody>
</table>
Example

\[ Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x) \]

When \( N_1 = N_2 = N_3 = N \), then the optimal load is \( L_{opt} = O\left(\frac{N}{p^{2/3}}\right) \)

What if their sizes are different?

<table>
<thead>
<tr>
<th>Fractional edge packing</th>
<th>( \left( \frac{N_1^{w_1} \cdot N_2^{w_2} \cdot N_3^{w_3}}{p} \right)^{1/(w_1+w_2+w_3)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{1}{2}, \frac{1}{2}, \frac{1}{2} )</td>
<td>( \left( \frac{N_1 \cdot N_2 \cdot N_3}{p^{2/3}} \right)^{1/3} )</td>
</tr>
<tr>
<td>1,0,0</td>
<td>( \frac{N_1}{p} )</td>
</tr>
<tr>
<td>0,1,0</td>
<td>( \frac{N_2}{p} )</td>
</tr>
<tr>
<td>0,0,1</td>
<td>( \frac{N_3}{p} )</td>
</tr>
<tr>
<td>0,0,0</td>
<td>0 (why?)</td>
</tr>
</tbody>
</table>

Optimal load \( L_{opt} \) is the maximum of this column
Example (cont’d)

\[ Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x) \]

Need max of \( \left( \frac{N_1 \cdot N_2 \cdot N_3}{p^{2/3}} \right)^{1/3}, \frac{N_1}{p}, \frac{N_2}{p}, \frac{N_3}{p} \)

Suppose w.l.o.g. \( N_1 \geq N_2 \geq N_3 \)
Example (cont’d)

\[ Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x) \]

Need max of \( \frac{(N_1 \cdot N_2 \cdot N_3)^{1/3}}{p^{2/3}}, \frac{N_1}{p}, \frac{N_2}{p}, \frac{N_3}{p} \)

Suppose w.l.o.g. \( N_1 \geq N_2 \geq N_3 \)

• Case 1: \( \frac{(N_1 \cdot N_2 \cdot N_3)^{1/3}}{p^{2/3}} \leq \frac{N_1}{p} = L_{opt} \)

The share of \( z \) is \( p_3 = 1 \), hence “cartesian product \( S \times T \), distribute \( R \)”
Example (cont’d)

\[ Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x) \]

Need max of \( \frac{(N_1 \cdot N_2 \cdot N_3)^{1/3}}{p^{2/3}} \), \( \frac{N_1}{p} \), \( \frac{N_2}{p} \), \( \frac{N_3}{p} \)

Suppose w.l.o.g. \( N_1 \geq N_2 \geq N_3 \)

- Case 1: \( \frac{(N_1 \cdot N_2 \cdot N_3)^{1/3}}{p^{2/3}} \leq \frac{N_1}{p} = L_{opt} \)

  The share of \( z \) is \( p_3 = 1 \), hence
  “cartesian product \( S \times T \), distribute \( R \)”

Proof: Load due to \( R \):

\[ \frac{N_1}{p_{1}p_{2}} = L_{opt}, \text{ i.e. } \frac{N_1p_3}{p} = \frac{N_1}{p} \]
Example (cont’d)

$Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x)$

Need max of $\frac{(N_1 \cdot N_2 \cdot N_3)^{1/3}}{p^{2/3}}$, $\frac{N_1}{p}$, $\frac{N_2}{p}$, $\frac{N_3}{p}$

Suppose w.l.o.g. $N_1 \geq N_2 \geq N_3$

- Case 1: $\frac{(N_1 \cdot N_2 \cdot N_3)^{1/3}}{p^{2/3}} \leq \frac{N_1}{p} = L_{opt}$

  The share of $z$ is $p_3 = 1$, hence “cartesian product $S \times T$, distribute $R$”

- Case 2: “normal” hypercube $L_{opt} = \frac{(N_1 \cdot N_2 \cdot N_3)^{1/3}}{p^{2/3}}$

**Proof:** Load due to $R$: $\frac{N_1}{p_1 p_2} = L_{opt}$, i.e. $\frac{N_1 p_3}{p} = \frac{N_1}{p}$
Example (cont’d)

\[ Q(x, y, z) = R(x, y) \land S(y, z) \land T(z, x) \]

Need max of \[ \frac{(N_1 \cdot N_2 \cdot N_3)^{1/3}}{p^{2/3}}, \frac{N_1}{p}, \frac{N_2}{p}, \frac{N_3}{p} \]

Suppose w.l.o.g. \( N_1 \geq N_2 \geq N_3 \)

- Case 1: \[ \frac{(N_1 \cdot N_2 \cdot N_3)^{1/3}}{p^{2/3}} \leq \frac{N_1}{p} = L_{opt} \]

  The share of \( z \) is \( p_3 = 1 \), hence “cartesian product \( S \times T \), distribute \( R \)”

- Case 2: “normal” hypercube \( L_{opt} = \frac{(N_1 \cdot N_2 \cdot N_3)^{1/3}}{p^{2/3}} \)

Proof: Load due to \( R \):

\[ \frac{N_1}{p_1 p_2} = L_{opt}, \text{ i.e. } \frac{N_1 p_3}{p} = \frac{N_1}{p} \]

When \( p \leq \frac{N_1^2}{N_2 N_3} \) then Case 1, linear speedup; otherwise case 2, sublinear
Final Special Case

• When all cardinalities are equal, then:

\[
\left( \frac{N^{w_1} \ldots N^{w_m}}{p} \right)^{\frac{1}{w_1+\ldots+w_m}} = \frac{N}{p^{\frac{1}{w_1+\ldots+w_m}}}
\]

• For a graph G, the quantity

\[
\tau^* = \max_{\text{fractional edge packing}} (w_1 + \ldots w_m)
\]

is called the \textit{fractional edge packing number}

• \( L_{opt} = \frac{N}{p^{\tau^*}} \)
Conclusions

• The HyperCube algorithms combines two strategies: hash-partition, and broadcast

• When \( p \) is small, then it can broadcast the smaller relations;

• As \( p \) increases, “smaller” relations no longer help, and the load gets closer to the fractional edge covering number \( \tau^* \)