Query Processing with Optimal Communication Cost

Magdalena Balazinska and Dan Suciu

University of Washington
Context

Past: NSF Big Data grant
• PhD student Paris Koutris received the ACM SIGMOD Jim Gray Dissertation Award

Current: AiTF Grant
• PI’s Magda Balazinska, Dan Suciu
• Student: Walter Cai
Basic Question

• How much communication is needed to compute a query $Q$ on $p$ servers?

• Parallel data processing
  – Gamma, MapReduce, Hive, Teradata, Aster Data, Spark, Impala, Myria, Tensorflow
  – See Magda Balazinska’s current class
Background

• $Q$ conjunctive query;
  $\rho^* = \text{its fractional edge covering number}$

**Thm.** [Atserias, Grohe, Marx’2011] If every input relation has size $\leq m$ then $|\text{Output}(Q)| \leq m^{\rho^*}$

• $Q(x,y,z) :- R(x,y) \land S(y,z) \land T(z,x)$
  If $|R|, |S|, |T| \leq m$ then $|\text{Output}(Q)| \leq m^{3/2}$

$\rho^* = \frac{3}{2}$
Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

Input data = size $m$

Number of servers = $p$

Input (size=$m$) $\rightarrow$ Server 1 $\ldots$ $\rightarrow$ Server $p$

$O(m/p)$ $\rightarrow$ Server 1 $\ldots$ $\rightarrow$ Server $p$
Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]
Input data = size $m$
Number of servers = $p$
One round = Compute & communicate

Input (size=$m$) -> $O(m/p)$
$\leq L$ (Round 1)
$O(m/p)$
Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

Input data = size $m$

Number of servers = $p$

One round = Compute & communicate

Algorithm = Several rounds

Input (size=$m$) $\rightarrow$ $O(m/p)$

Round 1

Round 2

Round 3
Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

Input data = size $m$

Number of servers = $p$

One round = Compute & communicate

Algorithm = Several rounds

Max communication load / round / server = $L$

Input (size = $m$) → $O(m/p)$

$\leq L$ → $\leq L$ → $\leq L$

Round 1 → Round 2 → Round 3

$O(m/p)$ → $O(m/p)$
Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

Input data = size m

Number of servers = p

One round = Compute & communicate

Algorithm = Several rounds

Max communication load / round / server = L

Cost:
Load L
Rounds r
Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

Input data = size \( m \)

Number of servers = \( p \)

One round = Compute & communicate

Algorithm = Several rounds

Max communication load / round / server = \( L \)

Cost:

<table>
<thead>
<tr>
<th></th>
<th>Naïve 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load ( L )</td>
<td>( L = m )</td>
</tr>
<tr>
<td>Rounds ( r )</td>
<td>1</td>
</tr>
</tbody>
</table>
Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

Input data = size $m$

Number of servers = $p$

One round = Compute & communicate

Algorithm = Several rounds

Max communication load / round / server = $L$

Cost:

<table>
<thead>
<tr>
<th>Load $L$</th>
<th>Naïve 1</th>
<th>Naïve 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve 1: $L = m$</td>
<td>$p$</td>
<td></td>
</tr>
<tr>
<td>Naïve 2: $L = m/p$</td>
<td>$p$</td>
<td></td>
</tr>
<tr>
<td>Round $r$</td>
<td>1</td>
<td>$p$</td>
</tr>
</tbody>
</table>
Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

Input data = size $m$

Number of servers = $p$

One round = Compute & communicate

Algorithm = Several rounds

Max communication load / round / server = $L$

Cost:

<table>
<thead>
<tr>
<th></th>
<th>Ideal</th>
<th>Naïve 1</th>
<th>Naïve 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load $L$</td>
<td>$L = m/p$</td>
<td>$L = m$</td>
<td>$L = m/p$</td>
</tr>
<tr>
<td>Rounds $r$</td>
<td>1</td>
<td>1</td>
<td>$p$</td>
</tr>
</tbody>
</table>
Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

Input data = size $m$

Number of servers = $p$

One round = Compute & communicate

Algorithm = Several rounds

Max communication load / round / server = $L$

<table>
<thead>
<tr>
<th>Cost:</th>
<th>Ideal</th>
<th>Practical $\epsilon \in (0, 1)$</th>
<th>Naïve 1</th>
<th>Naïve 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load $L$</td>
<td>$L = m/p$</td>
<td>$L = m/p^{1-\epsilon}$</td>
<td>$L = m$</td>
<td>$L = m/p$</td>
</tr>
<tr>
<td>Rounds $r$</td>
<td>1</td>
<td>$O(1)$</td>
<td>1</td>
<td>$p$</td>
</tr>
</tbody>
</table>
A Naïve Lower Bound

• Query $Q$
• Inputs $R, S, T, \ldots$ s.t. $|\text{size}(Q)| = m^\rho^*$
• Algorithm with load $L$,
• After $r$ rounds, one server “knows” $\leq L^r\rho^*$ tuples: it can output $\leq (L^r)^{\rho^*}$ tuples (AGM)
• $p$ servers compute $|\text{size}(Q)| = m^\rho^*$, hence $p^*(L^r)^{\rho^*} \geq m^\rho^*$

**Thm.** Any $r$-round algorithm has $L \geq m / r^*p^{1/\rho^*}$
Speedup

Speed = \( O\left(\frac{1}{L}\right) \)

A load of \( L = \frac{m}{p} \) corresponds to linear speedup

A load of \( L = \frac{m}{p^{1-\epsilon}} \) corresponds to sub-linear speedup

What is the theoretically optimal load \( L = f(m,p) \)? Is this the right question in the field?
Join of Two Tables

Join(x,y,z) = R(x,y) ∧ S(y,z)
|R| = |S| = m tuples

In the field:
• Hash-join on y: \( L = \frac{m}{p} \) (w/o skew)
• Broadcast-join: \( L \approx m \)

In theory: \( L \geq \frac{m}{p^{1/2}} \)
Triangles

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

State of the art:

• Hash-join, two rounds:
  • Problem: intermediate result too big!

• Broadcast \( S, T \), one round:
  • Problem: two local tables are huge!
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 $(i, j, k)$

$\text{Triangles}(x, y, z) = R(x, y) \land S(y, z) \land T(z, x)$

$|R| = |S| = |T| = m$ tuples

[Afrati & Ullman’10]
[Beame’13,’14]
**Triangles in One Round**

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

### Output:
- Compute locally $R(x,y) \land S(y,z) \land T(z,x)$

### Example Table:

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fred</td>
<td>Alice</td>
<td>Jim</td>
</tr>
<tr>
<td>Fred</td>
<td>Alice</td>
<td>Jim</td>
<td>Jack</td>
</tr>
<tr>
<td>Jack</td>
<td>Jim</td>
<td>Alice</td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>Jim</td>
<td>Jack</td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>Alice</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$$\text{Triangles}(x,y,z) = R(x,y) \land S(y,z) \land T(z,x)$$

$|R| = |S| = |T| = m$ tuples
Communication load per server

**Theorem** Assuming “no skew”, HyperCube computes **Triangles** with $L = O(m/p^{2/3})$ w.h.p.

Can we compute **Triangles** with $L = m/p$?

No!

**Theorem** Any 1-round algo. has $L = \Omega(m/p^{2/3})$, even on inputs with no skew.
1.1M triples of Twitter data $\rightarrow$ 220k triangles; $p=64$

local 1 or 2-step hash-join; local 1-step Leapfrog Trie-join (a.k.a. Generic-Join)
1.1M triples of Twitter data $\rightarrow$ 220k triangles; $p=64$

$$\text{Triangles}(x,y,z) = \text{R}(x,y) \land \text{S}(y,z) \land \text{T}(z,x)$$

<table>
<thead>
<tr>
<th>shuffle</th>
<th>tuples sent</th>
<th>producer skew</th>
<th>consumer skew</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{R}(x, y) \rightarrow \text{h}(y)$</td>
<td>1,114,289</td>
<td>1</td>
<td>1.35</td>
</tr>
<tr>
<td>$\text{S}(y, z) \rightarrow \text{h}(y)$</td>
<td>1,114,289</td>
<td>1</td>
<td>1.72</td>
</tr>
<tr>
<td>$\text{RS}(x, y, z) \rightarrow \text{h}(z)$</td>
<td>50,862,578</td>
<td>20.8</td>
<td>1</td>
</tr>
<tr>
<td>$\text{T}(z, x) \rightarrow \text{h}(z)$</td>
<td>1,114,289</td>
<td>1</td>
<td>1.01</td>
</tr>
<tr>
<td>Total</td>
<td>54,205,445</td>
<td>N.A.</td>
<td>N.A.</td>
</tr>
</tbody>
</table>

Table 2: Load balance with regular shuffles in query Q1

<table>
<thead>
<tr>
<th>shuffles</th>
<th>tuples sent</th>
<th>producer skew</th>
<th>consumer skew</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCS $\text{R}(x, y)$</td>
<td>4,457,156</td>
<td>1</td>
<td>1.05</td>
</tr>
<tr>
<td>HCS $\text{S}(y, z)$</td>
<td>4,457,156</td>
<td>1</td>
<td>1.05</td>
</tr>
<tr>
<td>HCS $\text{T}(z, x)$</td>
<td>4,457,156</td>
<td>1</td>
<td>1.05</td>
</tr>
<tr>
<td>Total</td>
<td>13,371,468</td>
<td>N.A.</td>
<td>N.A.</td>
</tr>
</tbody>
</table>

Table 3: Load balance with HyperCube shuffles in query Q1
General Case

**Theorem** The optimal load for computing Q in one-round on skew-free data is \( L = O\left(\frac{m}{p^{1/\tau^*}}\right) \)

\( \tau^* = \text{fractional vertex cover of Q's hypergraph} \)

\[ \begin{align*}
\tau^* &= 1 & m / p \\
0 & 1 & 0
\end{align*} \]

\[ \begin{align*}
\tau^* &= 3/2 & m / p^{2/3}
\end{align*} \]

**Thm.** Any \( r \)-round algorithm has \( L \geq \frac{m}{r^* p^{1/\rho^*}} \)

\( \rho^* = \text{fractional edge cover of Q's hypergraph} \)

\[ \begin{align*}
\rho^* &= 2 & m / p^{1/2} \\
1 & 1
\end{align*} \]

\[ \begin{align*}
\rho^* &= 3/2 & m / p^{2/3}
\end{align*} \]
Skew

• Skewed data is major impediment to parallel data processing

• Practical solutions:
  – Deal with stragglers, hope they eventually terminate
  – Remove heavy hitters from computation

• Our approach:
  – Query $\rightarrow$ Residual Query
  – Join $R(x,y) \land S(y,z)$ $\rightarrow$ Cartesian Product $R(x) \land S(z)$
Skewed Values $\rightarrow$ New Query

**Join** \(x,y,z\) = \(R(x,y) \land S(y,z)\)

**No-skew:**
\[\tau^* = 1, \quad L = \frac{m}{p}\]

**Skewed:** \((y = \text{single value, degree} = m)\)
Join becomes **Product** \(x,z = R(x) \land S(z)\)
\[\tau^* = 2, \quad L = \frac{m}{p^{1/2}}\]
Summary of Results so Far

• 1 Round
  – No skew: optimal load = \( \frac{m}{p^{1/\tau^*}} \)
  – Skew: provably higher

• Multiple rounds
  – Lower bound: load >= \( \frac{m}{p^{1/\rho^*}} \)
  – All relations are binary: optimal load = \( \frac{m}{p^{1/\rho^*}} \)
    [PODS’2017a]
  – Arbitrary relations: optimal load = ?? Open

• Additional statistics: keys, degree constraints
  [PODS’2017b]

Thank you!