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?

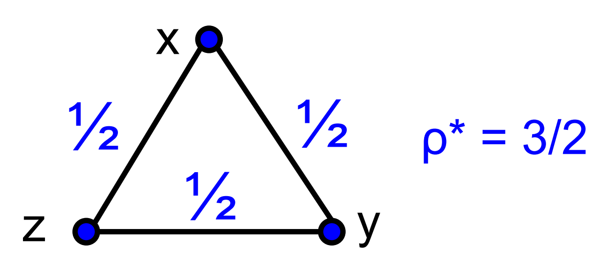
- Parallel data processing
  - Gamma, MapReduce, Hive, Teradata, Aster Data, Spark, Impala, Myria, Tensorflow
  - See Magda Balazinska's current class

Background

- Q conjunctive query;
  ρ^* = its fractional edge covering number

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

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

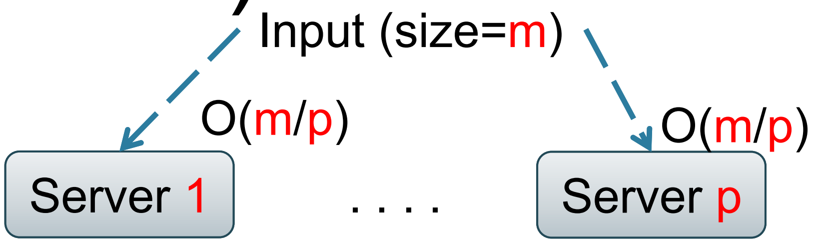


Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

Input data = size m

Number of servers = p



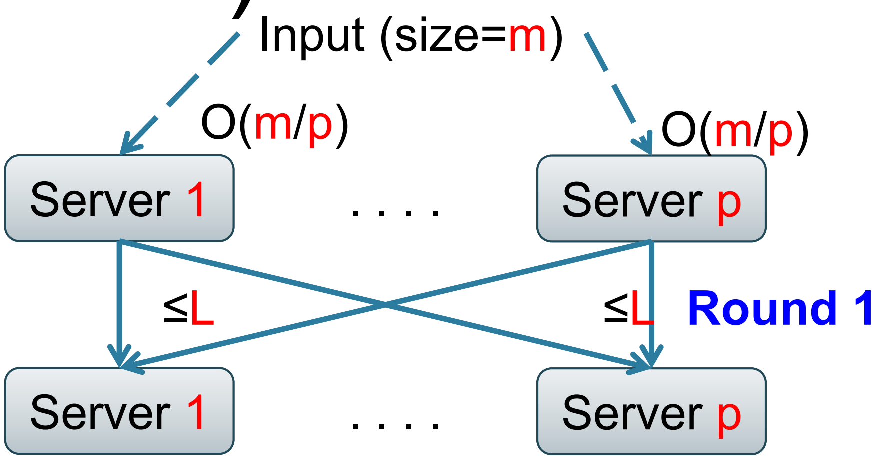
Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

Input data = size m

Number of servers = p

One round = Compute & communicate



Massively Parallel Communication Model (MPC)

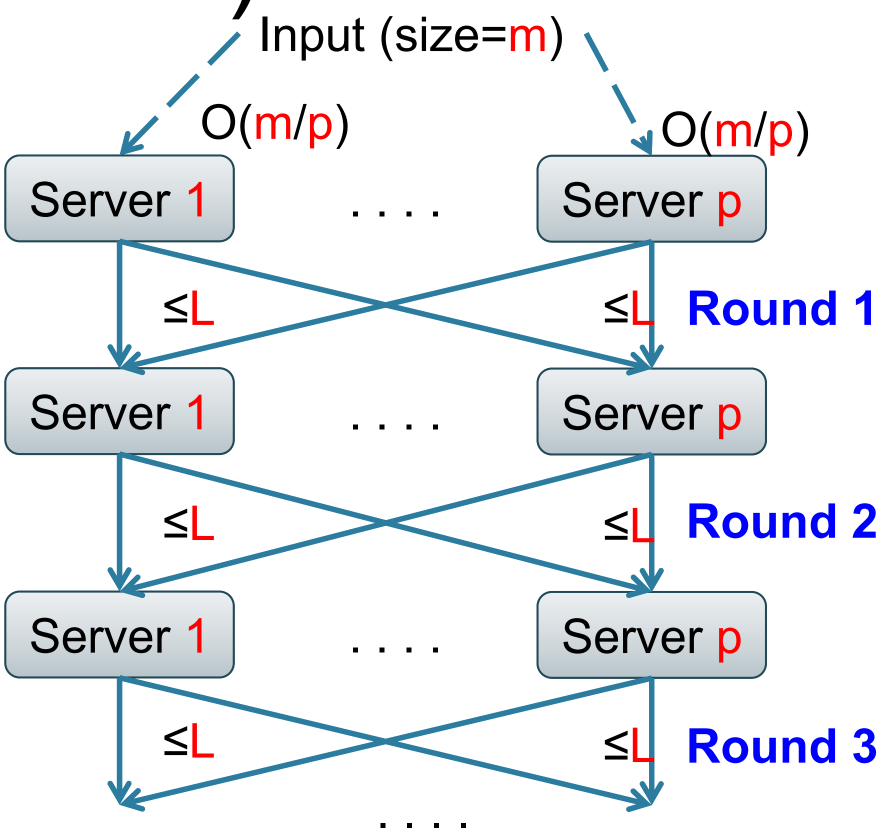
Extends BSP [Valiant]

Input data = size m

Number of servers = p

One round = Compute & communicate

Algorithm = Several rounds



Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

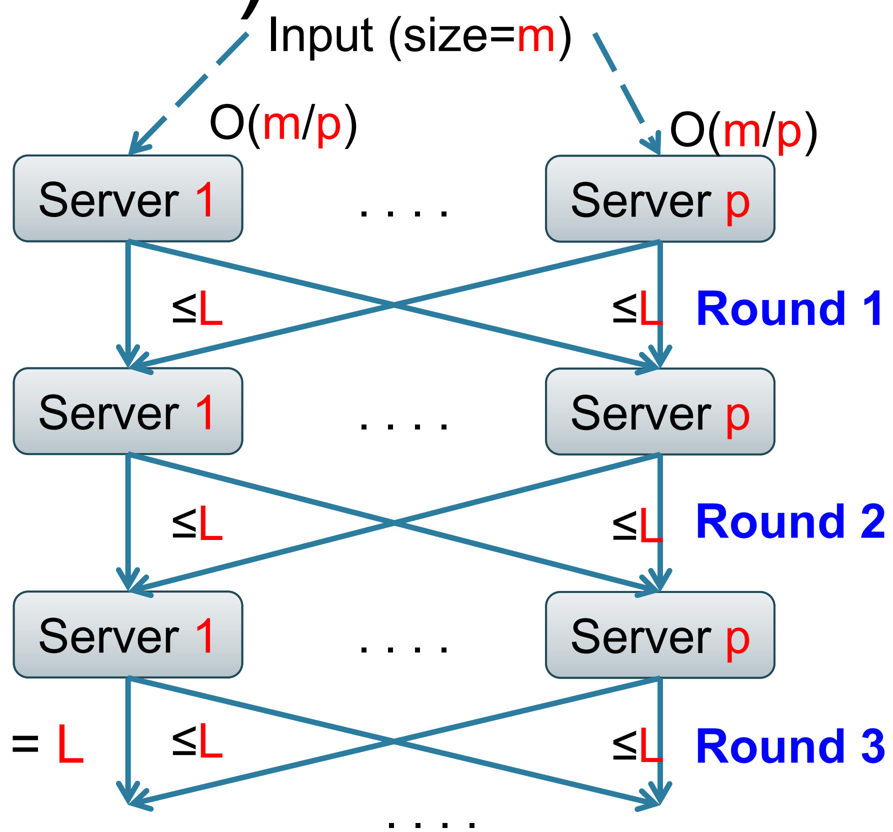
Input data = size m

Number of servers = p

One round = Compute & communicate

Algorithm = Several rounds

Max communication load / round / server = L



Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

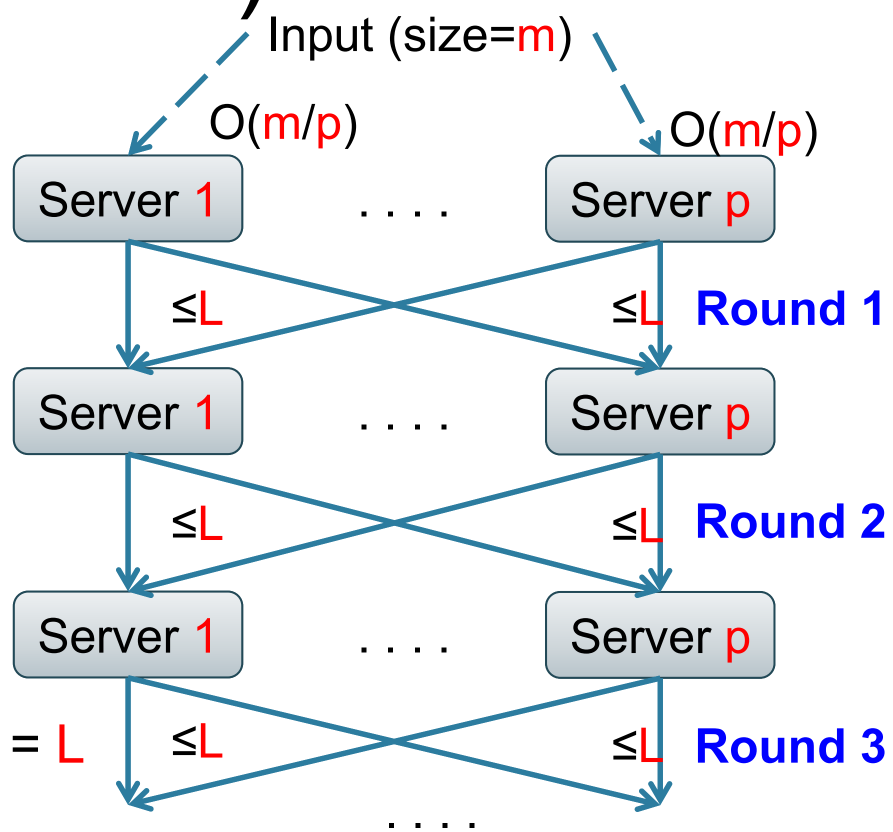
Input data = size m

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]

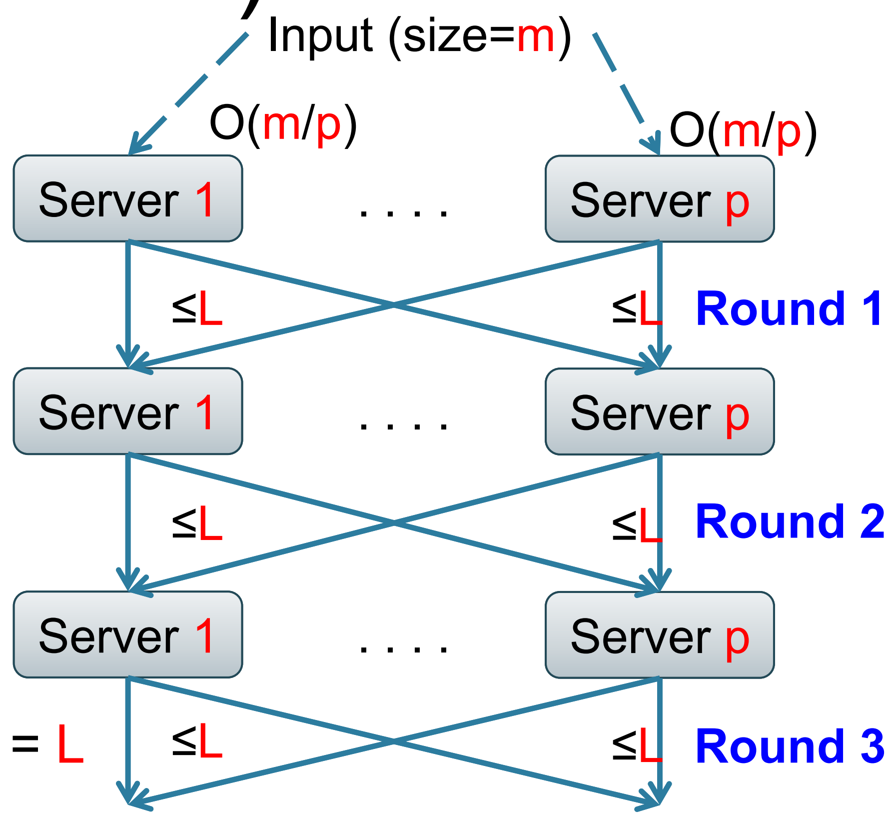
Input data = size m

Number of servers = p

One round = Compute & communicate

Algorithm = Several rounds

Max communication load / round / server = L



Cost:			Naïve 1	
Load L			L = m	
Rounds r			1	

Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

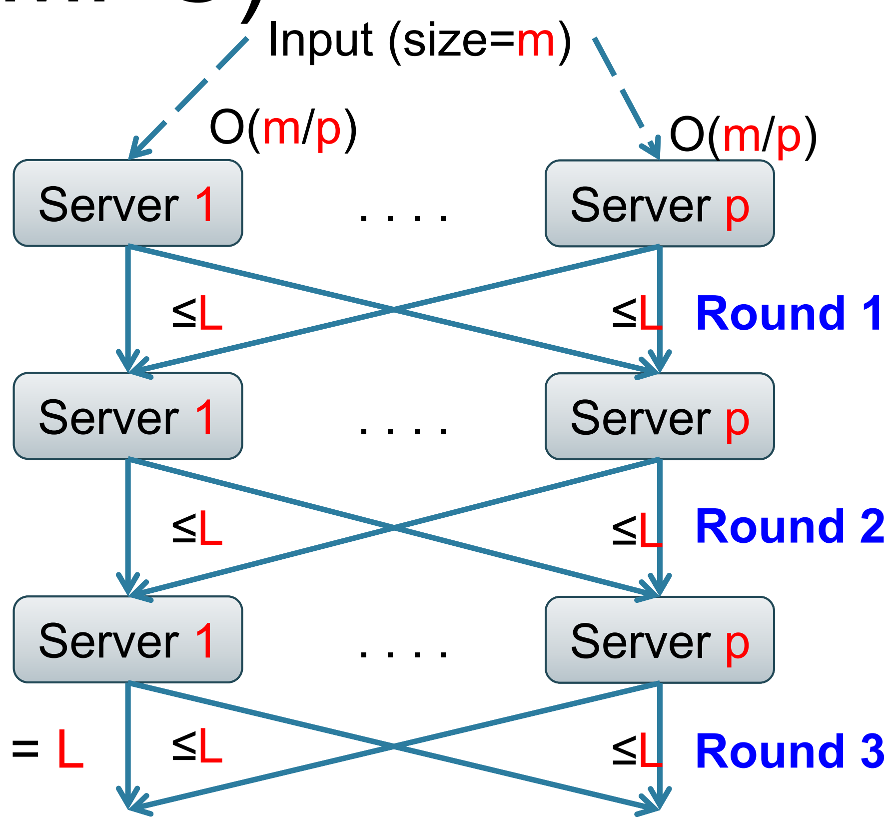
Input data = size m

Number of servers = p

One round = Compute & communicate

Algorithm = Several rounds

Max communication load / round / server = L



Cost:			Naïve 1	Naïve 2
Load L			L = m	L = m/p
Rounds r			1	p

Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

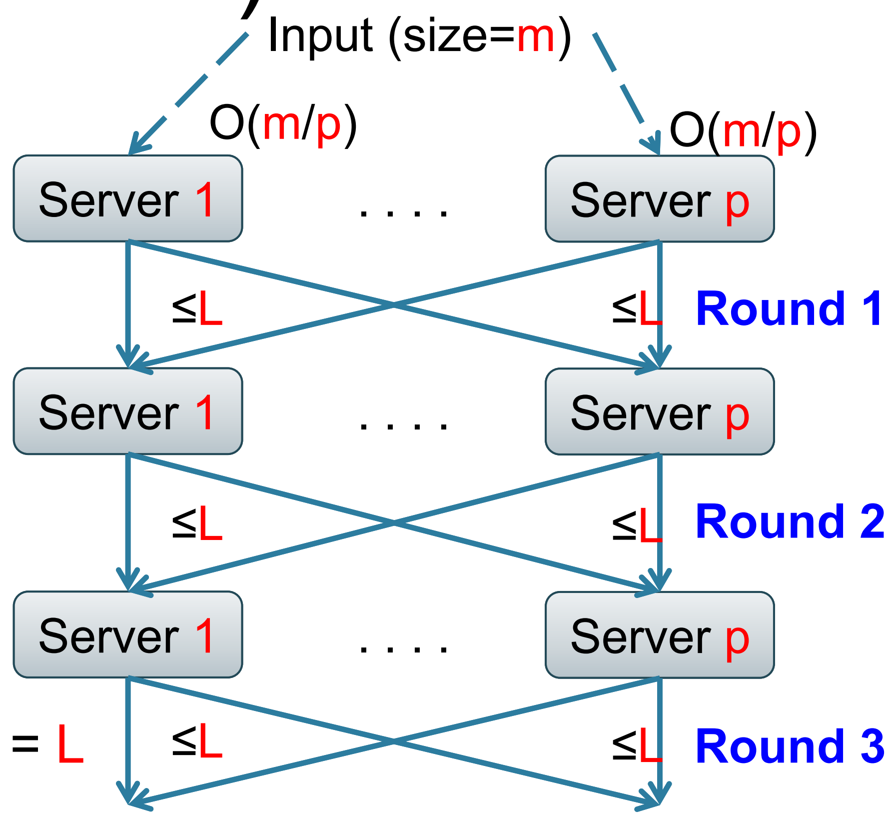
Input data = size m

Number of servers = p

One round = Compute & communicate

Algorithm = Several rounds

Max communication load / round / server = L



Cost:	Ideal		Naïve 1	Naïve 2
Load L	L = m/p		L = m	L = m/p
Rounds r	1		1	p

Massively Parallel Communication Model (MPC)

Extends BSP [Valiant]

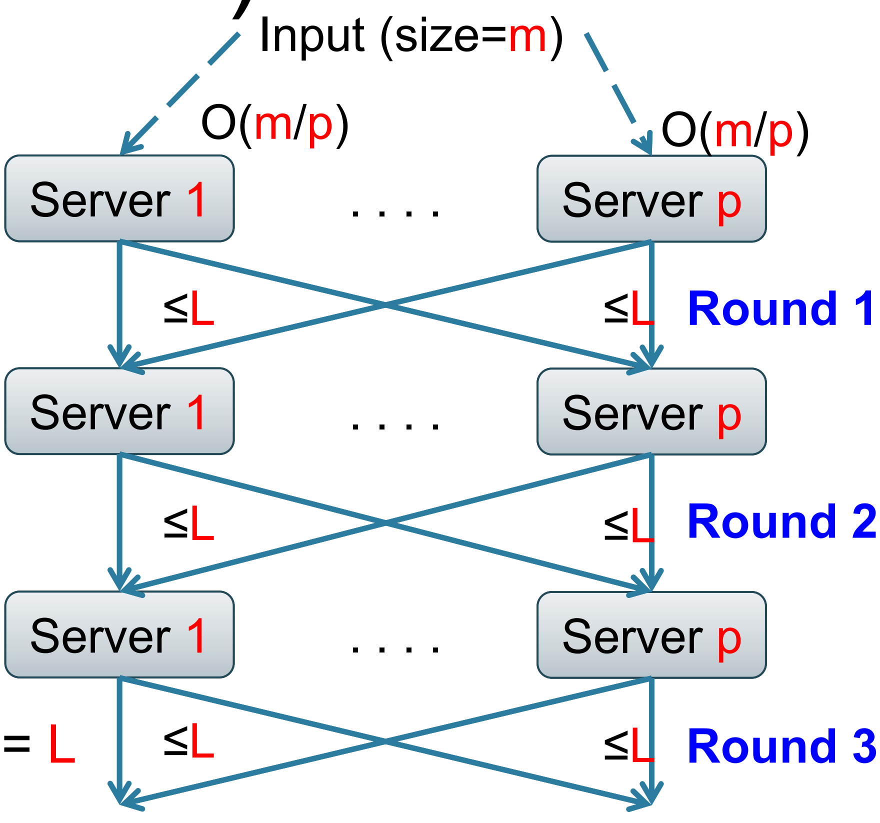
Input data = size m

Number of servers = p

One round = Compute & communicate

Algorithm = Several rounds

Max communication load / round / server = L



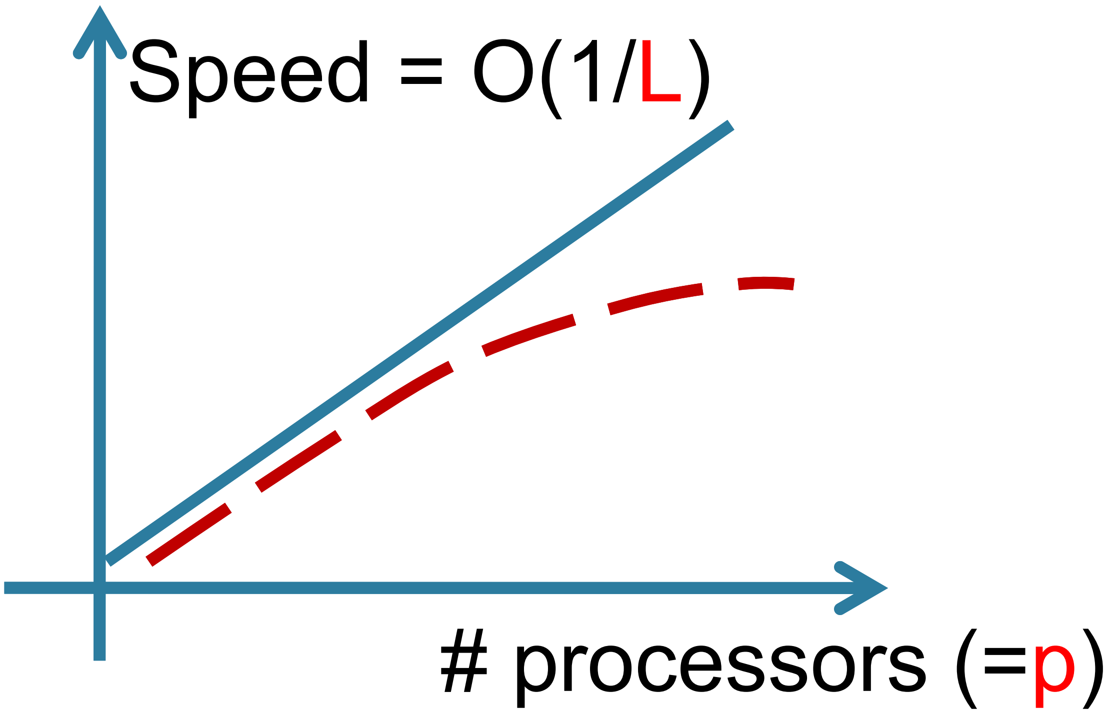
Cost:	Ideal	Practical $\varepsilon \in (0,1)$	Naïve 1	Naïve 2
Load L	$L = m/p$	$L = m/p^{1-\varepsilon}$	$L = m$	$L = m/p$
Rounds r	1	$O(1)$	1	p

A Naïve Lower Bound

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

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

Speedup



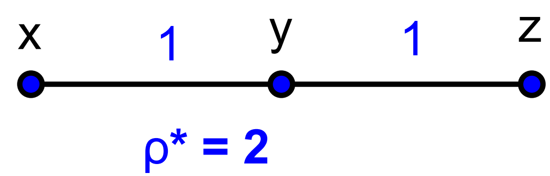
A load of $L = m/p$ corresponds to linear speedup

A load of $L = m/p^{1-\varepsilon}$ 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

$Join(x,y,z) = R(x,y) \wedge S(y,z)$



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

In the field:

- Hash-join on y: $L = m / p$ (w/o skew)
- Broadcast-join: $L \approx m$

In theory: $L \geq m / p^{1/2}$

Triangles

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

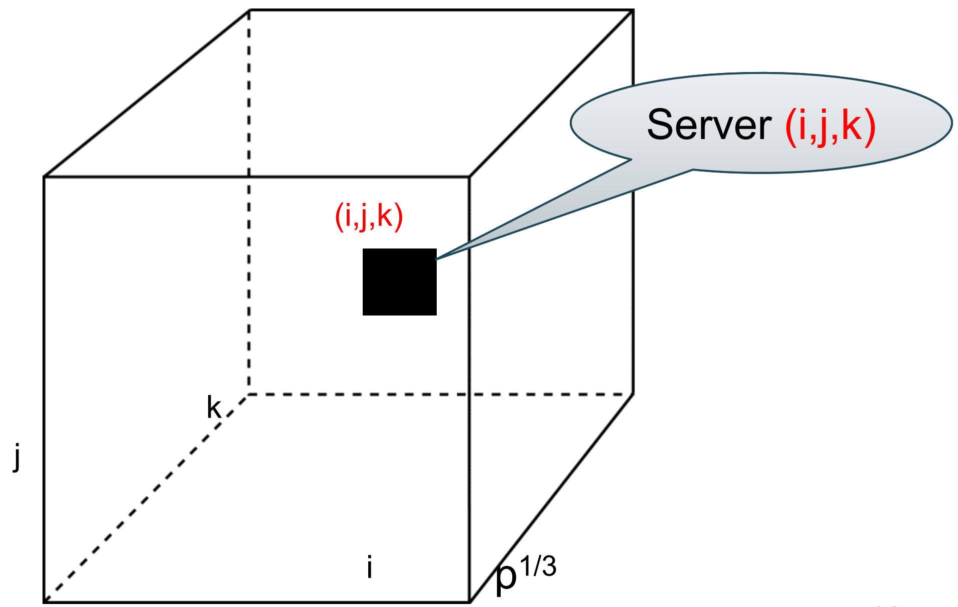
State of the art:

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

- Broadcast S,T, one round:
- Problem: two local tables are huge!

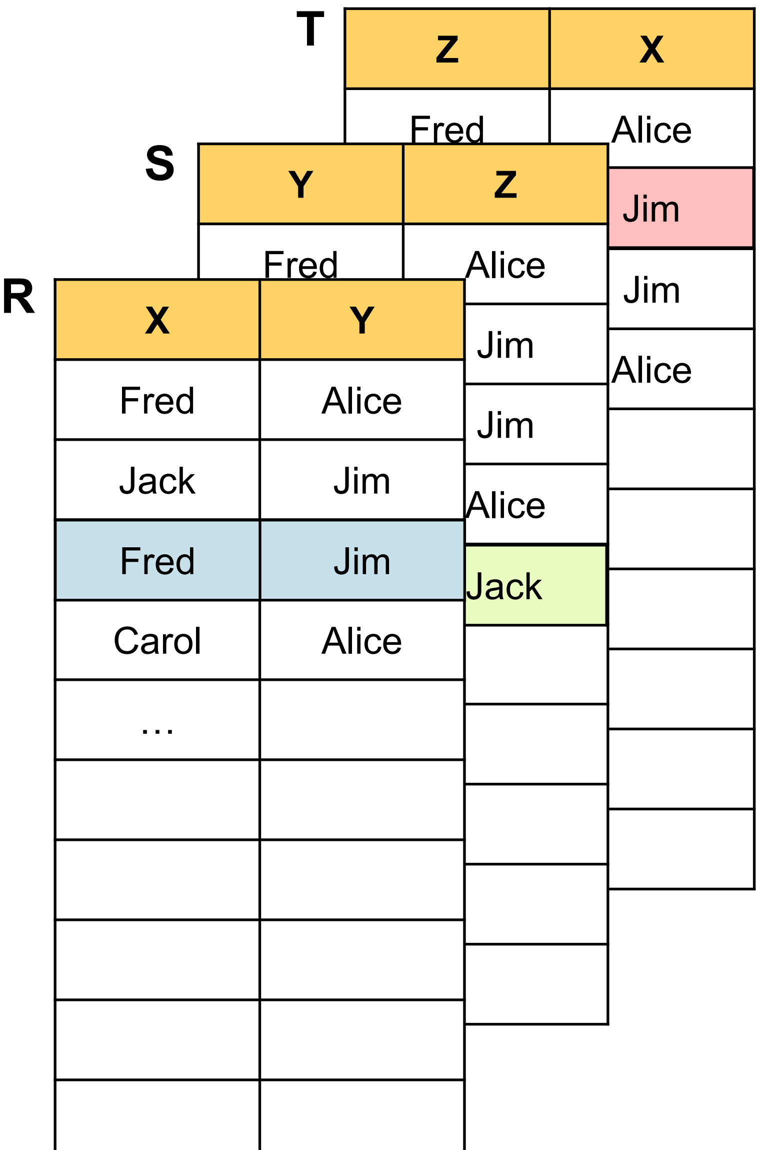
Triangles in One Round

- Place servers in a cube $p = p^{1/3} \times p^{1/3} \times p^{1/3}$
- Each server identified by (i,j,k)



[Afrati&Ullman'10]
[Beame'13,'14]

Triangles in One Round



Round 1:

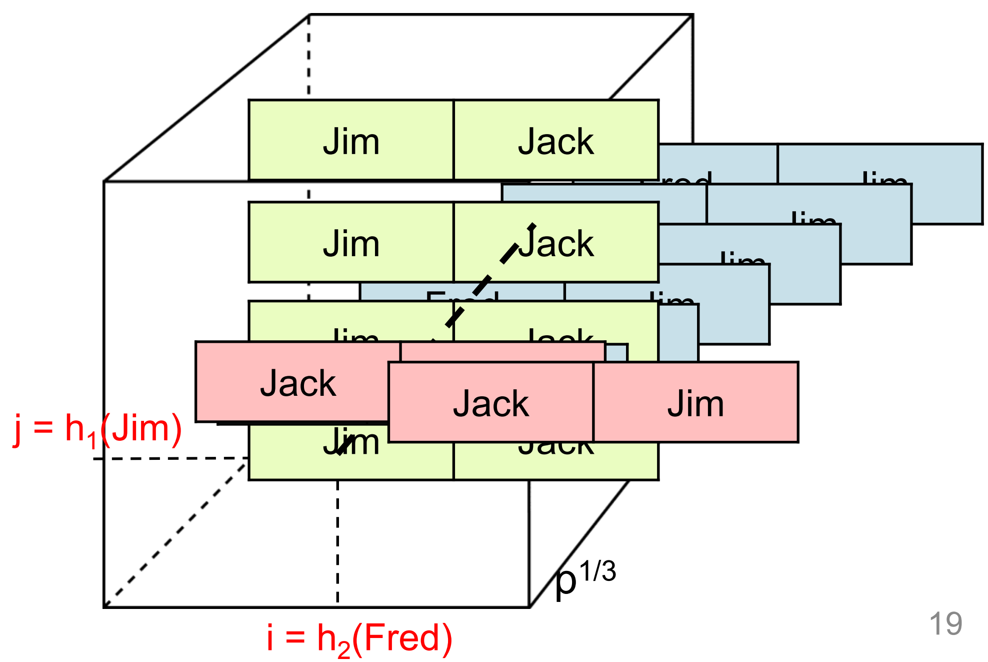
Send R(x,y) to all servers ($h_1(x)$,$h_2(y)$,*)

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)∧S(y,z)∧T(z,x)



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 $L = m/p$?

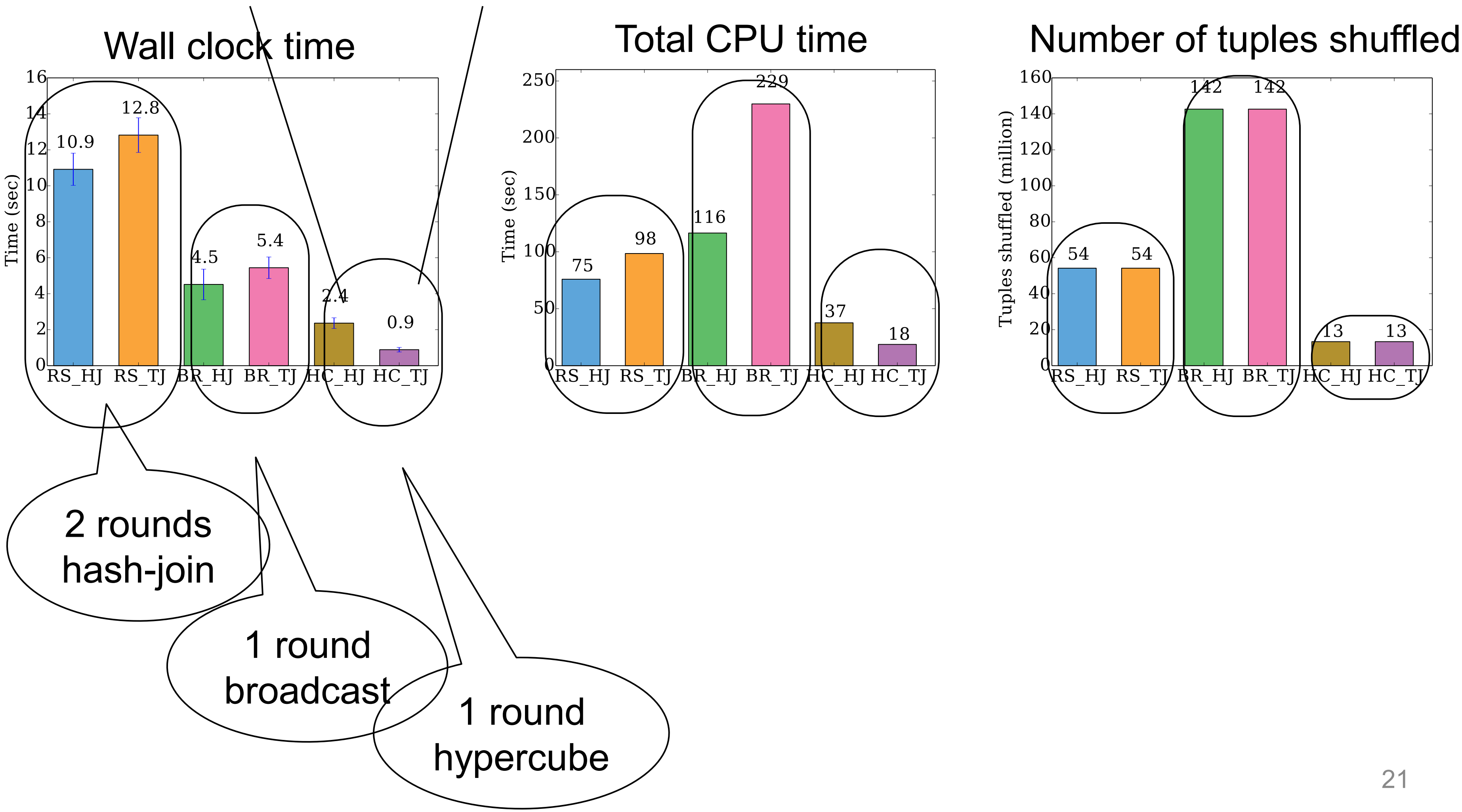
No!

Theorem Any 1-round algo. has $L = \Omega(m/p^{2/3})$, even on inputs with no skew.

Triangles(x,y,z) = R(x,y) ∧ S(y,z) ∧ T(z,x) |R| = |S| = |T| = 1.1M

1.1M triples of Twitter data → 220k triangles; p=64

local 1 or 2-step hash-join; local 1-step Leapfrog Trie-join (a.k.a. Generic-Join)



Triangles(x,y,z) = R(x,y) ∧ S(y,z) ∧ T(z,x) |R| = |S| = |T| = 1.1M

1.1M triples of Twitter data → 220k triangles;   p=64

shuffle	tuples sent	producer skew	consumer skew
R(x, y) ->h(y)	1,114,289	1	1.35
S(y, z) ->h(y)	1,114,289	1	1.72
RS(x, y, z) ->h(z)	50,862,578	20.8	1
T(z, x) ->h(z)	1,114,289	1	1.01
Total	54,205,445	N.A.	N.A.

Table 2: Load balance with regular shuffles in query Q1

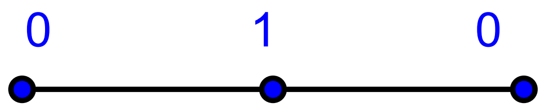
shuffles	tuples sent	producer skew	consumer skew
HCS R(x, y)	4,457,156	1	1.05
HCS S(y, z)	4,457,156	1	1.05
HCS T(z, x)	4,457,156	1	1.05
Total	13,371,468	N.A.	N.A.

Table 3: Load balance with HyperCube shuffles in query Q1

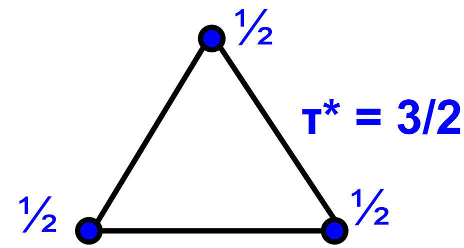
General Case

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

$\tau*$ = fractional vertex cover of Q's hypergraph



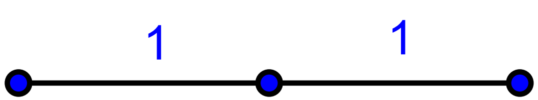
$\tau^* = 1$ m / p



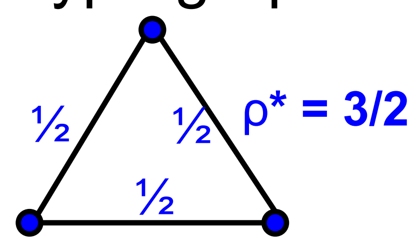
$m / p^{2/3}$

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

$\rho*$ = fractional edge cover of Q's hypergraph



$\rho^* = 2$ $m / p^{1/2}$



$m / p^{2/3}$

Skew

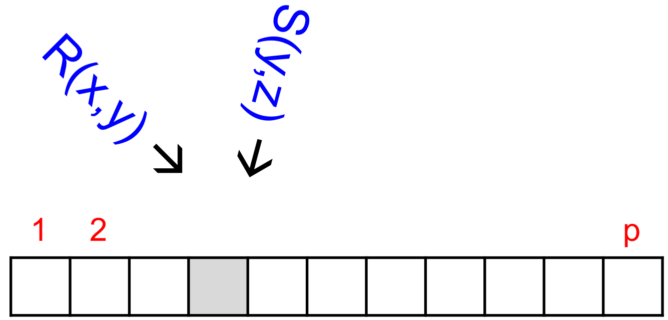
- Skewed data is major impediment to parallel data processing

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

- Our approach:
  - Query → Residual Query
  - Join $R(x,y) \wedge S(y,z)$ → Cartesian Product $R(x) \wedge S(z)$

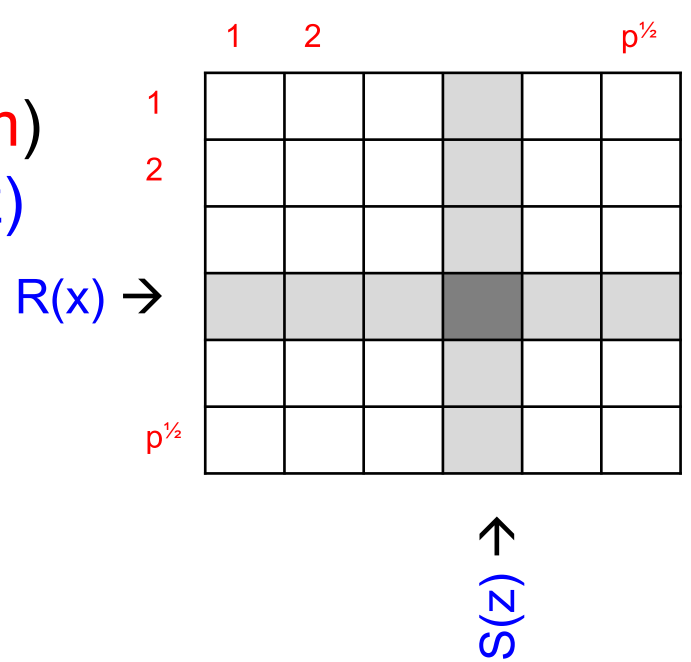
Skewed Values → New Query

Join(x,y,z) = R(x,y) ∧ S(y,z)



No-skew:
$\tau^* = 1$, $L = m/p$

Skewed: (y = single value, degree = m)
Join becomes Product(x,z) = R(x)∧S(z)
$\tau^* = 2$, $L = m/p^{1/2}$



Summary of Results so Far

- 1 Round
  - No skew: optimal load = $m / p^{1/\tau*}$
  - Skew: provably higher
- Multiple rounds
  - Lower bound: load >= $m / p^{1/\rho*}$
  - All relations are binary: optimal load = $m / p^{1/\rho*}$ [PODS'2017a]
  - Arbitrary relations: optimal load = ?? Open
- Additional statistics: keys, degree constraints [PODS'2017b]

Thank you!