Massively Parallel Algorithms for Small Subgraph Counting

Amartya Shankha Biswas
MIT CSAIL

Talya Eden
Boston University/ MIT → Bar-Ilan U

Quanquan C. Liu
Northwestern

Slobodan Mitrović
UC Davis

Ronitt Rubinfeld
MIT CSAIL

To Appear in **APPROX 2022**

Massively Parallel Computation (MPC)

- **Massively parallel systems**
  - Distributed cluster of **multiple machines**

Massively Parallel Computation (MPC)

- **Massively parallel systems**
  - Distributed cluster of **multiple machines**
  - Communicate with each other via **rounds of communication**

Massively Parallel Computation (MPC)

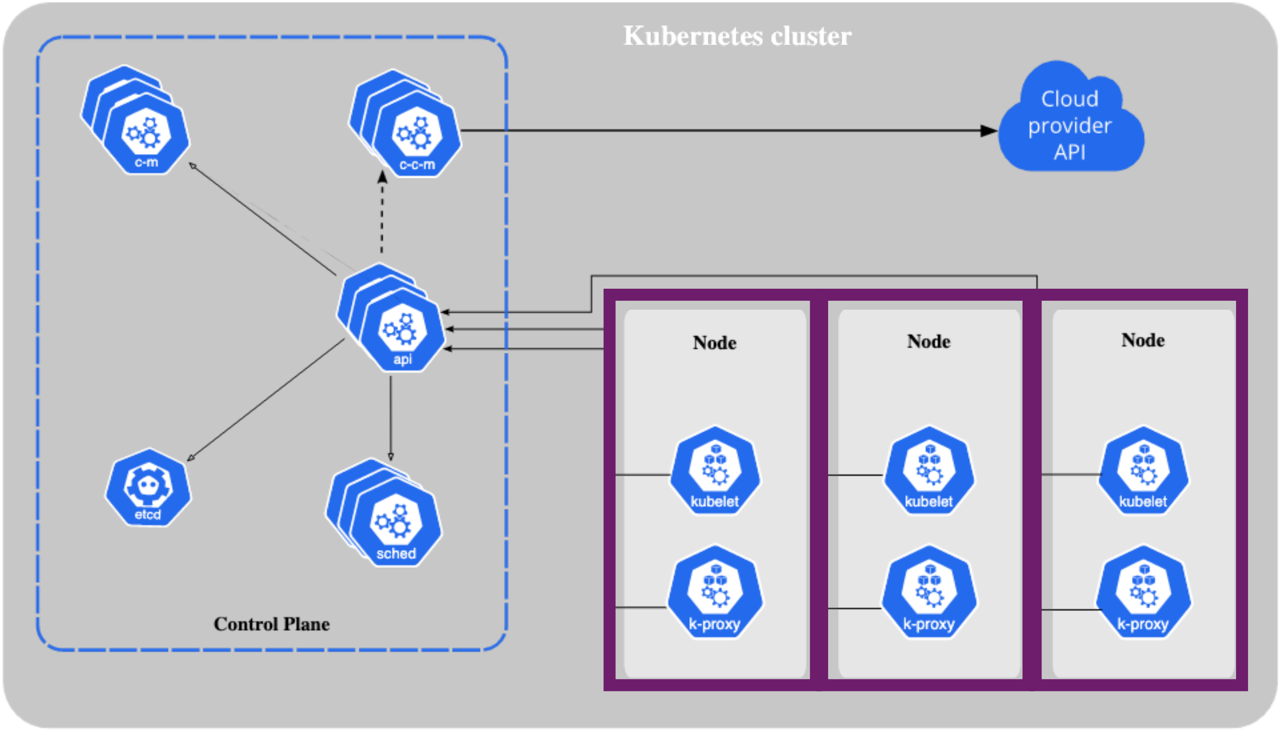
- **Massively parallel systems**
  - Distributed cluster of **multiple machines**
  - Communicate with each other via **rounds of communication**
  - **Limited space** in each individual machine

Commercial Data Centers

Google Kubernetes Engine

Commercial Data Centers

Google Kubernetes Engine



Machine 1 Machine 2 Machine 3

Massively Parallel Computation (MPC) Model

- Theoretical standard for studying parallel frameworks such as MapReduce, Hadoop, Spark, Dryad, and Google Cloud Dataflow

APACHE Spark

Microsoft Dryad

hadoop Map Reduce

Google Cloud Dataflow

Graph Algorithms in MPC Model

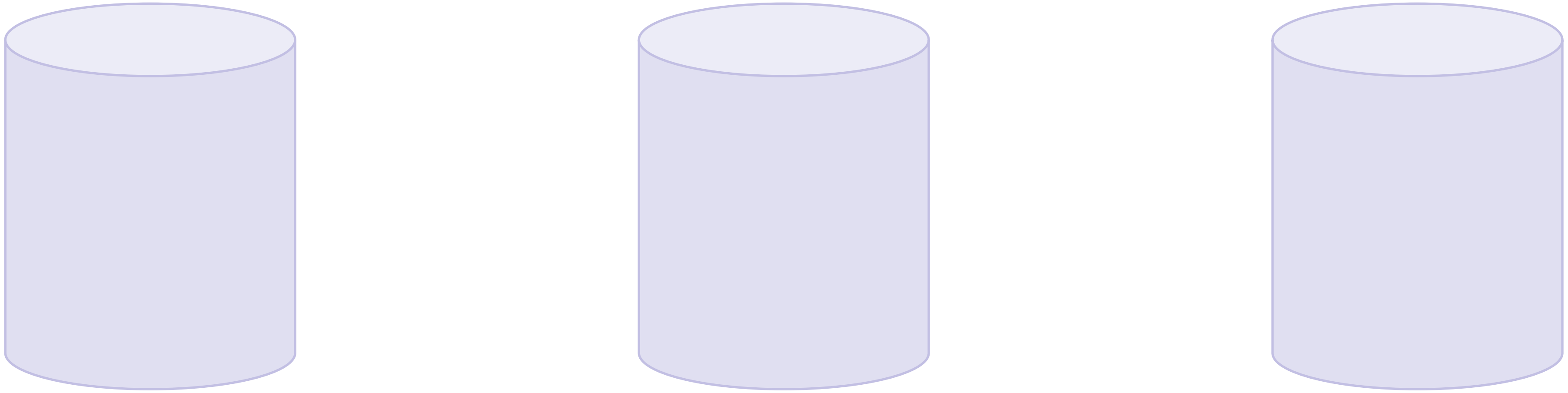
- Matching and MIS [BBDFHKU19, BHH19, GGKMR19, CLMMOS18, NO21]
- Connectivity [ASSWZ18, BDELM19, DDKPSS19]
- Graph sparsification [GU19, CDP20]
- Vertex cover [Assadi17, GGKMR18]
- MST and 2-edge connectivity [NO21]
- Well-connected components [ASW18, ASW19]
- Coloring [BDHKS19, CFGUZ19]

MPC Model Definition

- M machines
- Synchronous rounds

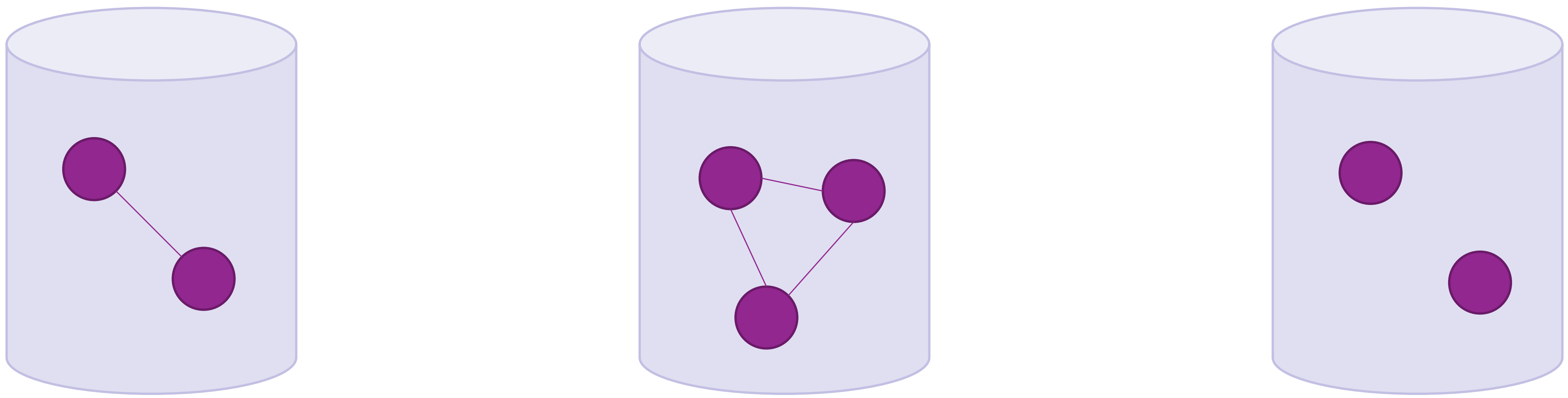
MPC Model Definition

- *M* machines
- Synchronous rounds



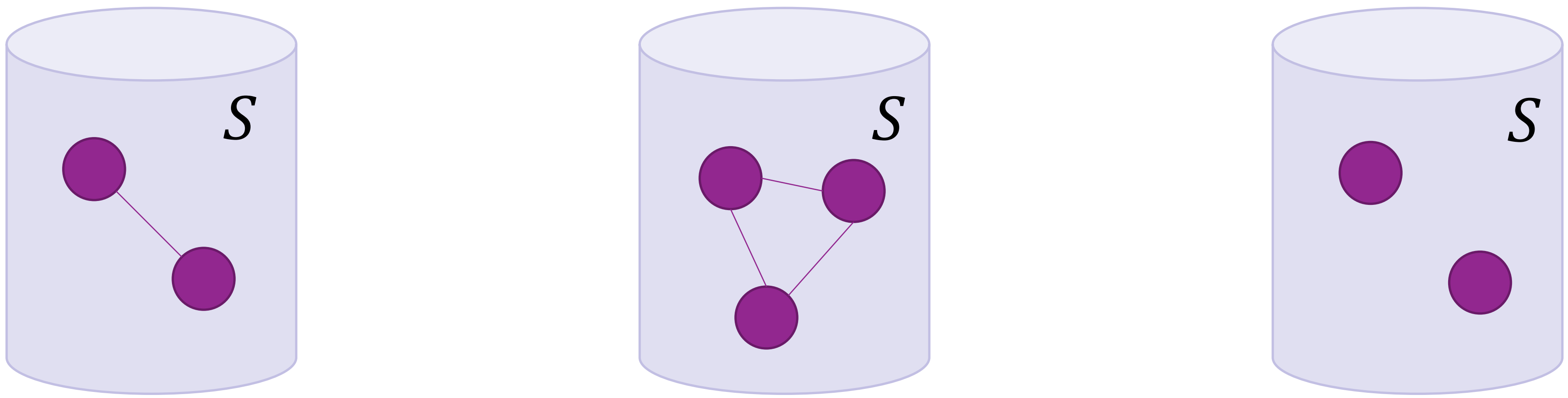
MPC Model Definition

- *M* machines
- Synchronous rounds



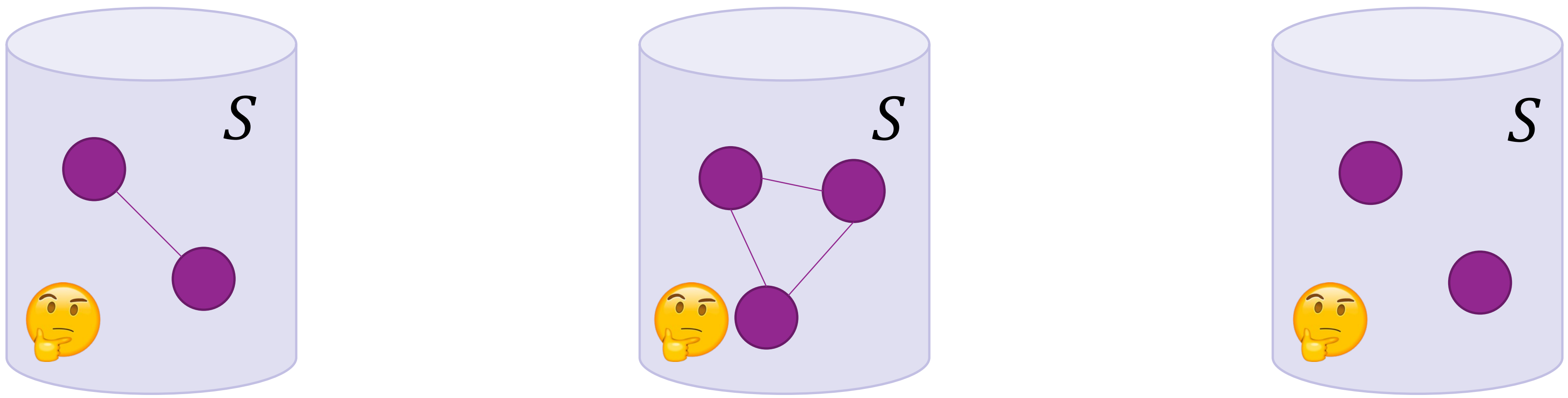
MPC Model Definition

- M machines
- Synchronous rounds



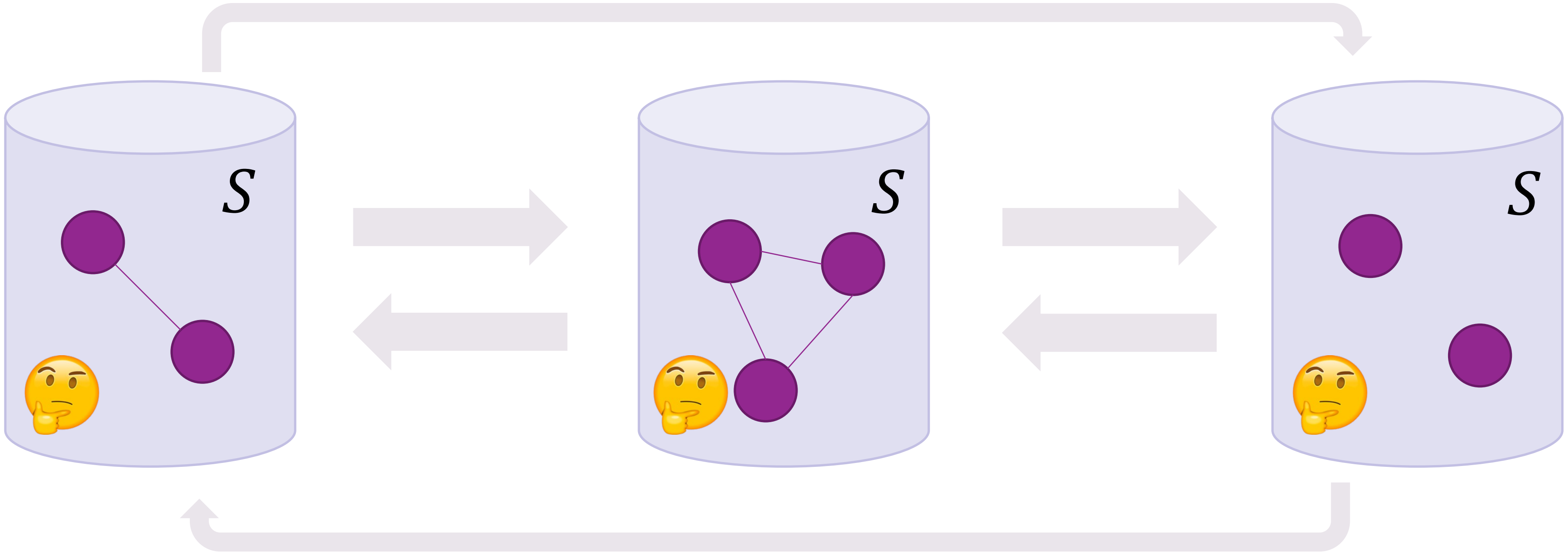
MPC Model Definition

- *M* machines
- Synchronous rounds



MPC Model Definition

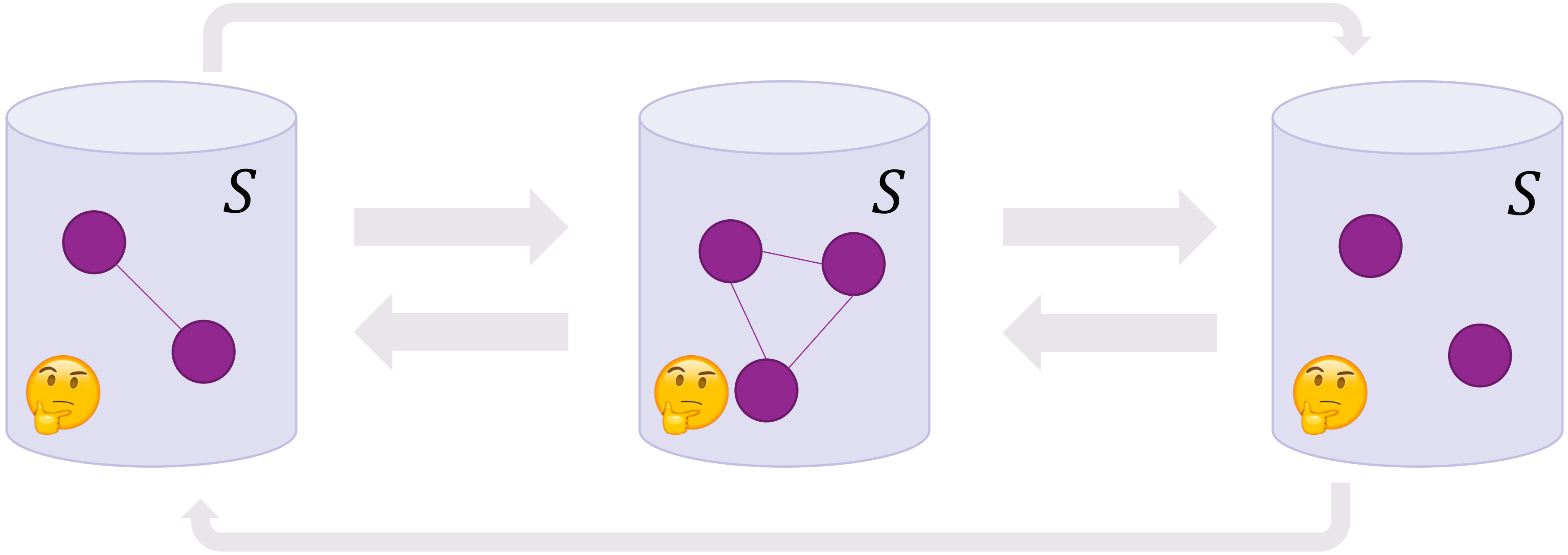
- *M* machines
- Synchronous rounds



MPC Model Definition

- M machines
- Synchronous rounds

Total Space: $\boldsymbol{M \cdot S}$



Space per Machine in MPC

- **Strongly sublinear memory**:
  - $S = n^{\delta}$ for some constant $\delta \in (0, 1)$

Space per Machine in MPC

- **Strongly sublinear memory**:
  - $S = n^{\delta}$ for some constant $\delta \in (0, 1)$
- **Near-linear memory**:
  - $S = \widetilde{\Theta}(n)$ (ignoring $\mathrm{poly}(\log(n))$ factors)

Space per Machine in MPC

- **Strongly sublinear memory**:
  - $S = n^{\delta}$ for some constant $\delta \in (0, 1)$
- **Near-linear memory**:
  - $S = \widetilde{\Theta}(n)$ (ignoring $\mathrm{poly}(\log(n))$ factors)
- **Strongly superlinear memory**:
  - $S = n^{1+\delta}$ for some constant $\delta > 0$

Space per Machine in MPC

- **Strongly sublinear memory**:
  - $S = n^{\delta}$ for some constant $\delta \in (0, 1)$
- **Near-linear memory**:
  - $S = \widetilde{\Theta}(n)$ (ignoring $\mathrm{poly}(\log(n))$ factors)
- **Strongly superlinear memory**:
  - $S = n^{1+\delta}$ for some constant $\delta > 0$

Also want: $\boldsymbol{O}(\log \log \boldsymbol{n})$ or $\boldsymbol{O}(\mathbf{1})$ rounds

Space per Machine in MPC

- **Strongly sublinear memory**:
  - $S = n^{\delta}$ for some constant $\delta \in (0, 1)$
- **Near-linear memory**:
  - $S = \widetilde{\Theta}(n)$ (ignoring $\mathrm{poly}(\log(n))$ factors)
- **Strongly superlinear memory**:
  - $S = n^{1+\delta}$ for some constant $\delta > 0$

Also want: $\boldsymbol{O}(\log \log \boldsymbol{n})$ or $\boldsymbol{O}(\mathbf{1})$ rounds

Also want: $\widetilde{\boldsymbol{O}}(\boldsymbol{n} + \boldsymbol{m})$ total space

Space per Machine in MPC

- **Strongly sublinear memory**:
  - $S = n^{\delta}$ for some constant $\delta \in (0, 1)$
- **Near-linear memory**:
  - $S = \widetilde{\Theta}(n)$ (ignoring $\mathrm{poly}(\log(n))$ factors)
- **Strongly superlinear memory**:
  - $S = n^{1+\delta}$ for some constant $\delta > 0$

Also want: $\boldsymbol{O}(\log \log \boldsymbol{n})$ or $\boldsymbol{O}(\mathbf{1})$ rounds

Also want: $\widetilde{\boldsymbol{O}}(\boldsymbol{n} + \boldsymbol{m})$ total space

All are **sublinear in number of edges *m*** in graph

Triangle Counting in MPC Model

Exact Setting			
Previous Work	MPC Rounds	Space Per Machine	Total Space
[SV11]	1	$O(m/\rho^2)$	$O(\rho m)$
[CC11]	$O(n)$	$O(n)$	$O(m)$
Folklore [CN85]	$O(\log n)$	$O(\alpha^2)$	$O(m\alpha)$
BELMR22	$\boldsymbol{O(\log\log n)}$	$\boldsymbol{O(n^\delta)}$	$\boldsymbol{O(m\alpha)}$

$\boldsymbol{\delta > 0}$ is **any constant**

[SV11]: Suri and Vassilvitski, WWW '11

[CC11]: Chu and Cheng KDD '11

[CN85]: Chiba and Nishizeki SICOMP '85

Triangle Counting in MPC Model

Exact Setting		
Previous Work	MPC Rounds	Space Pe
[SV11]	1	$O(m$
[CC11]	$O(n)$	$O($
Folklore [CN85]	$O(\log n)$	$O($
BELMR22	$\boldsymbol{O(\log \log n)}$	$\boldsymbol{O(}$

Arboricity α: number of forests that edges can be partitioned into

Real-world graphs: **arboricity generally** $\text{poly}(\log \boldsymbol{n})$

$\boldsymbol{\delta > 0}$ is **any constant**

[SV11]: Suri and Vassilvitski, WWW ‘11

[CC11]: Chu and Cheng KDD ’11

[CN85]: Chiba and Nishizeki SICOMP ‘85

Triangle Counting in MPC Model

Arboricity α: number of forests that edges can be partitioned into

Exact Setting			
Previous Work	MPC Rounds	Space Per Machine	Total Space
[SV11]	1	$O(m/\rho^2)$	$O(\rho m)$
[CC11]	$O(n)$	$O(n)$	$O(m)$
Folklore [CN85]	$O(\log n)$	$O(\alpha^2)$	$O(m\alpha)$
BELMR22	$\boldsymbol{O(\log\log n)}$	$\boldsymbol{O(n^{\delta})}$	$\boldsymbol{O(m\alpha)}$

Better space per machine or better total space when $\boldsymbol{\alpha \le m^{1/2-\varepsilon}}$, but worse number of rounds

Triangle Counting in MPC Model

Arboricity α: number of forests that edges can be partitioned into

Exact Setting			
Previous Work	MPC Rounds	Space Per Machine	Total Space
[SV11]	1	$O(m/\rho^2)$	$O(\rho m)$
[CC11]	$O(n)$	$O(n)$	$O(m)$
Folklore [CN85]	$O(\log n)$	$O(\alpha^2)$	$O(m\alpha)$
BELMR22	$\boldsymbol{O(\log\log n)}$	$\boldsymbol{O(n^{\delta})}$	$\boldsymbol{O(m\alpha)}$

Better rounds and space per machine,

but total space when $\alpha = \omega(1)$

Triangle Counting in MPC Model

Arboricity α: number of forests that edges can be partitioned into

Exact Setting			
Previous Work	MPC Rounds	Space Per Machine	Total Space
[SV11]	1	$O(m/\rho^2)$	$O(\rho m)$
[CC11]	$O(n)$	$O(n)$	$O(m)$
Folklore [CN85]	$O(\log n)$	$O(\alpha^2)$	$O(m\alpha)$
BELMR22	$\boldsymbol{O(\log\log n)}$	$\boldsymbol{O(n^\delta)}$	$\boldsymbol{O(m\alpha)}$

Smaller number of rounds, but worse space per machine when $\alpha < n^{o(1)}$

Triangle Counting in MPC Model

Arboricity α: number of forests that edges can be partitioned into

Exact Setting			
Previous Work	MPC Rounds	Space Per Machine	Total Space
[SV11]	1	$O(m/\rho^2)$	$O(\rho m)$
[CC11]	$O(n)$	$O(n)$	$O(m)$
Folklore [CN85]	$O(\log n)$	$O(\alpha^2)$	$O(m\alpha)$
BELMR22	$\boldsymbol{O(\log \log n)}$	$\boldsymbol{O(n^\delta)}$	$\boldsymbol{O(m\alpha)}$

Strictly sublinear setting

Triangle Counting in MPC Model

$(1+\varepsilon)$-Approximate Setting				
Previous Work	MPC Rounds	Space Per Machine	Total Space	Triangles Lower Bound
[PT12]	$O(1)$	$O\left(\frac{m\Delta_e}{T}\right)$	$O(m)$	$\Omega(d_{avg})$
[SPK13]	$O(1)$	$O(n^{\delta})$	$O(m)$	$\Omega\left(\sum_{v\in V} \deg(v)^2\right)$
BELMR22	$\boldsymbol{O(1)}$	$\boldsymbol{\tilde{O}(n)}$	$\boldsymbol{\tilde{O}(m)}$	$\Omega\left(\sqrt{d_{avg}}\right)$

[PT12]: Pagh and Tsourakakis, IPL '12

[SPK13]: Seshadhri, Pinar, Kolda, ICDM '13

Triangle Counting in MPC Model

$(1+\varepsilon)$-Approximate Setting				
Previous Work	MPC Rounds	Space Per Machine	Total Space	Triangles Lower Bound
[PT12]	$O(1)$	$O\left(\frac{m\Delta_e}{T}\right)$	$O(m)$	$\Omega(d_{avg})$
[SPK13]	$O(1)$	$O(n^\delta)$	$O(m)$	$\Omega\left(\sum_{v\in V}\deg(v)^2\right)$
BELMR22	$\boldsymbol{O(1)}$	$\boldsymbol{\tilde{O}(n)}$	$\boldsymbol{\tilde{O}(m)}$	$\Omega\left(\sqrt{d_{avg}}\right)$

Better triangle lower bounds,
but slightly worse total space

Triangle Counting in MPC Model

$(1+\varepsilon)$-Approximate Setting				
Previous Work	MPC Rounds	Space Per Machine	Total Space	Triangles Lower Bound
[PT12]	$O(1)$	$O\left(\frac{m\Delta_e}{T}\right)$	$O(m)$	$\Omega(d_{avg})$
[SPK13]	$O(1)$	$O(n^{\delta})$	$O(m)$	$\Omega\left(\sum_{v\in V}\deg(v)^2\right)$
BELMR22	$\boldsymbol{O(1)}$	$\tilde{\boldsymbol{O}}(\boldsymbol{n})$	$\tilde{\boldsymbol{O}}(\boldsymbol{m})$	$\Omega\left(\sqrt{d_{avg}}\right)$

Worse space per machine than SPK13

Triangle Counting in MPC Model

$(1+\varepsilon)$-Approximate Setting				
Previous Work	MPC Rounds	Space Per Machine	Total Space	Triangles Lower Bound
[PT12]	$O(1)$	$O\left(\frac{m\Delta_e}{T}\right)$	$O(m)$	$\Omega(d_{avg})$
[SPK13]	$O(1)$	$O(n^{\delta})$	$O(m)$	$\Omega\left(\sum_{v\in V} \deg(v)^2\right)$
BELMR22	$\boldsymbol{O}(\mathbf{1})$	$\tilde{\boldsymbol{O}}(\boldsymbol{n})$	$\tilde{\boldsymbol{O}}(\boldsymbol{m})$	$\Omega\left(\sqrt{d_{avg}}\right)$

Better space per machine than PT12 when $\boldsymbol{n = o\left(\frac{m\Delta_e}{T}\right)}$

Results in This Presentation

- Strongly sublinear memory:
    - **Exact** triangle counting:
        - Bounded arboricity
        - $O(\log \log n)$ rounds
        - $O(m\alpha)$ total space

Results in This Presentation

- Strongly sublinear memory:
  - **Exact** triangle counting:
    - Bounded arboricity
    - $O(\log\log n)$ rounds
    - $O(m\alpha)$ total space
- Near-linear memory:
  - **Approximate** triangle counting
  - $(1+\varepsilon)$-approximation when $T \geq \sqrt{m/n}$
  - $O(1)$ rounds, $\tilde{O}(n+m)$ total space

Results in This Presentation

- Strongly sublinear memory:
  - **Exact** triangle counting:
    - Bounded arboricity
    - $O(\log \log n)$ rounds
    - $O(m\alpha)$ total space
- Near-linear memory:
  - **Approximate** triangle counting
  - $(1 + \varepsilon)$-approximation when $T \geq \sqrt{m/n}$
  - $O(1)$ rounds, $\tilde{O}(n + m)$ total space

Results in Our Paper

- Strongly sublinear memory:
  - Extensions to clique counting

Results in This Presentation

- Strongly sublinear memory:
  - **Exact** triangle counting:
    - Bounded arboricity
    - $O(\log \log n)$ rounds
    - $O(m\alpha)$ total space
- Near-linear memory:
  - **Approximate** triangle counting
  - $(1+\varepsilon)$-approximation when $T \geq \sqrt{m/n}$
  - $O(1)$ rounds, $\tilde{O}(n+m)$ total space

Results in Our Paper

- Strongly sublinear memory:
  - Extensions to clique counting
- Linear memory:
  - Extensions to clique counting

Results in This Presentation

- Strongly sublinear memory:
  - **Exact** triangle counting:
    - Bounded arboricity
    - $O(\log \log n)$ rounds
    - $O(m\alpha)$ total space
- Near-linear memory:
  - **Approximate** triangle counting
  - $(1+\varepsilon)$-approximation when $T \geq \sqrt{m/n}$
  - $O(1)$ rounds, $\tilde{O}(n+m)$ total space

Results in Our Paper

- Strongly sublinear memory:
  - Extensions to clique counting
- Linear memory:
  - Extensions to clique counting
- Counting **all** small subgraphs of size at most 5 using Bera, Pashanasangi and Seshadhri [ITCS 2020]

Results in This Presentation

- Strongly sublinear memory:
  - **Exact** triangle counting:
    - Bounded arboricity
    - $O(\log\log n)$ rounds
    - $O(m\alpha)$ total space
- Near-linear memory:
  - **Approximate** triangle counting
  - $(1+\varepsilon)$-approximation when $T \geq \sqrt{m/n}$
  - $O(1)$ rounds, $\tilde{O}(n+m)$ total space

Results in Our Paper

- Strongly sublinear memory:
  - Extensions to clique counting
- Linear memory:
  - Extensions to clique counting
- Counting **all** small subgraphs of size at most 5 using Bera, Pashanasangi and Seshadhri [ITCS 2020]
- Simulations on real-world graphs:
  - Improvements in number of rounds
  - Improvements in approximation

Results in This Presentation

- Strongly sublinear memory:
  - **Exact** triangle counting:
    - Bounded arboricity
    - $O(\log \log n)$ rounds
    - $O(m\alpha)$ total space
- Near-linear memory:
  - **Approximate** triangle counting
  - $(1+\varepsilon)$-approximation when $T \geq \sqrt{m/n}$
  - $O(1)$ rounds, $\tilde{O}(n+m)$ total space

Results in Our Paper

- Strongly sublinear memory:
  - Extensions to clique counting
- Linear memory:
  - Extensions to clique counting
- Counting **all** small subgraphs of size at most 5
- Simulations on real-world graphs:
  - Improvements in number of rounds
  - Improvements in approximation

Exact Triangle Counting Bounded Arboricity

Arboricity α: number of forests that edges can be partitioned into

Exact Triangle Counting Bounded Arboricity

There exists a MPC algorithm that outputs the exact count of triangles in a graph with arboricity α in $\boldsymbol{O(\log \log n)}$ **rounds**, $\boldsymbol{O(n^{\delta})}$ **space per machine** for any constant $\delta > 0$ and $\boldsymbol{O(m\alpha)}$ **total space**.

Massively Parallel Algorithms for Small Subgraph Counting
Amartya Shankha Biswas, Talya Eden, Quanquan C. Liu, Slobodan Mitrovic, Ronitt Rubinfeld
[arxiv.org/2002.08299]

Arboricity α: number of forests that edges can be partitioned into

Exact Triangle Counting Bounded Arboricity

There exists a MPC algorithm that outputs the exact count of triangles in a graph with arboricity α in $\boldsymbol{O(\log\log n)}$ **rounds**, $\boldsymbol{O(n^{\delta})}$ **space per machine** for any constant $\delta > 0$ and $\boldsymbol{O(m\alpha)}$ **total space**.

Massively Parallel Algorithms for Small Subgraph Counting

Amartya Shankha Biswas, Talya Eden, Quanquan C. Liu, Slobodan Mitrovic, Ronitt Rubinfeld

[arxiv.org/2002.08299]

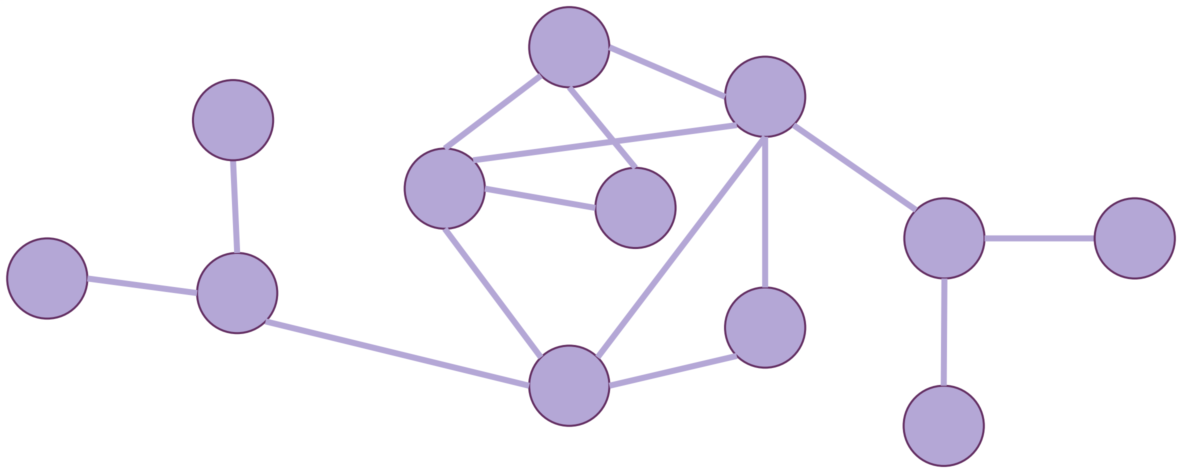
Standard Triangle Counting:
$\boldsymbol{O(\log n)}$ **rounds**
$\boldsymbol{\Omega(\alpha^2)}$ **space per machine**
$\boldsymbol{O(m\alpha)}$ **total space**

$\alpha \leq \sqrt{m}$

Arboricity α: number of forests that edges can be partitioned into

Sequential Triangle Algorithms Directly to MPC

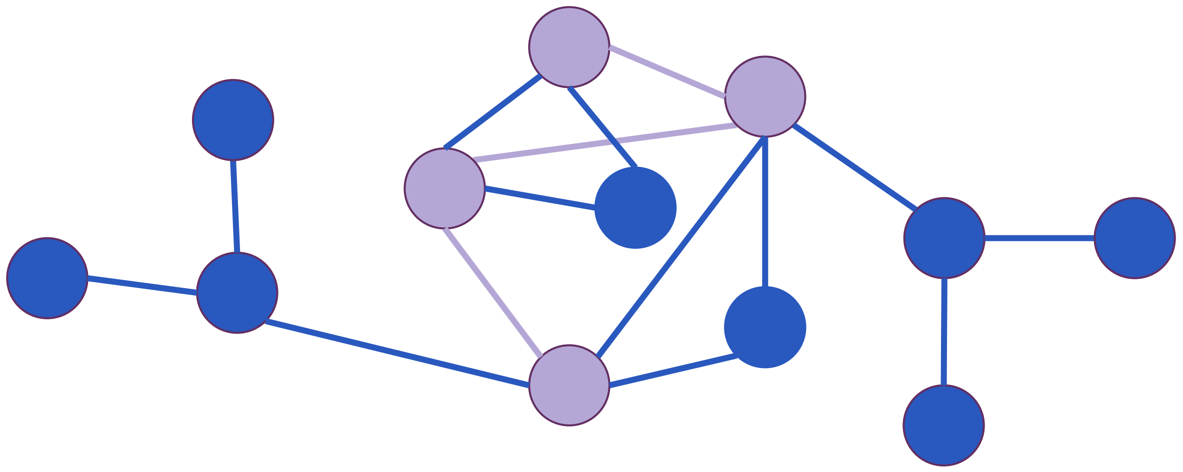
$\alpha = 2$



- Successively remove vertices with degree less than 2α and count number of triangles adjacent to the removed vertices
  - Maintain total count

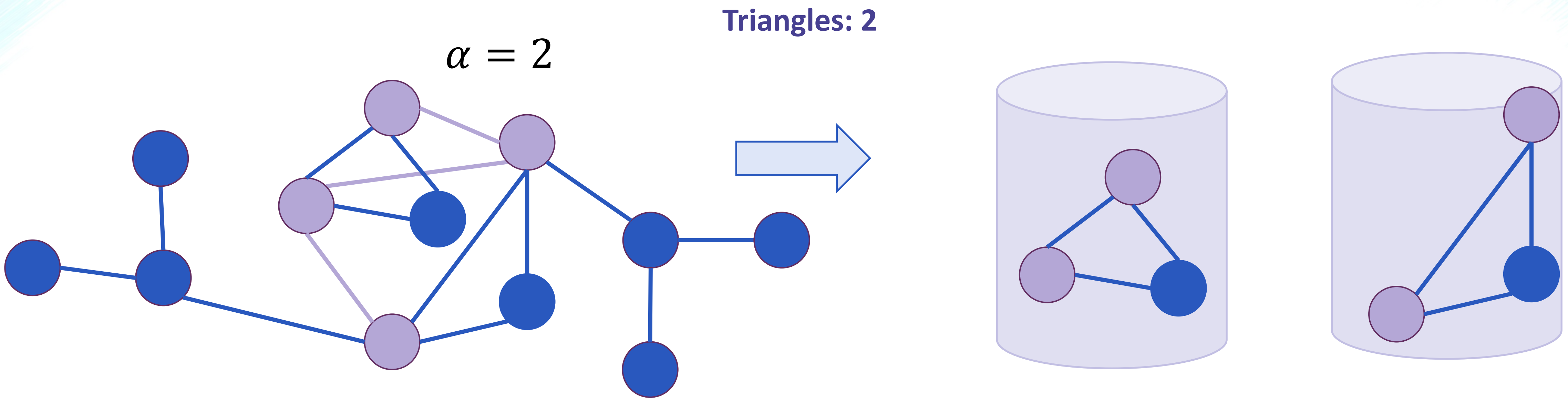
Sequential Triangle Algorithms Directly to MPC

$\alpha = 2$



- Successively remove vertices with degree less than 2α and count number of triangles adjacent to the removed vertices
  - Maintain total count

Sequential Triangle Algorithms Directly to MPC

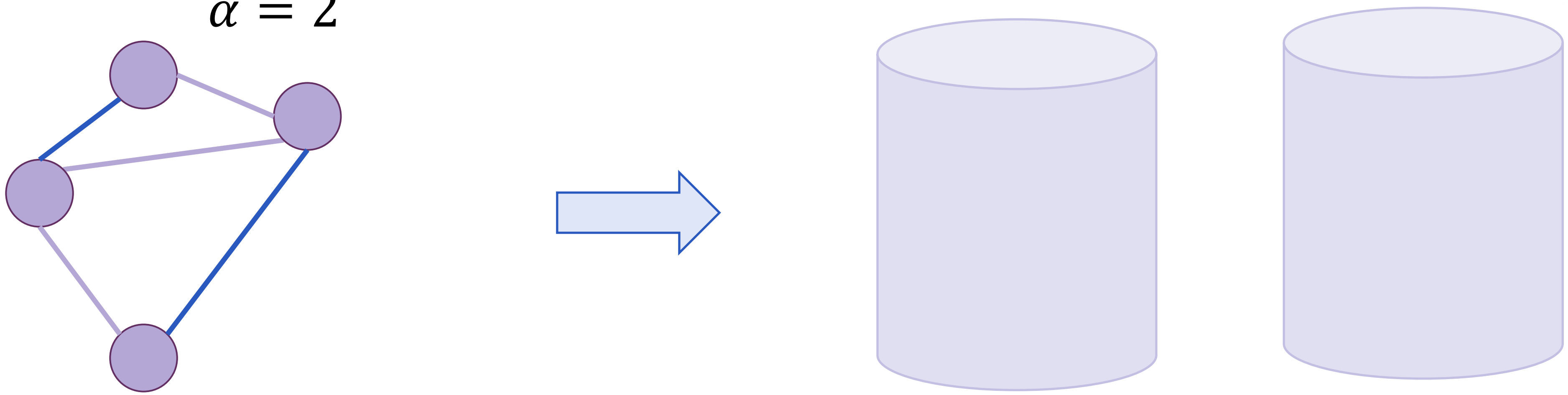


- Successively remove vertices with degree less than 2α and count number of triangles adjacent to the removed vertices
    - Maintain total count

Sequential Triangle Algorithms Directly to MPC

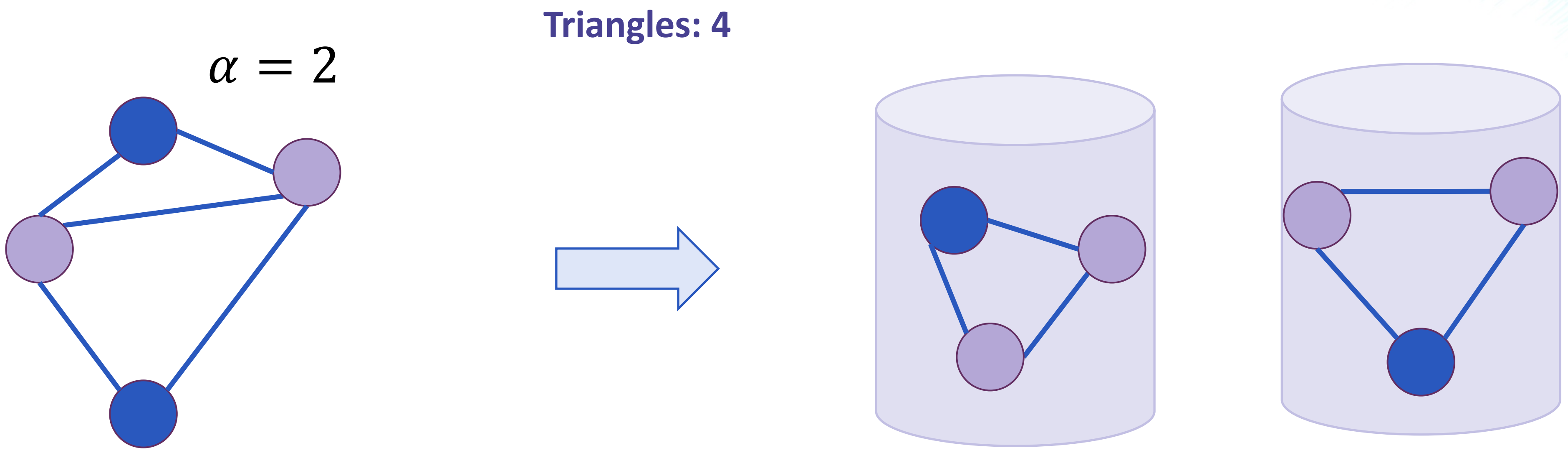
Triangles: 2

$\alpha = 2$



- Successively remove vertices with degree less than 2α and count number of triangles adjacent to the removed vertices
- Maintain total count

Sequential Triangle Algorithms Directly to MPC



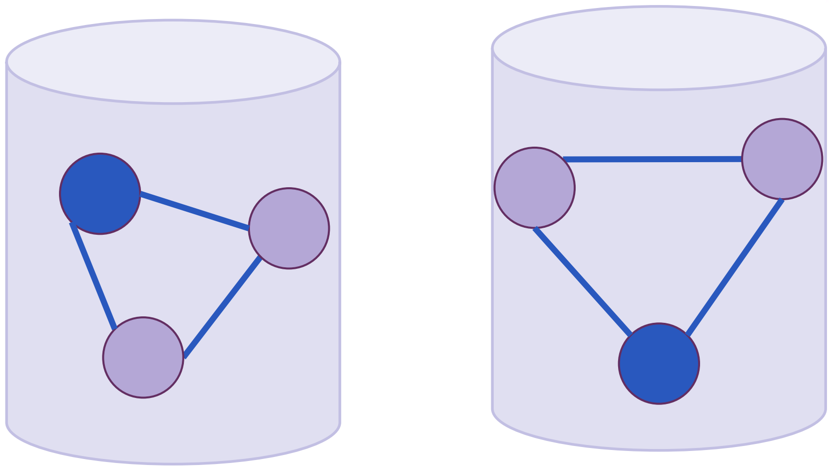
- Successively remove vertices with degree less than 2α and count number of triangles adjacent to the removed vertices
  - Maintain total count

Sequential Triangle Algorithms Directly to MPC

Maximum number of edges in the graph: $m \leq n\alpha$

Number of vertices remaining: $\frac{n\alpha}{2\alpha} = \frac{n}{2}$

Number of rounds needed: $O(\log n)$



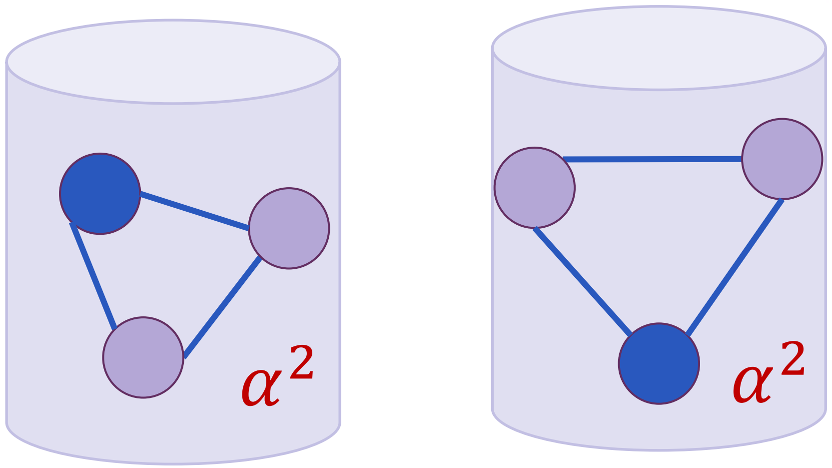
- Successively remove vertices with degree less than 2α and count number of triangles adjacent to the removed vertices
- Maintain total count

Sequential Triangle Algorithms Directly to MPC

Maximum number of edges in the graph: $m \leq n\alpha$

Number of vertices remaining: $\frac{n\alpha}{2\alpha} = \frac{n}{2}$

Number of rounds needed: $O(\log n)$



- Successively remove vertices with degree less than 2α and count number of triangles adjacent to the removed vertices
- Maintain total count

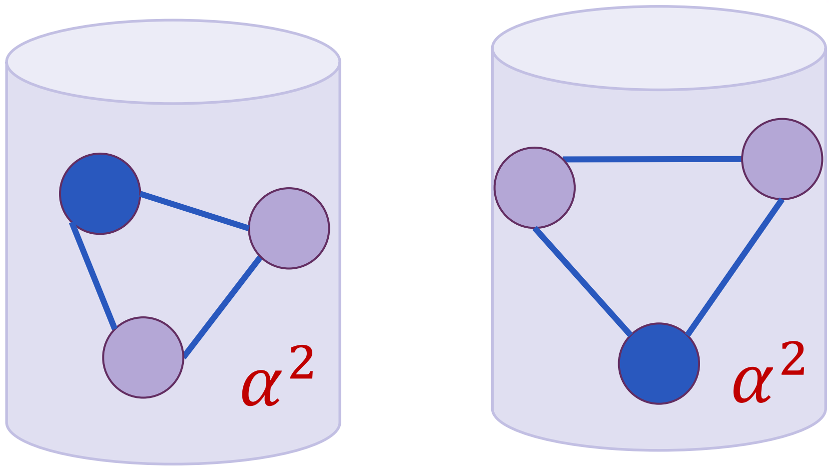
Sequential Triangle Algorithms Directly to MPC

Total space used: $O(m\alpha)$

Maximum number of edges in the graph: $m \leq n\alpha$

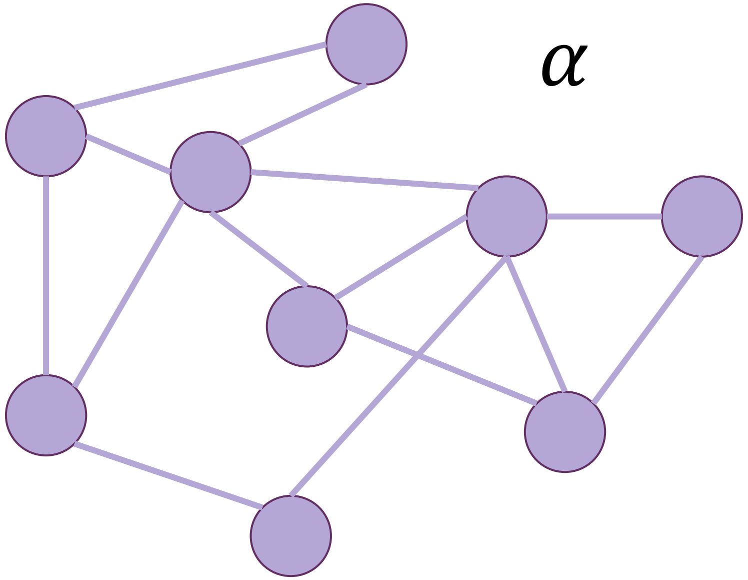
Number of vertices remaining: $\frac{n\alpha}{2\alpha} = \frac{n}{2}$

Number of rounds needed: $O(\log n)$



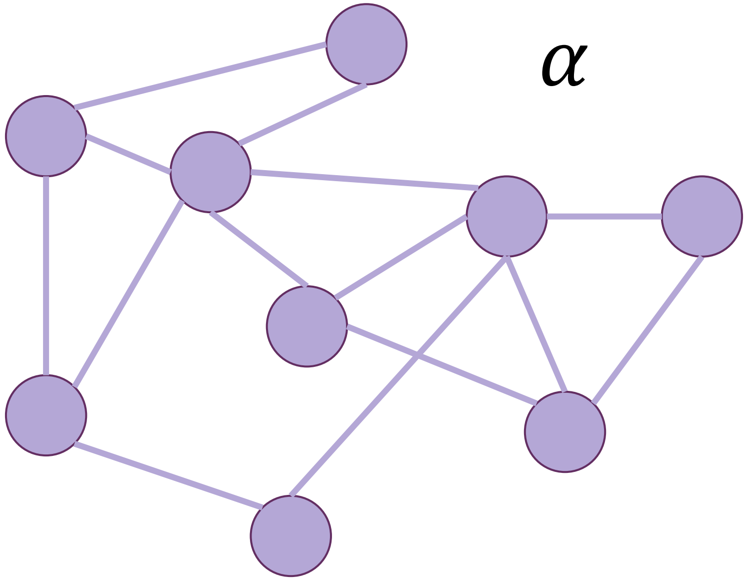
- Successively remove vertices with degree less than 2α and count number of triangles adjacent to the removed vertices
- Maintain total count

Our Exact Triangle Counting Algorithm



α

Our Exact Triangle Counting Algorithm

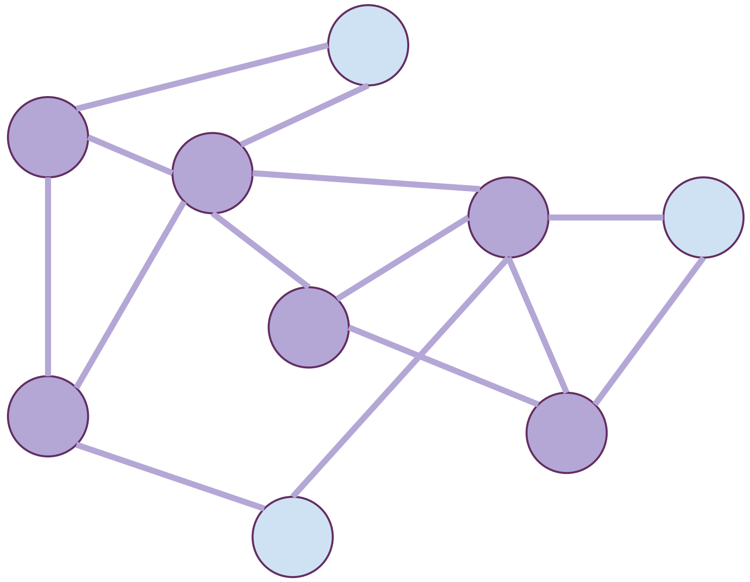


α

$i = 0$

$$\deg(v) \leq 2^{\left(\frac{3}{2}\right)^i} \cdot 2\alpha$$

Our Exact Triangle Counting Algorithm

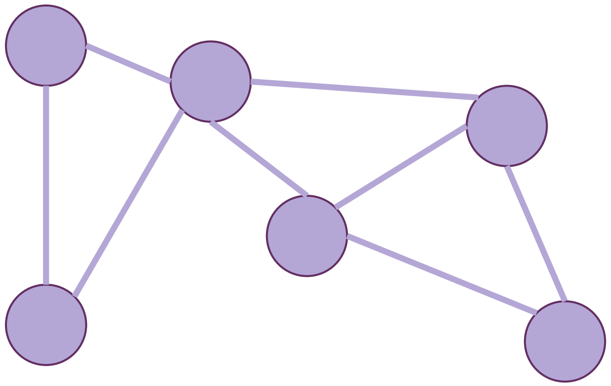


$i = 0$

$\deg(v) \leq 4\alpha$

Our Exact Triangle Counting Algorithm

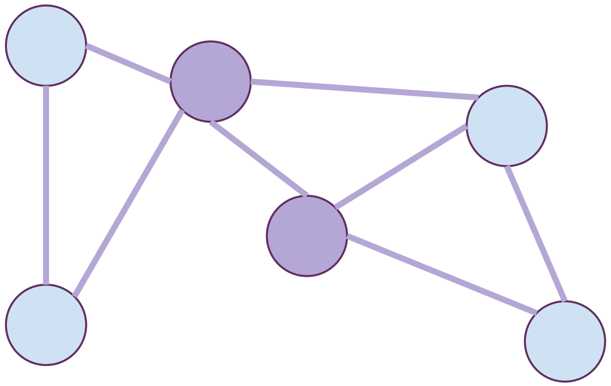
$$\deg(v) \leq 4\alpha$$



$$i = 0$$

2 Triangles

Our Exact Triangle Counting Algorithm

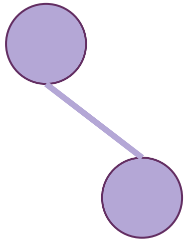


$i = 1$

$$\deg(v) \leq 6\alpha$$

2 Triangles

Our Exact Triangle Counting Algorithm

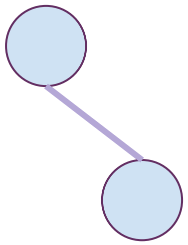


$i = 1$

$\deg(v) \leq 6\alpha$

5 Triangles

Our Exact Triangle Counting Algorithm



$i = 2$

$\deg(v) \leq 10\alpha$

5 Triangles

Our Exact Triangle Counting Algorithm

$$O(\log \log n)$$

$i = 2$

$\deg(v) \leq 10\alpha$

5 Triangles

Our Exact Triangle Counting Algorithm

- Number of vertices left after first round: X

Our Exact Triangle Counting Algorithm

- Number of vertices left after first round: X
- Total number of edges left after first round:

$$m \geq \frac{1}{2} \cdot X \cdot 4\alpha = 2\alpha X$$

Our Exact Triangle Counting Algorithm

- Number of vertices left after first round: X
- Total number of edges left after first round:

$$m \geq \frac{1}{2} \cdot X \cdot 4\alpha = 2\alpha X$$

$$m_1 \leq X\alpha$$

Our Exact Triangle Counting Algorithm

- Number of vertices left after first round: X
- Total number of edges left after first round:

$$m \geq \frac{1}{2} \cdot X \cdot 4\alpha = 2\alpha X$$

$$m_1 \leq X\alpha$$

$$m_1 \leq \frac{m}{2}$$

Our Exact Triangle Counting Algorithm

- Number of vertices left after i-th round: X
- Total number of edges left after first round:

$$m \geq \frac{1}{2} \cdot X \cdot 4\alpha = 2\alpha X$$

$$m_1 \leq X\alpha$$

$$m_1 \leq \frac{m}{2}$$

$$\boldsymbol{m_{i-1}} \geq \frac{1}{2} \cdot X \cdot \boldsymbol{2^{\left(\frac{3}{2}\right)^{i-1}}} \cdot \boldsymbol{2\alpha}$$

$$m_i \leq X \cdot \alpha$$

$$m_i \leq \frac{m_{i-1}}{2^{\left(\frac{3}{2}\right)^{i-1}}} < \frac{m}{2^{\left(\frac{3}{2}\right)^{i}}}$$

Our Exact Triangle Counting Algorithm

- Number of vertices left after i-th round: X
- Total

$$\boldsymbol{m_i \cdot \left(2^{\left(\frac{3}{2}\right)^i} \cdot 2\alpha\right)} \leq \frac{m}{2^{\left(\frac{3}{2}\right)^i}} \cdot \left(2^{\left(\frac{3}{2}\right)^i} \cdot 2\alpha\right) = 2m\alpha$$

$$m \geq \frac{1}{2}$$

$$m_1 \leq X\alpha$$

$$m_1 \leq \frac{m}{2}$$

$$\cdot\, \boldsymbol{2\alpha}$$

$$m_i \leq X \cdot \alpha$$

$$m_i \leq \frac{m_{i-1}}{2^{\left(\frac{3}{2}\right)^{i-1}}} < \frac{m}{2^{\left(\frac{3}{2}\right)^i}}$$

Our Exact Triangle Counting Algorithm

- Number of vertices left after i-th round: X
- Total

$$\boldsymbol{m_i \cdot \left(2^{\left(\frac{3}{2}\right)^i} \cdot 2\alpha\right)} \leq \frac{m}{2^{\left(\frac{3}{2}\right)^i}} \cdot \left(2^{\left(\frac{3}{2}\right)^i} \cdot 2\alpha\right) = 2m\alpha$$

$$m \geq \frac{1}{2}$$

$$m_1 \leq X\alpha$$

$$m_1 \leq \frac{m}{2}$$

$$\cdot \boldsymbol{2\alpha}$$

$$m_i \leq X \cdot \alpha$$

$$m_i \leq \frac{m_{i-1}}{2^{\left(\frac{3}{2}\right)^{i-1}}} < \frac{m}{2^{\left(\frac{3}{2}\right)^i}}$$

Our Exact Triangle Counting Algorithm

- Number of vertices left after i-th round: X
- Total

$$\boldsymbol{m_i} \cdot \left(\boldsymbol{2}^{\left(\frac{\boldsymbol{3}}{\boldsymbol{2}}\right)^{\boldsymbol{i}}} \cdot \boldsymbol{2\alpha}\right) \leq \frac{m}{2^{\left(\frac{3}{2}\right)^{i}}} \cdot \left(2^{\left(\frac{3}{2}\right)^{i}} \cdot 2\alpha\right) = \boldsymbol{2m\alpha}$$

$$m \geq \frac{1}{2}$$

$$\boldsymbol{2\alpha}$$

$$m_i \leq X \cdot \alpha$$

$$m_1 \leq X\alpha$$

$$m_1 \leq \frac{m}{2}$$

$$m_i \leq \frac{m_{i-1}}{2^{\left(\frac{3}{2}\right)^{i-1}}} < \frac{m}{2^{\left(\frac{3}{2}\right)^{i}}}$$

Exact Triangle Counting Space Per Machine

- Last Challenge: Cannot count on one machine because that is too much space

Exact Triangle Counting Space Per Machine

- Last Challenge: Cannot count on one machine because that is too much space
  - Solution: Reduce to a problem where we merge several lists, sort, and find duplicates

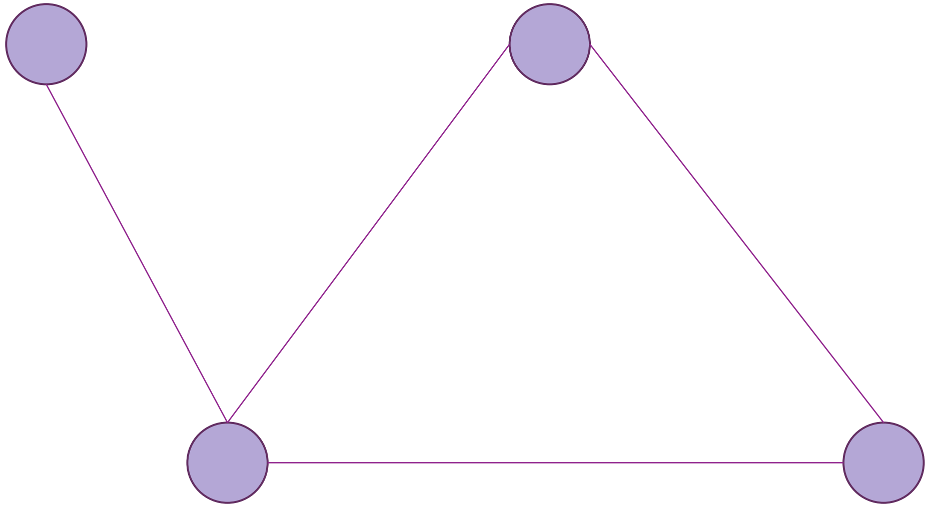
Exact Triangle Counting Space Per Machine

- Last Challenge: Cannot count on one machine because that is too much space
  - Solution: Reduce to a problem where we merge several lists, sort, and find duplicates
    - Every removed node sends its adjacency list to its neighbors

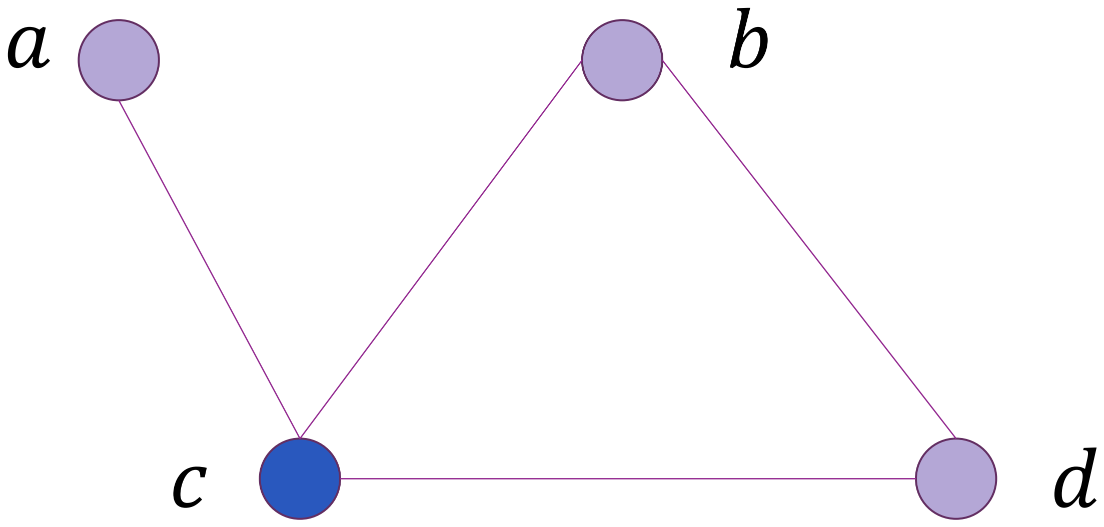
Exact Triangle Counting Space Per Machine

- Last Challenge: Cannot count on one machine because that is too much space
  - Solution: Reduce to a problem where we merge several lists, sort, and find duplicates
    - Every removed node sends its adjacency list to its neighbors
    - Each neighbor which receives adjacency lists merges received lists with its own adjacency list

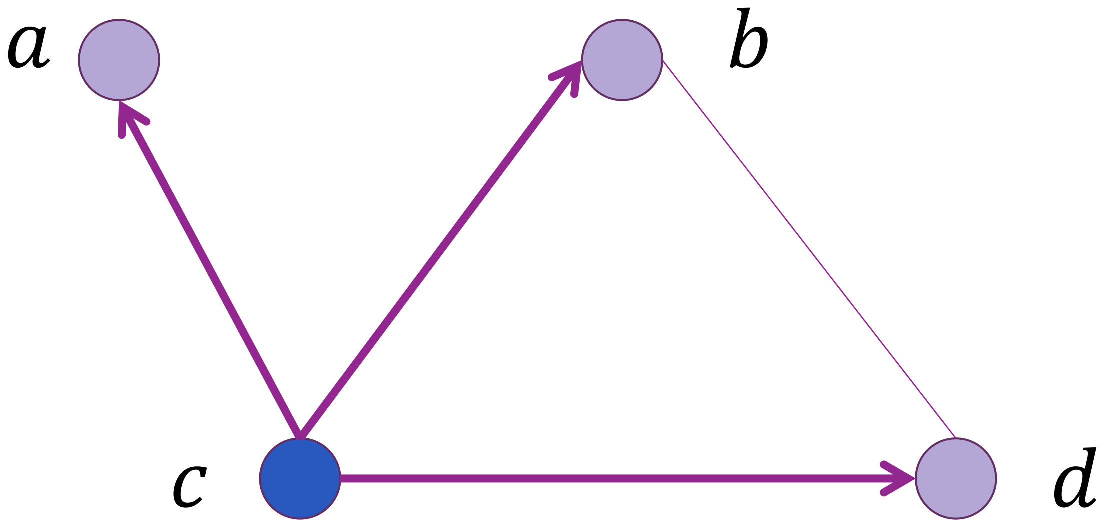
Exact Triangle Counting Space Per Machine



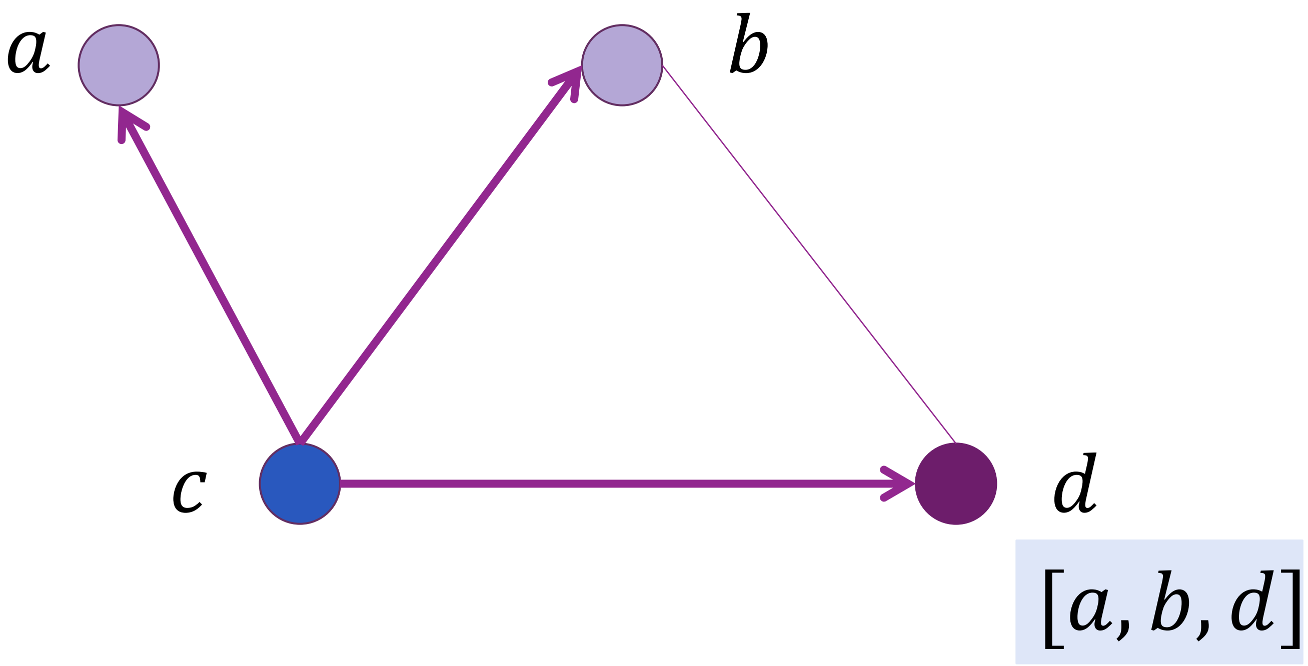
Exact Triangle Counting Space Per Machine



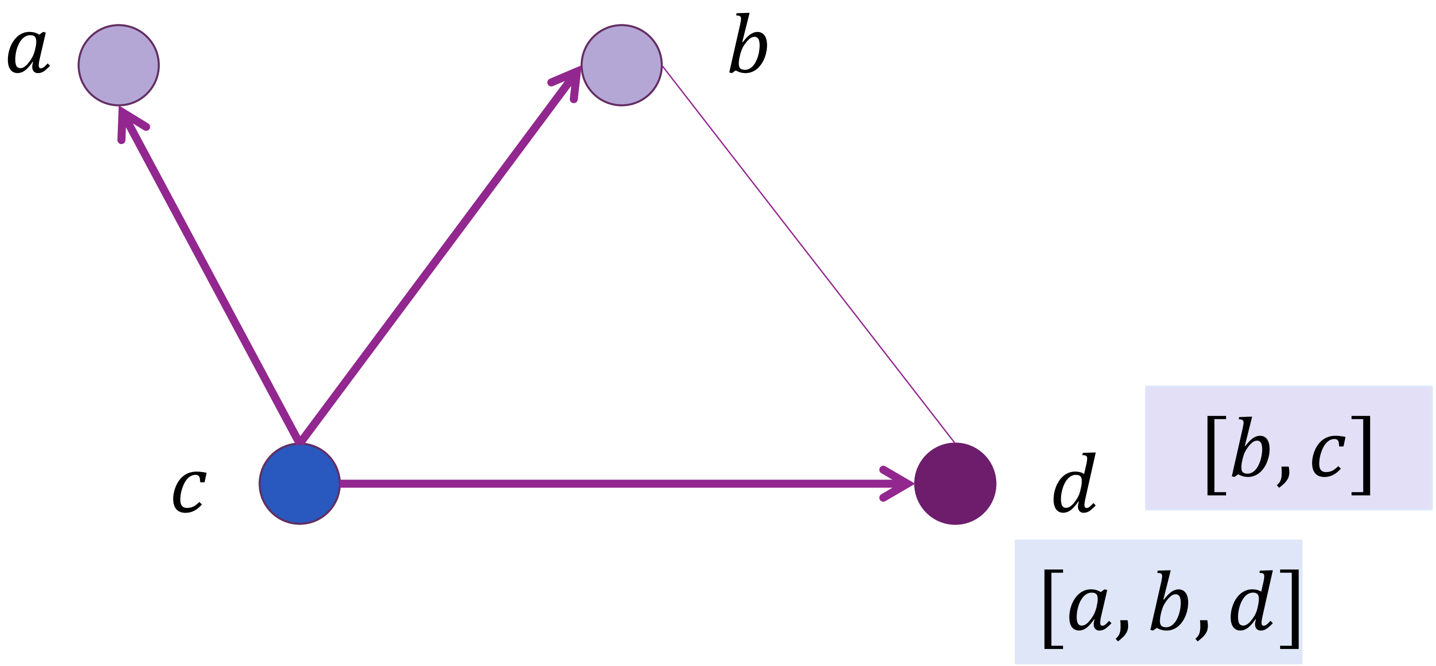
Exact Triangle Counting Space Per Machine



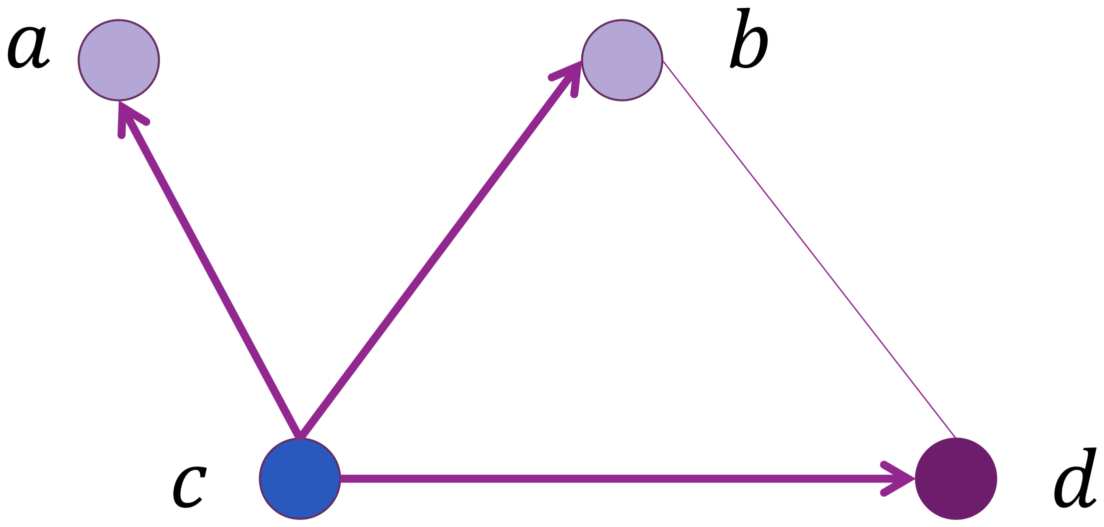
Exact Triangle Counting Space Per Machine



Exact Triangle Counting Space Per Machine



Exact Triangle Counting Space Per Machine



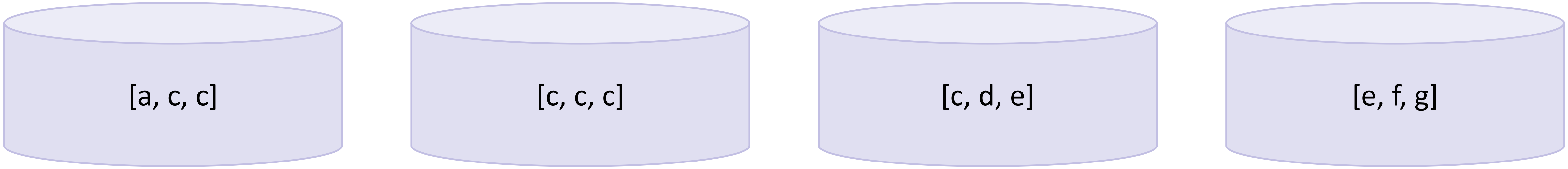
$[a, b, b, c, d]$

Exact Triangle Counting Space Per Machine

- MPC sorting algorithm of [GSZ11] to sort lists in $O(1)$ rounds
- Find duplicates using new MPC primitive

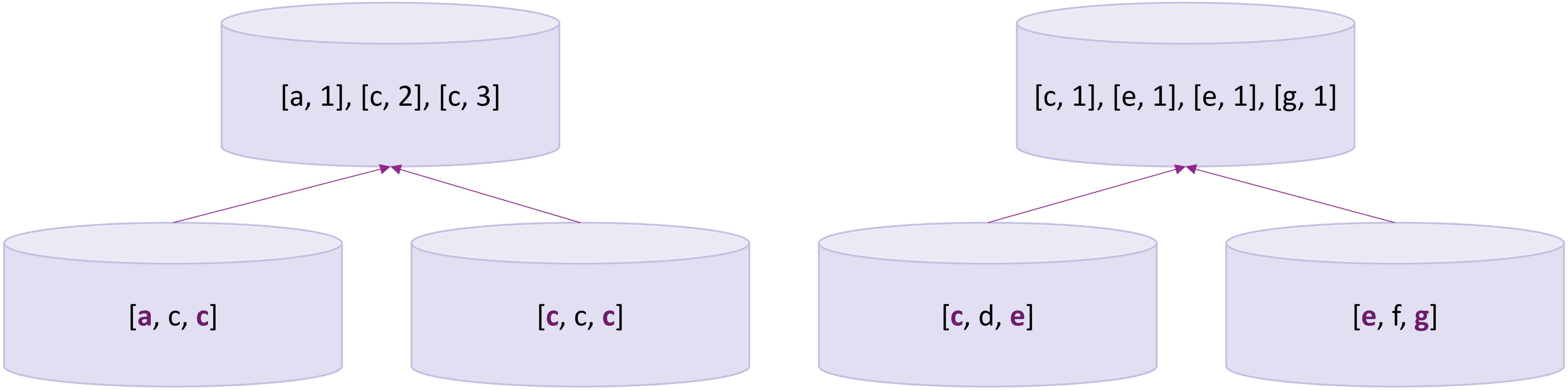
Exact Triangle Counting Space Per Machine

- Find duplicates using new MPC primitive



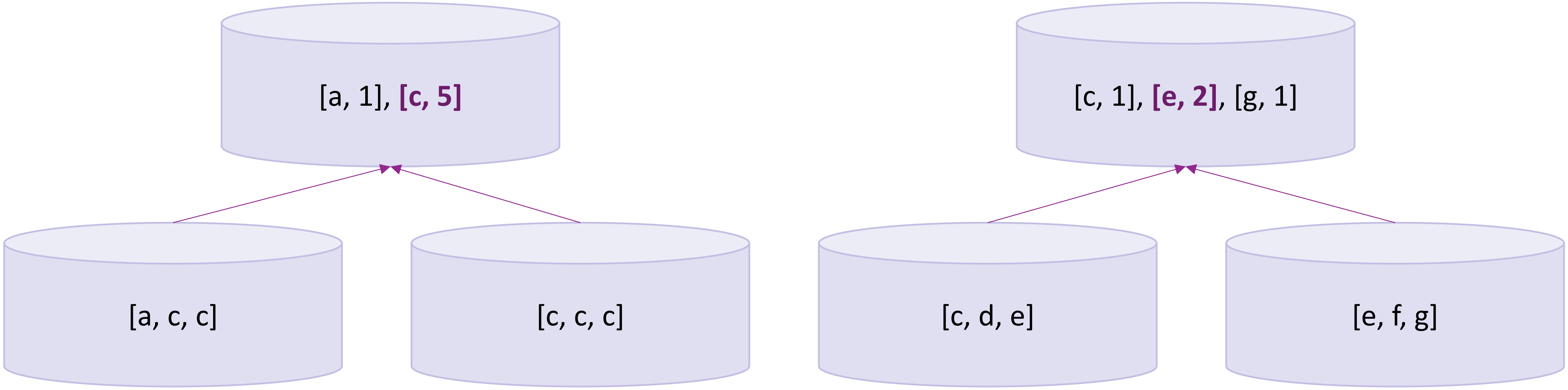
Exact Triangle Counting Space Per Machine

- Find duplicates using new MPC primitive



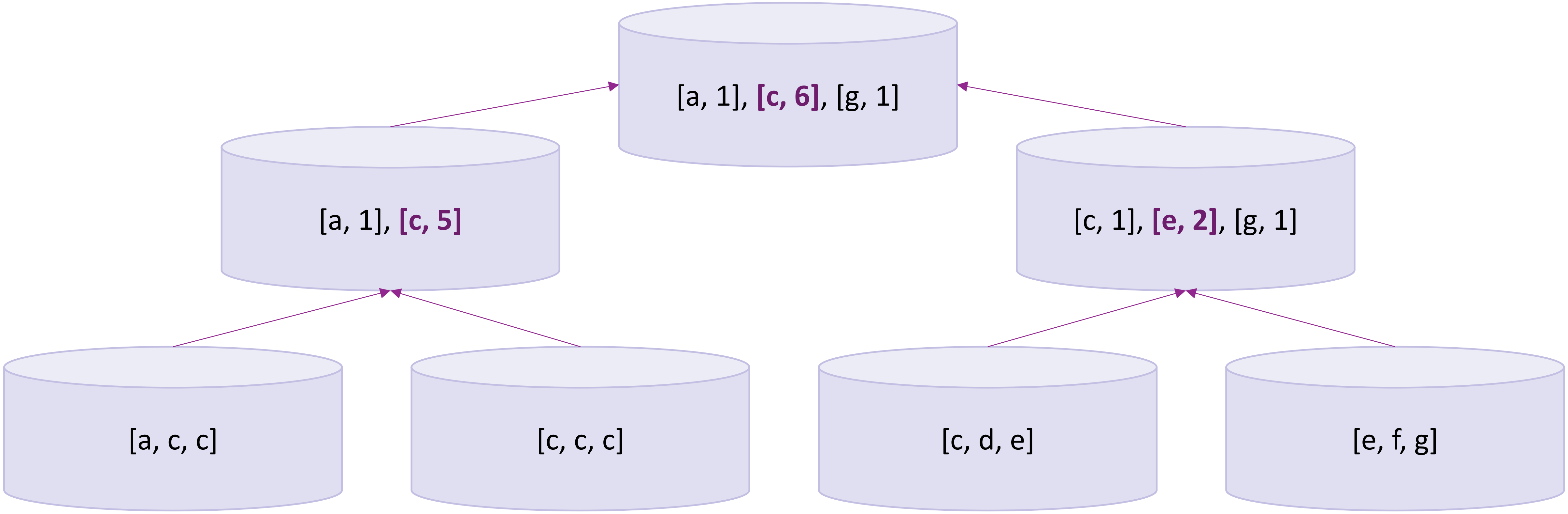
Exact Triangle Counting Space Per Machine

- Find duplicates using new MPC primitive



Exact Triangle Counting Space Per Machine

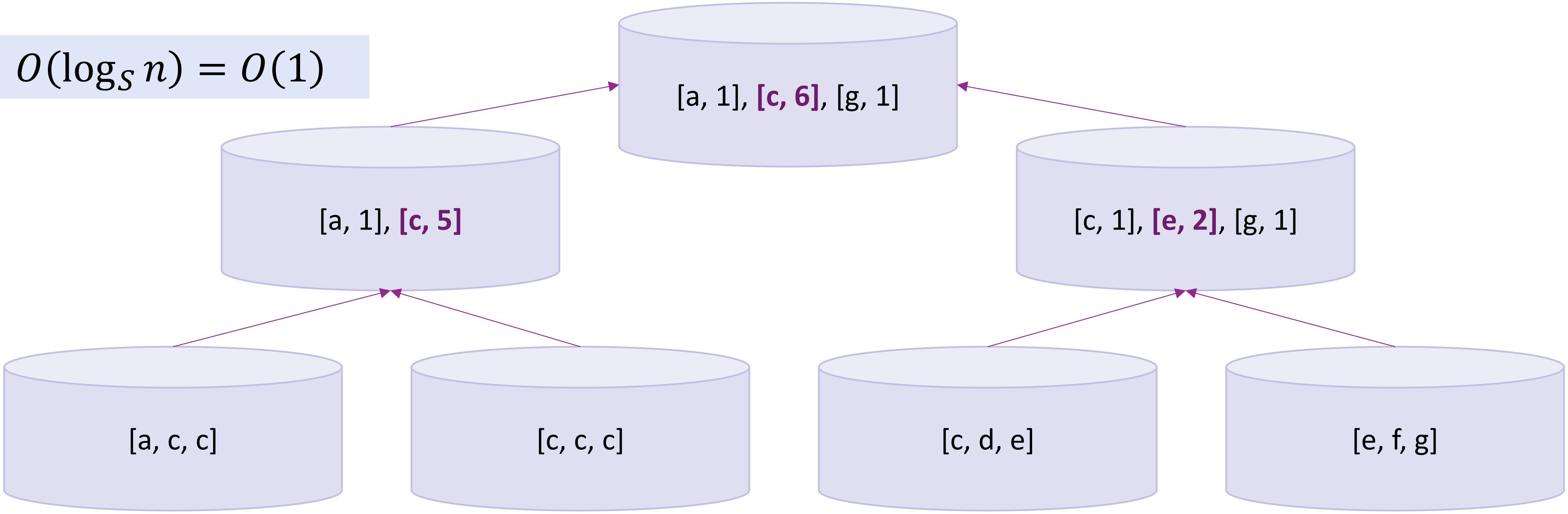
- Find duplicates using new MPC primitive



Exact Triangle Counting Space Per Machine

- Find duplicates using new MPC primitive

$O(\log_S n) = O(1)$



Exact Triangle Counting

- Challenge: Cannot count on one machine because that is too much space
  - Need to have an MPC specific counting procedure
  - Removed nodes send list of neighbors to all neighbors
  - MPC sorting algorithm of [GSZ11] to sort lists
  - Find duplicates using new MPC primitive

There exists a MPC algorithm that outputs the exact count of triangles in a graph with arboricity α in $O(\log \log n)$ rounds, $O\left(n^{\delta}\right)$ space per machine for any constant $\delta > 0$ and $O(m\alpha)$ total space.

Exact Triangle Counting

- Challenge: Cannot count on one machine because that is too much space
  - Need to have an MPC specific counting procedure
  - Removed nodes send list of neighbors to all neighbors
  - MPC sorting algorithm of [GSZ11] to sort lists
  - Find duplicates using new MPC primitive

Somewhat resembles **round compression** technique although simpler on bounded arboricity graphs and deterministic: do not need to do sampling

Results in This Presentation

- Strongly sublinear memory:
  - **Exact** triangle counting:
    - Bounded arboricity
    - $O(\log \log n)$ rounds
    - $O(m\alpha)$ total space
- Near-linear memory:
  - **Approximate** triangle counting
  - $(1+\varepsilon)$-approximation when $T \geq \sqrt{m/n}$
  - $O(1)$ rounds, $\tilde{O}(n+m)$ total space

Results in Our Paper

- Strongly sublinear memory:
  - Extensions to clique counting
- Linear memory:
  - Extensions to clique counting
- Counting **all** small subgraphs of size at most 5
- Simulations on real-world graphs:
  - Improvements in number of rounds
  - Improvements in approximation

Advantages and Disadvantages of Approximate Counting

- Main Advantage:
  - Small runtime, fast and requires little space
- Main Disadvantage:
  - Requires lower bound on the number of triangles

Approximate Triangle Counting

There exists a MPC algorithm that outputs a $(1+\epsilon)$-approximation for the number of triangles if the number of triangles $T \geq \sqrt{d_{avg}}$ and uses $\tilde{O}(m)$ total space and $\widetilde{\Theta}(n)$ space per machine, $O(1)$ MPC rounds.

Massively Parallel Algorithms for Small Subgraph Counting

Amartya Shankha Biswas, Talya Eden, Quanquan C. Liu, Slobodan Mitrovic, Ronitt Rubinfeld

[arxiv.org/2002.08299]

Approximate Triangle Counting

There exists a MPC algorithm that outputs a $(1+\epsilon)$-approximation for the number of triangles if the number of triangles $T \geq \sqrt{d_{avg}}$ and uses $\tilde{O}(m)$ total space and $\widetilde{\Theta}(n)$ space per machine, $O(1)$ MPC rounds.

Massively Parallel Algorithms for Small Subgraph Counting

Amartya Shankha Biswas, Talya Eden, Quanquan C. Liu, Slobodan Mitrovic, Ronitt Rubinfeld

[arxiv.org/2002.08299]

Previous: $T \geq d_{avg}$ [Pagh and Tsourakakis '12]

Approximate Triangle Counting

There exists a MPC algorithm that outputs a $(1+\epsilon)$-approximation for the number of triangles if the number of triangles $T \geq \sqrt{d_{avg}}$ and uses $\tilde{O}(m)$ total space and $\widetilde{\Theta}(n)$ space per machine, $O(1)$ MPC rounds.

Massively Parallel Algorithms for Small Subgraph Counting

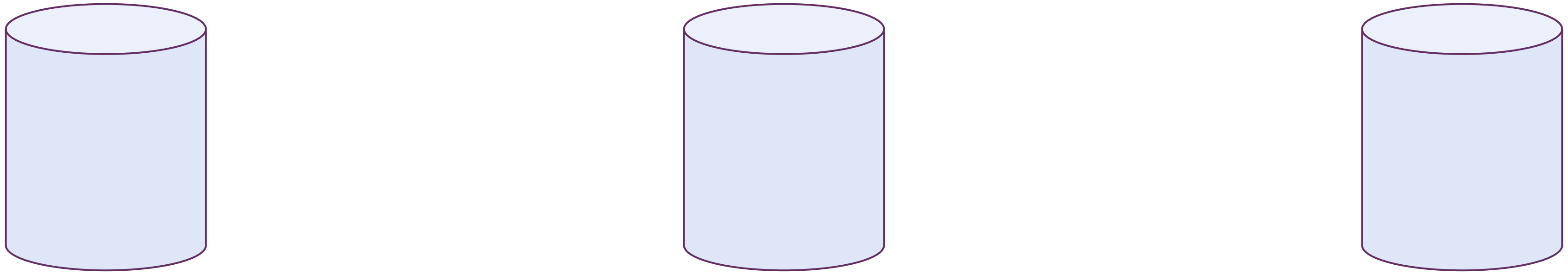
Amartya Shankha Biswas, Talya Eden, Quanquan C. Liu, Slobodan Mitrovic, Ronitt Rubinfeld

[arxiv.org/2002.08299]

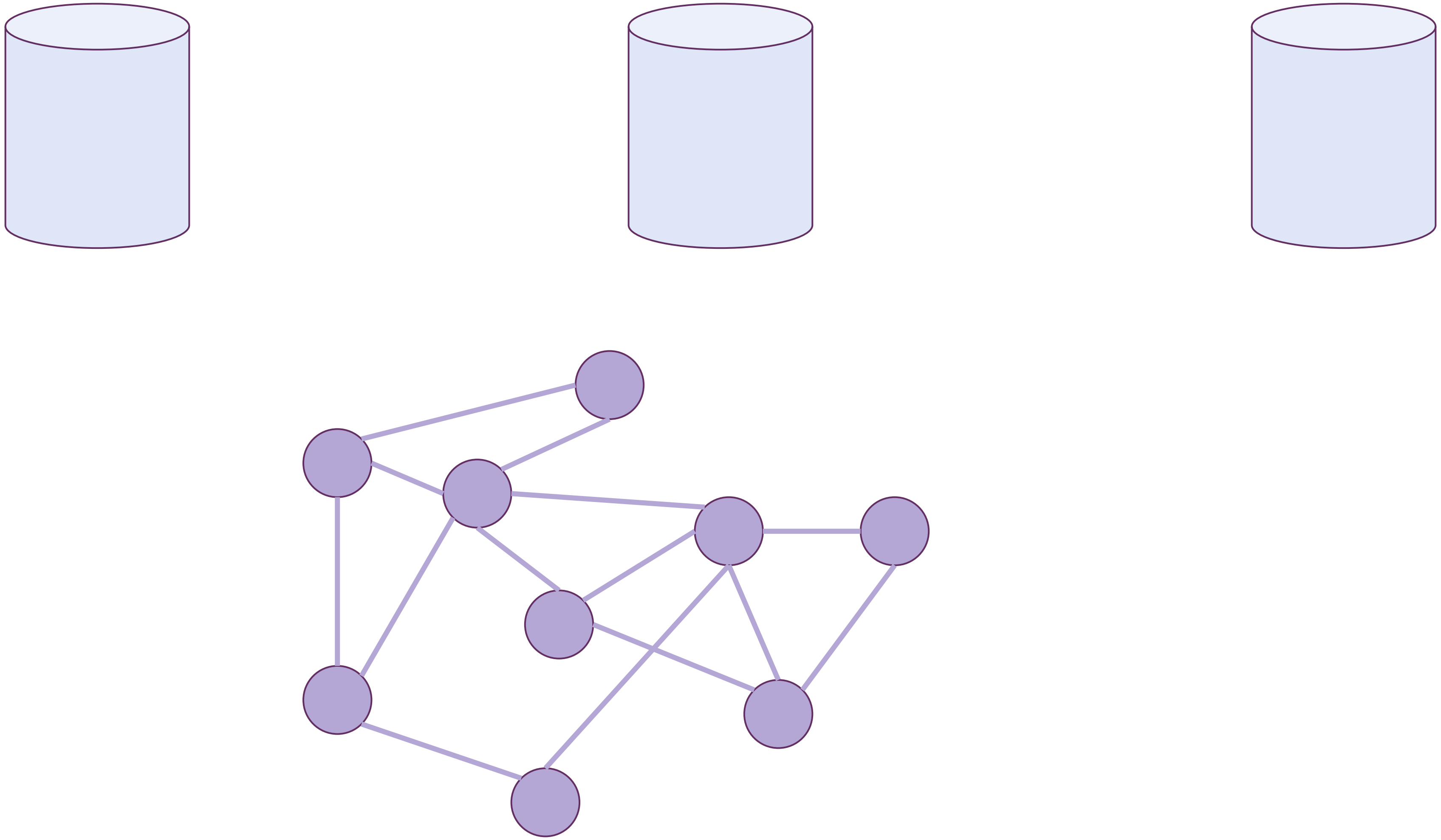
Previous: $T \geq d_{avg}$ [Pagh and Tsourakakis '12]

[Seshadhri, Pinar, Kolda '13] can get better near-linear space per machine

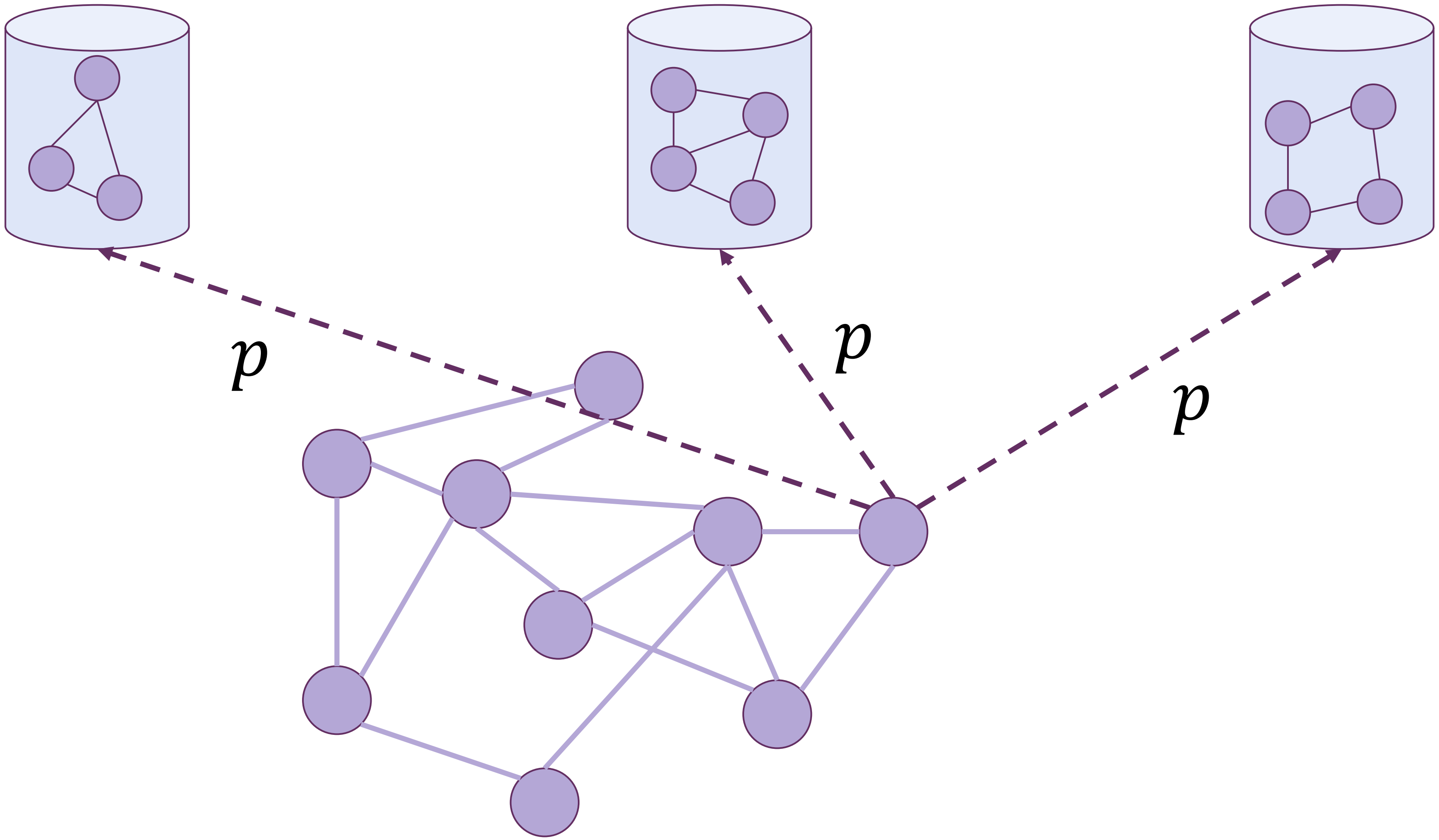
Approximate Triangle Counting



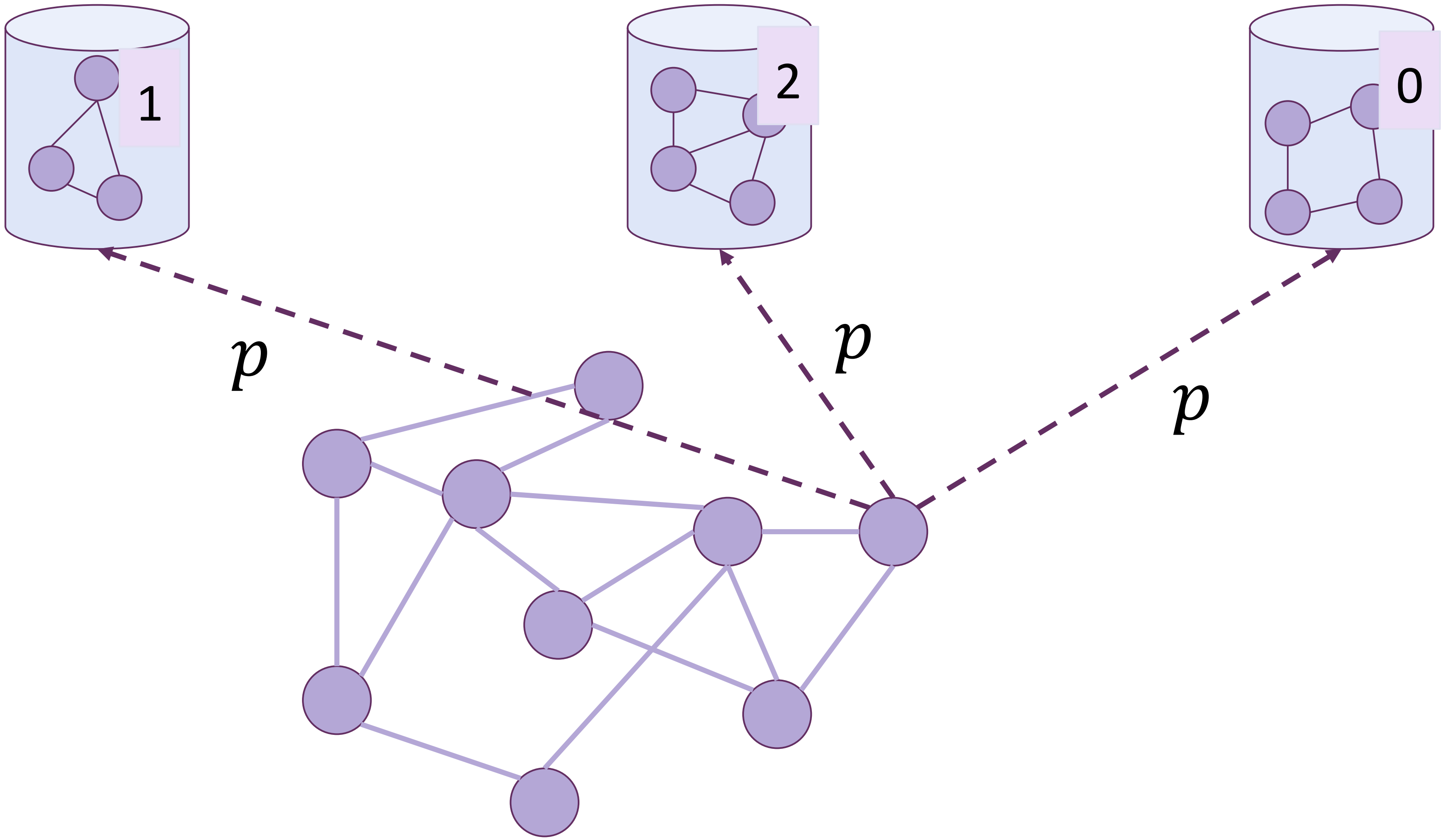
Approximate Triangle Counting



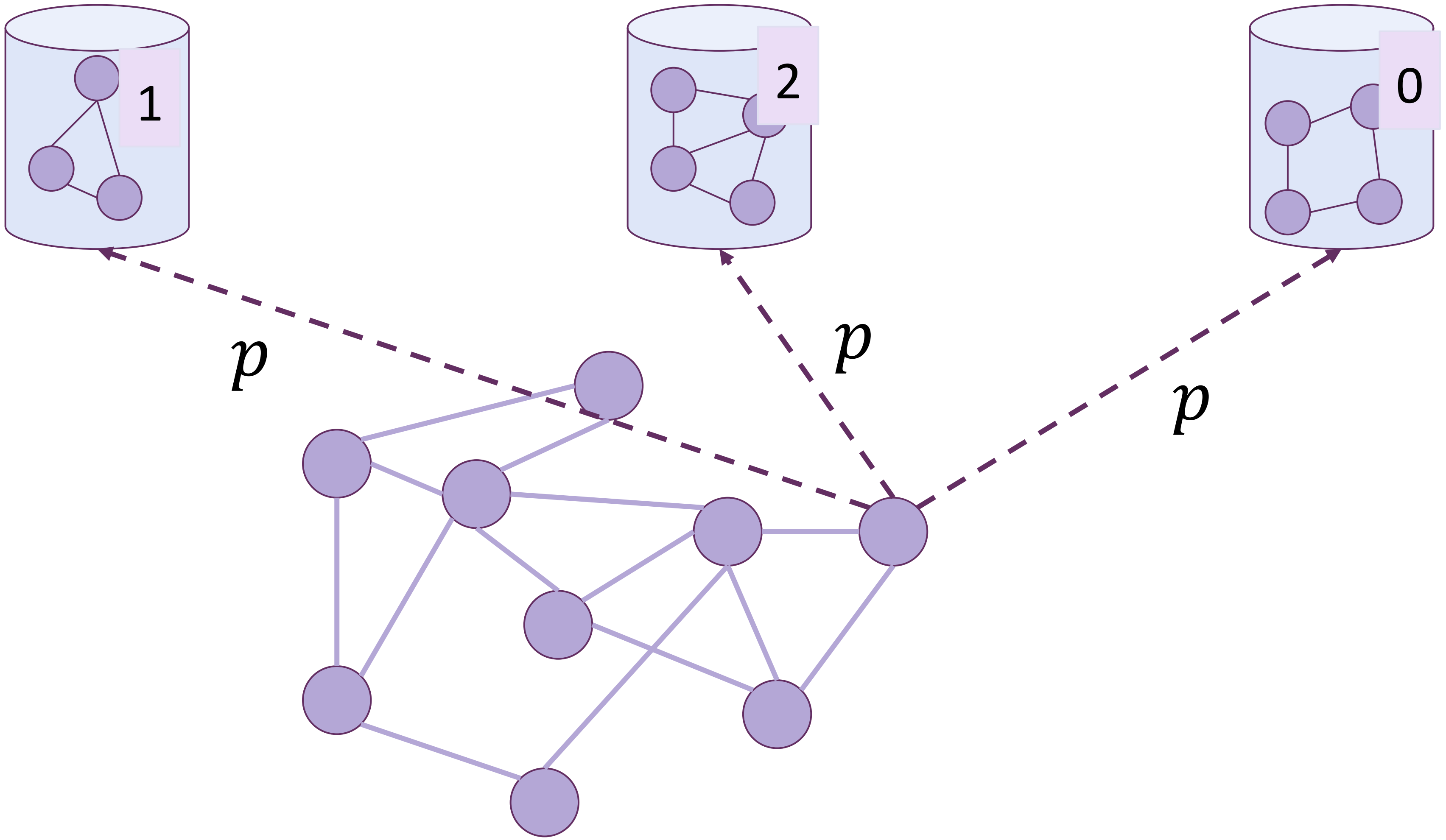
Approximate Triangle Counting



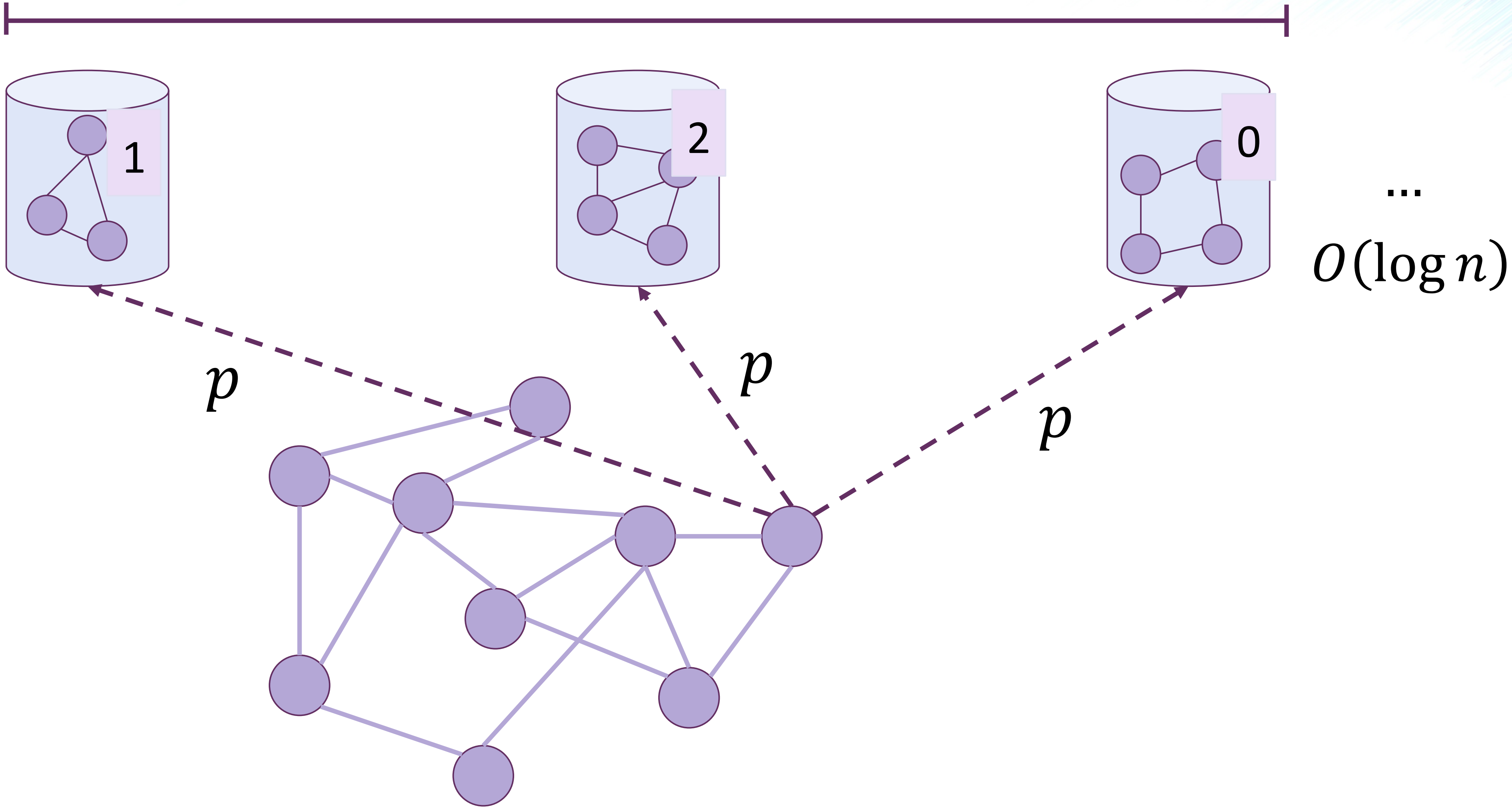
Approximate Triangle Counting



Approximate Triangle Counting



Approximate Triangle Counting



Challenges

Challenges

- Challenge 1: Induced subgraphs do not exceed the space per machine

Challenges

- Challenge 1: Induced subgraphs do not exceed the space per machine
- Challenge 2: How to compute the induced subgraph in each machine when one vertex *can appear on multiple machines*?

Challenges

- Challenge 1: Induced subgraphs do not exceed the space per machine
- Challenge 2: How to compute the induced subgraph in each machine when one vertex *can appear on multiple machines*?
- Challenge 3: The number of triangles across the machines concentrates

Challenges

- Challenge 1: Induced subgraphs do not exceed the space per machine

Careful setting of p

- Challenge 2: How to compute the induced subgraph in each machine when one vertex *can appear on multiple machines*?

k-wise independent hash function for small k

- Challenge 3: The number of triangles across the machines concentrates

Constant probability of success and median trick

Challenges

- Challenge 1: Induced subgraphs do not exceed the space per machine

Careful setting of p

- Challenge 2: How to compute the induced subgraph in each machine when one vertex *can appear on multiple machines*?

k-wise independent hash function for small k

- Challenge 3: The number of triangles across the machines concentrates

Constant probability of success and median trick

Open Questions and Future Directions

- Small subgraph counting for a **broader class of small subgraphs**

Open Questions and Future Directions

- Small subgraph counting for a **broader class of small subgraphs**
  - Recent works of Bressan ‘19 and Bera, Pashanasangi, and Seshadhri ‘21 use **DAG tree decomposition**

Open Questions and Future Directions

- Small subgraph counting for a **broader class of small subgraphs**
  - Recent works of Bressan ‘19 and Bera, Pashanasangi, and Seshadhri ‘21 use **DAG tree decomposition**
  - Can we implement in MPC?

Open Questions and Future Directions

- Small subgraph counting for a **broader class of small subgraphs**
  - Recent works of Bressan ‘19 and Bera, Pashanasangi, and Seshadhri ‘21 use **DAG tree decomposition**
  - Can we implement in MPC?
- Counting in the **adaptive MPC model (AMPC)**

Open Questions and Future Directions

- Small subgraph counting for a **broader class of small subgraphs**
    - Recent works of Bressan ‘19 and Bera, Pashanasangi, and Seshadhri ‘21 use **DAG tree decomposition**
    - Can we implement in MPC?
- Counting in the **adaptive MPC model (AMPC)**
- Approximate triangle counting in $O(1)$ rounds and strictly sublinear space in **sparse graphs** where $m = \tilde{O}(n)$