Sparse Matrix Multiplication and Triangle Listing in the Congested Clique Model

Keren Censor-Hillel, **Dean Leitersdorf**, Elia Turner (Technion)

OPODIS 2018

This project received funding from the European Union's Horizon 2020 Research and Innovation Program under grant agreement no. 755839

Overview

The Congested Clique

Overlay Network

Input Graph

- n nodes in both graphs
- Synchronous, $O(\log n)$ bits per message
- All-to-All Communication
- Goal: Minimize # communication rounds

Sparse Algorithms

- Sparse input graphs common in practice
- Leverage sparsity, reduce runtime
- Congested Clique: Does not decrease model strength

Sparse Algorithms - Our Results

- New load balancing building blocks
- New algorithms for sparse matrix multiplication, triangle listing
- Implies sparse graph algorithms
  - Triangle, 4-cycle counting
  - APSP

Sparse Matrix Multiplication (Sparse MM)

Previous Work on MM

- $O(n^{\omega-2}) = O(n^{0.372})$ boolean MM

[Drucker, Kuhn, Oshman, PODC 2014]

- $O(n^{1-2/\omega}) = O(n^{0.158})$ *ring* MM

[Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela, PODC 2015]

- $O(n^{1/3})$ *semiring* MM

[Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela, PODC 2015]

- Rectangular matrices and multiple instances of MM concurrently

[Le Gall, DISC 2016]

ω = exponent of sequential MM < 2.372864

Previous Work on MM

- $O(n^{\omega-2}) = O(n^{0.372})$ boolean MM

[Drucker, Kuhn, Oshman, PODC 2014]

- $O(n^{1-2/\omega}) = O(n^{0.158})$ *ring* MM

[Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela, PODC 2015]

- $O(n^{1/3})$ *semiring* MM

[Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela, PODC 2015]

- Rectangular matrices and multiple instances of MM concurrently

[Le Gall, DISC 2016]

ω = exponent of sequential MM < 2.372864

Previous Work on MM

- $O(n^{\omega-2}) = O(n^{0.372})$ boolean MM

[Drucker, Kuhn, Oshman, PODC 2014]

- $O(n^{1-2/\omega}) = O(n^{0.158})$ *ring* MM

[Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela, PODC 2015]

- $O(n^{1/3})$ *semiring* MM

[Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela, PODC 2015]

- Rectangular matrices and multiple instances of MM concurrently

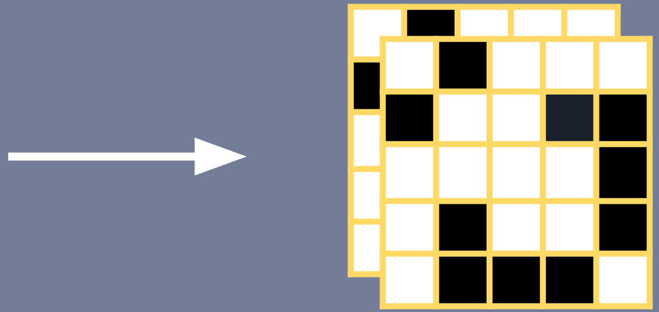
[Le Gall, DISC 2016]

ω = exponent of sequential MM < 2.372864

Matrix Multiplication (MM)

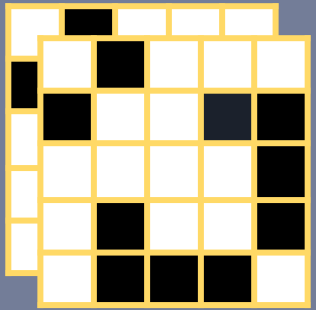
- Input: Square matrices *S*, *T*. Output: *P* = *S***T*
- Node *i*: row *i* of each matrix
- Example: S = T = Adjacency matrix of graph

Input Graph



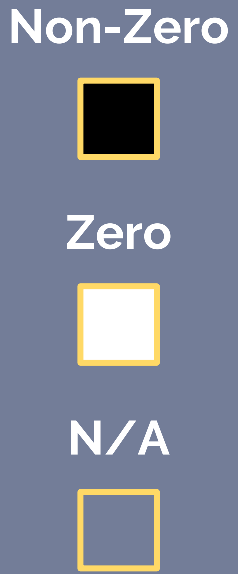
Input Matrices

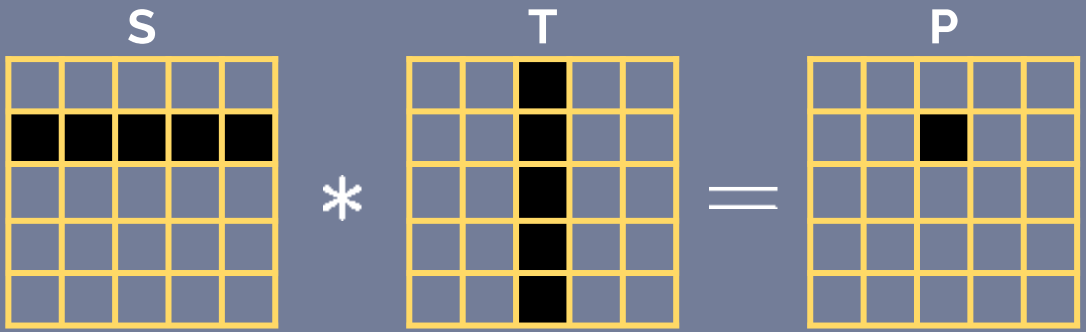
Sparse MM



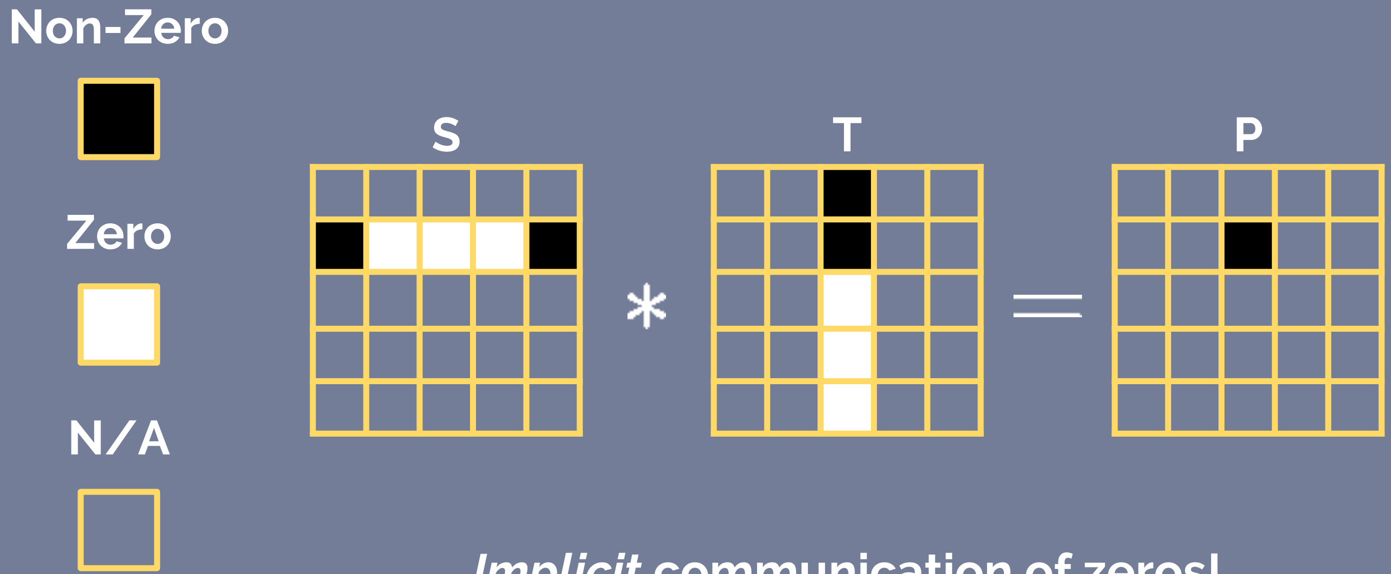
- Many beautiful works in sequential and parallel. Typically different runtime measures
- New algorithm: deterministic & dynamic communication pattern w.r.t. sparsity structure

Sparse MM





Sparse MM



Implicit **communication of zeros!**

Sparse MM - Our Main Result

➔ *P = S*T:*

$$O(nz(S)^{1/3}nz(T)^{1/3}/n + 1)$$

nz(A) = number of non-zero elements in *A*

$$O(nz(S)) = O(nz(T)) = m \longrightarrow O(m^{2/3}/n + 1)$$

$$m = O(n^{3/2}) \Rightarrow O(1)$$

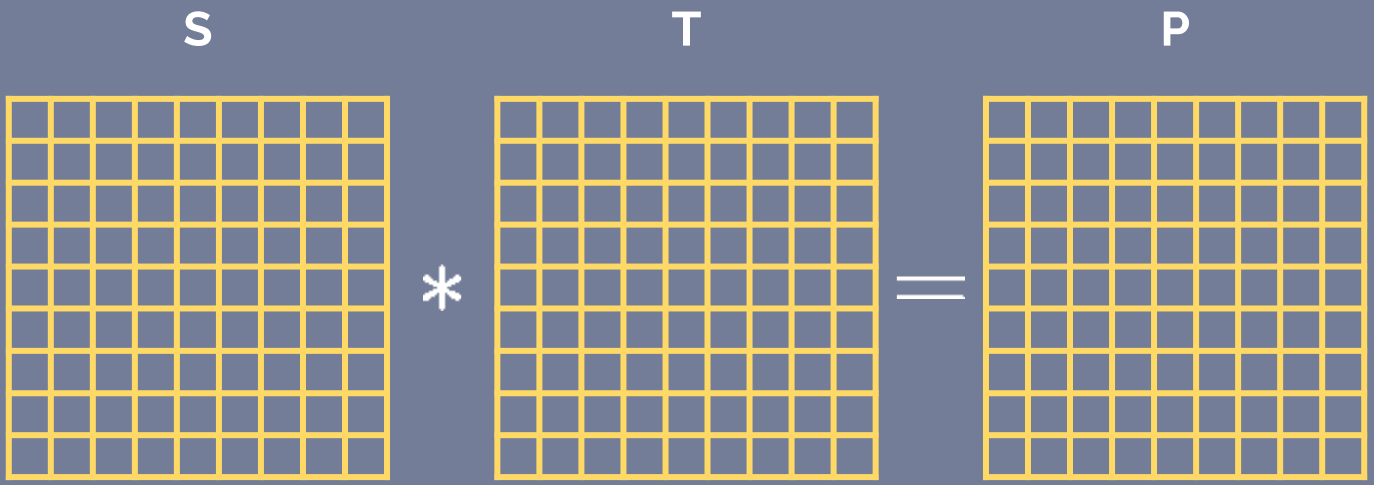
$$\min\{O(nz(S)), O(nz(T))\} = m \longrightarrow O(m^{1/3}/n^{1/3} + 1)$$

Lets see it!

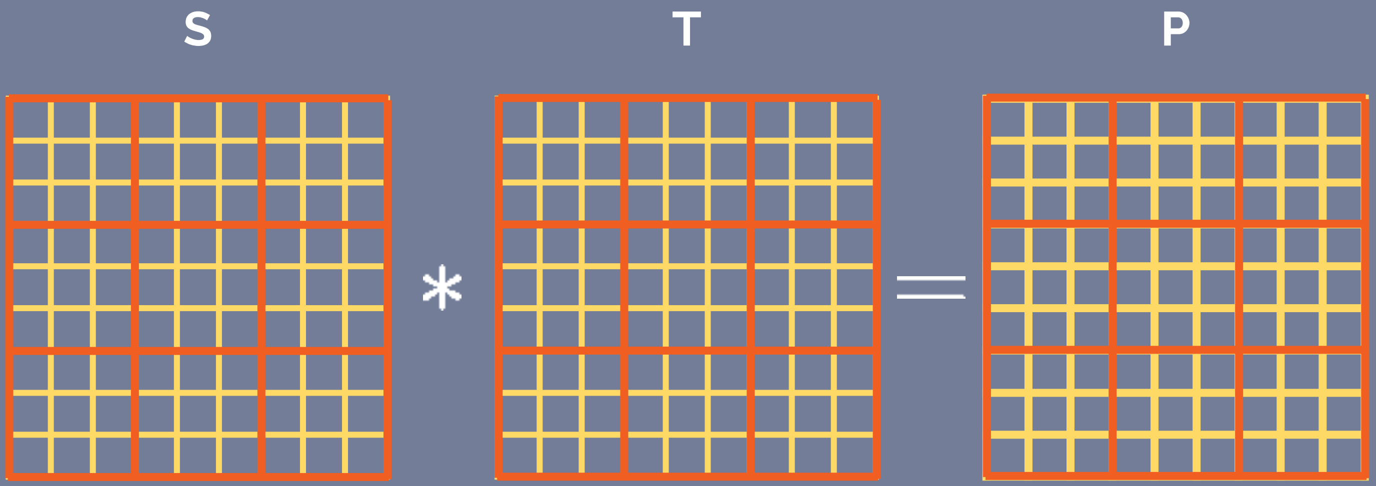
Semiring MM

- $O(n^{1/3})$ [Censor-Hillel, Kaski, Korhonen, Lenzen, Paz, Suomela, PODC 2015]
- 3 Parts:
  1. Distribute matrix entries
  2. Locally compute partial products
  3. Sum partial products
- Our novelty: 1, 3 in a sparsity aware manner

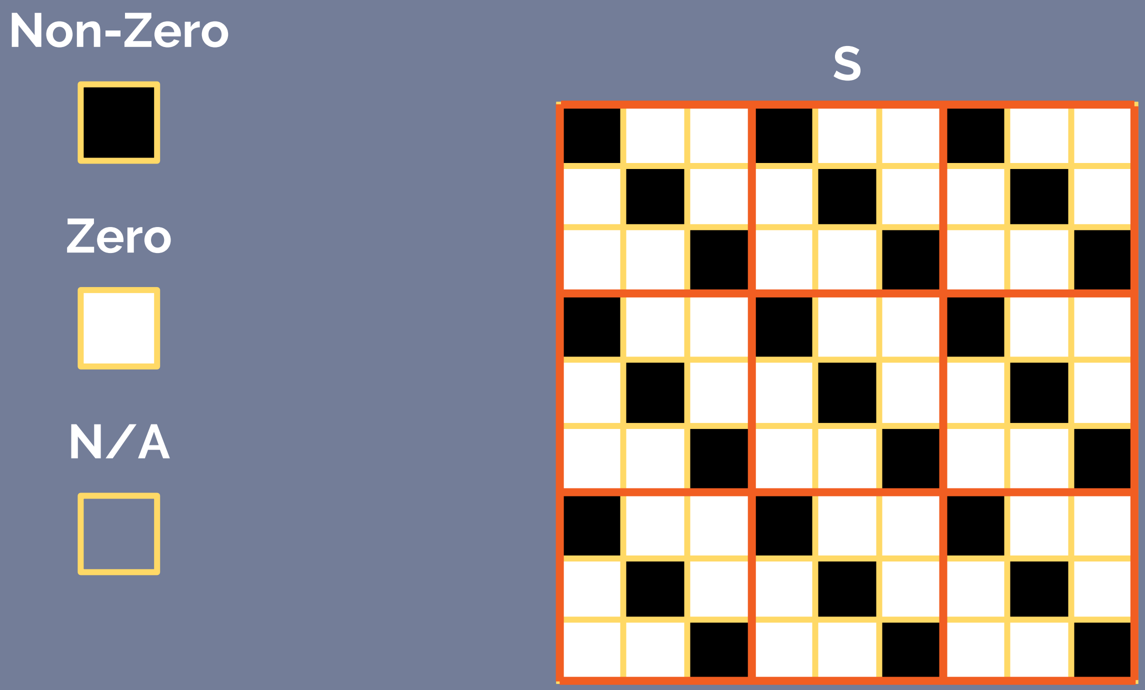
The Challenges



The Challenges



The Challenges

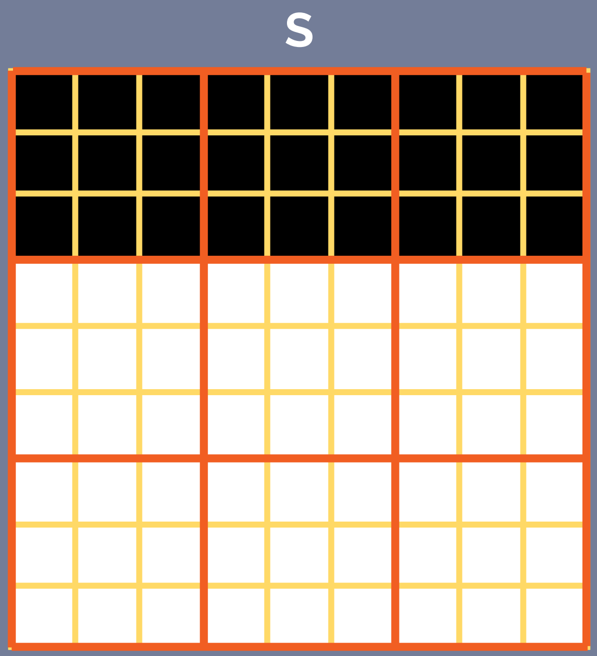


The Challenges

Non-Zero

Zero

N/A



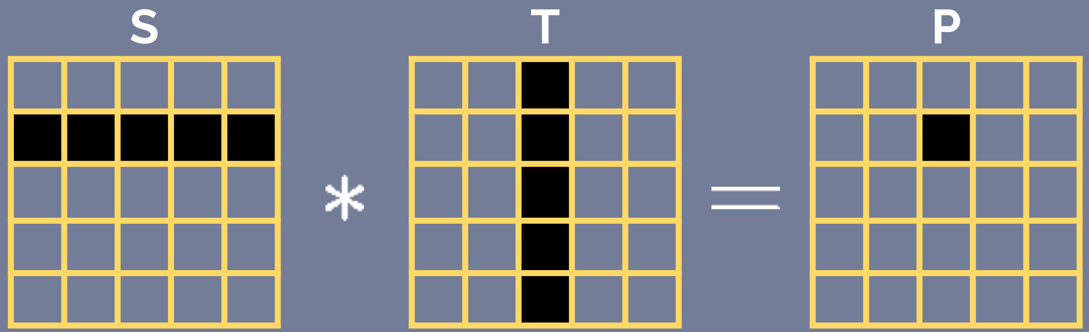
Sparse MM: *Two* Challenges

- [Lenzen, 2013] - runtime depends on max messages
- *Receiving Challenge*: Every node receives ~same # messages
- *Sending Challenge*: Every node sends ~same # messages

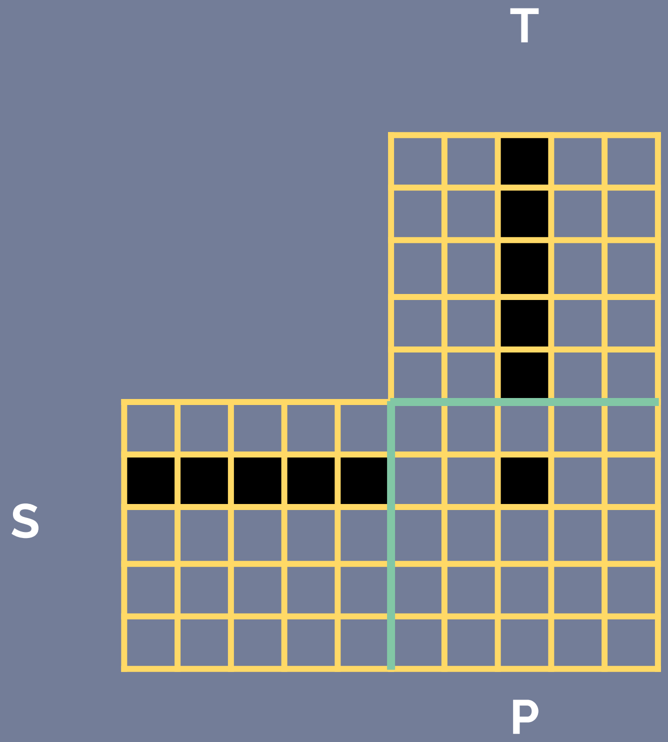
Step 1: (a, b)-split

Square MM → Several Instances of Rectangular MM

Step 1: (a, b)-split

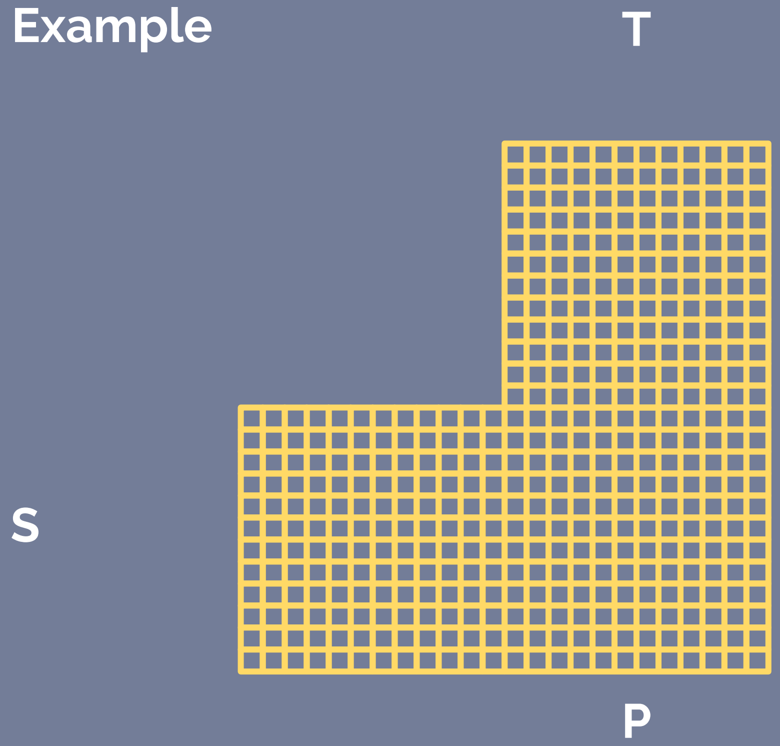


Step 1: (a, b)-split



Step 1: (a, b)-split

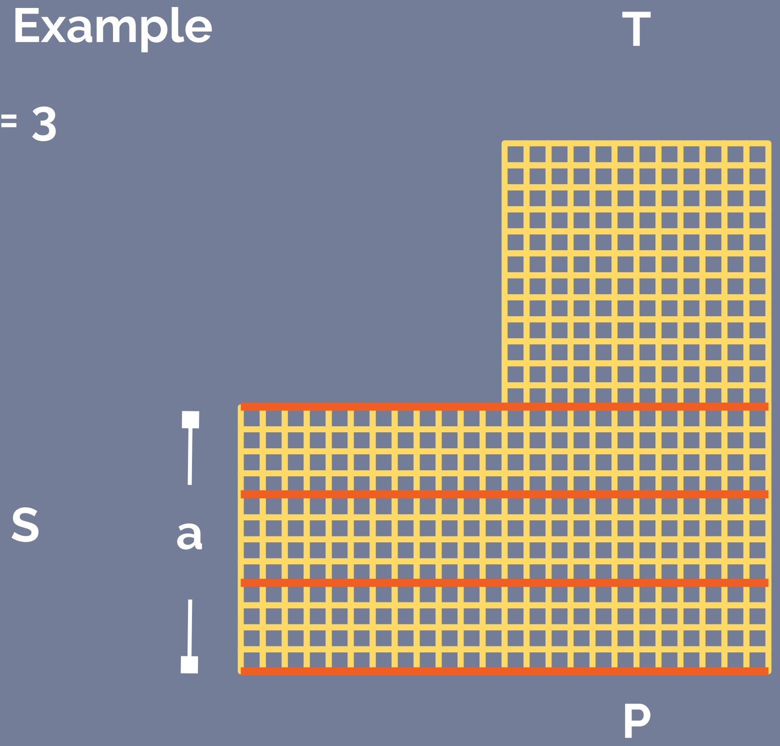
Detailed Example



Step 1: (a, b)-split

Detailed Example

a = 3

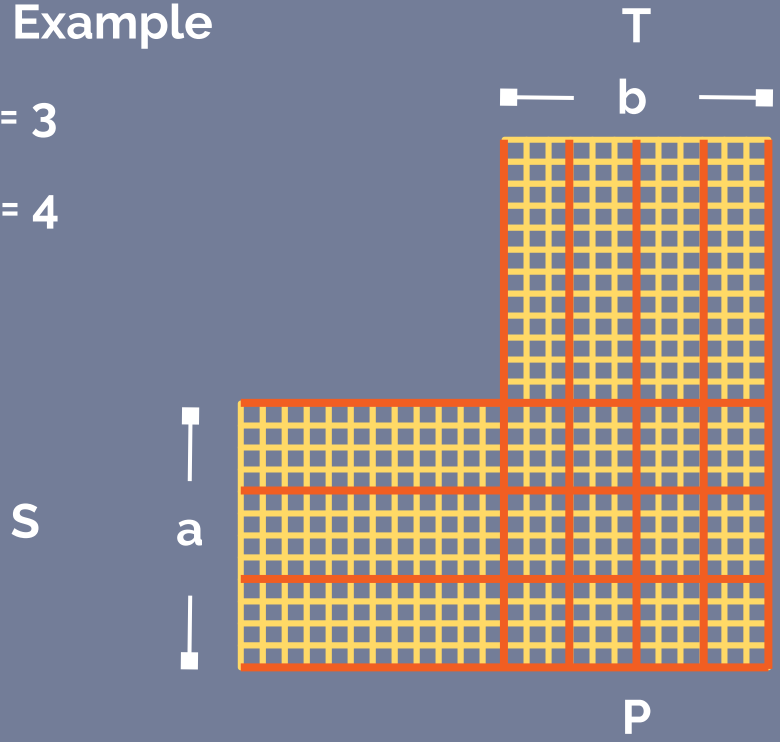


Step 1: (a, b)-split

Detailed Example

a = 3

b = 4

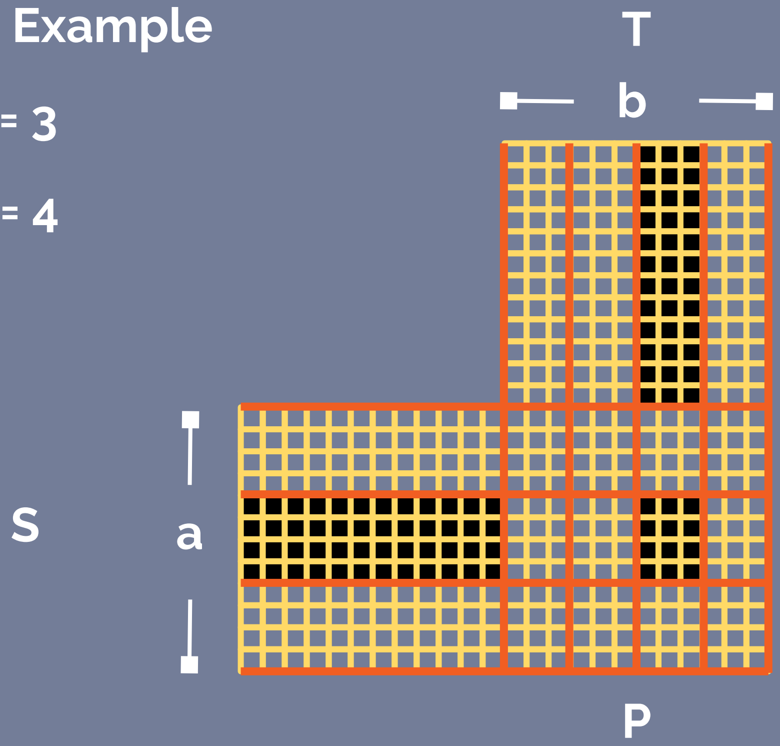


Step 1: (a, b)-split

Detailed Example

a = 3

b = 4

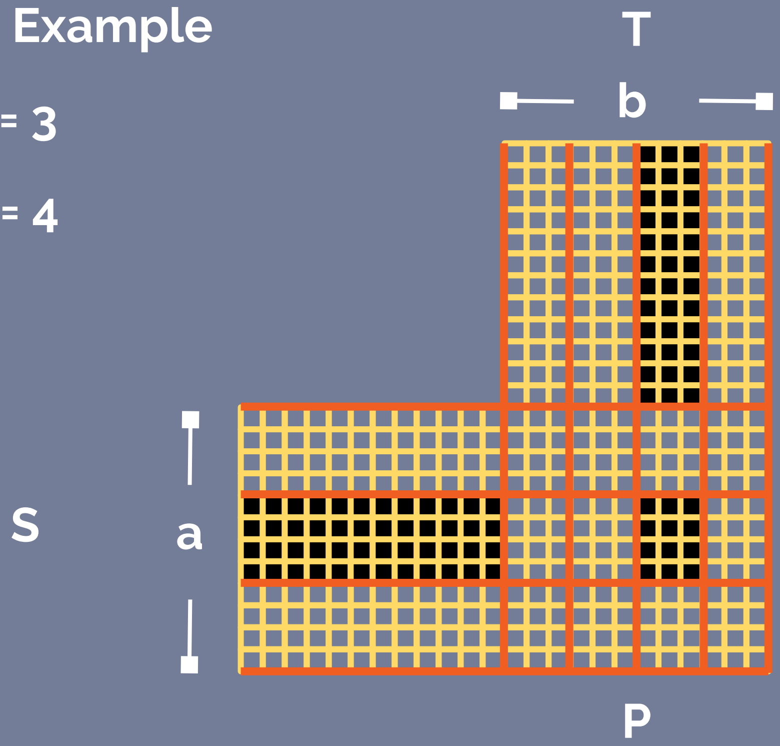


Step 1: (a, b)-split

Detailed Example

a = 3

b = 4



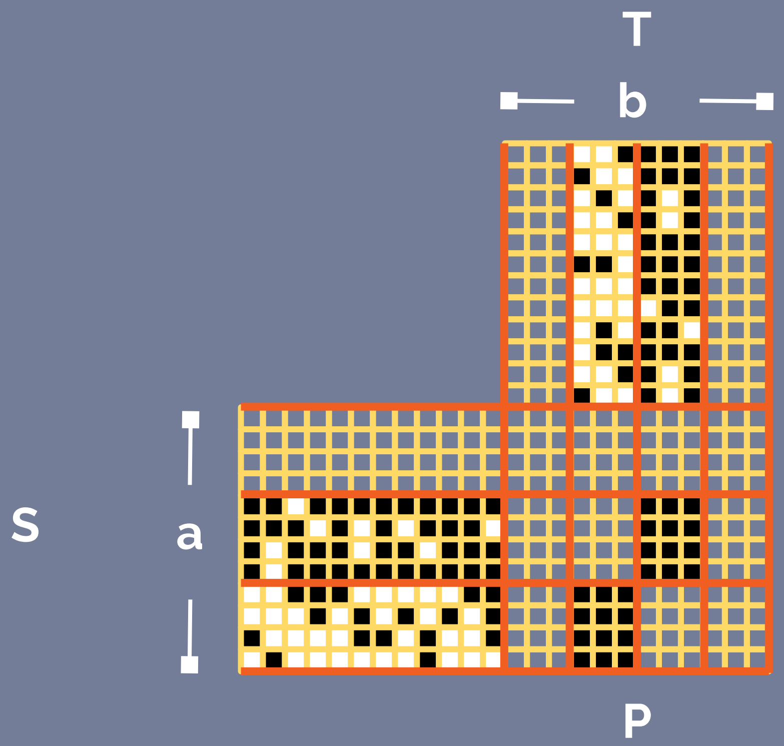
Finally:

- There are a*b rectangular MM
- Assign n/(ab) nodes to compute each

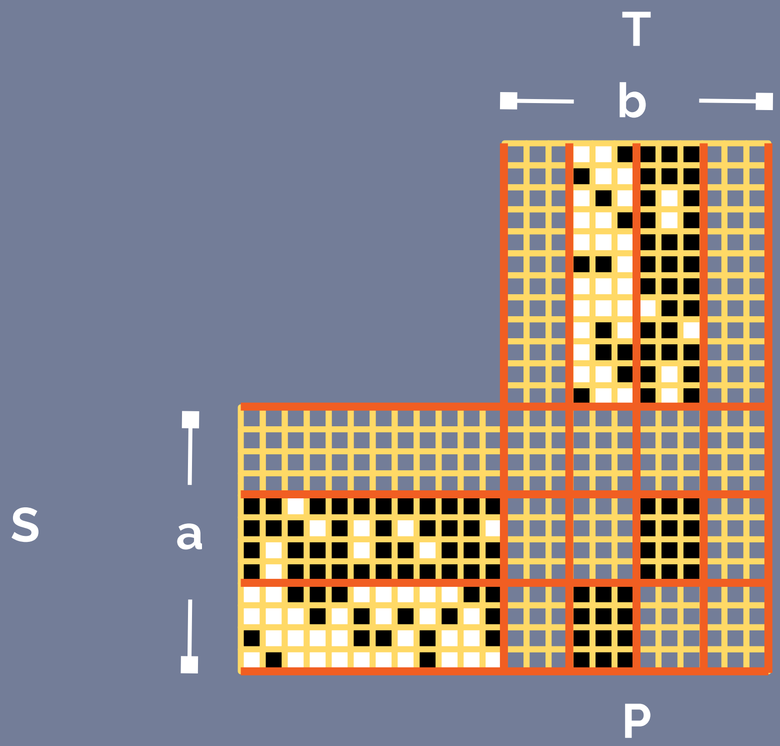
Step 2: Receiving Challenge

- Step 2.1: Roughly similar rectangular MM (density-wise)
- Step 2.2: Sparsity awareness within rectangular MM

Step 2.1: Similar Rectangular MM



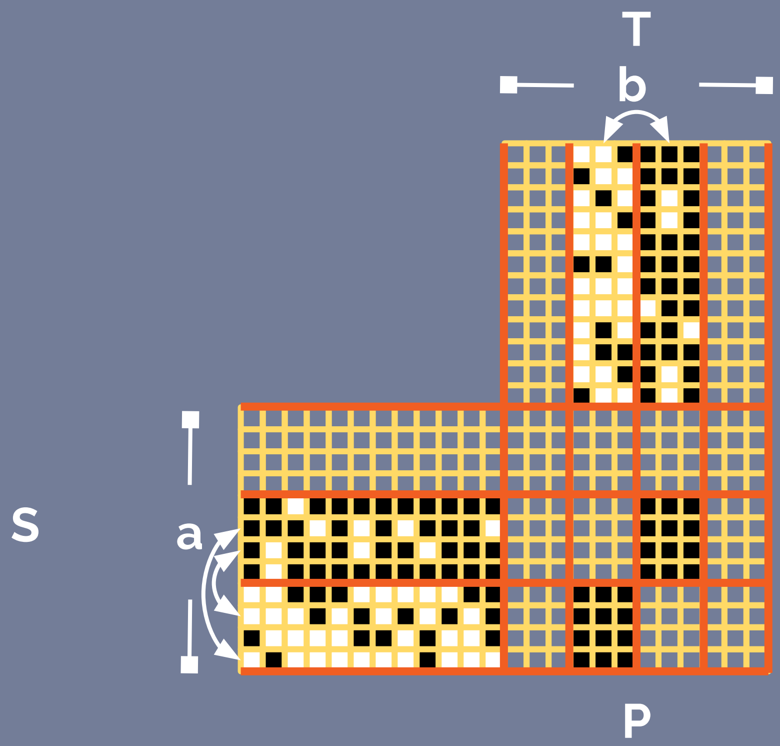
Step 2.1: Similar Rectangular MM



Observation:

- Ok reorder S-rows, T-cols
- Reorder to achieve similar rectangular MMs
- O(1) in congested clique
- Deterministic

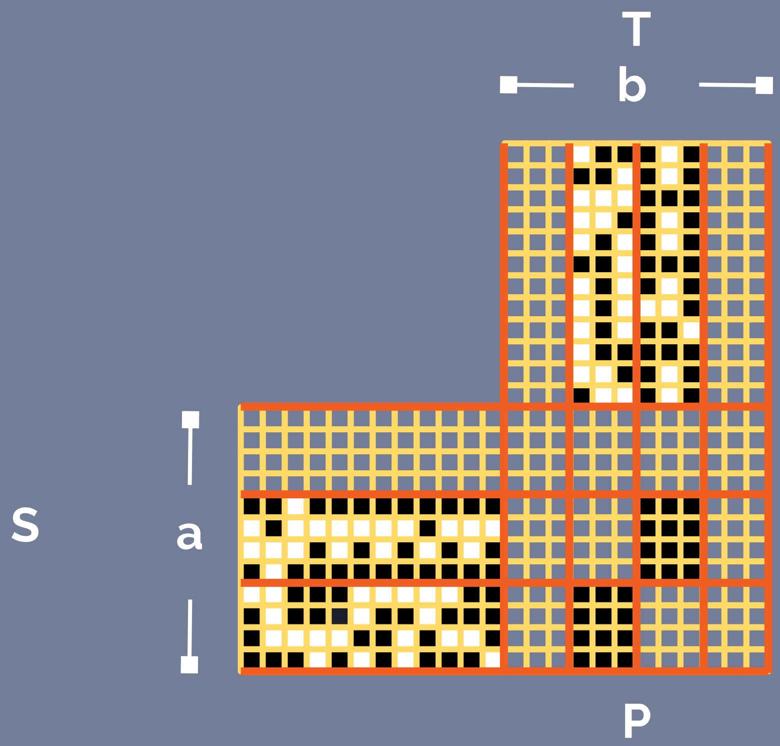
Step 2.1: Similar Rectangular MM



Observation:

- Ok reorder S-rows, T-cols
- Reorder to achieve similar rectangular MMs
- O(1) in congested clique
- Deterministic

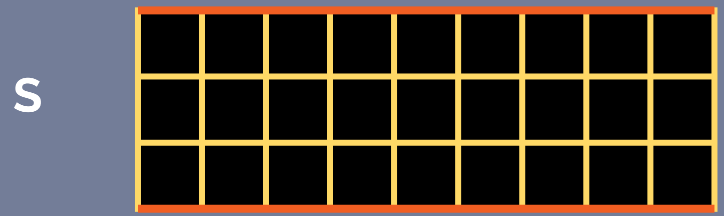
Step 2.1: Similar Rectangular MM

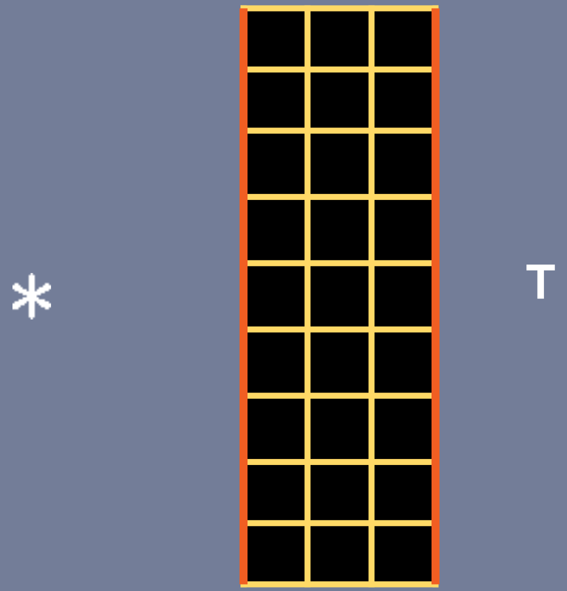


Observation:

- Ok reorder S-rows, T-cols
- Reorder to achieve similar rectangular MMs
- O(1) in congested clique
- Deterministic

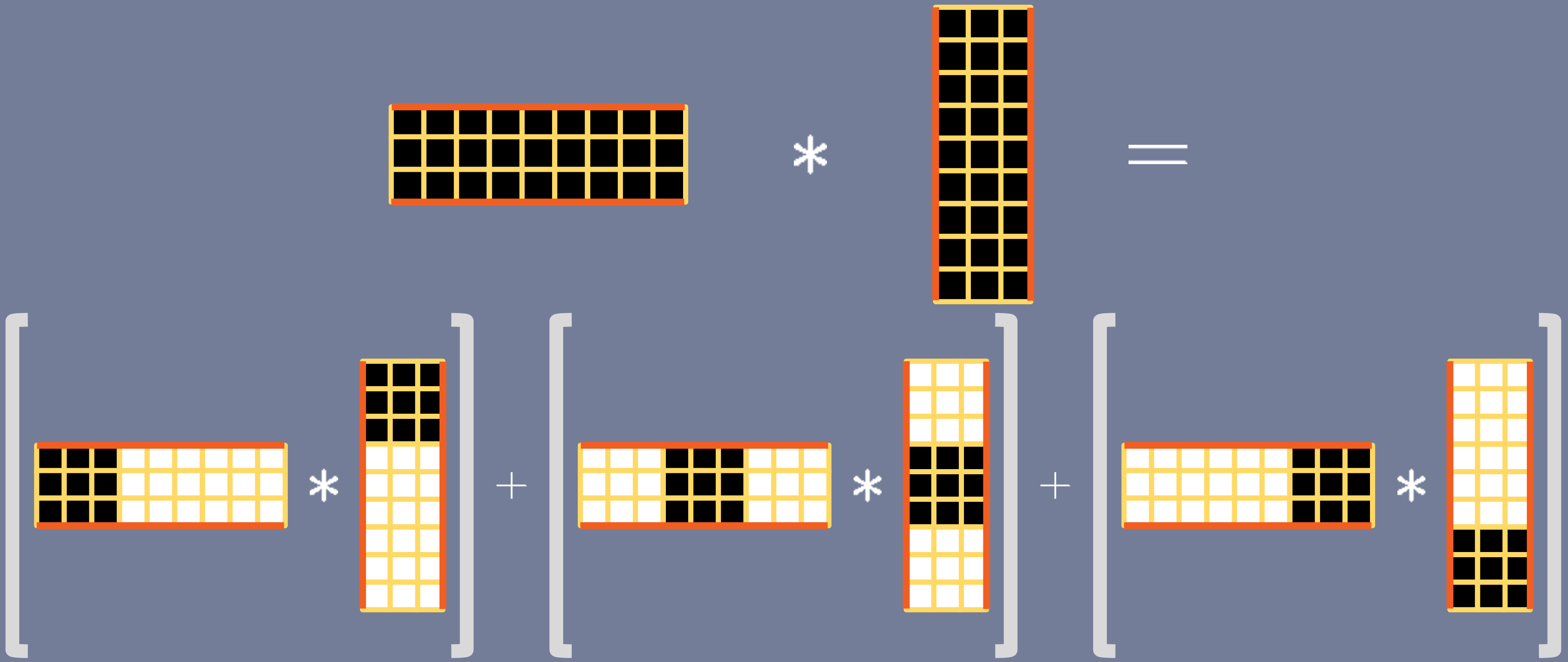
Step 2.2: Sparsity Aware Rectangular MM



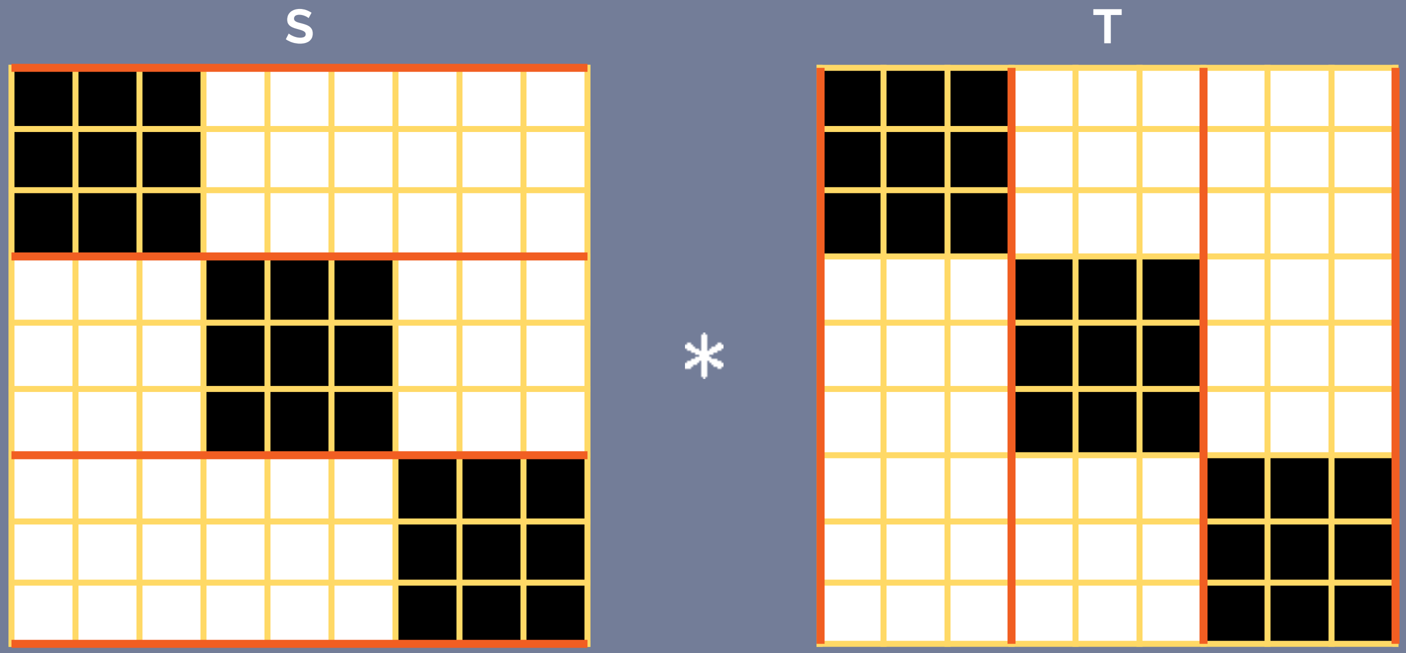


How do we split this between the n/(ab) nodes?

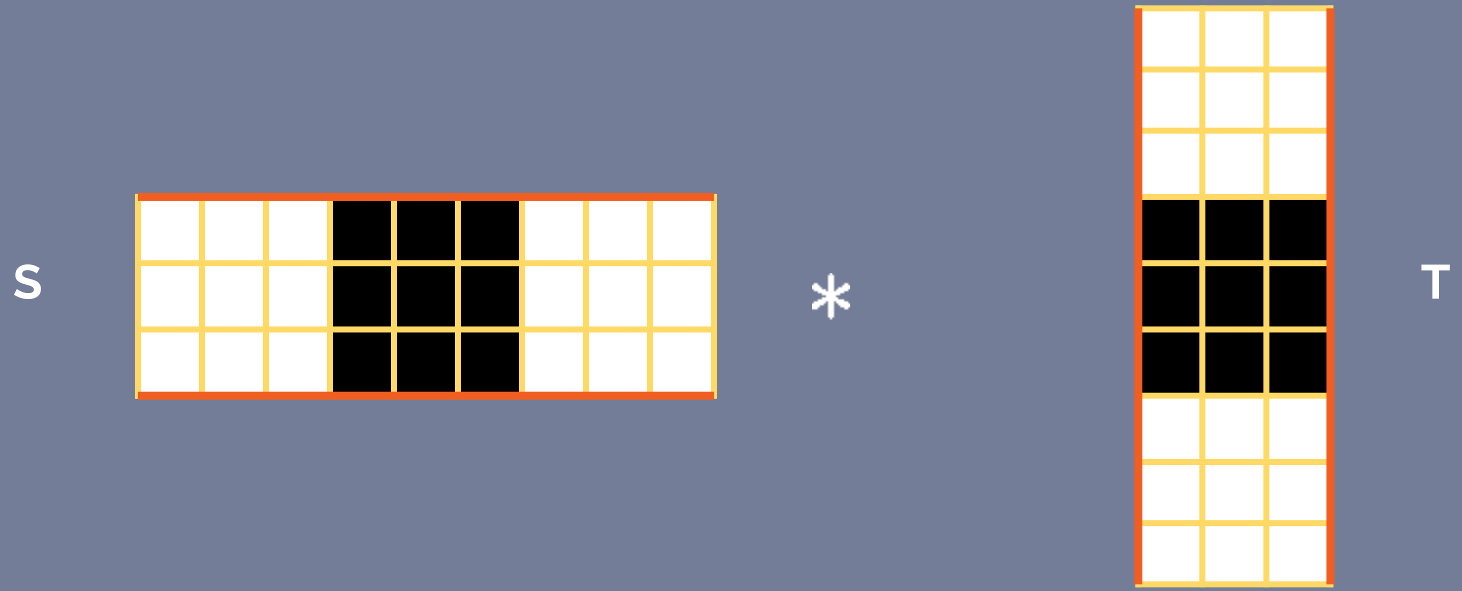
Step 2.2: Sparsity Aware Rectangular MM



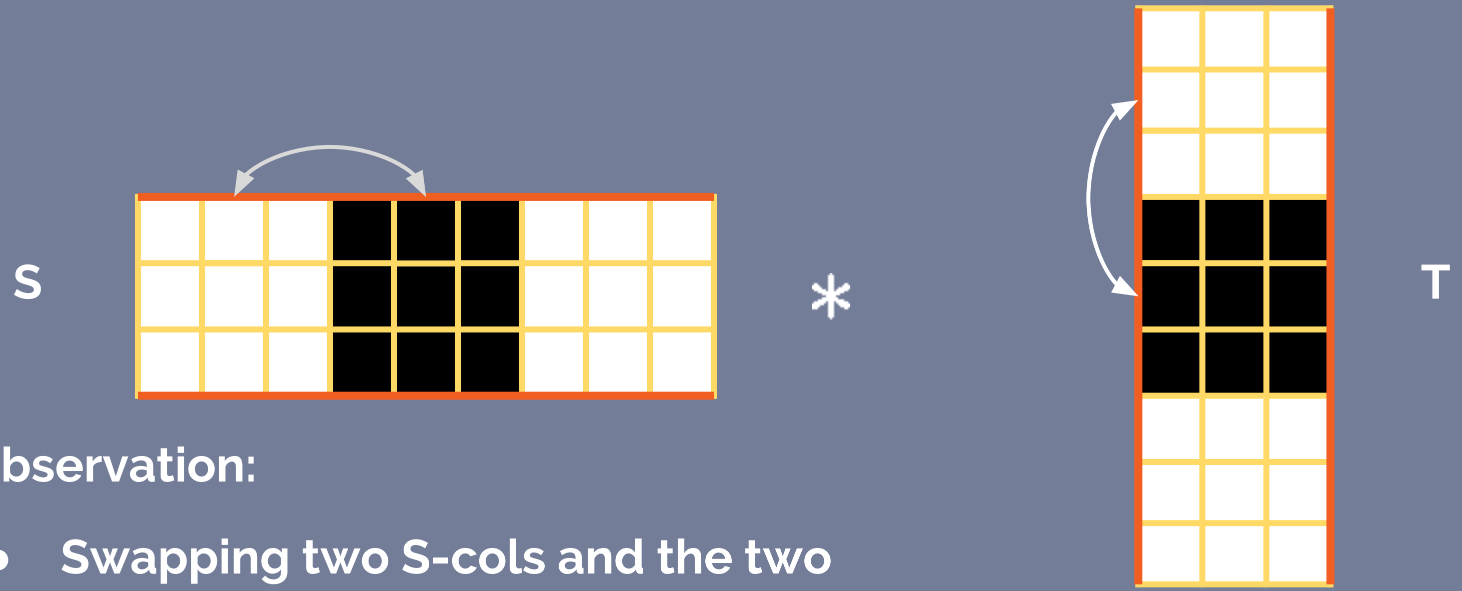
Step 2.2: Sparsity Aware Rectangular MM



Step 2.2: Sparsity Aware Rectangular MM



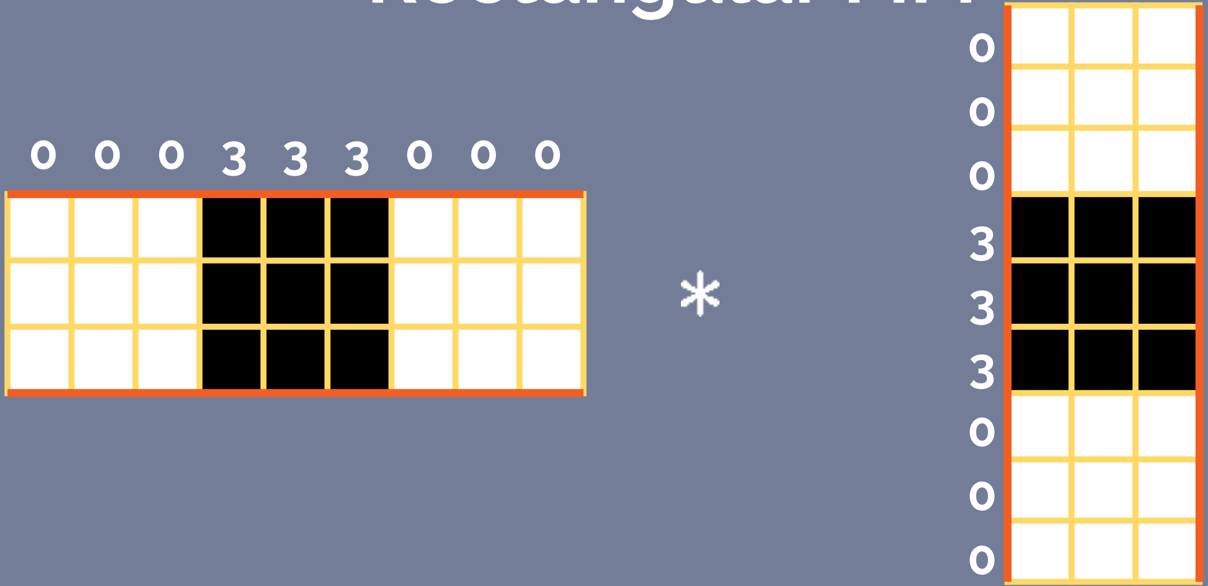
Step 2.2: Sparsity Aware Rectangular MM



Observation:

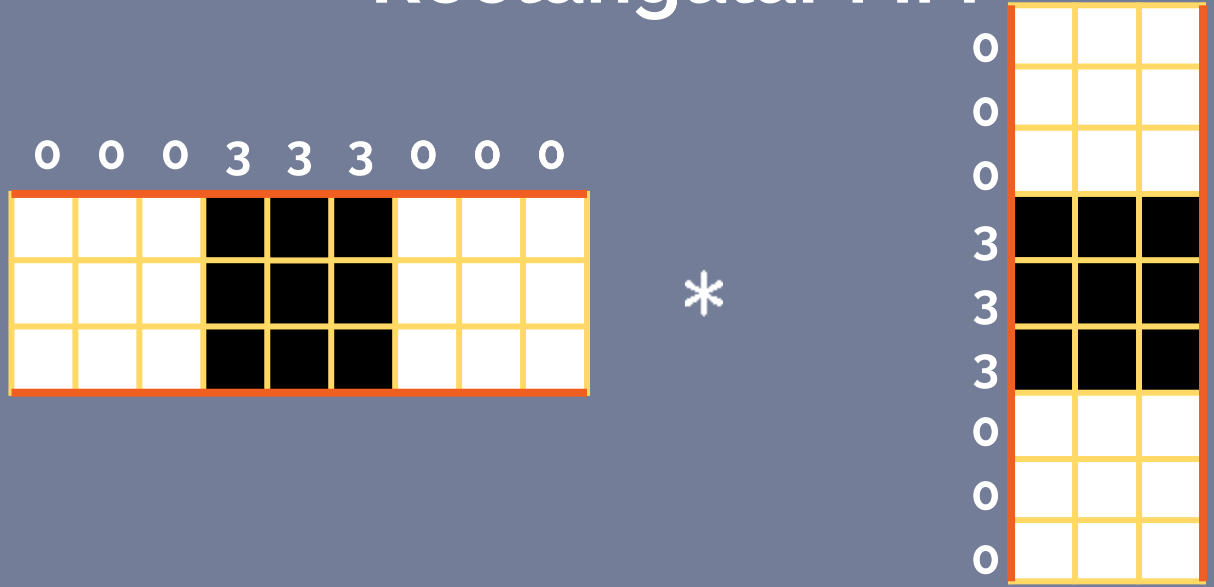
- Swapping two S-cols and the two respective T-rows cancels out

Step 2.2: Sparsity Aware Rectangular MM



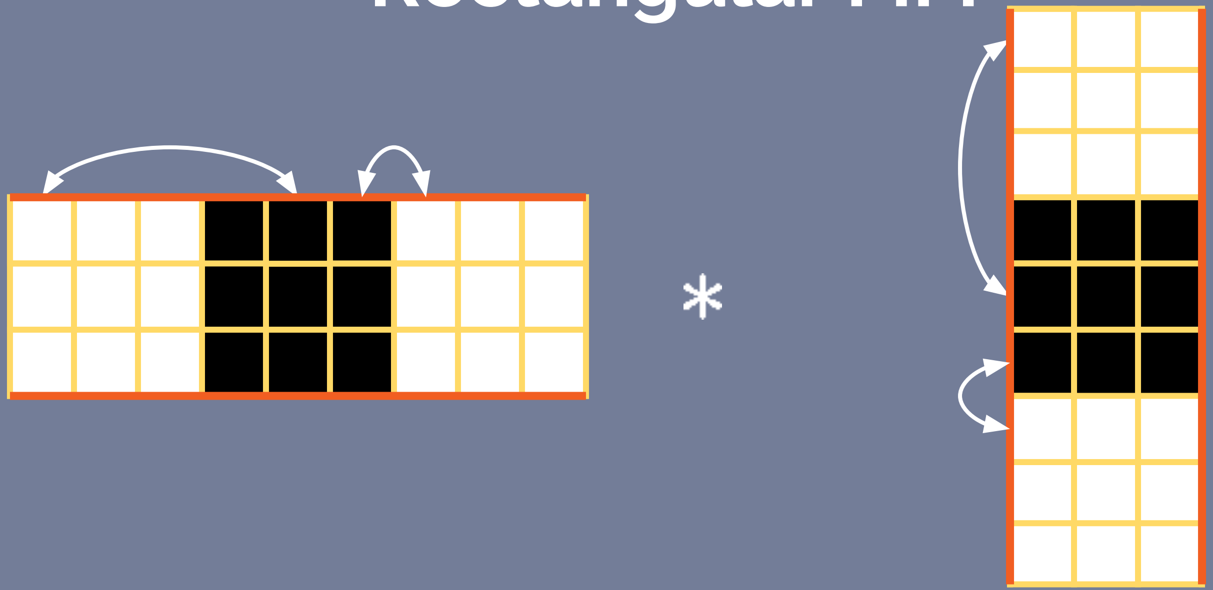
- Phase 1: Count non-zeros in S-cols, T-rows

Step 2.2: Sparsity Aware Rectangular MM



- Phase 1: Count non-zeros in S-cols, T-rows
- Phase 2: Sum counts

Step 2.2: Sparsity Aware Rectangular MM

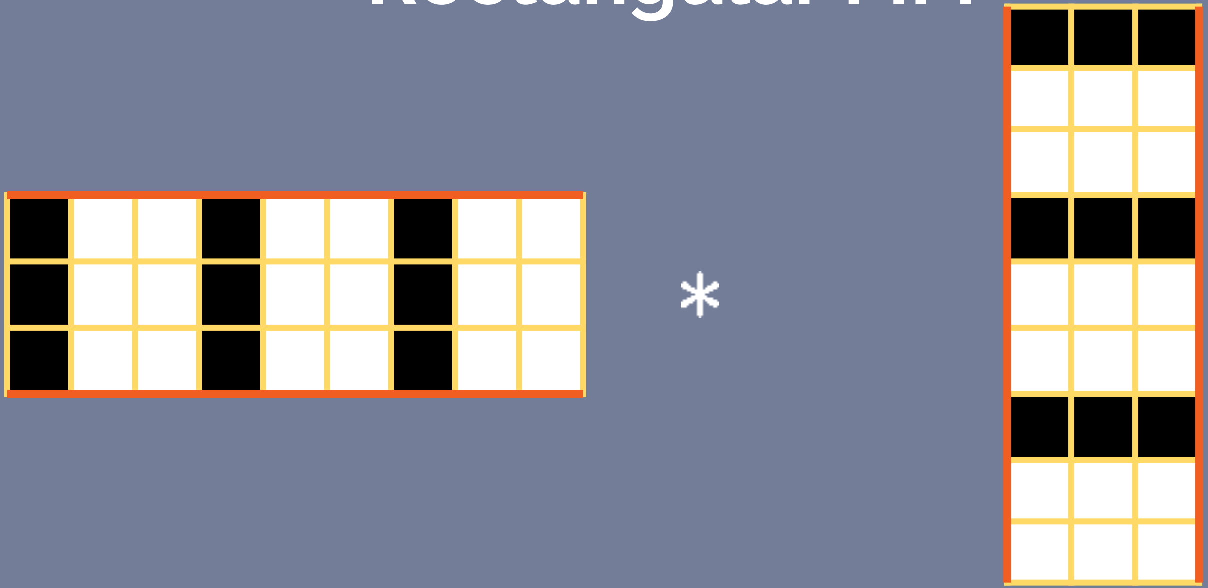


- Phase 1: Count non-zeros in S-cols, T-rows
- Phase 2: Sum counts
- Phase 3: Reorder

0 0 0 6 6 6 0 0 0

6 0 0 6 0 0 6 0 0

Step 2.2: Sparsity Aware Rectangular MM



- Phase 1: Count non-zeros in S-cols, T-rows
- Phase 2: Sum counts
- Phase 3: Reorder

0 0 0 6 6 6 0 0 0

6 0 0 6 0 0 6 0 0

Step 2: Receiving Challenge

SOLVED!

Step 2.2: Sparsity Aware Rectangular MM

Notice!

- a*b different rectangular MM
- n/(ab) nodes in each
- Inner reorderings = local knowledge of n/ab nodes

  Making them global knowledge = too expensive!

- Will be problematic soon

Step 3: Sending Challenge

- **Need every node to send roughly same**
- **Solution: balancing message duplication**

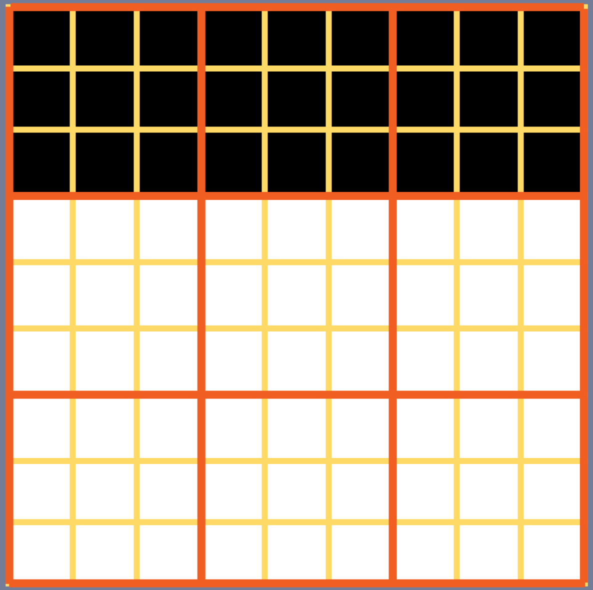
Dense Nodes Will Be Slow

Non-Zero

Zero

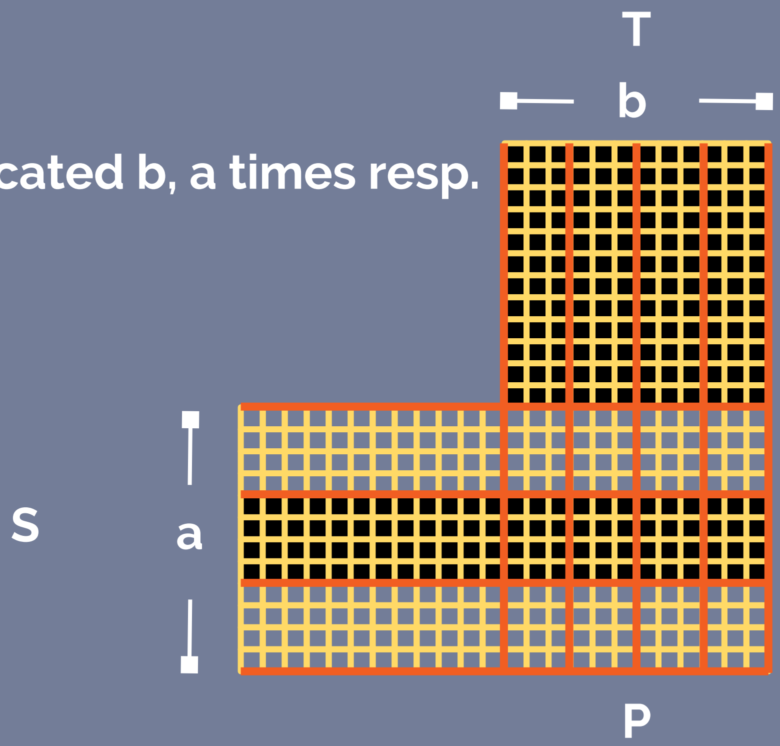
N/A

T



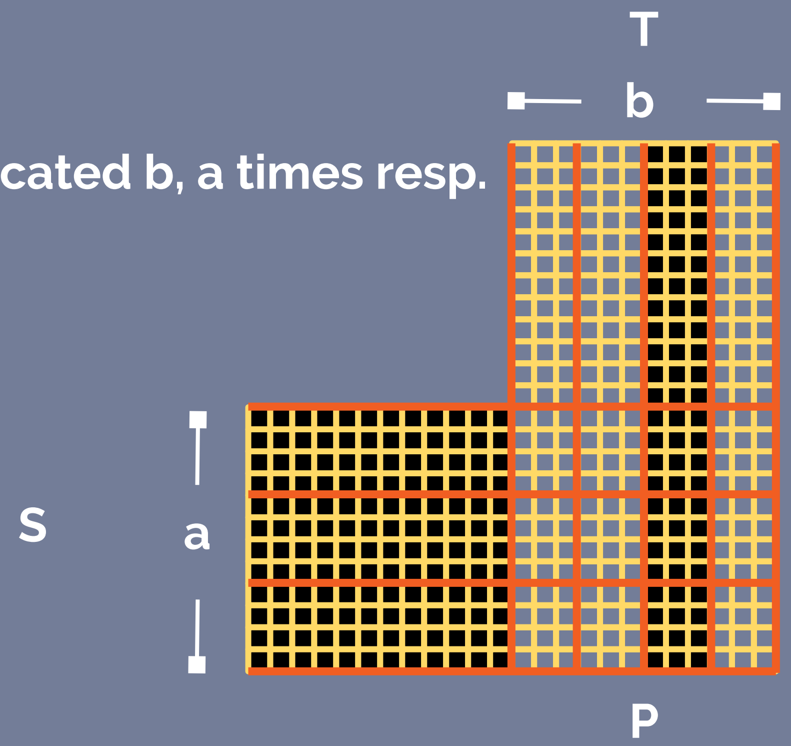
Message Duplication

- S, T duplicated b, a times resp.



Message Duplication

- S, T duplicated b, a times resp.



Step 3: Sending Challenge

- Key Point 1: Duplication is expensive!
- Key Point 2: Very easily load balanced - sparse nodes help dense nodes

Step 4: Knowledge Challenge

- Problem: Inner reorderings = local knowledge
- Senders do not know who to send messages to
- We show O(1) solution
    - Requires specific redistribution of elements in sending challenge - receivers know who needs to message them
    - Nodes ***request*** messages

Summary

- **For any (a, b), total runtime:**

$$O(b \cdot nz(S)/n^2 + a \cdot nz(T)/n^2 + n/ab)$$

- **Optimal (a, b):**

$$a = n \cdot nz(S)^{1/3}/nz(T)^{2/3}, b = n \cdot nz(T)^{1/3}/nz(S)^{2/3}$$

- **Resulting overall runtime:**

$$O(nz(S)^{1/3}nz(T)^{1/3}/n + 1)$$

Sparse MM - Our Main Result

- ➔ *P = S*T:*

$$O(nz(S)^{1/3} nz(T)^{1/3}/n + 1)$$

nz(A) = number of non-zero elements in *A*

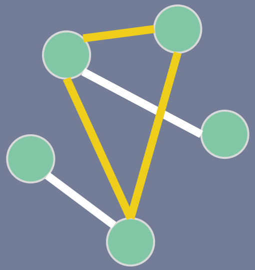
$$O(nz(S)) = O(nz(T)) = m \longrightarrow O(m^{2/3}/n + 1)$$

$$m = O(n^{3/2}) \Rightarrow O(1)$$

$$\min\{O(nz(S)), O(nz(T))\} = m \longrightarrow O(m^{1/3}/n^{1/3} + 1)$$

Sparse Triangle Listing

Triangle Listing



- Every triangle must be known to at least one node

Previous Work on Triangle Listing

- $O(n^{1/3})$ [Dolev, Lenzen, Peled DISC 2012]
- $\tilde{\Omega}(n^{1/3})$ [Izumi, Le Gall, PODC 2017,

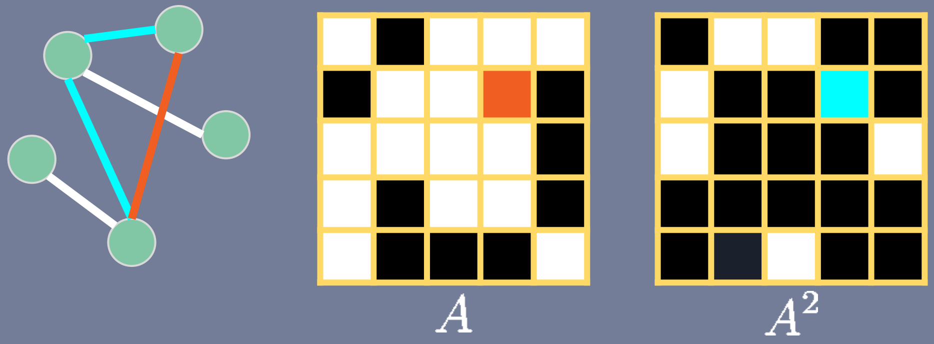
  Pandurangan, Robinson, Scquizzato, SPAA 2018]
- $\tilde{O}(m/n^{5/3})$ *w.h.p.*

  [Pandurangan, Robinson, Scquizzato, SPAA 2018]

Our Result: $O(m/n^{5/3} + 1)$ *deterministic*

Our Result

- Triangle = path length 1 (v→u) + path length 2 (u→v)



- Our runtime:

$$O(m/n^{5/3} + 1)$$

- <u>Notice:</u> this is faster than the time for squaring!
    - No need to compute all of A^2

Conclusion

Conclusion

Our Work

- New load balancing building blocks in Congested Clique
- New algorithms for Sparse MM, Triangle Listing

Open Questions

- Can the complexity of Sparse MM be improved in the clique? Sparse Ring MM?
- Lower bound for Sparse Triangle Listing?
- Using these algorithms/techniques in other models