PROJECT NAME:

Communication Avoiding All-Pairs Shortest Paths Algorithm for Sparse Graphs

CS 309
PARALLEL COMPUTING

14. NOVEMBER, 2021

TEAM MEMBERS:

- Keren Tudu -- 190001023
- Krishanu Saini -- 190001029
- Kuldeep Singh -- 190001030
- Rahul Kumar -- 190001049

SUBMITTED TO:

Dr. Chandresh Kumar Maurya

TABLE OF CONTENTS

01

Introduction

A

LATENCY:

The time is taken to receive the first packet of data between two points in a network

.

Latency depends on the topology of the network.

- BUS - O(n)
- RING - O(n/2)* (i7 uses this)
- Crossbar - O(1)
- Mesh - O(sqrt(N))
- H-tree - O(log(N))
- Hypercube - O(log(N)) well suited for the problem
- N is number of processors / cores

We will consider the value to be O(1) for calculations

B

BANDWIDTH:

Bandwidth is another concept that is often associated with latency. Bandwidth describes the maximum capacity of a network/internet connection. The less bandwidth a network has, the more latency.

C

PARALLEL ALGORITHM MODEL:

Parallel Algorithm Model
= Decomposition + Mapping + Minimize task-interaction

For our algorithm, we utilize a hybrid approach
DISTRIBUTION -- Block distribution
MODEL -- The Pipelined Task Graph Model
MAPPING -- Decentralized dynamic mapping

02

Background

A

THE ALL-PAIRS SHORTEST-PATHS PROBLEM:

An undirected weighted graph $G = \{V, E, W\}$

- vertex set V containing $n = |V|$ vertices
- Edge set E with $m = |E|$ edges
- Weights W

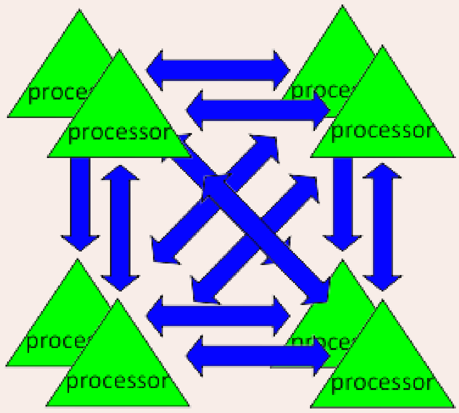
The all-pairs shortest path problem computes the length of the shortest paths between every pair of vertices in the graph G.

B

THE DISTRIBUTED-MEMORY COST MODEL:

We quantify interprocessor bandwidth (the number of words) and latency (the number of messages) costs of a parallelization via a network model processor processor processor processor.

- The architecture is homogeneous.
- A processor can only send/receive a message to/from one other processor at a time.
- There is a link between each processor pair (all-to-all network)



C

GOALS:

- To design an APSP algorithm with minimum communication cost for sparse graphs.
- To give the lower bound of bandwidth cost and latency cost.
- To design an APSP algorithm with minimum communication cost for sparse graphs.
  - Bandwidth cost: $O(\frac{n^2 log^2 P}{P} + |S|^2 log^2 P)$
  - Latency cost: $O(log^2 P)$
- To give the lower bound of bandwidth cost and latency cost
  - Bandwidth lower bound: $\Omega(\frac{n^2}{P} + |S|^2)$
  - Latency lower bound: $\Omega(log^2 P)$

D

FLOYD-WARSHALL ALGORITHM (FW) :

Floyd-Warshall algorithm

At each iteration k, the distance matrix A is updated

- $A(i,j) = A(i,j) \oplus A(i,k) \otimes A(k,j)$
- $x \oplus y = \min\{x, y\}, x \otimes y = x + y$

Floyd-Warshall algorithm

Divide A into $n/b \times n/b$ blocks, each of size $b \times b$ At each iteration k

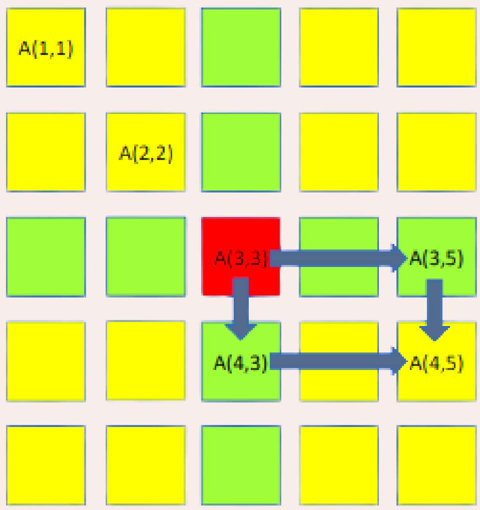
$A(k,k) = FW(A(k,k))$

$A(:,k) = A\ :,k \oplus A(:,k) \otimes A(k,k)$

$A(k,:) = A\ k,: \oplus A(k,k) \otimes A(k,:)$

$A(i,j) = A\ i,j \oplus A(i,k) \otimes A(k,j)$

For example: k=3



All blocks of A need to be updated

How to avoid the update of certain blocks for sparse graphs ?

03

Several Algorithmic Techniques

A

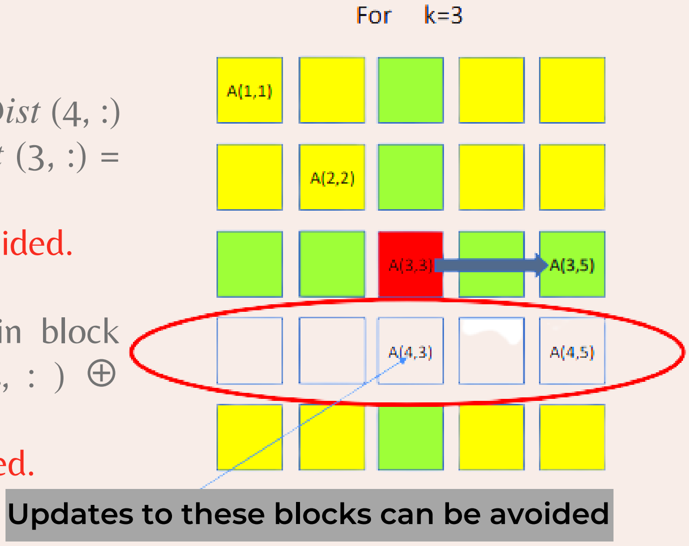
PROJECTED DURATION:

For k = 3, if Dist (4,3) = ∞, then Dist (4, :) = min{Dist (4, :), Dist (4,3) + Dist (3, :) = Dist (4, :)

The update of Dist (4, :) can be avoided.

Similar, for k = 3, if all entries in block A(4,3) is ∞, then A(4, :) = A(4, :) ⊕ A(4,3) ⊗ A(3, :)

The update of A(4, :) can be avoided.



However, A is irregular and there may not be all infinite values in a block.

B

NESTED-DISSECTION ORDERING (ND PROCESS):

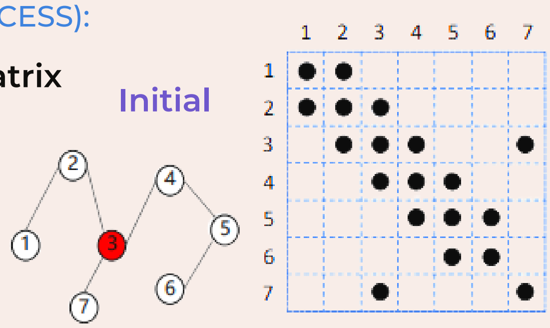
ND process: reorder the adjacency matrix

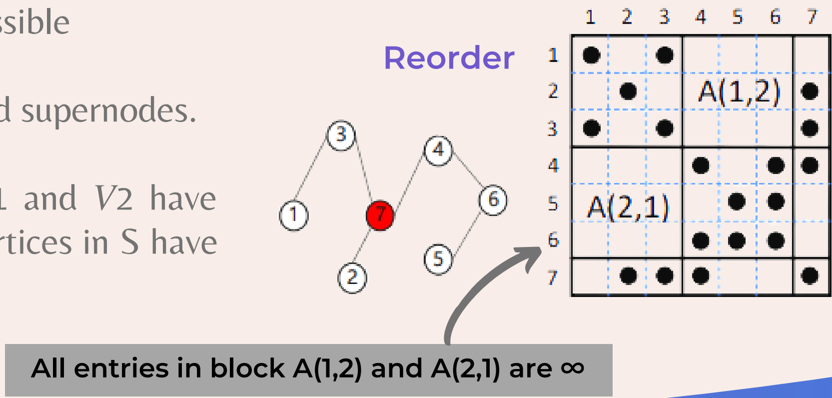
Find the vertex separator S, S partitions V into three disjoints sets, $V = V1 \cup S \cup V2$, and

- No edges between $V1$ and $V2$
- $|V1| = |V2|$
- S is as small as possible

$V1$, $V2$, and S are called supernodes.

The vertices within $V1$ and $V2$ have consecutive indices; vertices in S have a higher index.





C

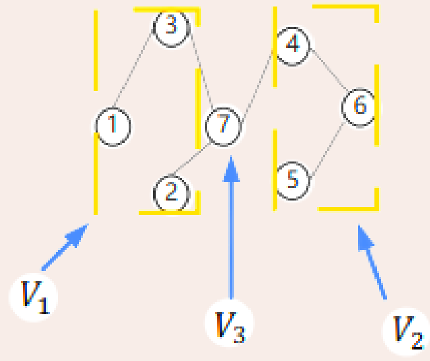
ELIMINATION TREE (ETREE) :

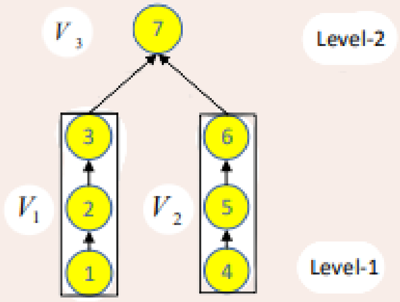
By computing the separator of graph G, we can get a two-level elimination tree (eTree).

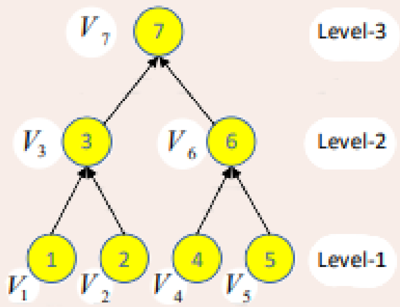
By recursively computing the separators of $V1$ and $V2$, we can obtain a multi-level eTree.

The eTree can guide parallelism.

- The elimination of supernodes in the same level is independent.







B

CHALLENGE:

The computational cost of the FW algorithm is $O(n^3)$. Using ND process and eTree techniques, the computational cost can be reduced to $O(n^2S)$

How to reduce communication cost in the distributed memory model?

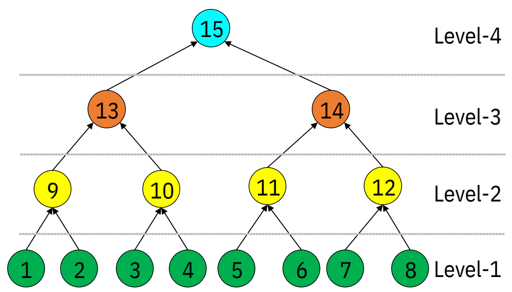
04

Our Method

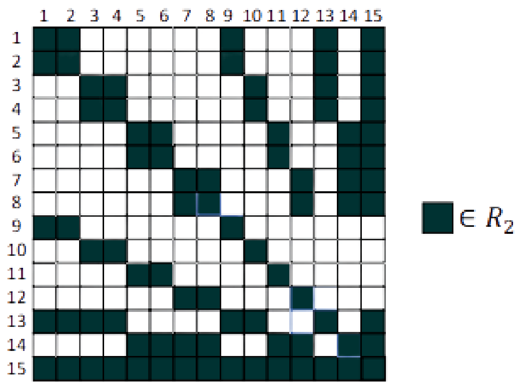
A

SYMBOL DESCRIPTION:

We map the supermodel block sparse matrix A to a $\sqrt{P} \times \sqrt{P}$ grid in a block layout.
Symbol description:
$A(k)$: The set of all ancestors of supernode k
$D(k)$: The set of all descendants of supernode k
$C(k)$: The set of all cousins of supernode k
Ql: The collection of the l-th level supernodes
Rl: The updated region of A during the elimination of the l-th level supernodes.



$$R_l = \cup_{k \in Q_l} (k \cup A(k) \cup D(k), k \cup A(k) \cup D(k))$$



B

OUR METHOD:

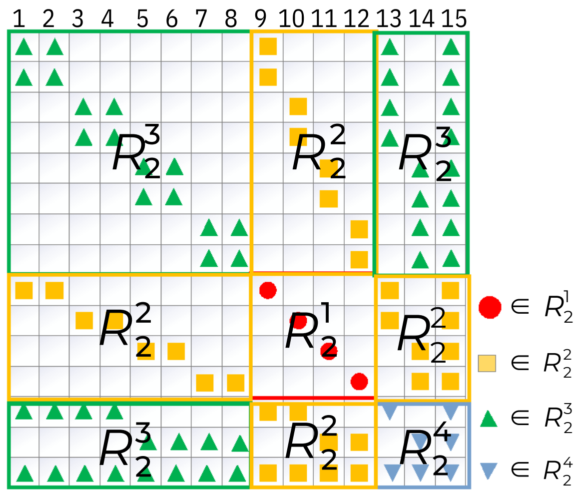
Divide Rl into four subsets:

$R_l^1 = \cup_{k \in Q_l} (k, k)$
$R_l^2 = \cup_{k \in Q_l} (A(k) \cup D(k), k) \cup (k, A(k) \cup D(k))$
$R_l^3 = \cup_{k \in Q_l} (A(k) \cup D(k), D(k)) \cup (D(k), A(k))$
$R_l^4 = \cup_{k \in Q_l} (A(k), A(k))$

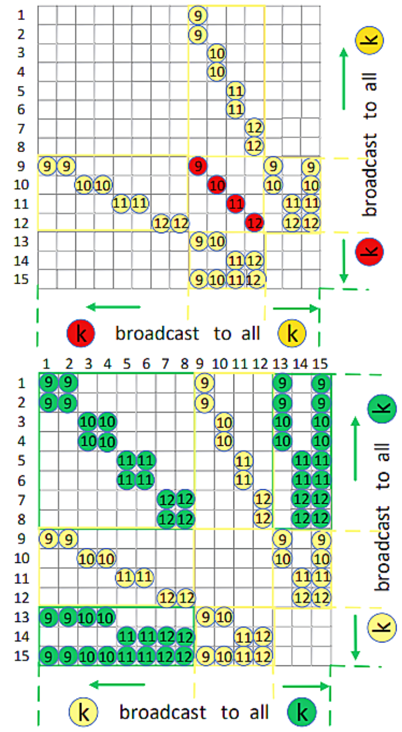
For each $(i, j) \in R_l$, the update of A(i, j) is

$$A(i, j) = A(i, j) \oplus \sum_{k}^{\oplus} A(i, k) \otimes A(k, j)$$

where $k \in (A(i) \cup D(i)) \cap (A(j) \cup D(j)) \cap Q_l$



THE UPDATE OF R_i^1, R_i^2, R_i^3 AND R_i^4



The update of R_i^1: P_{kk} performs local updates

The update of R_i^2:

- P_{kk} broadcast to all P_{ik}, where $i \in A(k) \cup D(k)$
- P_{kk} broadcast to all P_{kj}, where $j \in A(k) \cup D(k)$

The update of R_i^3:

- For each $(i,k) \in R_i^2$, P_{ik} broadcast to all P_{ij}, $j \in A(k) \cup D(k)$
- For each $(k,j) \in R_i^2$, P_{kj} broadcast to all P_{ij}, $i \in A(k) \cup D(k)$

The update of R_i^4 :

If $|Ai \cup Di \cap Aj \cup Dj \cap Q\, l| = q$, then $A(i,j)$ needs to be updated q times,
i.e., $A(i,j) = A(i,j) \oplus A(i,1) \otimes A(1,j) \oplus A(i,2) \otimes A(2,j) \ldots \oplus A(i,q) \otimes A(q,j)$

A trivial strategy: $Pi1$, $Pi2$ ..., Piq send local data to Pij in sequential

- Latency cost: $\Omega(q)$

Optimal strategy: $Pi1$, $Pi2$ ..., Piq send local data to q different processors, each processor performs a computing unit and then reduces it to $P(i,j)$.

- Latency cost: $O(\log q)$

There is more than one block $A(i,j)$ in R_i^4 needs to be updated.

In order to update all the blocks in R_i^4 with a maximum degree of parallelization, the optimal strategy is to allocate each computing unit that updates R_i^4 to a separate processor one-to-one.

The number of computing units required to update R_i^4 is $O(P)$.

- By summing the number of units of each Aij

We get such a one-to-one mapping from the computing units for updating R_i^4 to the processors.

- For each A(i,j) in R_i^4, each computing unit is all A(i,k) ⊗ A(k,j), where $k \in Q_l \cap D(i) \cap D(j)$.
- Each computing unit can be assigned to a separate processor P_{fg}, where $f = \sum_{b=h+a-c}^{h-1} 2^b + (a-l)$, $g = k - \sum_{b=h-l+1}^{h-1} 2^b$, $a \in \{l+1, l+2 \ldots, h\}$ and $c \in \{a, a+1 \ldots, h\}$.

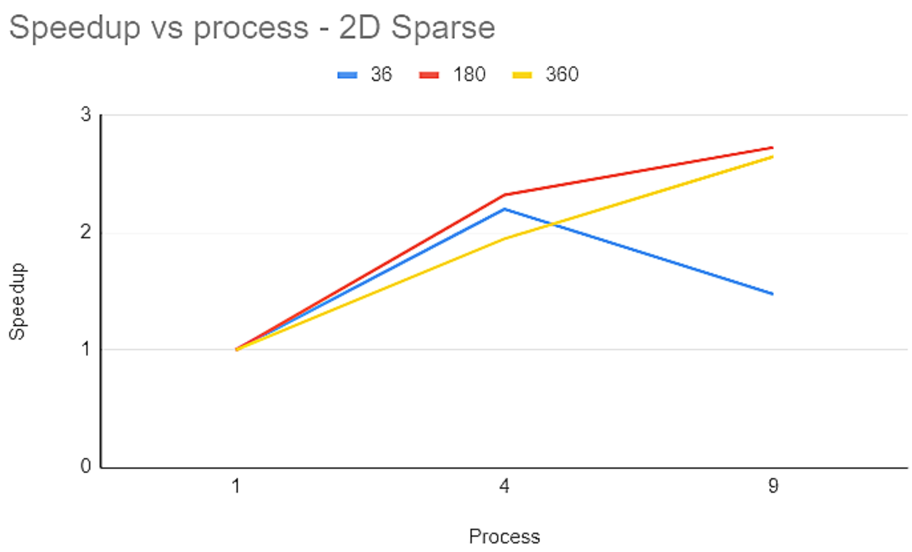
C

GRAPH:

2D-Sparse-algorithm

nxn - graph size	18	36	90	180	270	360
P(process)	time(us)					
1	43.15	327.34	4596.94	35968.78	131376.98	284208.77
4	57.45	148.77	2029.65	15497.20	59540.98	146014.45
9	79.87	221.96	1821.51	13204.57	46028.61	107426.16
S(Speedup)	18	36	90	180	270	360
1	1	1	1	1	1	1
4	0.7511	2.2003	2.2649	2.3210	2.2065	1.9464
9	0.5403	1.4748	2.5237	2.7240	2.8542	2.6456
Efficiency(S / P)	0.1351	0.3687	0.6309	0.6810	0.7136	0.6614

SPARSE GRAPH ----- around log^2(n) edges



05

Proof of Lower Bound

3NL COMPUTATION MODEL:

The computation of matrix multiplication and APSP can be expressed in a three-nested-loop (3NL) way. multiplying two $n \times n$ matrices:

$$Cij = Cij + Aik \cdot Bkj$$

Informally, the 3NL computation model is defined as follows:

- There are two non-trivial parameter-dependent functions f_{ij}, g_{ijk} such that $Cij = f_{ij}(g_{ijk}(Aik, Bkj))$
- The elements in A, B, and C are mapped to memory locations one by one.

3NL computation model lower bound:

- bandwidth lower bounds $\Omega(\frac{F}{P\sqrt{M}})$
- latency lower bounds $\Omega(\frac{F}{PM^{2/3}})$

F: the number of computation operations
M: per-process memory size.

PROOF:

Computing the APSP of a sparse graph is a 3NL computation.

The total number of operations to compute the APSP is $\Omega(n^2|S|)$:

- By calculating the number of computation operations required in the elimination of the top-level supernodes, which is a part of the total operations.

The bandwidth and latency lower bounds for solving the APSP of sparse graphs are $\Omega(n^2P + |S|^2)$ and $\Omega(\log^2 p)$, respectively.

- By applying the lower bound of operations and M to the 3NL computation model lower bound.
- By summing the lower bound of the latency cost during the elimination of each level of supernodes.

06

Improved Algorithm

A

ALGORITHMS:

- Floyd-Warshall classicalFW
- Johnson's algorithm
- BlockedFW
- superFW - Sao et al.
- 2D-DC-APSP / 2.5D-DC-APSP
- 2D-SPARSE-APSP

B

OUR APPROACH:

OPTIMIZING COMMUNICATION:

We propose a method using the latest findings in communication-avoiding algorithms to improve the method given in the research paper. (E.Solomonik et al)

Communication-avoiding '2.5D' algorithms take advantage of the extra available memory and reduce the bandwidth cost of many algorithms in numerical linear algebra. Generally, 2.5D algorithms can use a factor of c more memory to reduce the bandwidth cost by a factor of √c.

Table:

PARAMETER	2D-DC-APSP	2.5D-DC-APSP	3D-DC-APSP
Per-process memory	$O(\frac{n^2}{P})$	O(c * n² / √p)	O(n² / p^{1/2})
Bandwidth cost (B)	$O(\frac{n^2}{\sqrt{P}})$	$O(n^2/\sqrt{cp})$	O(n² / p^{2/3})
Latency cost (L)	$O(\sqrt{P}log^2P)$	$O(\sqrt{cp}\log^2(p))$	O(log²(p))

We observe -

- The 2.5D algorithm works asymptotically as fast as the 2D algorithm. For large problem sizes, the factor of √c is prominent.
- The 3D algorithm we chose utilizes a parallel implementation of Strassen matrix multiplication on a distributed platform and reduces communication bandwidth and latency by a factor of p1/6 wrt 2D algorithms. (Agarwal et al).
- We can theoretically utilize the above algorithm with 2.5D-DC-algorithm to achieve improved communication latency and bandwidth, we call the algorithm 3D-DC-algorithm.

PARAMETER	AWARENESS	CONSIDERATION
PER-PROCESS MEMORY	$O(n^2 / p^{2/3})$	$O(\frac{n^2}{P} + \|S\|^2)$
BANDWIDTH COST (B)	$O(n^2 / p^{2/3})$	$O(\frac{n^2log^2P}{P} + \|S\|^2log^2P)$
LATENCY COST (L)	$O(log^2P)$	$O(log^2P)$

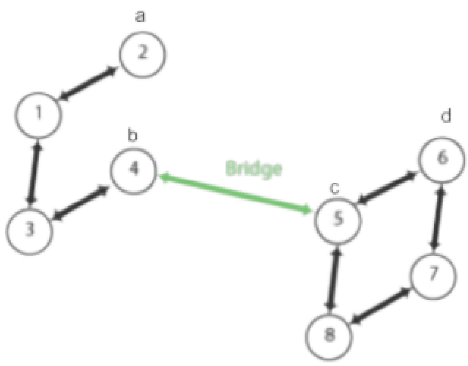
In fact, the improved algorithm outperforms 2D-SPARSE-APSP(research paper algorithm) in per-process memory usage in terms of |S| supernodes. Bandwidth cost for our algorithm will become better if |S| ~ O(n), that is the graph is dense.

IMPROVEMENT FOR SPECIFIC CASES:

CUT-EDGE ALGORITHM:

Step1:

- Find bridges in graphs or cut-edges(bridges).
- Using Karger's algorithm similar to Tarjan's algorithm.



Step2:

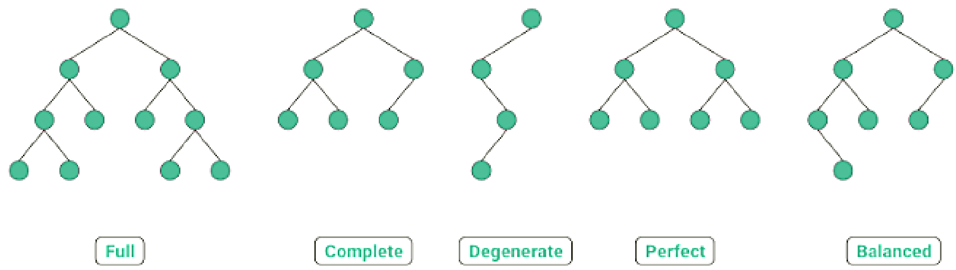
- ASPS can be calculated for 2 components as follows.

d(a,d) = d(a,b) + 1 + d(c,d), d is minimum path

Property of cut-edges.

Steps:

1. Solve 2 components in parallel.
2. Fill matrix for nodes across cut-edges using the above formula.



Recursively perform the above steps:

- Latency is the number of times a message is sent using MPI in the critical path.
- The latency is O(log P), which is better than previous algorithms.
- The bandwidth is O(n^2).

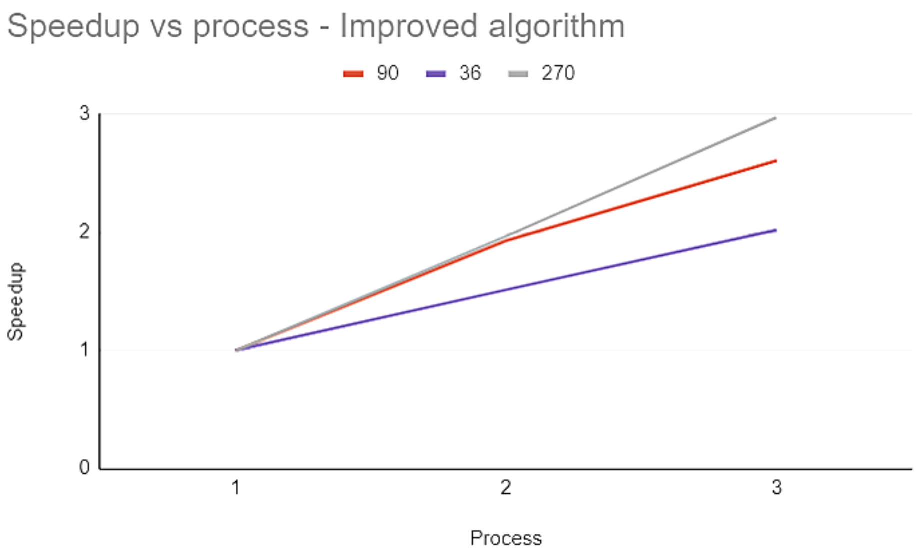
B

GRAPH:

Improved-algorithm

nxn - graph size	18	36	90	180	270	360
P(process)	time(us)					
1	21.22	130.89	1924.03	35968.78	14751.91	110912.79
2	28.13	86.54	1013.80	20497.20	7500.17	57510.37
3	19.07	64.84	665.42	13204.57	4963.40	42553.66
S(Speedup)	18	36	90	180	270	360
1	1	1	1	1	1	1
2	0.7544	1.5125	1.8978	1.7548	1.9669	1.9286
3	1.1127	2.0187	2.8915	2.7240	2.9721	2.6064
Efficiency(S / P)	0.3772	0.7562	0.9489	0.8774	0.9834	0.9643

SPARSE GRAPH ---- around log^2(n) edges



07

Summary

PARAMETER	2D-DC-APSP	SPARSE-APSP	LOWER BOUND	
			Dense graph	Sparse graph
Per-process memory (M)	$O(\frac{n^2}{P})$	$O(\frac{n^2}{P} + \lvert S \rvert^2)$	$\Omega(\frac{n^2}{P})$	$\Omega(\frac{n^2}{P})$
Bandwidth cost (B)	$O(\frac{n^2}{\sqrt{P}})$	$O(\frac{n^2 log^2 P}{P} + \lvert S \rvert^2 log^2 P)$	$\Omega(\frac{n^2}{\sqrt{P}})$	$\Omega(\frac{n^2}{P} + \lvert S \rvert^2)$
Latency cost (L)	$O(\sqrt{P} log^2 P)$	$O(log^2 P)$	$\Omega(\sqrt{P})$	$\Omega(log^2 P)$


THANK
YOU