DREAM:Lab

A Partition-centric Distributed Algorithm for Identifying Euler Circuits in Large Graphs

Siddharth D Jaiswal & Yogesh Simmhan
Department of Computational and Data Science,
Indian Institute of Science, Bangalore, India

INDIAN INSTITUTE OF SCIENCE

Department of Computational and Data Sciences

2019 IEEE International Workshop on High-Performance Big Data, Deep Learning, and Cloud Computing

DREAM:Lab

Introduction and Algorithm

- Euler Circuits and Use Cases
- Preliminaries
- Algorithmic walkthrough

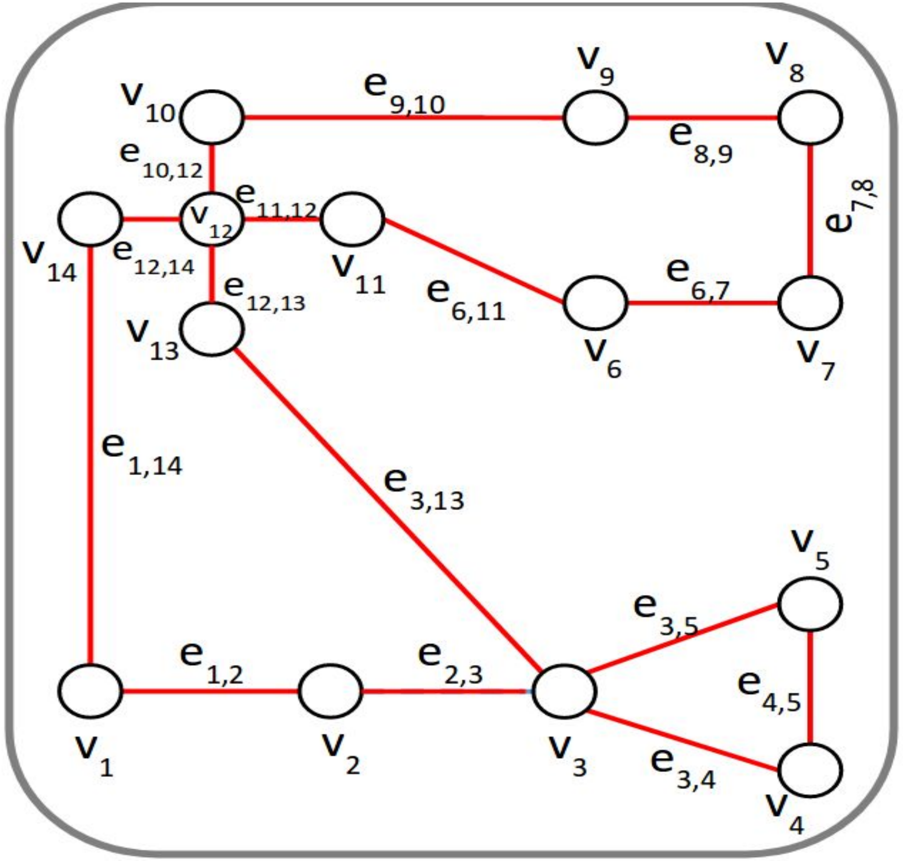
Introduction

Eulerian Circuits-

A circuit starting at some vertex $\boldsymbol{v_i}$ in a graph such that every edge is visited exactly once and the circuit ends at the same vertex, $\boldsymbol{v_i}$.

These circuits find uses in Biological Networks, Transportation Networks, IoT applications and electrical circuit design.

Example Eulerian Circuit



All vertices are even degree

A potential Eulerian Circuit-

$(v_1, v_2, v_3, v_4, v_5, v_3, v_{13}, v_{12}, v_{10}, v_9, v_8, v_7, v_6, v_{11}, v_{12}, v_{14}, v_1)$

Introduction

Existing Parallel Algorithms are impractical for commodity clusters and VMs.

We propose an algorithm which identifies partial paths and cycles within partitions of a large graph, iteratively over multiple Supersteps, using a **B**ulk **S**ynchronous **P**arallel model.

Partition Centric Euler Circuit

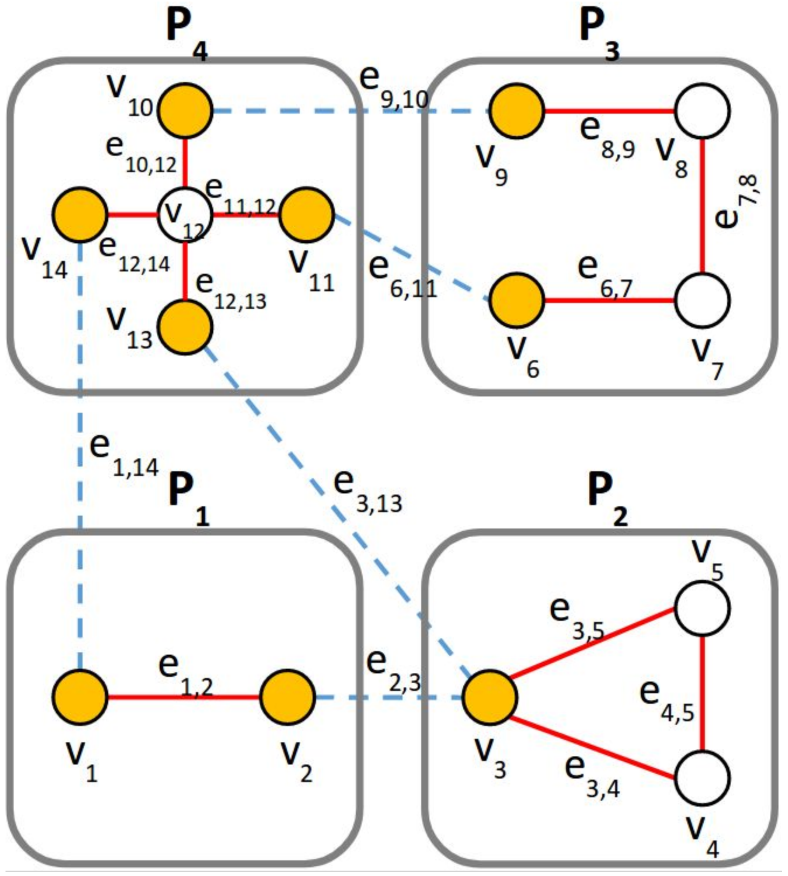
To find an Eulerian Circuit on a large distributed graph using a partition-centric model of computation.

- Undirected Eulerian graph partitioned into **P** partitions.
- A partition has odd**(OB)** and even**(EB)** degree Boundary vertices and even degree Internal vertices**(IV)**.
- Meta graph-
  - Each partition is a meta vertex.
  - All edges between a pair of partitions compose a single meta edge.

Partition Centric Euler Circuit

- Identify pairwise edge-disjoint partial paths locally in each partition.
- Recursively merge the partial paths in pairs of partitions to coarsen them.
- Construct the final circuit

Partition Centric Euler Circuit



Partitions: P_1-P_4

OB: $v_1, v_2, v_6, v_9, v_{10}, v_{11}, v_{13}, v_{14}$

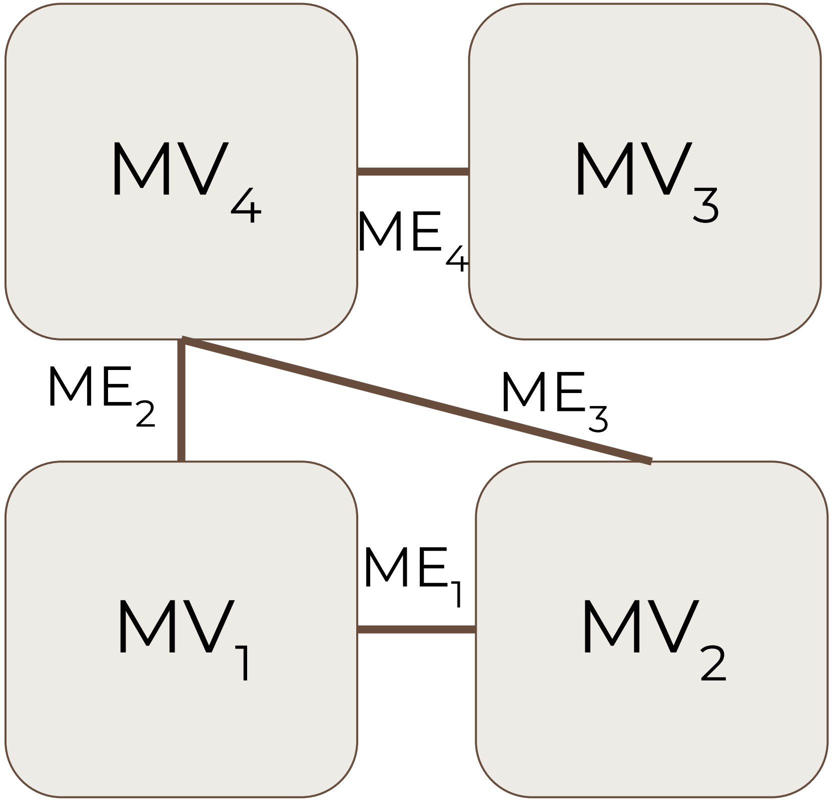
EB: v_3

IV: v_4, v_5, v_7, v_8

Local Edges(in Red)

Remote Edges(in Blue)

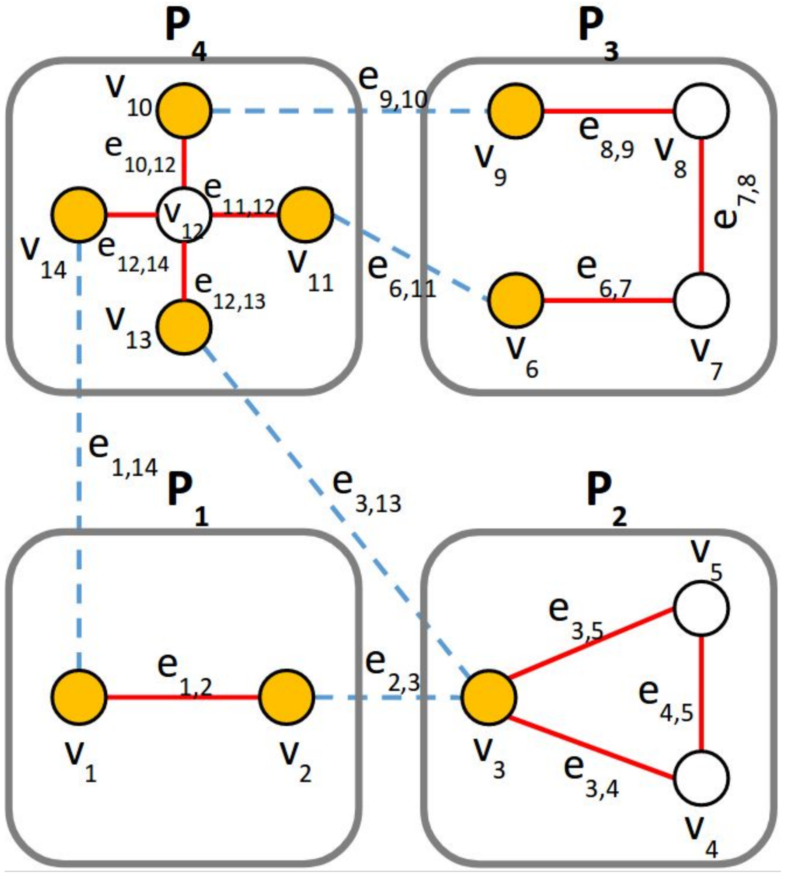
Partition Centric Euler Circuit-Meta Graph View

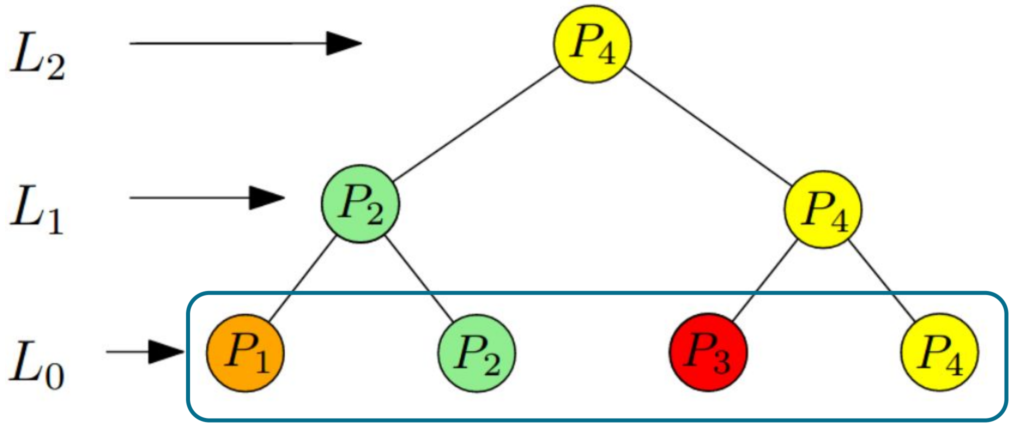


Meta Vertices: {MV_1, MV_2, MV_3, MV_4}

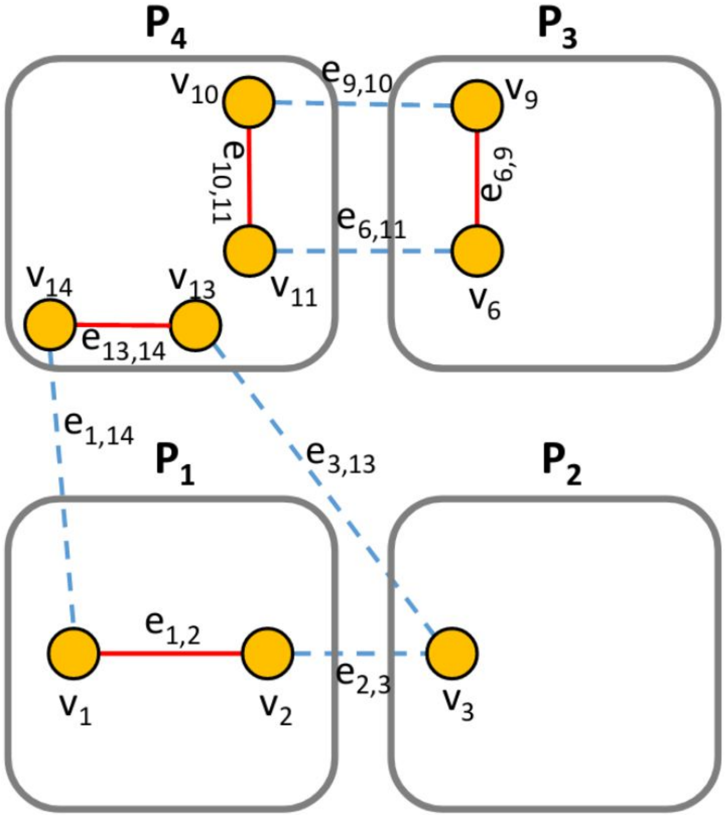
Meta Edges: {ME_1, ME_2, ME_3, ME_4}

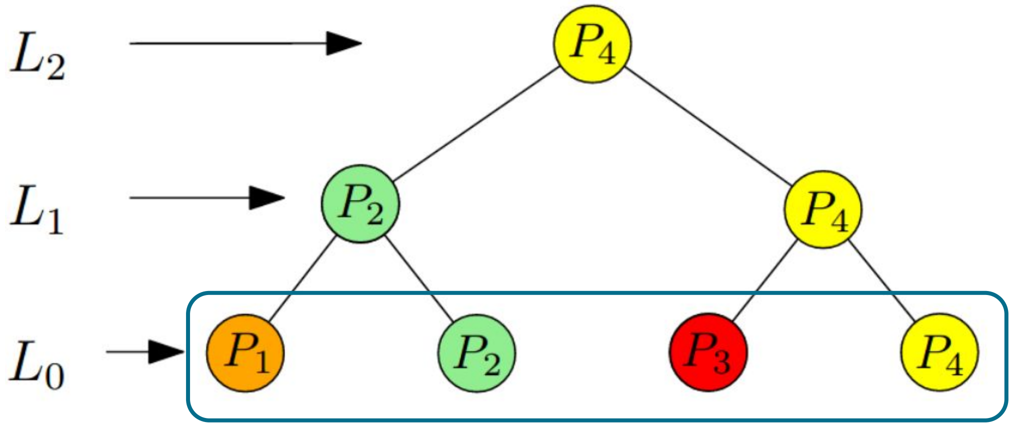
Partition Centric Euler Circuit



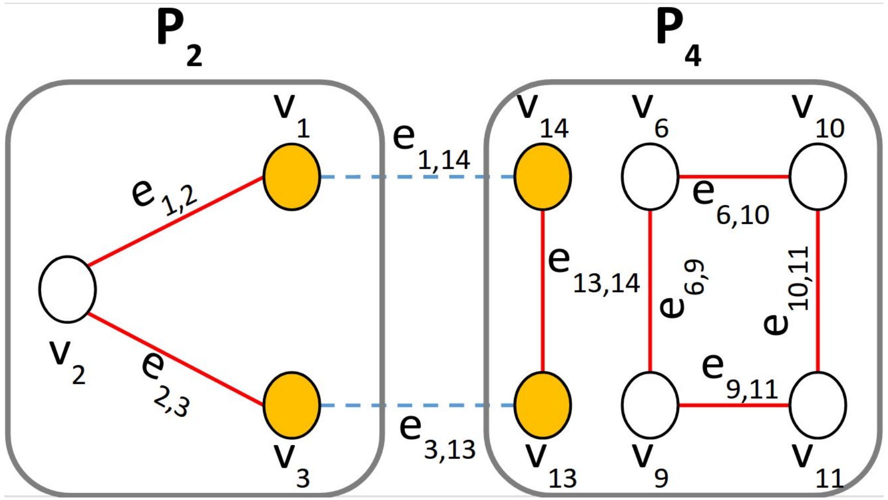


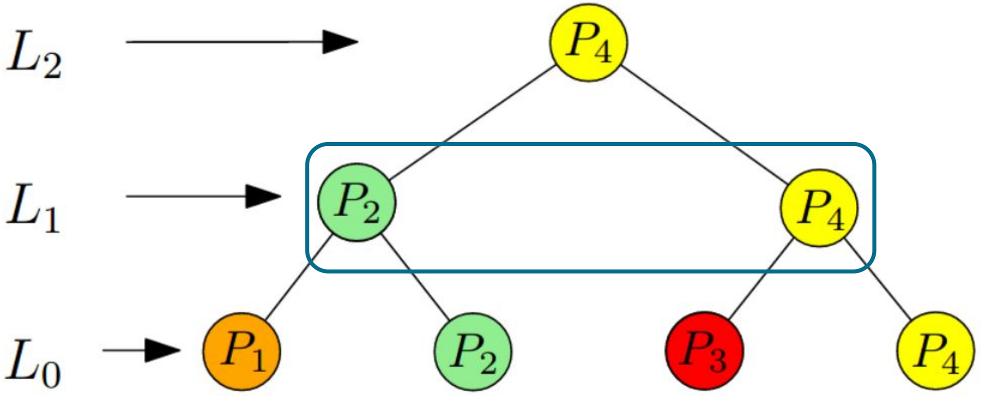
Partition Centric Euler Circuit



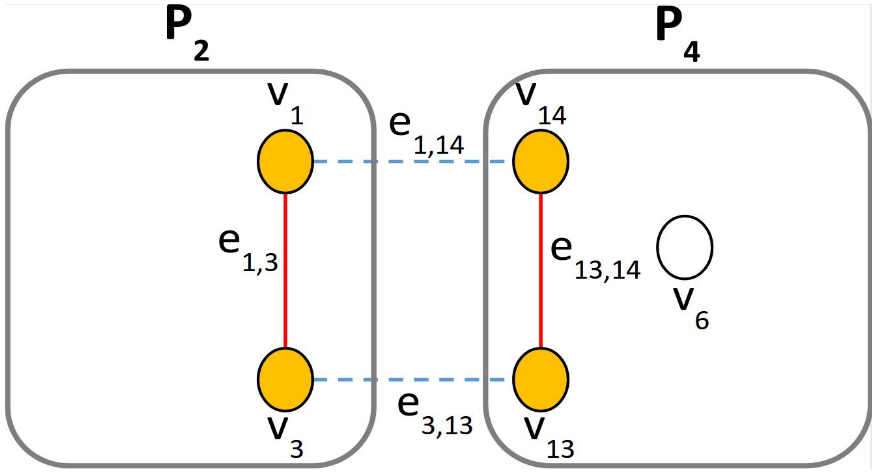


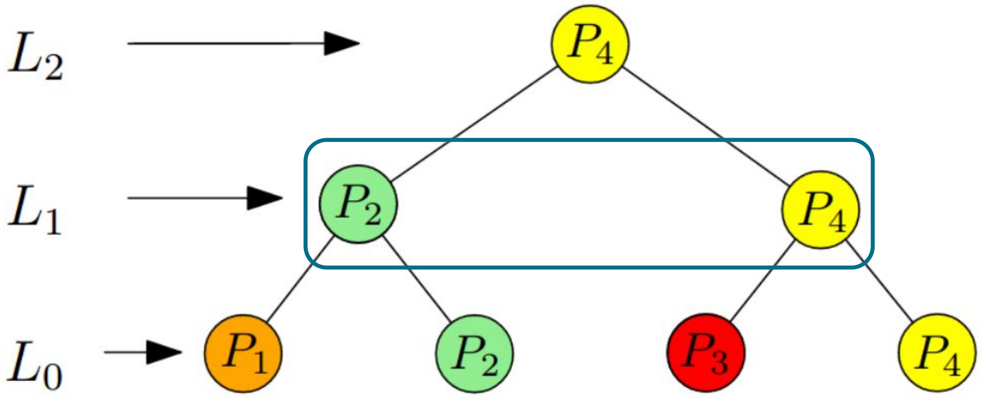
Partition Centric Euler Circuit



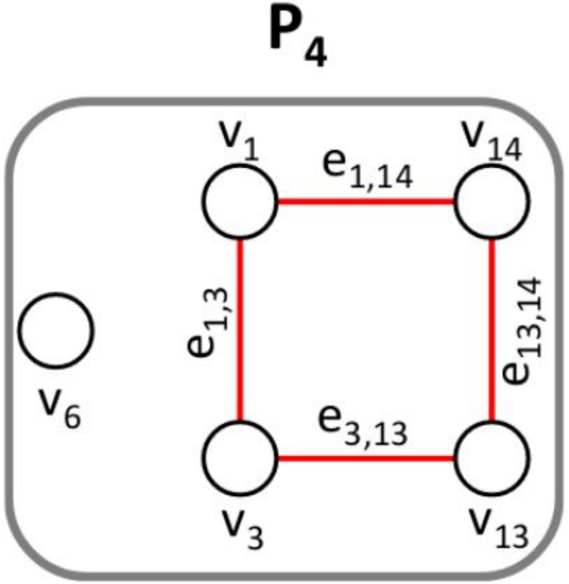


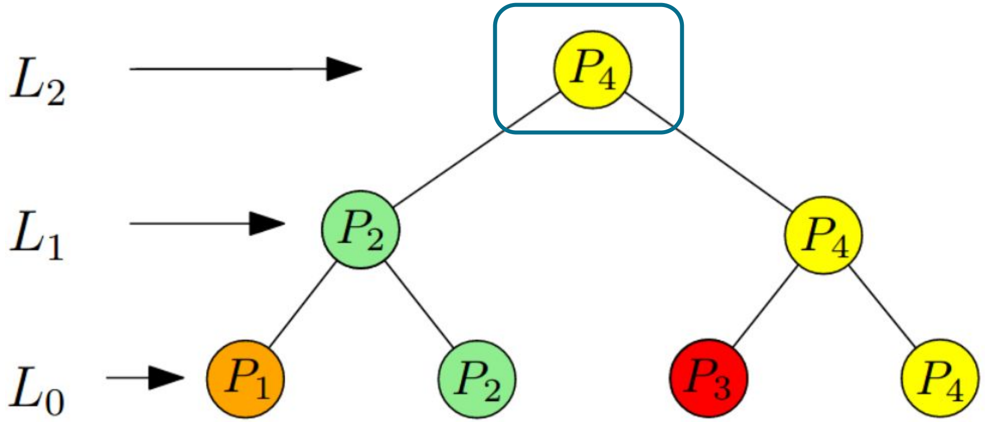
Partition Centric Euler Circuit





Partition Centric Euler Circuit





Complexity Measures

- Coordination
- Computation
- Communication

Coordination Cost

Number of Supersteps-

Path Identification within a partition - 1 Superstep

Levels of the Binary Merge Tree- log(n)

$\boxed{\log(n) + 1}$

for *n partitions.*

Computation Cost

Within a partition at every level-

Time to compute the partial paths(cycles) from OB, EB and (if required) IV vertices.

$$\boxed{O(|BV_i| + |IV_i| + |LE_i|)}$$

where, $BV_i = OB_i + EB_i$ and LE_i = Local Edges.

Communication Cost

Merging of pairs of partitions at every level-

OB + EB + OB Paths + Remote Edges compose this message exchange between partitions.

$$\boxed{O(|BV_i| + |RE_i|)}$$

where, $BV_i = OB_i + EB_i$ and RE_i = Remote Edges

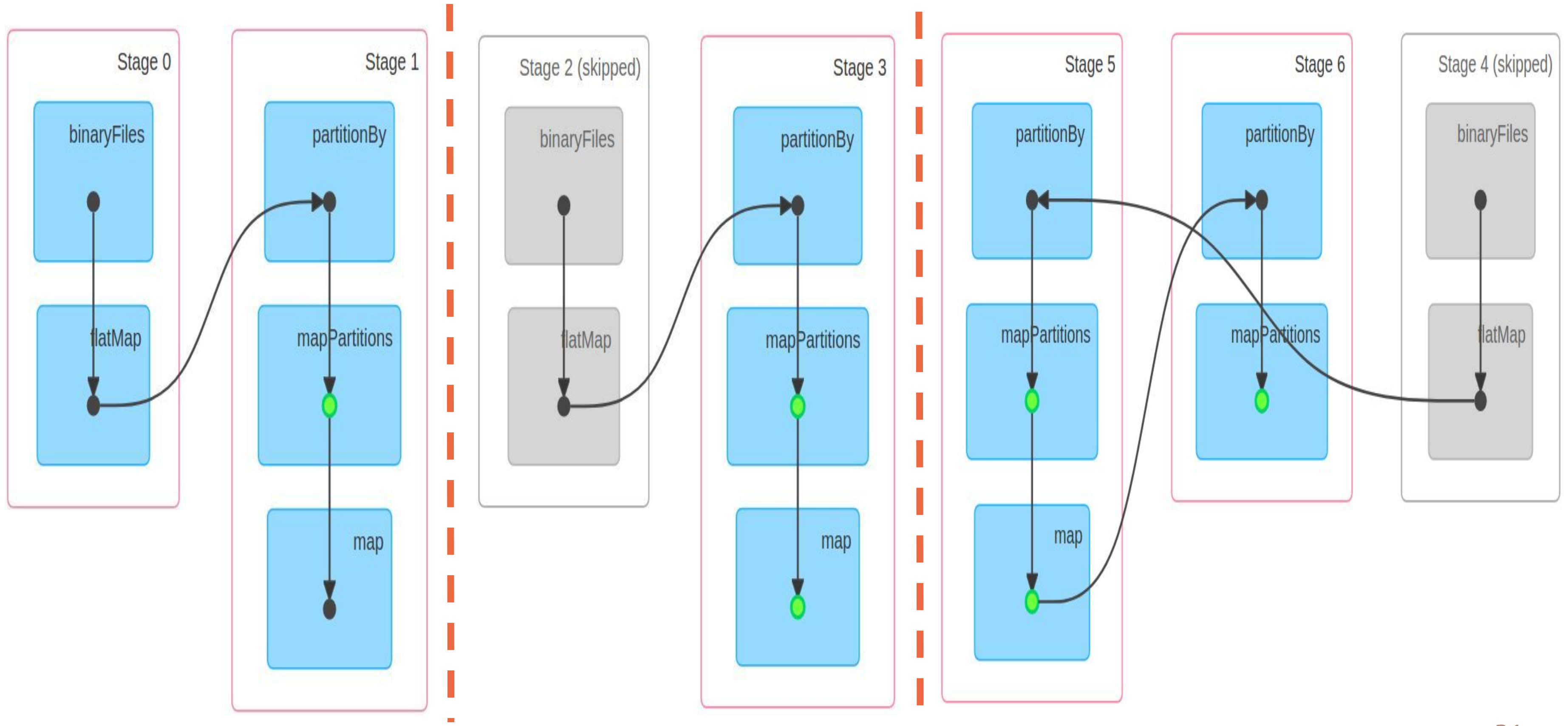
Results

- Experimental Setup
- Graph Characteristics
- Discussion

Results- Experimental Setup

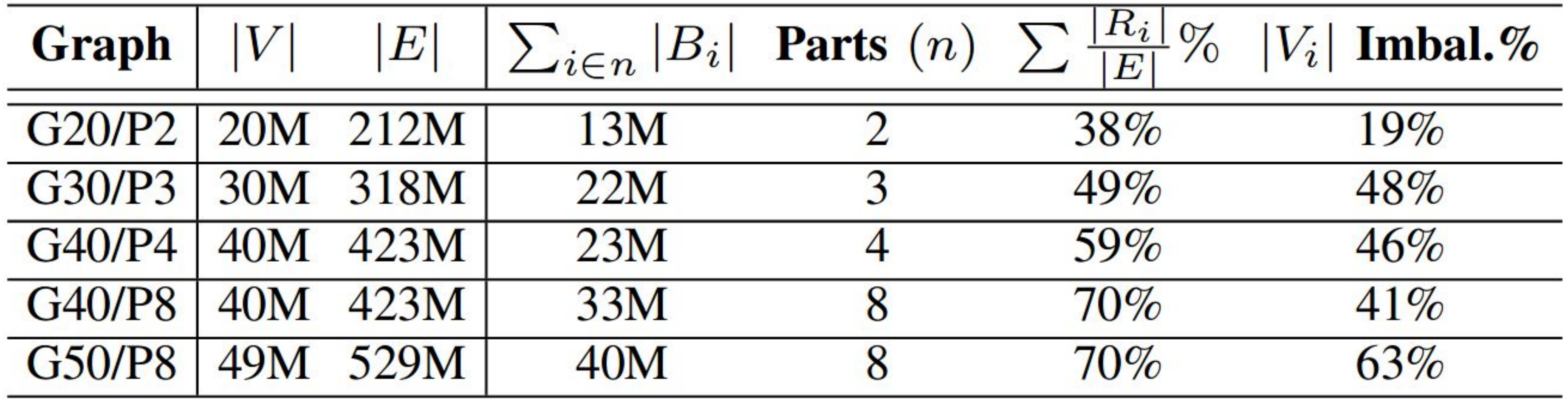
- Apache Spark 2.2 with Java 8
- 8 Microsoft Azure E8/v3 VMs.
  - Each has 8 cores(Intel Xeon E5-2673, 2.3 GHz)
  - 64 GB RAM, 128 GB Local Storage
- 1 Spark Executor/VM assigned 45 GB RAM
- 1 Spark Executor/Partition of the Graph
- Graph Generator Tool: PaRMat
- Partitioning Tool: ParHip
- Extra Edges: ~5%

Apache Spark Pipeline



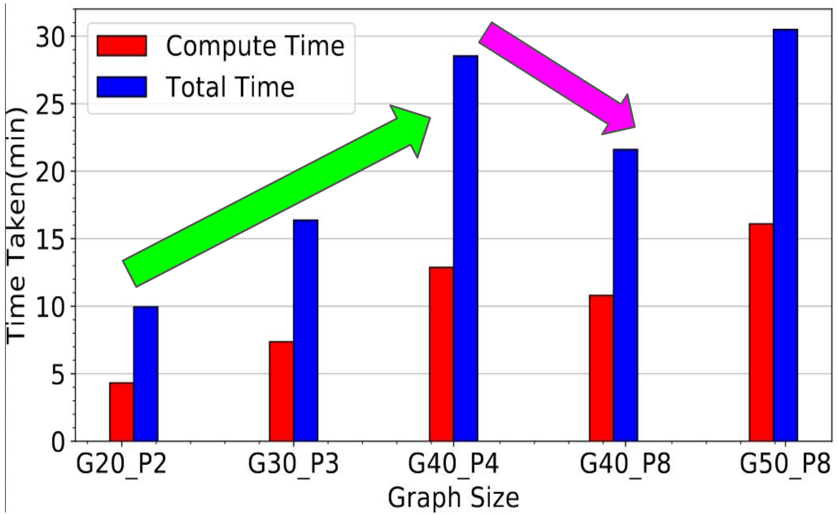
DREAM:Lab

Results- Graph Characteristics



Graph	$\lvert V\rvert$	$\lvert E\rvert$	$\sum_{i\in n}\lvert B_i\rvert$	**Parts** (n)	$\sum\frac{\lvert R_i\rvert}{\lvert E\rvert}\%$	$\lvert V_i\rvert$ **Imbal.%**
G20/P2	20M	212M	13M	2	38%	19%
G30/P3	30M	318M	22M	3	49%	48%
G40/P4	40M	423M	23M	4	59%	46%
G40/P8	40M	423M	33M	8	70%	41%
G50/P8	49M	529M	40M	8	70%	63%

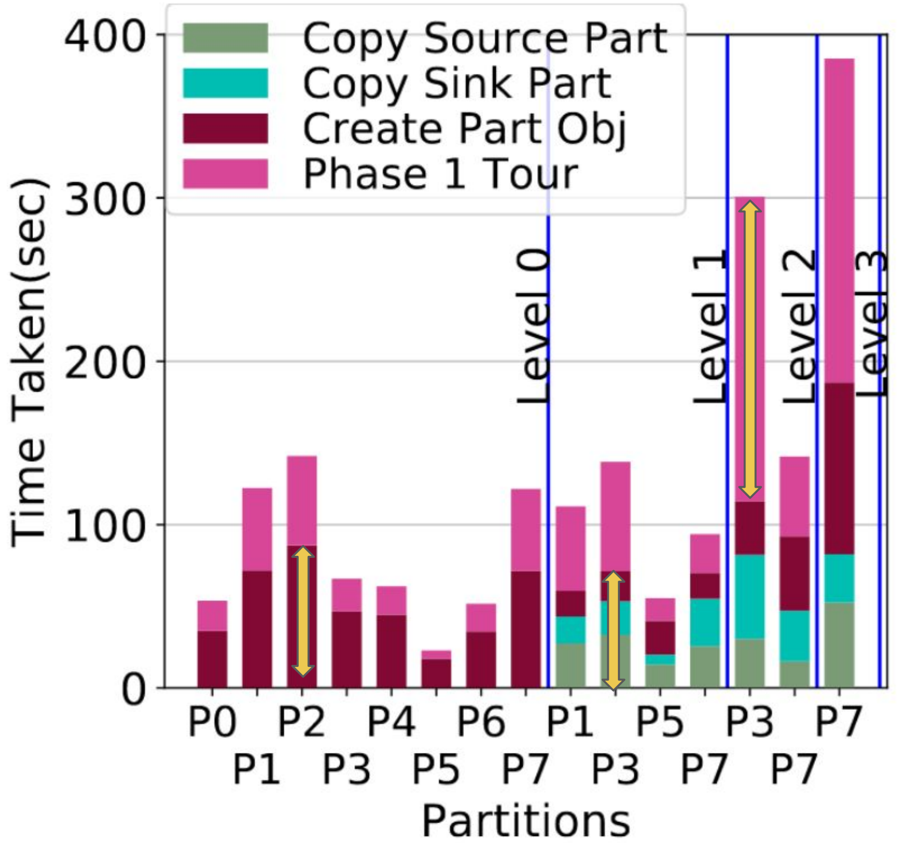
Results- Total Time



- Weak Scaling is inefficient!
- Strong Scaling is inefficient!
- Tradeoff between compute and, coordination(more supersteps) & data transfer.
- User compute time is **~50%** of total time.

Spark's overheads dominate.

Results- Split of User Compute Time- G50/P8



Level 0- Object Creation time dominates (~67% of total time).

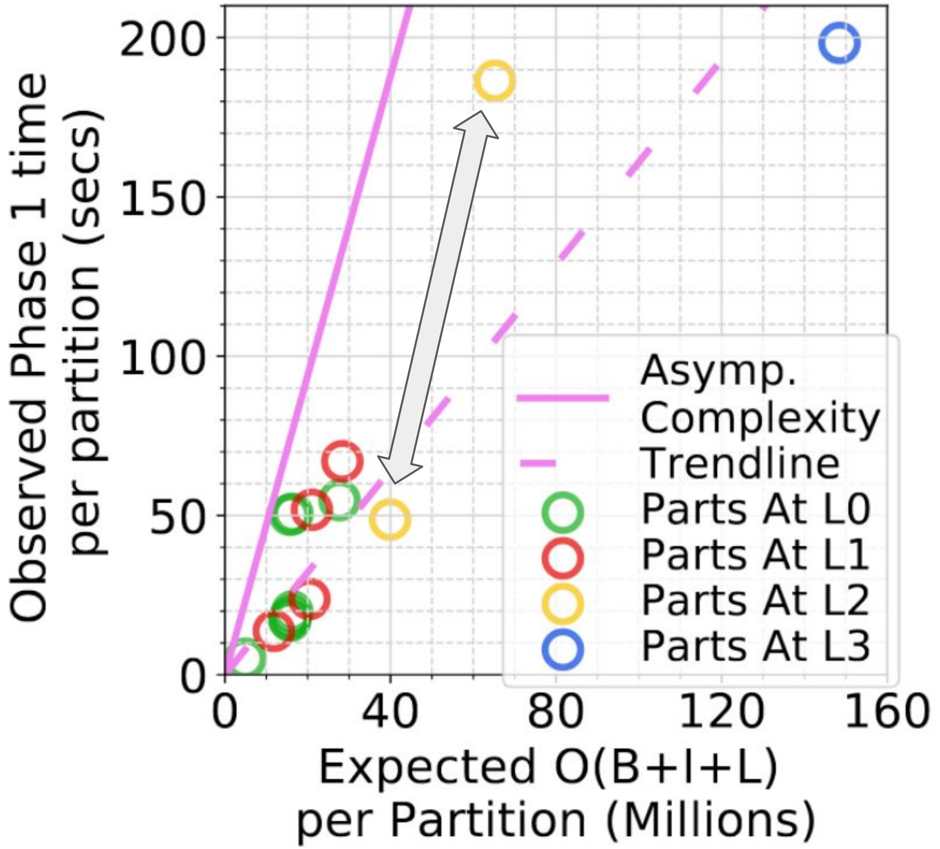
Level 1- (De)serialising + Object creation + Merging of partitions dominates.

Level 2 and **3**- Phase 1 dominates(~48% and ~51% resp).

Overall compute time increases with level.

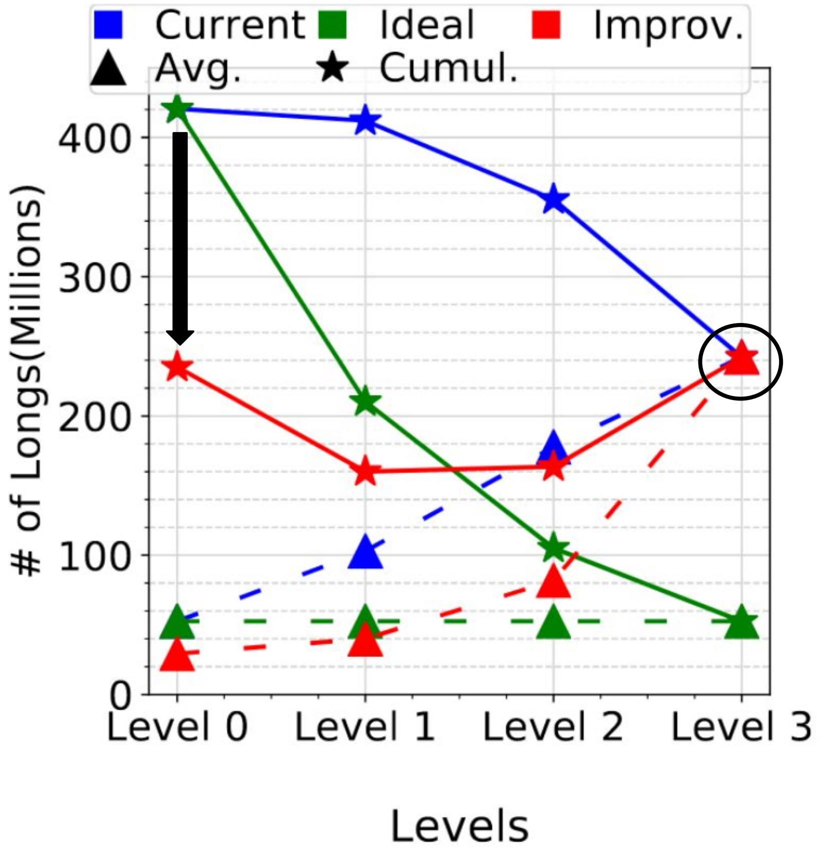
DREAM:Lab

Results- Expected and Observed Phase 1 Time- G50/P8



- Expected time matches observed time with few outliers.
- Outliers have skew in data.

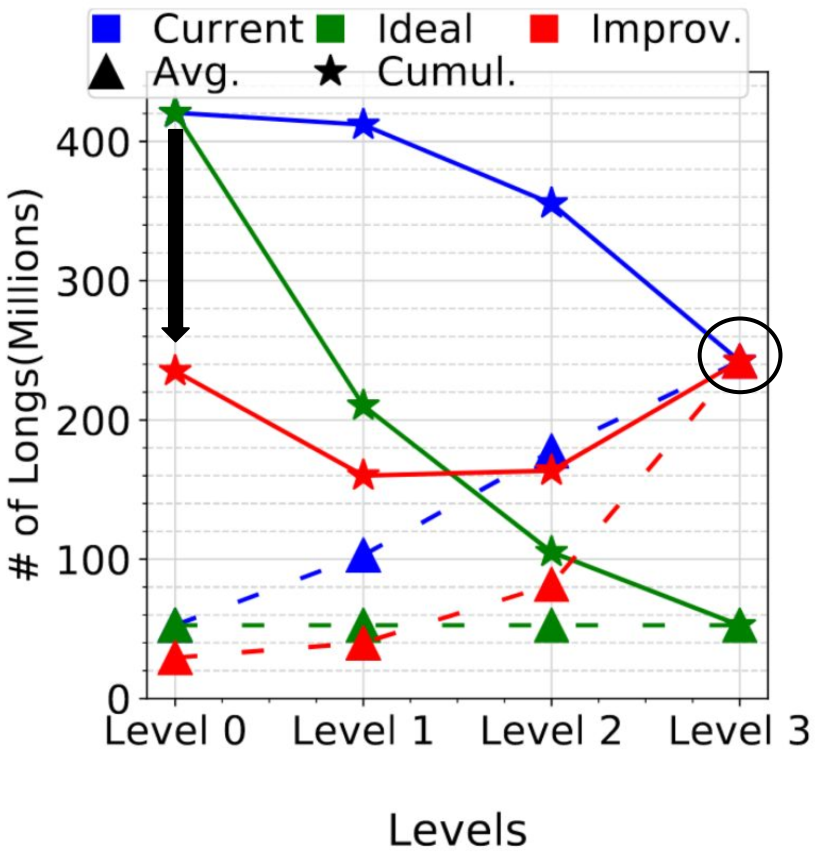
Results- Cumulative and Average Memory- G50/P8



Monotonic drop in cumulative state. Rate of drop is **INADEQUATE**.

Cumulative Drop % is higher with increasing level.

Results- Cumulative and Average Memory- G50/P8



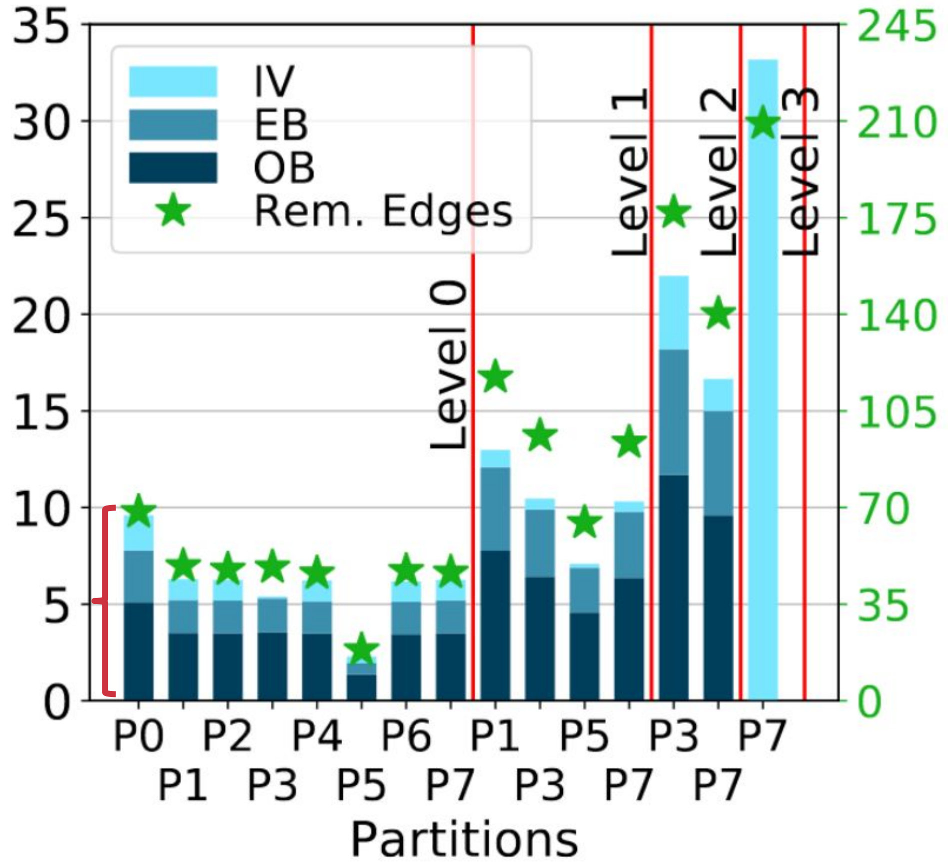
Two techniques which can improve memory bottleneck-

- Avoiding Duplication of Remote Edges
- Defer Transfer of Remote Edges

At **Level 0**, Initial Memory Maintained drops by ~43%.

At **Level 3**, Memory remains same as current implementation because all Edges are Local.

Results- Vertices and Edges per Partition- G50/P8



- Number of boundary vertices and Remote edges grow with merging.
- Remote edge count is ~7x vertex count.
- Drop in state(IV and LE) is low. IV is a small fraction of total vertices.

References

1. *Jaiswal, Siddharth D., and Yogesh Simmhan. "A Partition-centric Distributed Algorithm for Identifying Euler Circuits in Large Graphs." arXiv preprint arXiv:1903.06950 (2019).*
2. Euler, Leonhard. "Solutio problematis ad geometriam situs pertinentis." Commentarii academiae scientiarum Petropolitanae (1741): 128-140.
3. Atallah, Mikhail, and Uzi Vishkin. "Finding Euler tours in parallel." Journal of Computer and System Sciences 29.3 (1984): 330-337.
4. Makki, S. A. M. "A distributed algorithm for constructing an Eulerian tour." 1997 IEEE International Performance, Computing and Communications Conference. IEEE, 1997.
5. Awerbuch, Baruch, Amos Israeli, and Yossi Shiloach. "Finding Euler circuits in logarithmic parallel time." Proceedings of the sixteenth annual ACM symposium on Theory of computing. ACM, 1984.

Thank You!
Questions?

DREAM:Lab
dream-lab.in
Indian Institute of Science

Contact- siddharthj@iisc.ac.in

Checkout- http://dream-lab.cds.iisc.ac.in/
https://github.com/dream-lab/

Department of Computational and Data Sciences