Second Workshop on Software Challenges to Exascale Computing
SCEC 2018

Overcoming MPI Communication Overhead for Distributed Community Detection

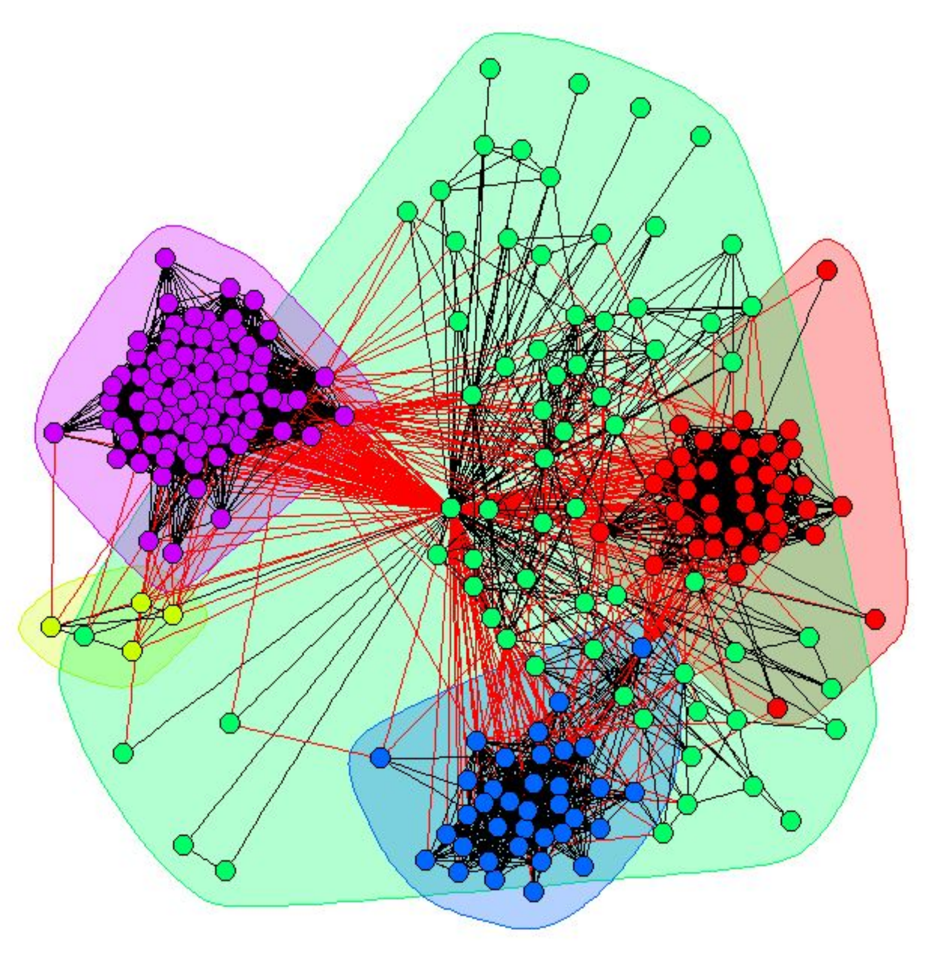
NAW SAFRIN SATTAR
SHAIKH ARIFUZZAMAN

Big Data and Scalable Computing Research Lab

THE UNIVERSITY of NEW ORLEANS
New Orleans, LA 70148 USA

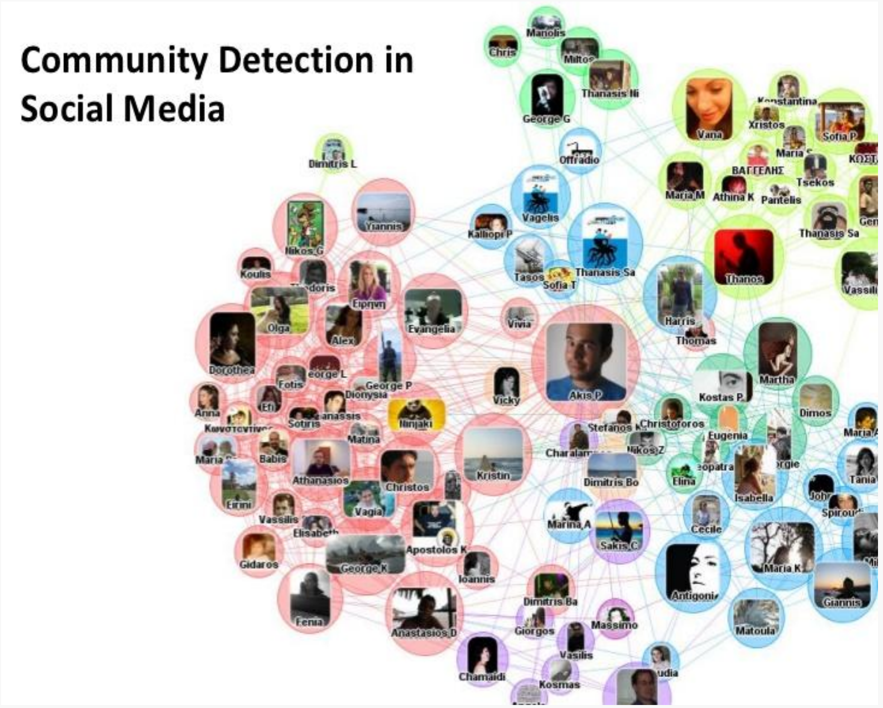
Introduction

- Louvain algorithm
  - A well-known and efficient method for detecting communities
- Community
  - a subset of nodes having more inside connections than outside



Motivation

- Community Detection Challenges
  - Large networks emerging from online social media
    - Facebook
    - Twitter
  - Other scientific disciplines
    - Sociology
    - Biology
    - Information & technology
- Load balancing
  - Minimize communication overhead
  - Reduce idle times of processors leading to increased speedup



Parallelization Challenges

Shared Memory

- Merits
  - Conventional multi-core processors
- Demerits
  - Scalability limited by moderate no. of available cores
  - Physical cores limited for the scalable chip size restriction
  - Shared global address space size limited for memory constraint

Distributed Memory

- Merits
  - utilize a large number of processing nodes
  - freedom of communication among processing nodes through passing messages
- Demerits
  - An efficient communication scheme required

Louvain Algorithm

THE UNIVERSITY *of* NEW ORLEANS

- Detects community based on modularity optimization
- Better than other community detection algorithms in terms of
    - Computation time and
    - Quality of the detected communities

❑ Modularity Calculation

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i c_j)$$

Here,

Q = Modularity

A_{ij} = Link weight between nodes i and j

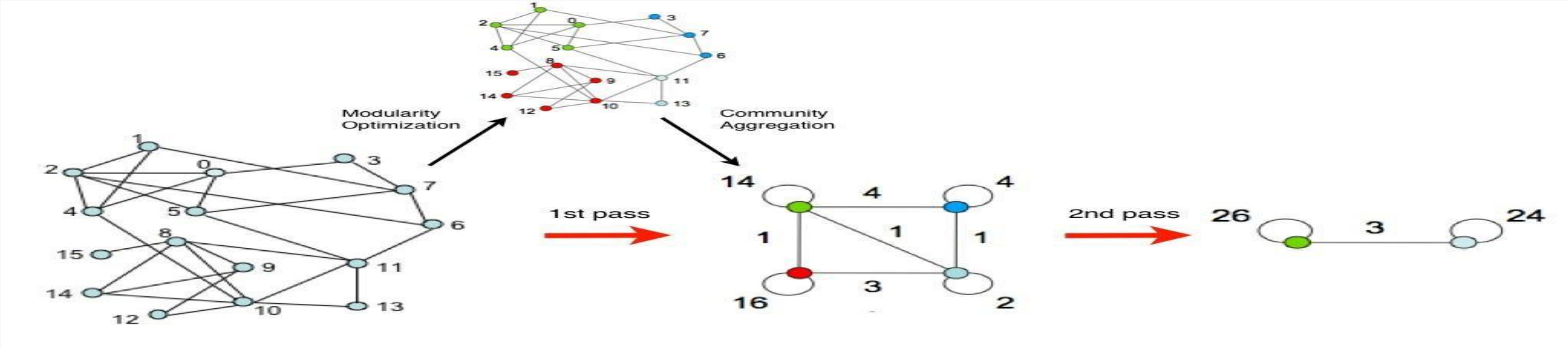
m = Total link weight in the network

k_i = Sum of the link weights attached to node i

c_i = Community to which node i is assigned

$\delta(c_i c_j)$ = Kronecker delta. Value is 1 when nodes i and j are assigned to the same community. Otherwise, the value is 0

Louvain Algorithm



- 2 Phases
  - Modularity Optimization- looking for "small" communities by local optimization of modularity
  - Community Aggregation- aggregating nodes of the same community a new network is built with the communities as nodes

Shared Memory Parallel Algorithm

- Parallelize computational task-wise
  - iterate over the full network
  - the neighbors of a node
- Work done by multiple threads
  - minimize the workload
  - do the computation faster

Distributed Memory Parallel Algorithm

Algorithm 1: Our Parallel Louvain using MPI

```
Data: Input Graph G(V,E)
Result: (Vertex, Community) Pair
while increase in modularity do
    G (V, E) is divided into p processes;
    Each graph_i.bin contains ⌈n/p⌉ vertices and
     corresponding edges in adjacency list format;
    for Each processor P_i (executing in parallel) do
        Gather_Neighbour_Info();
        Compute_Community();
        Exchange_Updated_Community();
        Resolve_Community _Duality();
        Exchange_Duality _Resolved _Community();
        Find_Unique_Communities();
        Compute_Modularity();
        Generate_NextLevel_Graph();
        if number_of_communities < i then
            i ← number_of_communities / 2;
        end
    end
end
```

Hybrid
Parallel Algorithm

- Both MPI and OpenMP together
- Flexibility to balance between both shared and distributed memory system

❑ Challenge

  ➢ Demerits of Distributed Memory Overweigh the performance

DPLAL- Distributed Parallel Louvain Algorithm with Load-balancing

- Similar approach as Distributed Memory Parallel Algorithm
- Load balancing of Input Graph using Graph-partitioner METIS
- Re-computation required for each function being calculated from Input Graph

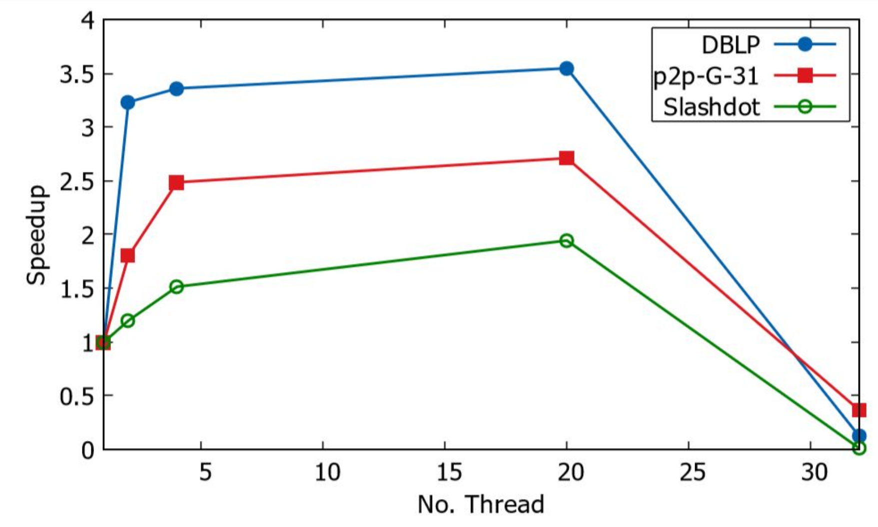
Experimental Setup

- Language
  - C++
- Libraries
  - Open Multi-Processing (OpenMP)
  - Message Passing Interface (MPI)
  - METIS
- Environment
  - Louisiana Optical Network Infrastructure (LONI) QB2 compute cluster
    - 1.5 Petaflop peak performance
    - 504 compute nodes
    - over 10,000 Intel Xeon processing cores of 2.8 GHz

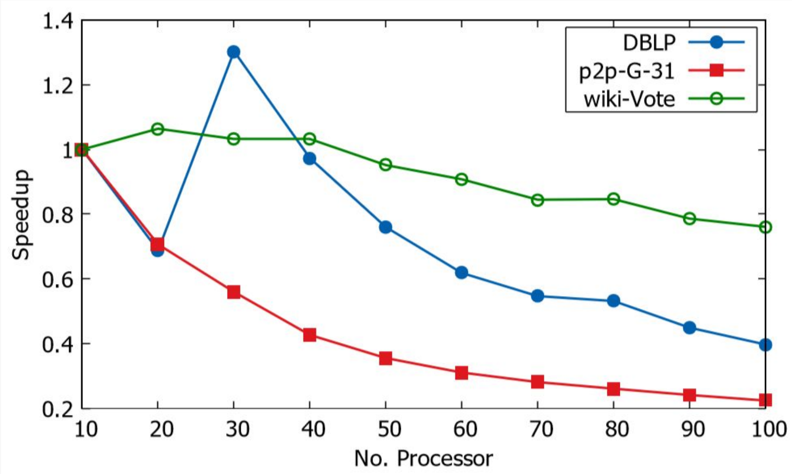
Dataset

Network	Vertices	Edges	Description
email-Eu-core	1,005	25,571	Email network from a large European research institution
ego-Facebook	4,039	88,234	Social circles ('friends lists') from Facebook
wiki-Vote	7,115	1,03,689	Wikipedia who-votes-on-whom network
p2p-Gnutella08, 09, 04, 25, 30, 31	6,301 - 62,586	20,777 - 1,47,892	A sequence of snapshots of the Gnutella peer-to-peer file sharing network for different dates of August 2002
soc-Slashdot0922	82,168	9,48,464	Slashdot social network from February 2009
com-DBLP	3,17,080	10,49,866	DBLP collaboration(co-authorship) network
roadNet-PA	1,088,092	1,541,898	Pennsylvania road network

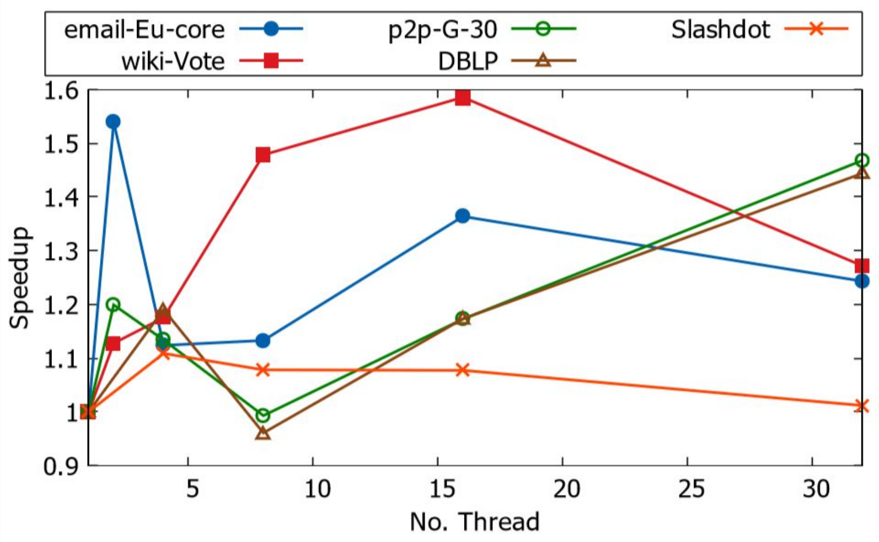
Speedup Factors of Parallel Louvain Algorithms



(a) Shared Memory Algorithm

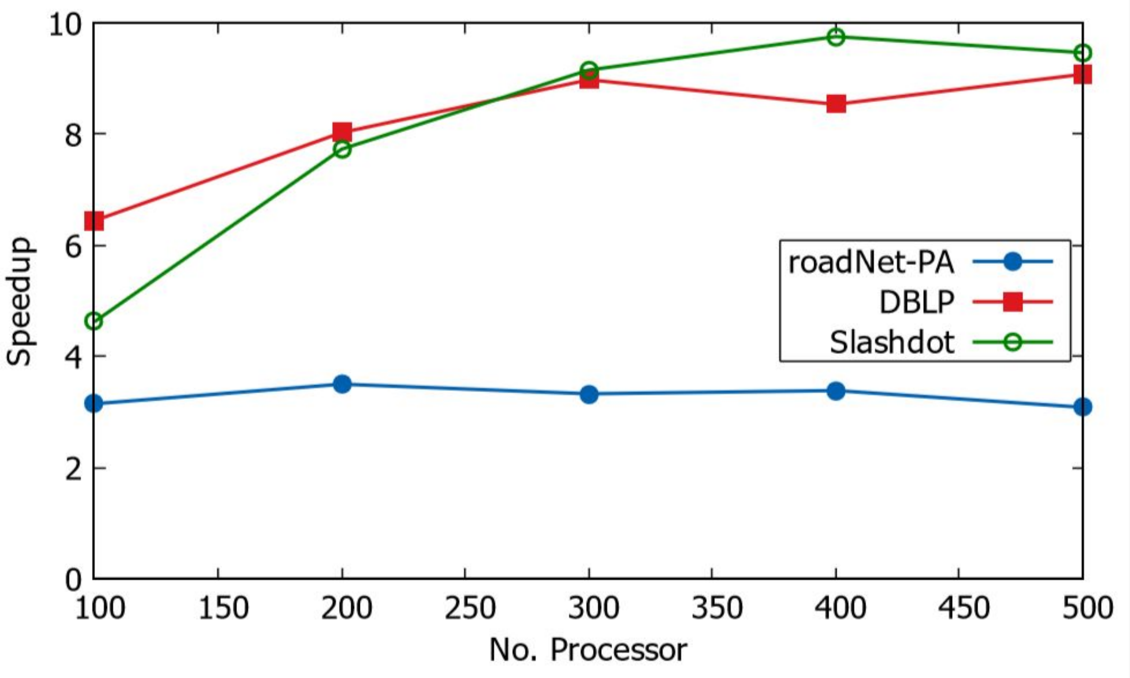


(b) Distributed Memory Algorithm

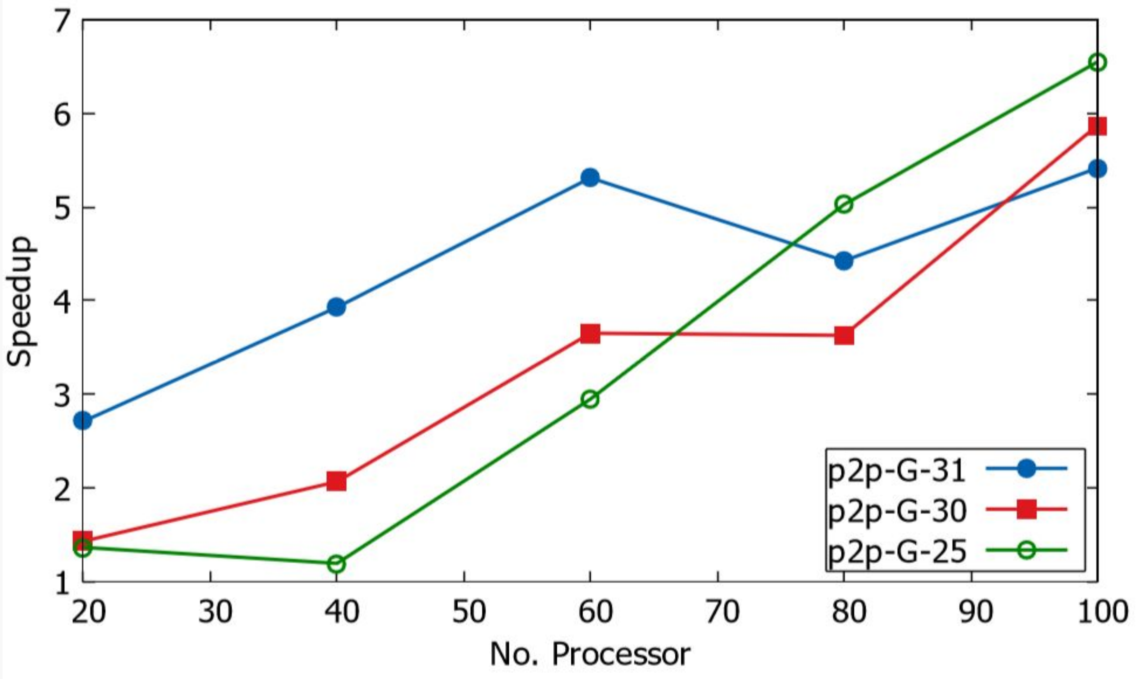


(c) Hybrid Algorithm

Speedup Factor of DPLAL-Distributed Parallel Louvain Algorithm with Load Balancing

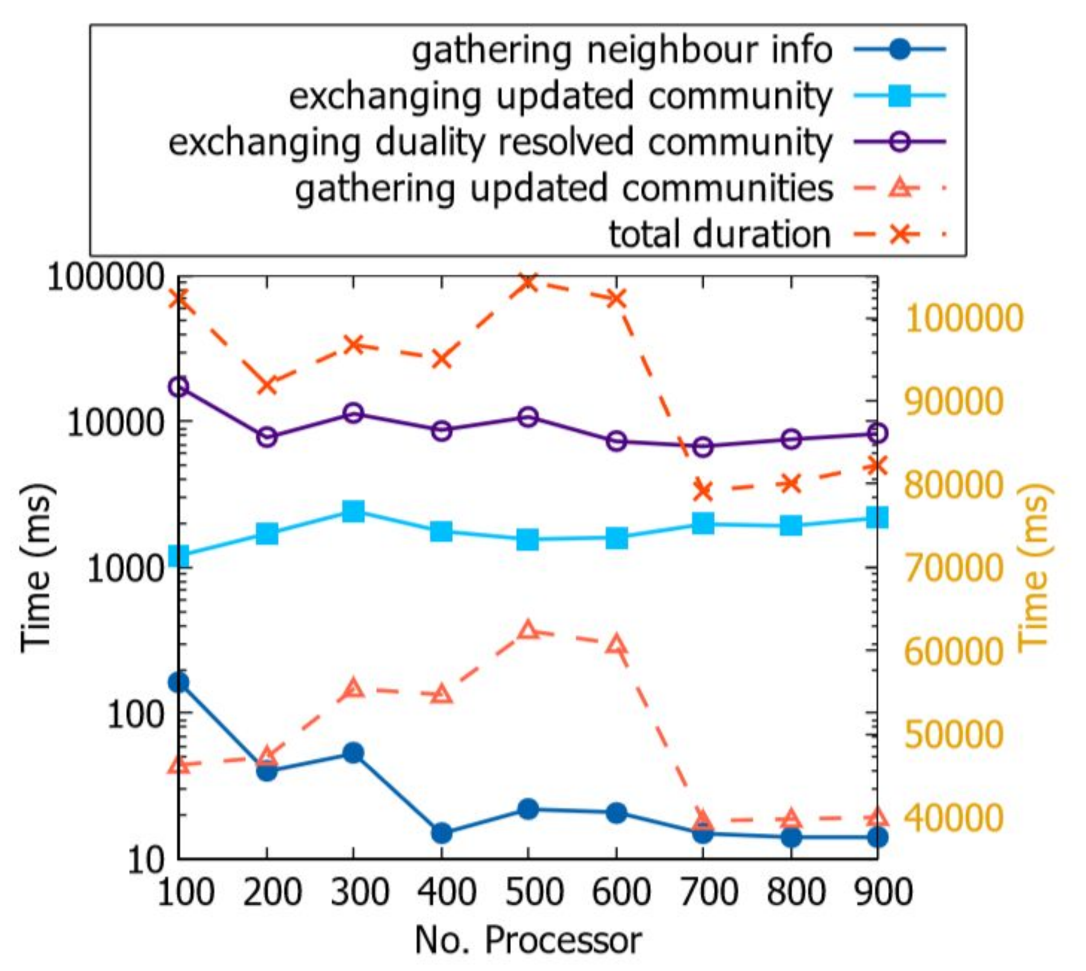


(a) Speedup results for large graphs

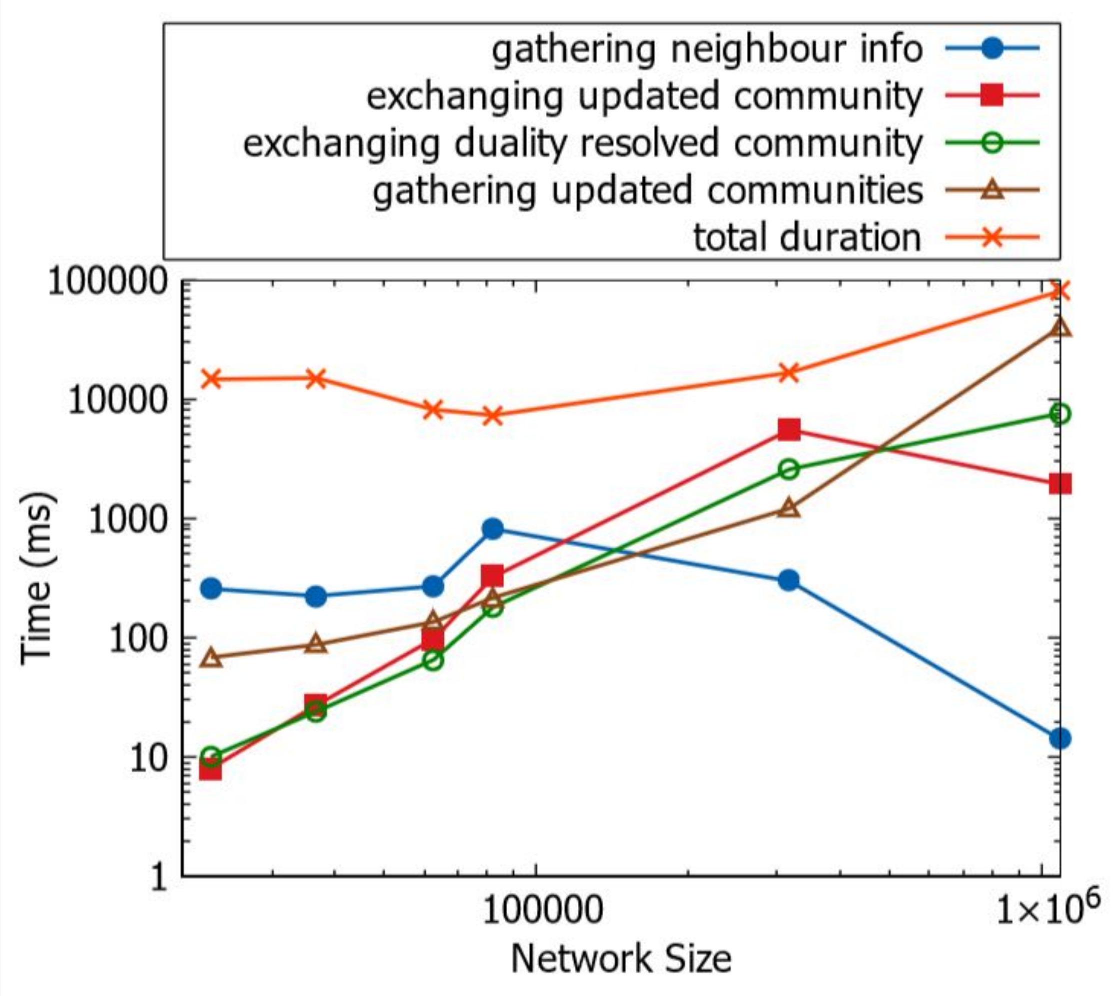


(b) Speedup results for relatively small graphs

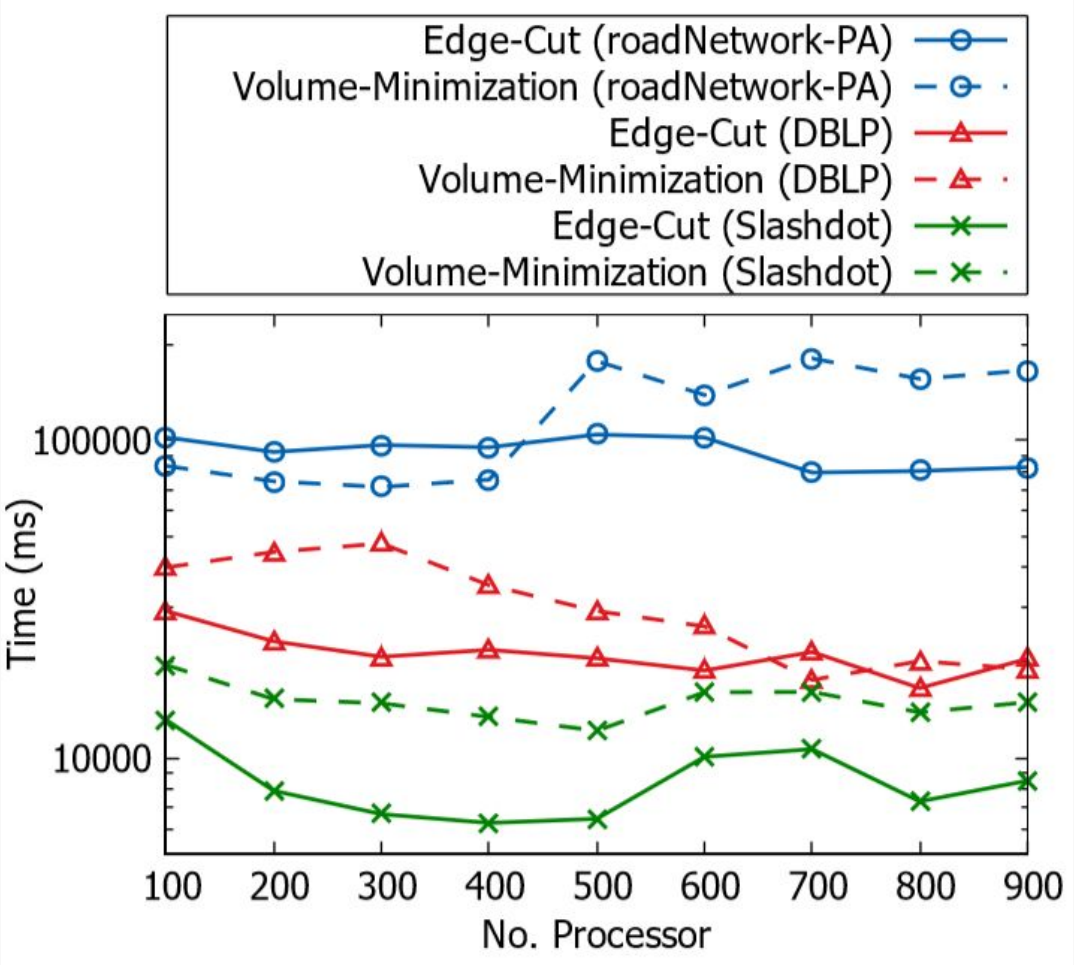
Runtime Analysis of RoadNet-PA Graph with DPLAL algorithm



Runtime of DPLAL Algorithm with Increasing Network Sizes



Comparison of METIS Partitioning Approaches



Performance Analysis

Sequential Algorithm

Algorithm	Network			
	com-DBLP		wiki-Vote	
	Comm. No.	Dev. (%)	Comm. No.	Dev. (%)
Sequential	109,104	-	1,213	-
Shared	109,102	.0006	1,213	0
Distributed	109,441	0.106	1,216	0.042
Hybrid	104,668	1.39	1,163	0.71
DPLAL	109,063	0.0129	1,210	0.042

Another MPI based Parallel Algorithm

	DPLAL	**Charith *et.al***
Network (node) size – Speedup	317,080 – 12, almost double	500,000 - 6
Speedup for the largest network	4 (1M nodes), same	4 (8M nodes)
Scalability for Processors	Upto 1000	Upto 16

Conclusion

- Our parallel algorithms for Louvain method demonstrating good speedup on several types of real-world graphs
- Implementation of Hybrid Parallel Algorithm to tune between shared and distributed memory depending on available resources
- Identification of the problems for the parallel implementations
- An optimized implementation DPLAL
  - DBLP network 12-fold speedup.
  - Our largest network, roadNetwork-PA 4-fold speedup for same number of processors

Future Works

- Improve the scalability of our algorithm for large scale graphs with billions of vertices and edges
  - other load balancing schemes to find an efficient load balancing
- Eliminate the effect of small communities hindering the detection of meaningful medium sized communities
- Investigate the effect of node ordering on the performance
  - degree based ordering
  - kcores
  - clustering coefficients

Contact: nsattar@uno.edu