Scalable Algorithm Design and Performance Analysis for Graph Motifs Discovery

Md Abdul M Faysal, Shaikh Arifuzzaman (Ph.D. Advisor), **University of Nevada, Las Vegas**

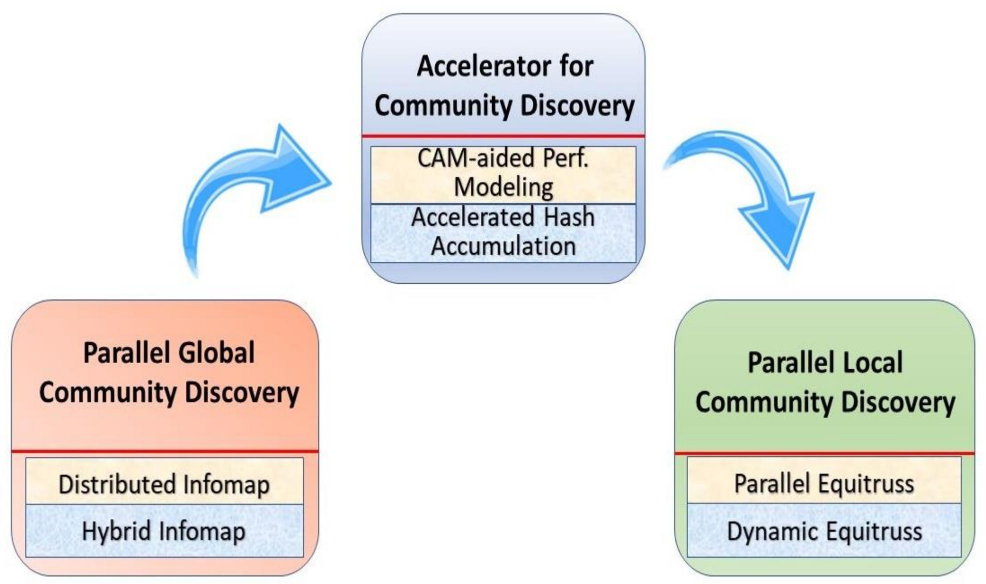
Motivation

Community discovery, a prominent graph application with usages in social, professional network analysis, health, bioinformatics and metagenomics, etc.

Limitations: State-of-the-art techniques suffer from inadequate scalability, poor performance, and methodological inaccuracy

Our Approach: Software-hardware co-design for high-performance, scalable, highly accurate parallel community discovery

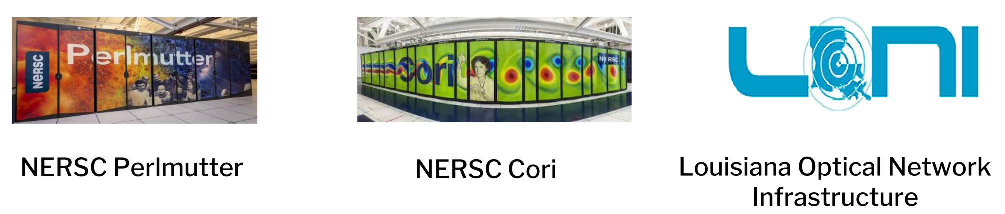
Research Scope



i) **Hybrid-memory** parallel global community discovery using information-theoretic approach
ii) **Accelerator** design for fast community discovery, and
iii) **k-truss**-oriented parallel algorithm design for local community discovery

Research Tools

- **Frameworks:** MPI, OpenMP, CUDA-C
- **Platforms:** NERSC Perlmutter, NERSC Cori, LONI



- **Language:** C++
- **Distributed Architecture:** CPU, GPU
- **Profiling Tools:** Vtune, OSU benchmark
- **Dataset:** SNAP Networks
- **Micro-architecture Simulator:** ZSim, Pin

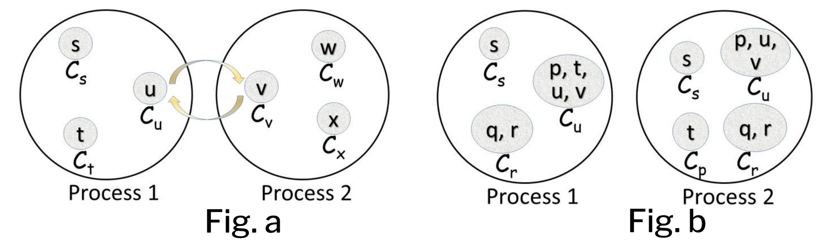
Parallel Global Community Discovery [1, 2]

Distributed Infomap

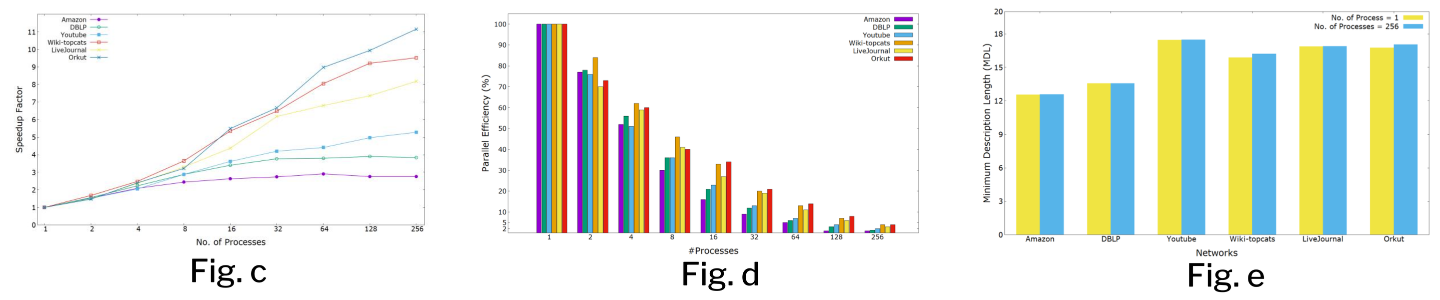
Distributed Challenges:
- Vertex bouncing problem (*Fig. a*)
- Inconsistent update ordering (*Fig. b*)
- Inactive vertices
- Distributed workload imbalance

Solution Heuristics:
- MPI for distributed computing
- Numeric ordering for vertex bouncing
- Priority-based assignment for inconsistency
- Maintaining and disregarding inactive vertices
- Metis partitioner for workload balancing

Key Takeaways:
- Better speedup than parallel state-of-the-arts
- Solution accuracy not compromised
- Scope for performance improvement



Figures: a) Shows the effect of vertex bouncing due to group of vertices having strong affinity, b) Incorrect communities from incorrect synchronization ordering



Figures c-e: The plots demonstrate the speedup (strong scalability) of the distributed Infomap (*fig. c*), the parallel efficiency of the algorithm (*fig. d*), and the accuracy of the converged solution in terms of Minimum Description Length (MDL) (*fig. e*)

Hybrid Infomap

Observations:
- Kernel breakdown shows (*Fig. f*) *CommunityOptimization* kernel needs to be further optimized
- Adopt OpenMP parallelism
- Use cache-friendly data-structure

Kernel Runtime Analysis:
- Cache optimized *unordered_map* data structure demonstrate better performance over regular *map*
- The biggest performance boost (*Fig. g*) obtained from multithreaded kernel

Optimized Hybrid Infomap Design:
- Delivers 25X speedup, accuracy maintained
- Scales to 1280 processing cores (*Fig. i*)

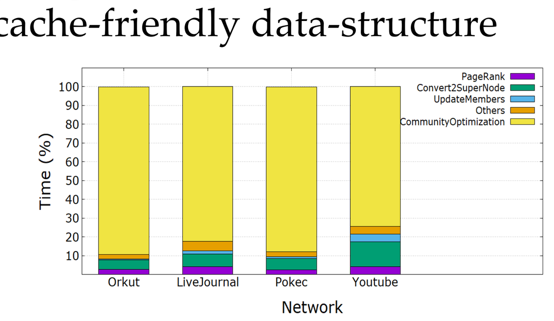


Figure f: Compute kernel breakdown of the distributed Infomap

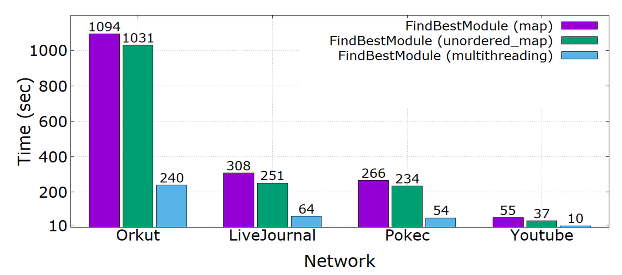


Figure g: Runtime effect of optimizations on dominant kernel

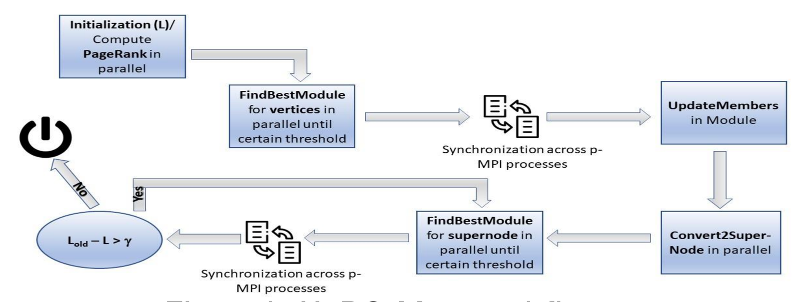


Figure h: HyPC-Map workflow

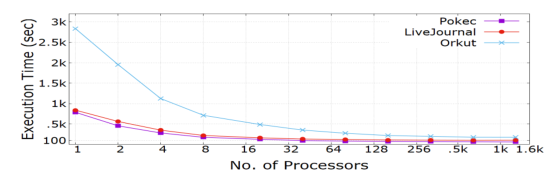
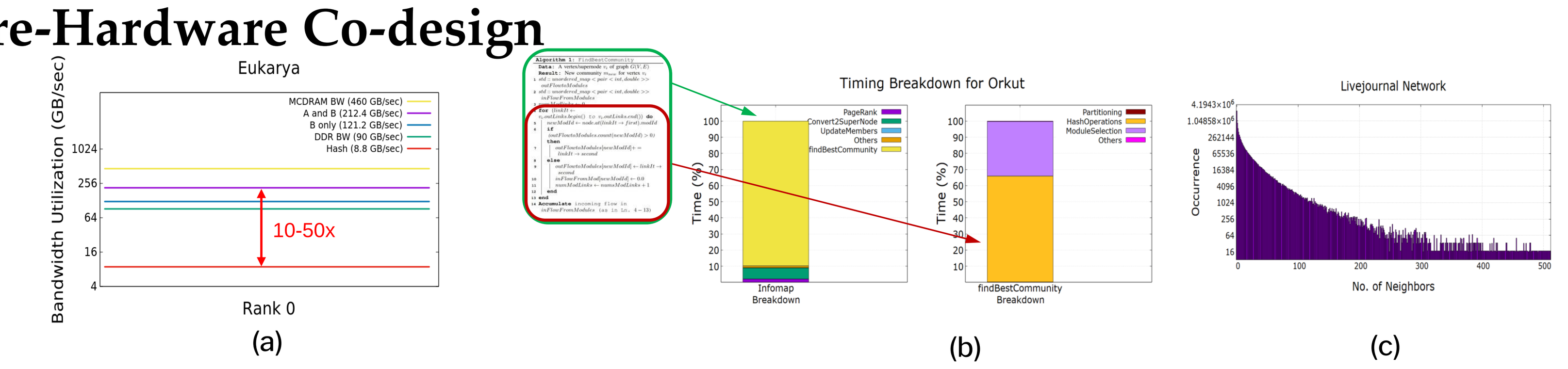


Figure i: Strong scalability (runtime)

Accelerator-aided Community Discovery [3]

Performance Modeling for Software-Hardware Co-design

Case Studies:
- Two community detection applications: ***HipMCL***, and ***HyPC-Map***
- Performance profiling and Modeling shows large gap between achievable and operational performance
- Performance limiting factor in both applications: *Software hash accumulation*



Figures **a-c**: (a) Performance gap between software hash (red) and streaming bandwidth (purple and blue) in *HipMCL* illustrates the potential impact (10-50X) for HW CAM support, (b) Timing breakdown shows impact of latency bound software hash in HyPC-Map, (c) Real-world networks exhibiting power-law degree distribution.

Fast HyPC-Map with Accelerator for Sparse Hash Accumulation (ASA)

3 main computation phases:
- Hardware accumulation
- Hardware CAM gather
- Software sort and merge

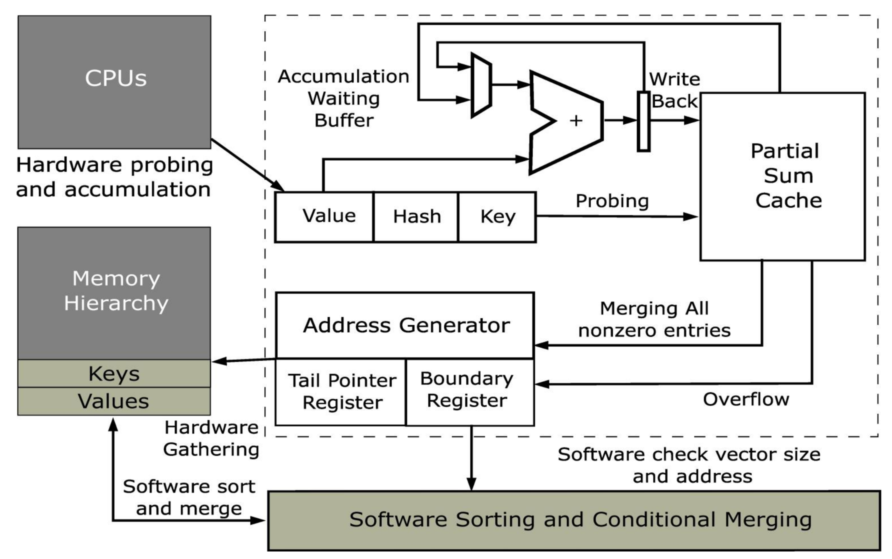
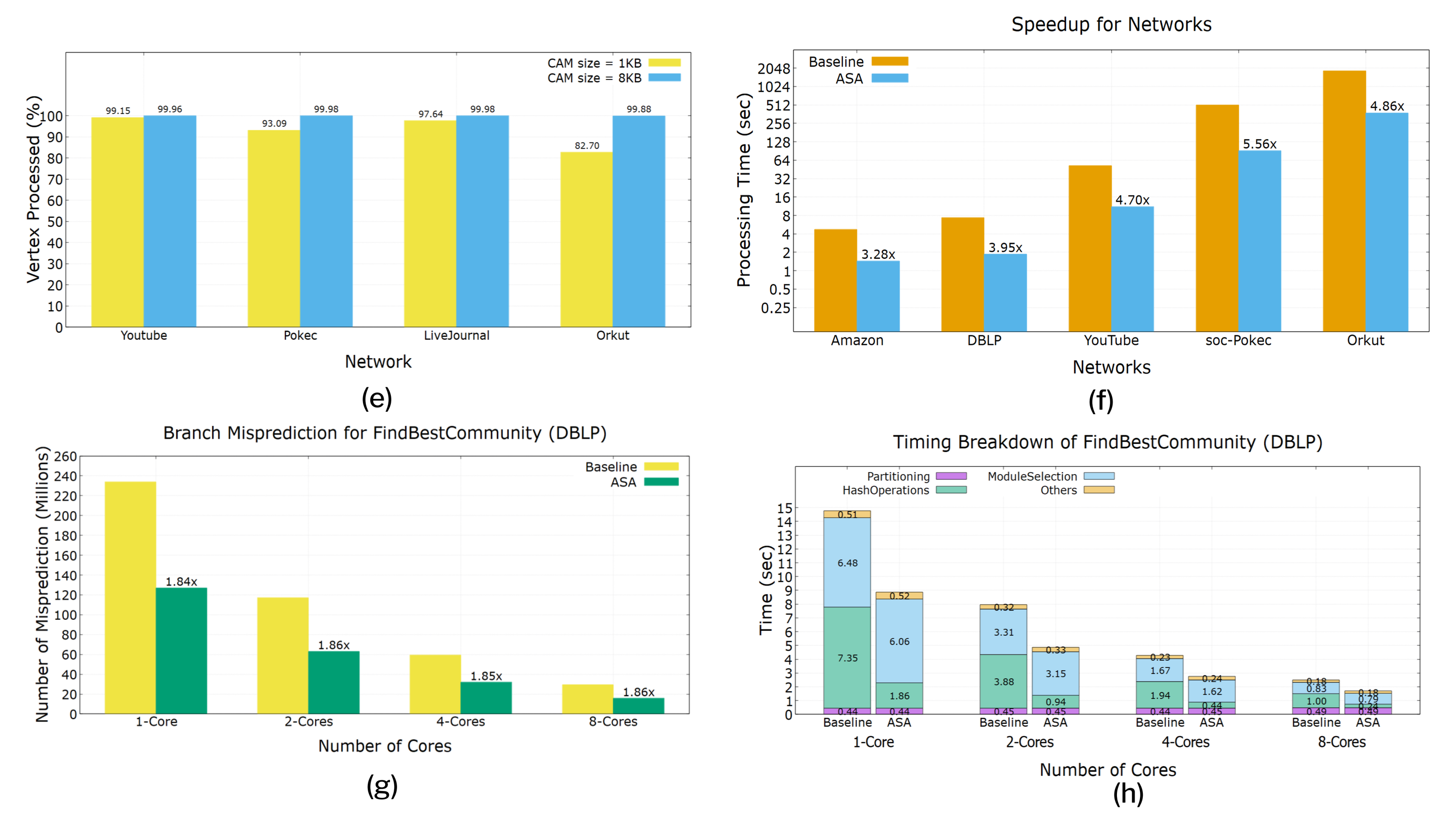


Figure **d**: Block diagram of the generalized ASA microarchitecture. Details of different modules of the architecture and their functionalities are described in Chao et al. [4].



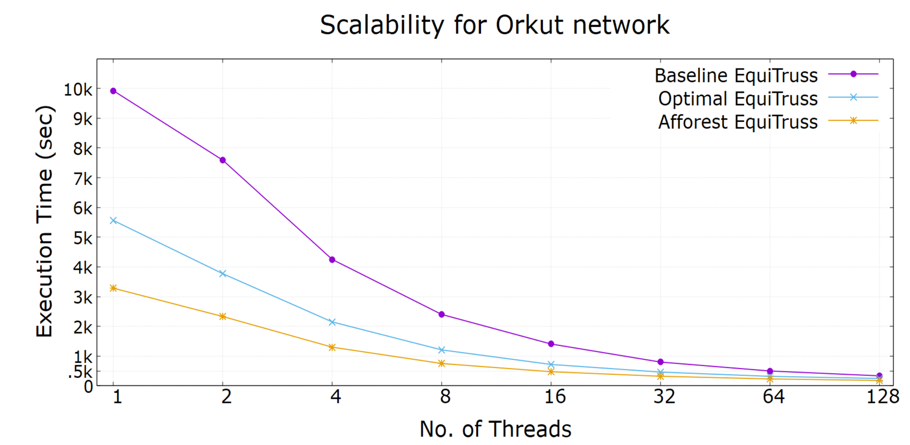
Figures **e-h**: (e) Limited capacity core-local Content Addressable Memory (CAM) can cover 99% of the vertices of power-law graphs, (f) ASA demonstrating up to 5.56x speedup over baseline, g) ASA exhibiting 47% reduction in branch misprediction, and h) strong scalability across varying number of cores

Parallel Local Community Search

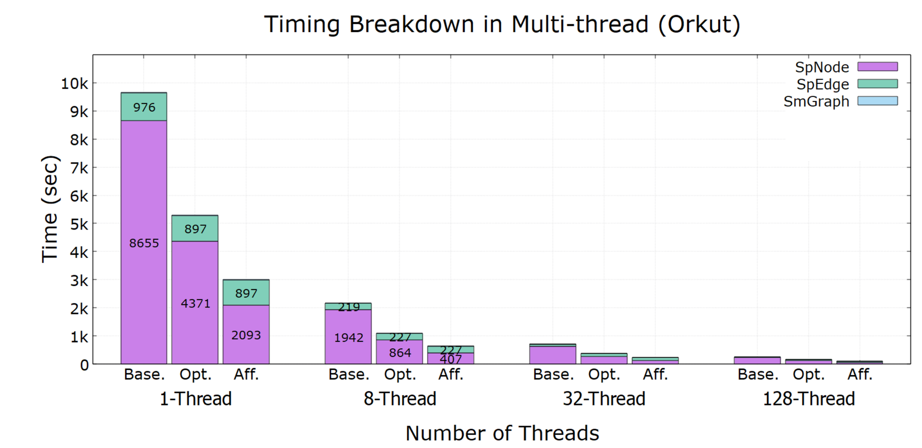
Specification: Real-world applications often demand community of a query entity

Limitations: Lack of scalable algorithms, methodological inefficiency

Our Approach: A k-truss-based scalable local community search, **parallel *EquiTruss***, using higher order graph motif (k-triangle connectivity) as building block



Strong scalability for our parallel ***EquiTruss*** in 3 different optimization phases for Orkut social network.



Timing breakdown of major compute kernels of parallel **EquiTruss** and reduction of execution time for different number of threads

ACKNOWLEDGMENTS

- **Funding Sources:** Dept. of Energy (DOE)/ Berkeley Lab, National Science Foundation (NSF), Nevada State of Higher Education (NSHE)/ UNLV, BoR/LA, and UNO
- **Collaborators:** John Shalf, Cy Chan, Maximilian Bremmer, and Doru Thom Popovici, Computer Architecture group (CAG), Berkeley Lab (LBNL)

SCHOLARLY ARTICLES

1) HyPC-Map: A Hybrid Parallel Community Detection Algorithm Using Information-Theoretic Approach, MAM Faysal, S Arifuzzaman, C Chan, M Bremer, D Popovici, J Shalf, 2021 IEEE High Performance Extreme Computing Conference (HPEC), 1-8
2) Distributed community detection in large networks using an information-theoretic approach, MAM Faysal, S Arifuzzaman, 2019 IEEE International Conference on Big Data (Big Data), 4773-4782
3) Md Abdul Motaleb Faysal, Maximilian Bremer, Shaikh Arifuzzaman, Doru Popovici, John Shalf and Cy Chan, "Fast Community Detection in Graphs with Infomap Method using Accelerated Sparse Accumulation," [Just Accepted] AsHES 2023 IPDPSW
4) C. Zhang, M. Bremer, C. Chan, J. Shalf, and X. Guo, "Asa:Accelerating sparse accumulation in column-wise spgemm," ACM Trans. Archit. Code Optim., May 2022, just Accepted. [Online]. Available: https://doi.org/10.1145/3543068