Accelerating Actor-based Distributed Triangle Counting

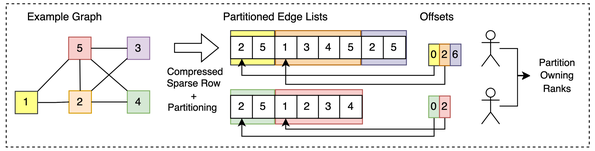
Aniruddha Mysore*, Kaushik Ravichandran*, Youssef Elmougy, Akihiro Hayashi, Vivek Sarkar

Problem

- Counting triangles in large graphs
- Build atop actor-based distributed computing runtime [1]
- Set-operation based implementation
- Optimized message passing/communication
- Important optimization candidate because of wide-usage:
  - Improves network utilization
  - Exploits data parallelism

Background

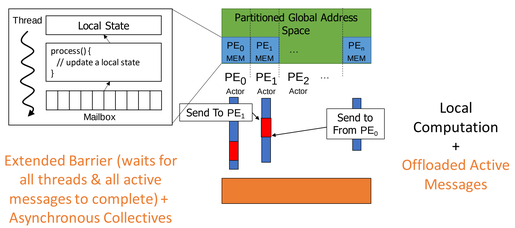
Graph Primitives



- Triangle Node set: {{1, 2, 5}, {2, 4, 5}, {2, 3, 5}}
- Triangle Edge set: {{(1, 2), (1, 5), (5, 0)}.....}

Distributed Actors for PGAS Applications

- Actor (PE) = thread/rank owning a slice of global address space
- Asynchronous active messages (Actor mailboxes) - PE comms.
- Blocking extended barrier for active message completion
- Single Program Multiple Data (SPMD) model for scalable programming



Evaluation

Computation Hardware

Cluster	PACE (Georgia Tech)	Perlmutter (NESRC)
CPU Architecture	Intel Xeon Gold 6226	AMD EPYC 7763
PEs per Node	24	128
Vector Instruction Set	AVX512	AVX2

Dataset: Graph500 R-MAT generator

Dataset	Small.mtx	Medium.mtx	Med-Large.mtx
R-MAT Scale (N)	N = 16	N = 20	N = 22
R-MAT Params	A=57, B=C=19, D=5	A=57, B=C=19, D=5	A=55, B=C=20, D=5
Vertices	65,536 (2^{16})	1,048,576 (2^{20})	4,194,304 (2^{22})
Edges	909,917 (~$2^{19.7}$)	15,701,646 (~$2^{23.9}$)	67,108,864 (2^{26})

- The graphs Small.mtx and Medium.mtx have a higher degree of imbalance than Med-Large.mtx

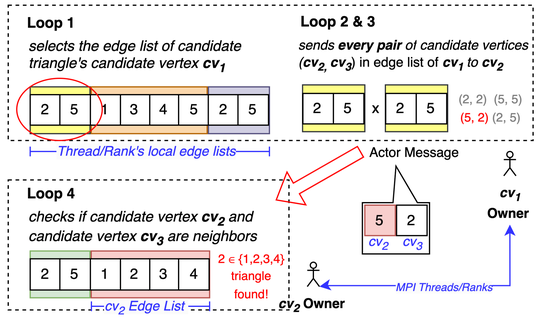
*equal contribution

Approach

Baseline approach [3]

```
for n1 in local_edge_list:
    for n2 in neighbors(n1):
        for n3 in neighbors(n1):
            for n4 in neighbors(n2):
                if (n3 == n4):
                    count++;
```

Actor message sent to host(n2) with parameters n2, n3



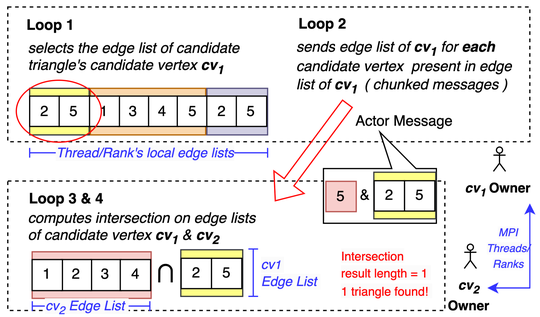
- **Baseline**: Simple, Relies on efficient fine-grained asynchronous messaging. **Other options:** Linear-algebraic - sparse matrix operations are inherently expensive

Our Approach: Distributed Set-Intersection on Actors

```
for n1 in local_edge_list:
    for n2 in neighbors(n1):
        for n3 in neighbors(n1):
            for n4 in neighbors(n2):
                if (n3 == n4):
                    count++;
```

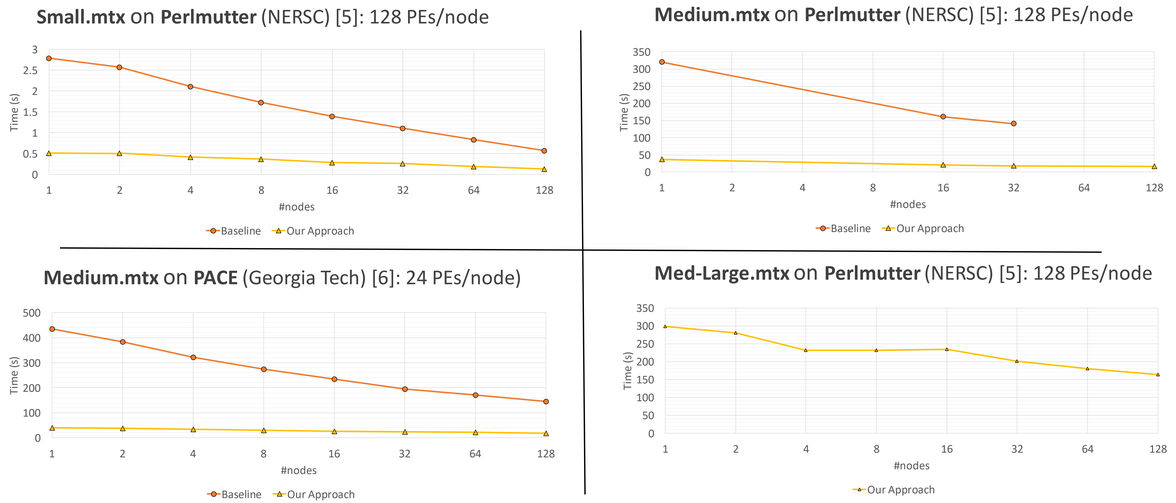
```
count += intersection (neighbors (n1), neighbors(n2))
```

Actor message sent to host(n2) with parameters n2, neighbors(n1)

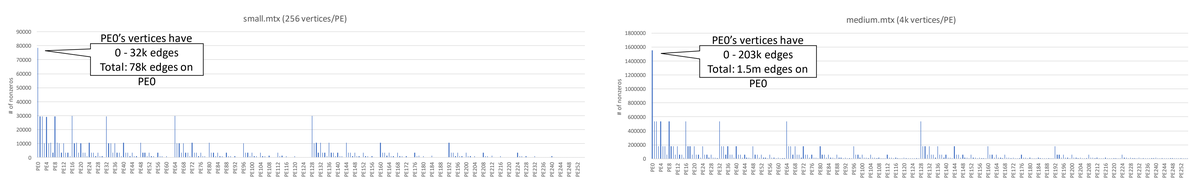


- General: duplicate counting + re-scaling is a performance overhead
- **Intersection**: Fixed tunable message size w. chunking, Vectorizable

Strong Scaling Results

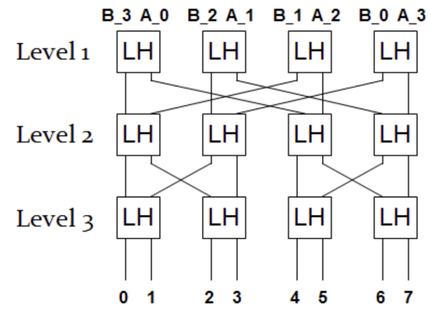


Load Imbalance



Vectorization

Vectorized Set-Intersection – Bitonic Merge



- Set operation implementations are vectorization friendly [2, 4]
- Can be achieved using SIMD units which:
  - are extremely efficient
  - take advantage of data parallelism
- Since graphs are sparse and irregular, improvement due to vectorization is limited (currently up to 1.2x)

Conclusions & Future Work

- Actor-based set-intersection version of triangle counting improves absolute performance over existing baseline version, but the scalability of the current version is poorer
- Early results from vectorization shows 1.2x performance improvement, but load-imbalance of RMAT-generated graph is a bottleneck.
- **Current Focus**: edge redistribution to mitigate imbalanced graphs

Acknowledgements

This research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the Advanced Graphical Intelligence Logical Computing Environment (AGILE) research program, under Army Research Office (ARO) contract number W911NF22C0083. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. We are grateful to Jiawei Yang for their contribution to this work.

References

[1] Sri Raj Paul, Akihiro Hayashi, Kun Chen and Vivek Sarkar (2022), **A Productive and Scalable Actor-Based Programming System for PGAS Applications.** ICCS

[2] Zhang, J., Lu, Y., Spampinato, D.G. and Franchetti, F., (2020). **Fesia: A fast and simd-efficient set intersection approach on modern cpus**. IEEE ICDE

[3] Sri Raj Paul, Akihiro Hayashi, Kun Chen, Youssef Elmougy, Vivek Sarkar (2023), **A Fine-grained Asynchronous Bulk Synchronous parallelism model for PGAS applications**.. Journal of Computational Science, April 2023."

[4] Ravichandran, K., Subramaniasivam, A., Aishwarya, P.S. and Kumar, N.S., 2023. **Fast exact triangle counting in large graphs using SIMD acceleration**. In Advances in Computers (Vol. 128, pp. 233-250). Elsevier.

[5] Yang, C. and Deslippe, J., 2020. **Accelerate Science on Perlmutter with NERSC.** Bulletin of the American Physical Society, 65.

[6] PACE, D., 2017. **Partnership for an advanced computing environment (PACE).**