Parallel Graph Algorithms by Blocks

Abdurrahman Yaşar and Ümit V. Çatalyürek

Collaborators: Sivasankaran Rajamanickam and Jon Berry (Sandia National Laboratories)

Georgia Tech College of Computing — Computational Science and Engineering

TDAlab — tda.gatech.edu

Introduction

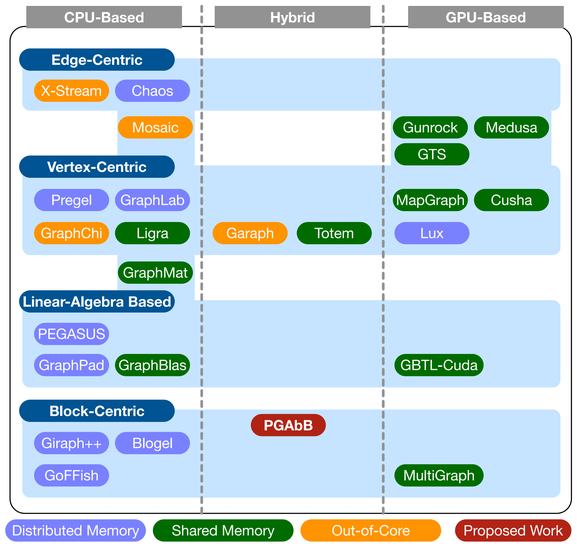
Graphs are very useful data-structures. They can represent different applications, such as social network, biological networks, and scientific simulations.

- Graphs are growing to billions of vertices and edges
- Fast, efficient analysis is crucial
- Many graph processing frameworks have been proposed

Goal: Developing an algorithmic framework (PGAbB) in which people can implement architecture agnostic graph algorithms that perform well on different architectures. Here we show our approach for the triangle counting problem as a use case.

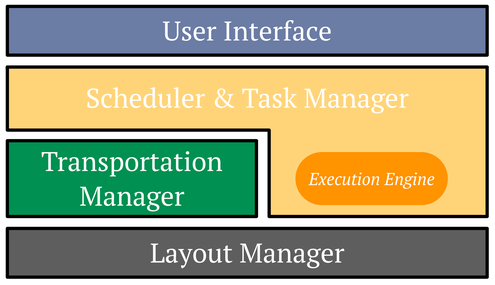
Graph Processing Systems

Programming abstractions of current systems can be broadly divided into five categories: vertex-centric; edge-centric; block-centric; linear-algebra based and domain-specific-language (DSL) based.



Parallel Graph Algorithm by Blocks (PGAbB)

PGAbB targets multi-core shared memory machines (host) with GPU accelerators (devices) and is composed by four components; Layout Manager, Scheduler, Transportation Manager Scheduler&Task Manager and User Interface.



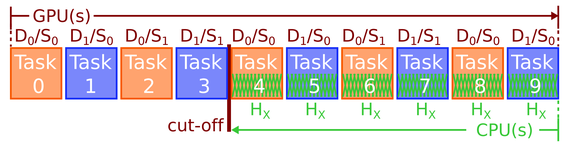
- **Layout Manager:** (1) Row-major, and (2) Hilbert order.
  - **Partitioner:** Symmetric Rectilinear Partitioning.
  - **Blocks:** (1) CSR and (2) COO style storage.
- **Transporter:** Communication overlapping.
- **Scheduler&Task Manager:** (1) locality based, and (2) workload estimation based.
- **User Interface:** Block-centric API and tools.

Use Case: Triangle Counting

Given a graph $G = \{V, E\}$, the triangle counting problem is to find the number (T) of all sets of three vertices, $u, v, w \in V$, such that:

$$T = \mid \{u, v, w \mid (u, v), (v, w), (w, u) \in E\} \mid .$$

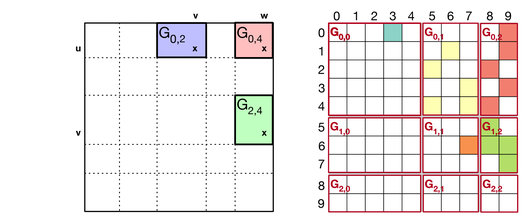
Heterogeneous Execution



To utilize the massive parallelism on GPUs:

- Tasks are ordered based on their estimations.
- GPUs: starts with the heaviest tasks, and CPUs start with lighter ones
- CUDA streams: simultaneous task execution and communication.

A Block-Based Triangle Counting Algorithm (bbTC)



Block CSR Layout

A Task: counting number of triangles in three subgraphs; $G_{i,j}$, $G_{j,k}$ and $G_{i,k}$ where $i \leq j \leq k$.

Algorithm 1: bbTC($G_{i,j}$, $G_{j,k}$, $G_{i,k}$)

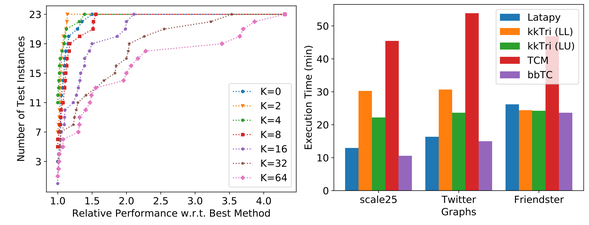
$\tau \leftarrow 0$ ▷ Initialize number of triangles to 0
for ***each*** $u \in V_S(G_{i,j})$ **do**
  for ***each*** $v \in N(G_{i,j}, u)$ **do**
    $\tau \leftarrow \tau +$ INTERSECT$(N(G_{i,k}, u), N(G_{j,k}, v))$
return τ

Triangle Counting Algorithms

	TCM	kkTri	TriCore	bbTC
Algorithmic properties				
List Intersect.	✓	✗	✗	✓
Map Intersect.	✗	✓	✗	✓
Search Intersect.	✗	✗	✓	✗
Multi-Core	✓	✓	✗	✓
Multi-GPU	✗	✗	✓	✓
Complexity	$O(m^{\frac{3}{2}})$	$O(m^{\frac{3}{2}})$	$O(m^{\frac{3}{2}} \log \sqrt{m})$	$O(\frac{\lambda p}{c} m^{\frac{3}{2}})$
Parallelization strategy				
One-dimension	✓	✓	✓	✗
Two-dimension	✗	✗	✓	✓
Used runtimes for implementation				
Pthread based	✓	✗	✗	✗
OpenMP	✓	✓	✗	✓
Cilk	✓	✓	✗	✓
TBB	✗	✗	✗	✓
CUDA	✗	✗	✓	✓

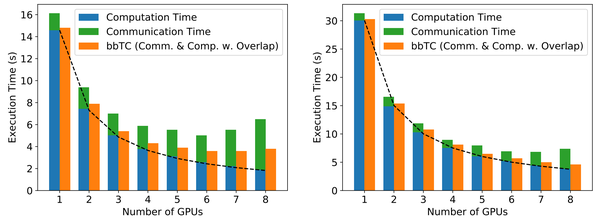
Experiments

bbTC has a better memory utilization and outperforms the best sequential algorithm.

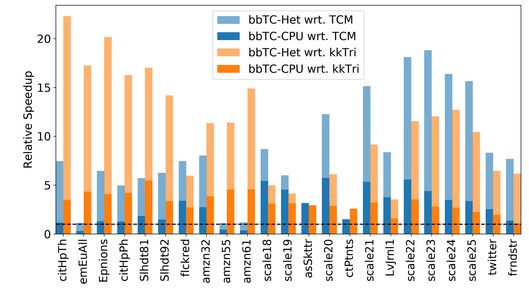


Perf. Profile of Latapy's Alg. | Sequential Execution Times

Overlap communication with computation with 10% overhead (5 GPUs).



bbTC outperforms TCM in 23 of 23 cases and kkTri in 20 of 23 cases.



bbTC outperforms kkTri and TCM 1.6× and 1.8 times, and processes the WDC-2014 graph in 116 seconds.

	Haswell	DGX	Newell
kkTri	423	Out of memory	Out of memory
TCM	476	Out of memory	Out of memory
TriCore	N/A	-	-
bbTC	265	116	1120

Table: Execution times (seconds) on a WDC graph.

References

[1] Y. Hu, H. Liu, and H. H. Huang. Tricore: Parallel triangle counting on gpus.

[2] M. Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs.

[3] J. Shun and K. Tangwongsan. Multicore triangle computations without tuning.

[4] M. M. Wolf, M. Deveci, J. W. Berry, S. D. Hammond, and S. Rajamanickam. Fast linear algebra-based triangle counting with kokkoskernels.

*This work was partially supported by the NSF grant CCF- 1919021.