HyGraph: High-Performance Graph Processing on Hybrid CPU-GPU Platforms using Dynamic Load-balancing

Stijn Heldens•, Ana-Lucia Varbanescu§, Alexandru Iosup‡¶

s.j.heldens@utwente.nl, a.l.varbanescu@uva.nl, a.iosup@vu.nl

§University of Amsterdam, ‡VU University Amsterdam, •University of Twente, ¶Delft University of Technology

Main Idea

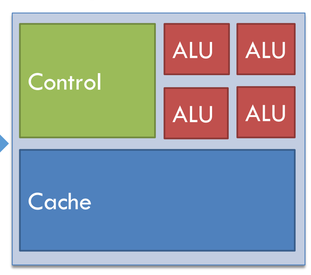
Processing massive graphs remains computationally expensive. We propose a graph processing system for hybrid platforms which delivers performance by using the CPU and the GPU *concurrently*. Its core feature is smart data distribution that enables dynamic workload scheduling, contrary to the state-of-the-art approach of statically partitioning the graph beforehand.

Challenges

Graphs processing means:

- Massive and diverse datasets.
- Fine-grained parallelism.
- Irregular workload.
- Random memory accesses.
- Low arithmetic intensity.

Graph Processing: CPU or GPU? Use Both!



CPU architecture

Studies [2] show there is no clear answer to which device is best for graph processing. **Let's use both!**

GPU architecture

ID	Name	Origin	Vertices	Edges	Size
R1	LiveJournal	Social	4.85M	69M	0.5GB
R2	Orkut	Social	3.07M	117M	0.9GB
R3	Wikipedia	WWW	12.1M	378M	2.8GB
G24	RMAT	Synt.	8.9M	260M	1.9GB
G25	RMAT	Synt.	17.1M	524M	3.9GB
G26	RMAT	Synt.	32.8M	1050M	7.8GB

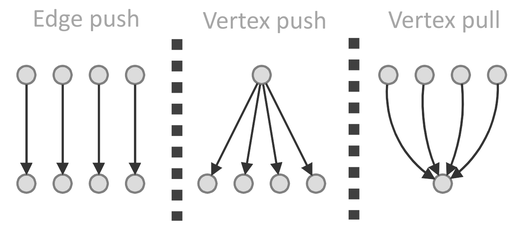
Design of HyGraph

A High-Level Model

HyGraph uses a **bulk-synchronous vertex-centric** model ("*think like a vertex*" [5]). An algorithm is defined using our API in C++.

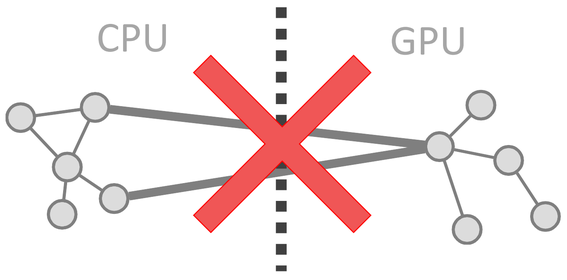
```
1 class VertexProgram<V,E> {
2   inline send_msg(…);
3   inline process_edge(…);
3  inline combine_msg(…);
4  inline process_vertex(…);
5 }
```

From this definition, **HyGraph** generates code for CPU and GPU, supporting various data structures (**vertex**/**edge**) and access patterns (**push**/**pull**).

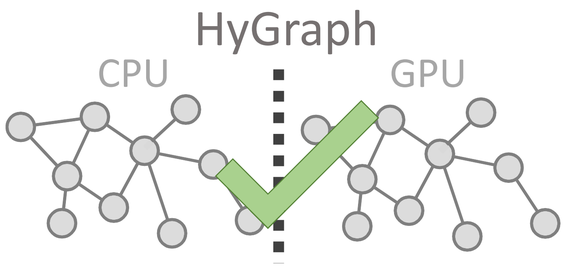


B Smart Data Distribution

Current systems [4] for hybrid processing use **static partitioning,** forcing users to fix the distribution.

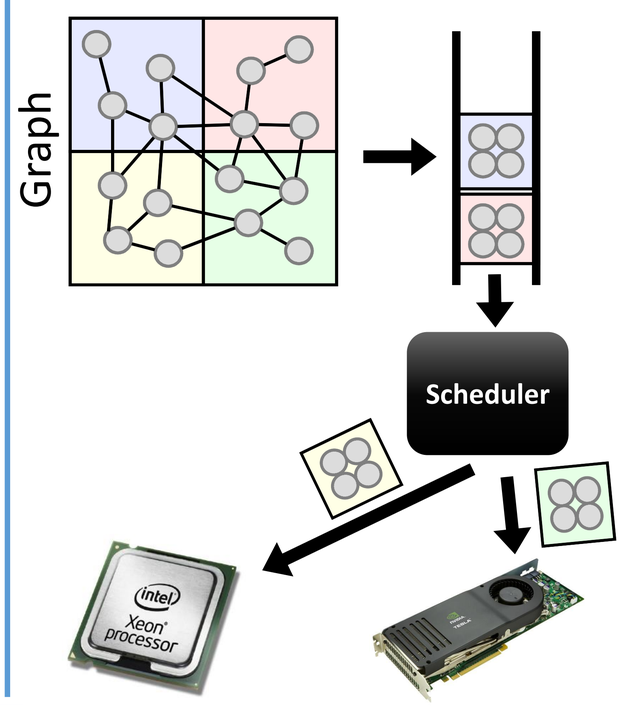


HyGraph has **smart caching** of vertices in the memory of the devices, enabling dynamic redistribution of work.



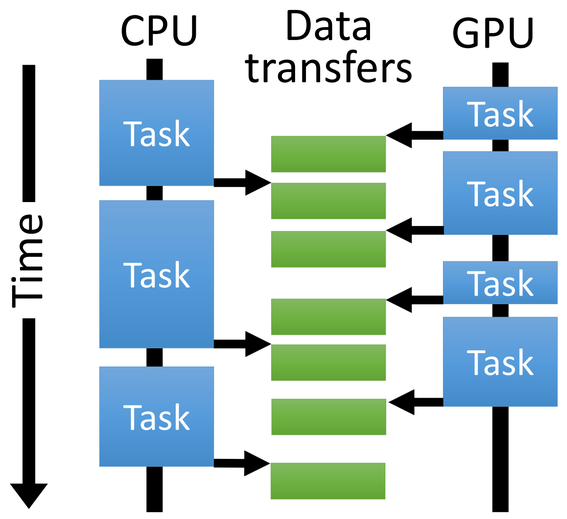
C Dynamic Load Balancing

HyGraph uses a customizable global **scheduler** to assign work to the devices at run-time. Both **static** and **dynamic** scheduling are supported out of the box.



D Efficient Communication

Current systems alternate computation with communication by copying the vertex state between devices after each superstep, which is costly. **HyGraph** removes the data transfer overhead by overlapping **communication** and **computation**.



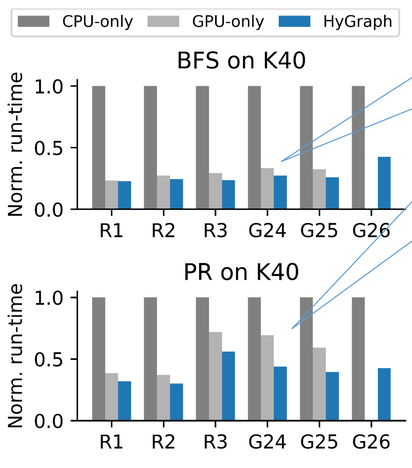
Results

HyGraph implemented in **C++11** and **CUDA**. Evaluated on **DAS5** [6] using data from **SNAP** [3]. Results shown only for **NVIDIA K40** (Kepler, 12GB).

Results for two algorithms presented (out of four):

- **Breadth-first search (BFS):** Find length of paths from root.
- **PageRank (PR)**: Used for ranking search engines results.

Our solution is faster than using only the CPU or only the GPU!

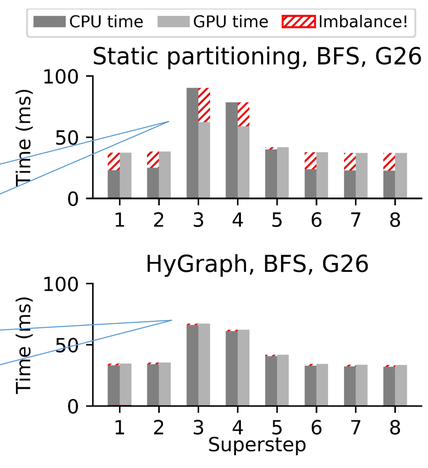


Up to 4.3x speedup over CPU-only. Up to 1.6x speedup over GPU-only

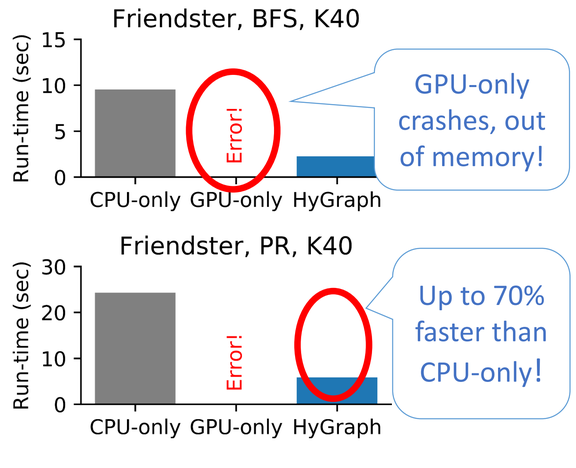
Naive static approach → load imbalance → performance loss.

Dynamic scheduling removes imbalance → better performance!

Our dynamic approach performs better than static partitioning!



Easily handles massive datasets (Friendster[3], ~2B edges, 13GB)!



Learn more?

See reference [1]

s.j.heldens@utwente.nl

github.com/atlarge-research/HyGraph

References

[1] S. Heldens, A. L. Varbanescu and A. Iosup, "Dynamic Load Balancing for High-Performance Graph Processing on Hybrid CPU-GPU Platforms," 6th Workshop on Irregular Applications: Architecture and Algorithms (IA3), 2016. (**goo.gl/6yLzHp**)

[2] Varbanescu et al., "Can Portability Improve Performance? An Empirical Study of Parallel Graph Analytics", ICPE, 2015.

[3] Jure Leskovec and Andrej Krevl, "SNAP Datasets: Stanford Large Network Dataset Collection", http://snap.stanford.edu/data

[4] Gharaibeh et al., "A Yoke of Oxen and a Thousand Chickens for Heavy Lifting Graph Processing", PACT, 2012.

[5] McCune et al., "Thinking like a vertex: a survey of vertex-centric frameworks for large-scale distributed graph processing." ACM Computing Surveys, 2015.

[6] Henri Bal et al., "A Medium-Scale Distributed System for Computer Science Research: Infrastructure", IEEE Computer, 2016.