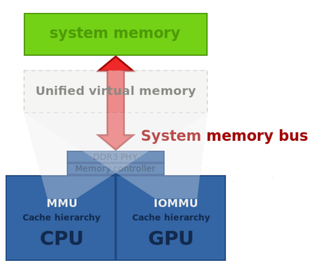
Energy-Efficient Graph Traversal on Integrated CPU-GPU Architecture

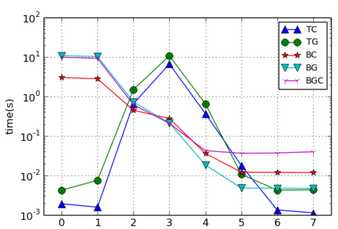
Heng Lin Jidong Zhai Wenguang Chen
linheng11@mails.tsinghua.edu.cn,{zhaijidong,cwg}@tsinghua.edu.cn
Tsinghua University

Motivation

- Energy become a key challenge recently. To deal with it, integrated CPU-GPU architecture is designed to reduce the data transfer cost between CPUs and GPUs.
- Graph applications are becoming increasingly important for big data analysis, Breadth-First Search(BFS) is the most representative one.
- Despite previous efforts, it remains an important problem to get optimal performance for BFS on integrated architectures.
- GPU can use the whole main memory, which is much larger than conventional discrete GPU



Architecture of AMD A10-7850K



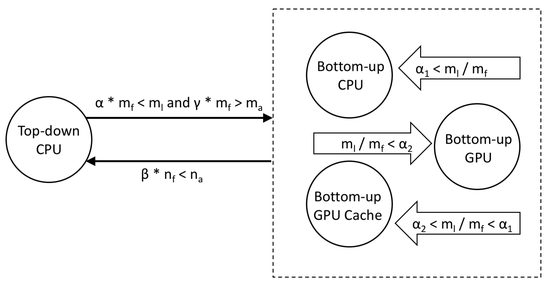
An comparison of different implementation. In this example, best combination can outperform simple *topdown_cpu* plus *bottomup_gpu* about 20%

Key Ideas

- Implement state-of-the-art algorithm both for CPU and GPU.
- Find an adaptive algorithm to atomically find the optimal combination of search algorithm and executing device for each level of BFS.

Item	Description
TC	Top-down direction + CPU
BC	Bottom-up direction + CPU
BG	Bottom-up direction + GPU
BGC	Bottom-up direction + GPU + Software Cache

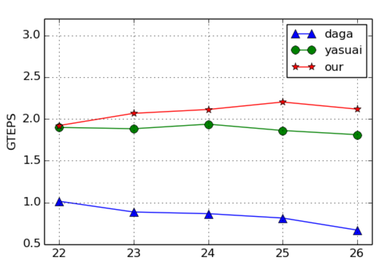
Candidate implementation for choice, some other combinations are omit because of performance.



Switch policy for candidate implementation. m_f is edge number in frontier, m_l is edge number left to visit, m_a is total edge number, n_a is total vertex number, and n_f is vertex in frontier. In our environment, we choose $\alpha = 100$, $\alpha_1 = 40$, $\alpha_2 = 5$, $\beta = 100$, $\gamma = 1000$.

Selected Methods

1. Graph layout. There are special treatments for zero-degree vertex and high-degree vertex.[1]
2. Direction optimization. Using bottom-up direction aside conventional top-down direction.[2]
3. Software cache for auxiliary data structure.
4. Devices choice.

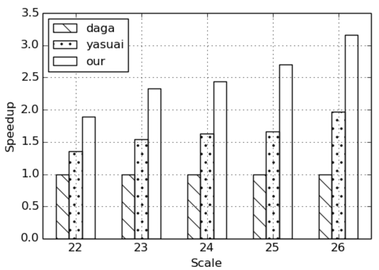


Our algorithm is weak scaled and better than Daga's[3] or Yasuai's[2]. Our algorithm is3.1X compare to Daga's and 1.2X compare to Yasuai's as largest runnable scale. Note at scale 26 the algorithm doesn't follow the former scalability because one of associatedata structure exceeds last level cache in CPU.

[1] Y. Yasui, K. Fujisawa, and Y. Sato. Fast and energy-efficient breadth-first search on a single numa system. ISC 2014.
[2] S. Beamer, K. Asanovi˙c, and D. Patterson. Direction-optimizing breadth-first search. SC 2012.
[3] M. Daga, M. Nutter, and M. Meswani. Efficient breadth-first search on a heterogeneous processor. Big Data 2014.

Environment

- GCC 4.8.2, OpenCL 2.0
- AMD A10-7850K Radeon R7, 4 CPU cores(3.7GHz), 8 GPU cores(0.7GHz)
- 32GB memory
- Graph generator is kronerker and edge_factor is set to 16.



In terms of energy, which measured by TESP/Watt, Our method is even more efficient because GPU is more energy efficiency than CPU. Our algorithm yields a speedup of 3.1X compare to Daga's and 1.6X compare to Yasuai's.

	Vertex	Edge	edge_factor	Diameter	MTEPS	MTEPS/W
kron	67.11M	1073.74M	16	12	2140.75	17.83
rmat	67.11M	1073.74M	16	12	1963.21	16.36
usa-road	23.95M	58.33M	2.4	8,098	50.55	0.31
friendster	65.61M	1,806.07M	27.5	25	397.55	3.32
livejournal	4.85M	68.99M	14.2	16	469.94	3.92
twitter	61.58M	1,468.37M	23.8	16	1148.48	9.57
wiki-talk	2.39M	5.02M	2.1	8	113.09	0.94

Experiments in various dataset including synthesized and real world datasets.

Conclusion and Future Work

The appearance of APU gives us an opportunity to use CPU and GPU concurrently without data transfer overhead. This paper discusses various BFS optimization methods and implements it efficiently in APU. We raise a switch policy to automatically choose the best implementation at runtime. At last, the algorithm is tested both in synthesized and real world datasets to show the efficiency in time and energy, finally get 1.6X speedup compared to previous state-of-the-art algorithms in energy consumption.

In the future, we will explore how different methods affect the BFS and try to optimize more application under this guide.